SAS® 9.1 Companion for UNIX Environments

The Power to Know®

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2004. *SAS® 9.1 Companion for UNIX Environments*. Cary, NC: SAS Institute Inc.

SAS® 9.1 Companion for UNIX Environments

ISBN 1–59047–210–1

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

1st printing, January 2004

SAS Publishing provides a complete selection of books and electronic products to help customers use SAS software to its fullest potential. For more information about our e-books, e-learning products, CDs, and hard-copy books, visit the SAS Publishing Web site at **support.sas.com/pubs** or call 1-800-727-3228.

Contents

What's New

Overview

New and enhanced features for Base SAS improve ease of use and SAS performance under the UNIX operating environment:

- SAS for the AIX, HP-UX, and Solaris operating environments is 64-bit only.
- SMTP (Simple Mail Transfer Protocol) is now the default mail handler.
- Sharing files between UNIX and Windows has been simplified.
- Using the MODULE family of SAS functions and CALL routines, you can invoke a routine that resides in an external shared library from within SAS.
- Universal Printing is the new default printing mechanism. SAS does not support host printing functionality.

Note:

- This section describes the features of SAS software under the UNIX operating environment that are new or enhanced since SAS 8.2.
- z/OS is the successor to the OS/390 operating system. Throughout this document, any reference to z/OS also applies to OS/390, unless otherwise stated.

△

Migrating 32-Bit SAS Files to 64–Bit SAS Files

Starting in SAS 9, SAS for the AIX, HP-UX, and Solaris operating environments is 64-bit only. Consequently, some SAS files (such as your SAS catalogs) that were created in 32-bit releases of SAS cannot be read by the V9 engine. You can read and write to your 32-bit SAS data sets, SAS/ACCESS views from Oracle or SYBASE, SQL views, or MDDB files from a 64-bit SAS session using CEDA. However, you cannot update these files. For more information, see "Migrating 32-Bit SAS Files to 64-Bit in UNIX Environments" on page 106.

You can use the MIGRATE procedure to convert all of your SAS files to 64-bit. For more information about the MIGRATE procedure, see the Migration Community at **`support.sas.com/rnd/migration`**.

Note: If you use Remote Library Services (RLS) to access SAS files on a server, see the *SAS/CONNECT User's Guide* for information about accessing Version 6 SAS files. △

Accessing SAS Files from Previous Releases

On the 64-bit AIX, HP-UX, and Solaris platforms, the V6 and V6TAPE read-only engines provide read-only access to your Release 6.12 data sets. See *SAS Language Reference: Concepts* for more information about the compatibility of V6 files with SAS 9.1.

With the Tru64 and Linux platforms, you still have read and write access to your Release 6.12 data sets. For more information about compatibility issues, see "Accessing SAS Files across Compatible Machine Types in UNIX Environments" on page 108.

Restricted System Options

Your site administrator can specify SAS system options for your site, a specific group, or an individual user in a restricted configuration file . Because these options are restricted, you cannot change the specified value. Use the new RESTRICT option in the OPTIONS procedure to see all the system options that have been restricted. For more information about how SAS processes the restricted configuration file, see "Order of Precedence for SAS Configuration Files" on page 17. For information about creating a restricted configuration file, see the *SAS System Configuration Guide for UNIX*.

Executing UNIX Commands within a SAS Session

umask is added to the list of UNIX commands **cd**, **pwd**, or **setenv** that SAS checks before executing the SAS equivalent in a session when you use the X command, X statement, CALL system routine, or %SYSEXEC. For more information, see "Executing a Single UNIX Command" on page 13.

Sending E-mail from within Your SAS Session

- The default mail handler is SMTP (Simple Mail Transfer Protocol), which supports attachments. For more information, see "Initializing Electronic Mail" on page 144.
- The new BCC option in the FILENAME statement enables you to send blind copy e-mails during a SAS session. For more information, see "Syntax of the FILENAME Statement for Electronic Mail" on page 144.
- Using the Send Mail dialog box, you can now do the following:
  - include the contents of an active SAS text window (such as the Program Editor or Log) in the body of your e-mail. For more information, see "Sending the Contents of a Text Window" on page 48.
  - attach the contents of a non-text window to your e-mail. Examples of non-text windows include a graph generated by SAS/GRAPH and an image in your PROC REPORT output. For more information, see "Sending the Contents of a Non-Text Window" on page 49.

Accessing Shared Executable Libraries from SAS

Shared libraries in UNIX contain executable programs that are written in various programming languages. These libraries store useful routines that might be needed by many applications. Using the MODULE family of SAS functions and CALL routines, you can invoke a routine that resides in an external shared library from within SAS. You can access the shared library routines by using a DATA step, the IML procedure, and SCL code. See Chapter 7, "Accessing Shared Executable Libraries from SAS," on page 169.

Changes to the cleanwork Command

The **cleanwork** command can now be used to delete utility directories whose associated SAS process has ended. For more information, see "cleanwork Command" on page 400.

SAS Resources

- The **SAS.useNativeXmTextTranslations** X resource specifies whether any XmText widget translations are inherited by all instances of the Text, Combo Box, and Spin Box widgets that are used by the SAS X Motif user interface. For more information, see "Miscellaneous Resources in UNIX Environments" on page 96.
- The **SAS.webBrowser** resource is no longer supported. The **SAS.helpBrowser** resource now specifies the pathname of the World Wide Web browser for use when viewing the online help or when the WBROWSE command is issued. For more information, see "Miscellaneous Resources in UNIX Environments" on page 96.
- The following resources that were used to control your ODS results are no longer supported:
  - **SAS.resultsHTML**
  - **SAS.resultsUseWork**
  - **SAS.resultsTmpDir**
  - **SAS.resultsHTMLStyle**
  - **SAS.resultsListing**
  - **SAS.resultsAutoNavigate**

  To set the values for your ODS output, use the **Results tab** in the Preferences dialog box. For more information, see "Modifying the Results Settings" on page 60.

SAS Language Elements

Commands

The following commands are obsolete:

- DLGPRT

- DLGPRTMODE
- DLGPRTPREVIEW
- DLGPRTSETUP.

Functions and CALL Routines

- To call a specific routine or module that resides in a shared library, you can use the MODULE function. For more information, see "MODULE Function" on page 251.
- You can store the contents of a memory address in a numeric variable on 32-bit and 64-bit platforms by using the PEEKLONG function. This function replaces the PEEK function, which was valid only on 32-bit platforms. For more information, see "PEEKLONG Function" on page 254.

Statements

- The following option is new in the FILE, FILENAME, and INFILE statements:
  - TERMSTR= enables the sharing of UNIX and PC formatted files.

  For more information, see "FILE Statement" on page 291, "FILENAME Statement" on page 293, and "INFILE Statement" on page 299.
- The following options are new in the %INCLUDE statement:
  - BLKSIZE= specifies the number of bytes that are physically read or written in an I/O operation.
  - ENCODING= specifies the encoding to use when reading from the specified source.
  - LRECL= specifies the record length (in bytes).
  - RECFM= controls the record format.

  For more information, see "%INCLUDE Statement" on page 298.

Procedures

- To see all the system options that have been set by your site administrator, use the RESTRICT option in the OPTIONS procedure. For more information, see "OPTIONS Procedure" on page 279.
- The BMDP procedure is obsolete.

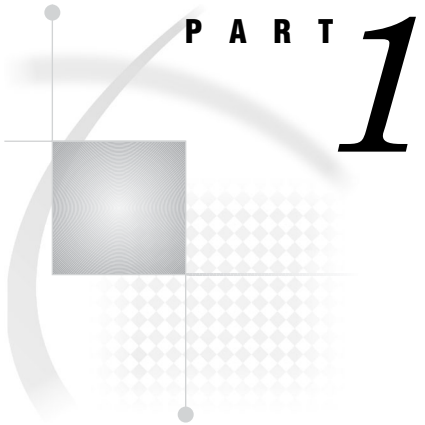
System Options

- The following system options are new:
  - You can specify the location of the Program Editor autosave file by using the AUTOSAVELOC system option. For more information, see"AUTOSAVELOC System Option" on page 317.
  - If you create a customized table of contents and index for the SAS Help and Documentation, use the HELPINDEX and HELPTOC system options to specify the file location. For more information, see "HELPTOC System Option" on page 335 and "HELPINDEX System Option" on page 333.

- SSLCALISTLOC, SSLCERTLOC, SSLCLIENTAUTH, SSLCRLCHECK, SSLCRLLOC, SSLPVTKEYLOC, and SSLPVTKEYPASS are new system options that support Secure Sockets Layer (SSL) authentication. For more information, see "SSLCALISTLOC System Option" on page 369, "SSLCERTLOC System Option" on page 370, "SSLCLIENTAUTH System Option" on page 371, "SSLCRLCHECK System Option" on page 372, "SSLCRLLOC System Option" on page 373, "SSLPVTKEYLOC System Option" on page 373, and "SSLPVTKEYPASS System Option" on page 374.
- To set permissions for the temporary Work library when it is created, use the WORKPERMS system option. For more information, see "WORKPERMS System Option" on page 382.

- The following system options are enhanced:
  - If you specify only a directory path for the ALTLOG, LOG, ALTPRINT, or PRINT system options during SAS invocation, then the default filename for your log or procedure output file is *filename*.LOG or *filename*.LST, where *filename* is the name of your SAS job. For more information, see "ALTLOG System Option" on page 313, "LOG System Option" on page 340, "ALTPRINT System Option" on page 314, and "PRINT System Option" on page 351.
  - SMTP (Simple Mail Transfer Protocol) is the new default for the EMAILSYS system option. For more information, see "EMAILSYS System Option" on page 326.
  - V9 is a new value for the ENGINE system option. For more information, see "ENGINE System Option" on page 327.
  - MAX is the new default for the SORTSIZE system option. The value of MAX is based on your operating environment. For more information, see "SORTSIZE System Option" on page 368.
- The following system options have values that are obsolete:
  - Because the CoSort utility is no longer supported, **`cosort`** is not a valid value for the SORTNAME system option. For more information, see "SORTNAME System Option" on page 367.
- The following system options are obsolete:
  - PROCLEAVE
  - SORTLIB
  - SYSLEAVE
  - XPRINTNM.

The UNBUFLOG system option has been replaced by the LOGPARM system option, which is available in all operating environments. For details, see *SAS Language Reference: Dictionary*



PART 1

Running SAS Software Under UNIX

CHAPTER

1

Getting Started with SAS in UNIX Environments

Starting SAS Sessions in UNIX Environments

Invoking SAS

The command that you use to invoke your SAS session is defined during the SAS installation process and is added to the list of commands that are recognized by the operating environment. Ask your system administrator what the command is that invokes SAS at your site. At many sites, the command to invoke SAS is simply **sas**, but a different command might have been defined during the SAS installation process at your site. This documentation assumes that SAS is invoked by the **sas** command.

Note: Before you start your SAS session, review the different techniques for interrupting and terminating your SAS session (see "Interrupting or Terminating Your SAS Session in UNIX Environments" on page 22). Also, if you cannot stop your session, contact your system administrator; do not turn off your machine, especially if your machine is part of a network. △

Syntax of the SAS Command

The general form of the SAS command is as follows:

sas <*-option1…-option-n*> <*filename*>

You can use these arguments with the SAS command:

-option1 ... -option-n
specifies SAS system options to configure your session or X command line options. See Chapter 17, "System Options under UNIX," on page 311 and "X Command Line Options" on page 11 for more information. If you omit any options (either on the command line or in the configuration file), the SAS (or site-specific) default options are in effect.

filename
specifies the name of the file containing the SAS program to be executed. Specifying a filename on the SAS command invokes a batch SAS session. Omit the filename to begin an interactive session.

If the file is not in the current directory, specify its full pathname.

Example: Invoking an Interactive SAS Session

To invoke an interactive SAS session, without specifying any SAS system options, enter

```
sas
```

The execution mode will depend on your default settings. For more information, see "Selecting a Method of Running SAS in UNIX Environments" on page 6.

To specify the NODATE and LINESIZE system options, you could enter

```
sas -nodate -linesize 80
```

To run a SAS program and pass parameters to it, enter

```
sas -sysparm 'A B C' progparm.sas
```

The value **A B C** is assigned to the SYSPARM macro variable, which can be read by the program ProgParm.sas.

What If SAS Does Not Start?

If SAS does not start, the SAS log might contain error messages that explain the failure. However, error messages that SAS issues before the SAS log is initialized are written to the SAS console log.

Under UNIX, the STDOUT fileref specifies the location of the console log.

Running SAS in a Foreground or Background Process

UNIX is a multitasking system, so you can run multiple processes at the same time. For example, you can have one process running in the foreground and three in the background. A *foreground process* executes while you wait for the prompt; that is, you cannot execute additional commands while the current command is being executed. After you enter a command, the shell starts a process to execute the command. After the system executes the command, the shell displays the prompt and you can enter

additional commands. A *background process* executes independently of the shell. After you enter a command, the shell starts a process to execute the command and then issues the system prompt. You can enter other commands or start other background tasks without waiting for your initial command to execute. You can run SAS in the foreground or in the background.

Note: Both the C shell and the Korn shell include commands that enable you to move jobs among three possible states: running in the foreground, running in the background, and suspended. △

Selecting a Method of Running SAS in UNIX Environments

You can run SAS in the following modes:

- SAS windowing environment
- interactive line mode
- batch mode.

Ask your system administrator which interface or mode of operation is the default at your site.

SAS Windowing Environment in UNIX Environments

Introduction to the SAS Windowing Environment

You interact with SAS through windows using your keyboard, mouse, pull-down menus, pop-up menus, and icons. The windowing environment includes, but is not limited to, the Explorer, Program Editor, Output, Log, and Results windows.

Your SAS session may default to the windowing environment interface. If you want to use the windowing environment, you can start your SAS session as a foreground process or as a background process (by adding an ampersand (&) to your SAS command line).

For more information about using the windowing environment, see Chapter 2, "Working in the SAS Windowing Environment," on page 29.

Note: If you are not using an X display, then you need to invoke SAS in interactive line mode using the NODMS system option. For more information, see "Interactive Line Mode in UNIX Environments" on page 7. △

What Is the Explorer Window?

Explorer is a windowing environment for managing basic SAS software tasks such as viewing and managing data sets, libraries, members, applications, and output. The SAS Explorer is a central access point from which you can do the following:

- manipulate SAS data through a graphical interface
- access the Program Editor, Output, and Log windows (as well as other windows)
- view the results of SAS procedure output in the Results window
- import files into SAS.

What Are the Program Editor, Output, and Log Windows?

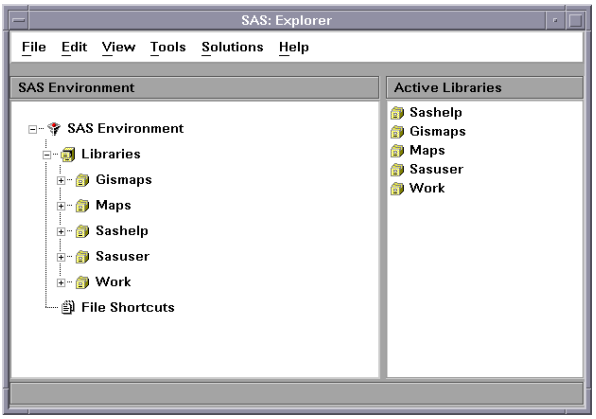
The Program Editor, Output, and Log windows enable you to edit and execute SAS programs and display output. For more information about these windows, see SAS Help and Documentation.

Invoking SAS in the Windowing Environment

You can use the following commands to specify which windows open when the SAS session starts.

- You can open the Explorer window by specifying the EXPLORER system option:

```
sas -explorer
```



- You can open the Program Editor, Output, and Log windows by specifying the DMS system option:

```
sas -dms
```

- You can use the DMSEXP system option to open the Program Editor, Output, Log, and Results windows and the Explorer:

```
sas -dmsexp
```

SAS also opens the toolbox from which you can open additional SAS windows. For more information about the toolbox, see to Chapter 2, “Working in the SAS Windowing Environment,” on page 29.

Exiting SAS in the Windowing Environment

To end your SAS session, enter the BYE or ENDSAS command on the command line or select

File ▶ Exit

from the pull-down menu of the session that you want to end.

Interactive Line Mode in UNIX Environments

Introduction to Interactive Line Mode

You enter SAS statements line by line in response to prompts issued by SAS. SAS reads the source statements from the terminal as you enter them. DATA and PROC steps execute when

- a RUN, QUIT, or DATALINES statement is entered
- another DATA or PROC statement is entered
- the ENDSAS statement is entered.

To use interactive line mode, you must run SAS in the foreground.

Invoking SAS in Interactive Line Mode

To start an interactive line mode session, invoke SAS with the NODMS or NODMSEXP system option:

```
sas -nodms
sas -nodmsexp
```

By default, SAS log and procedure output (if any) appear on your display as each step executes.

After you invoke SAS, the **1?** prompt appears, and you can begin entering SAS statements. After you enter each statement, a line number prompt appears.

Exiting SAS in Interactive Line Mode

You can end the session by pressing the EOF key (usually CTRL+D; see "Using Control Keys" on page 23) or by issuing the ENDSAS statement:

```
endsas;
```

Batch Mode in UNIX Environments

Introduction to Running SAS in Batch Mode

To run SAS in batch mode, you specify your SAS application name in the SAS command. You can run batch mode in the foreground, in the background by specifying an ampersand at the end of the SAS command, or submit your application to the batch queue by using the **batch**, **at**, **nohup**, or **cron** UNIX commands. (For more information, refer to the UNIX man pages for the **batch**, **at**, **nohup**, or **cron** commands.) If you start your application with one of these UNIX commands and you log off of your system, then your application will complete execution. If your application contains statements that start an interactive procedure such as FSEDIT, then you need to run your batch application in the foreground.

Invoking SAS in Batch Mode

To invoke SAS in batch mode, you must specify a filename in the SAS command. For example, if Weekly.rpt is the file containing the SAS statements to be executed, and you want to specify the NODATE and LINESIZE system options, you would enter

```
sas weekly.rpt -nodate -linesize 90
```

The command would run the program in the foreground. If you want to run the program in the background, add the ampersand to the end of the command:

```
sas weekly.rpt -nodate -linesize 90 &
```

You do not need to specify the SYSIN option as with some other platforms.

SAS creates a .log file and a .lst file in the current directory that contains the log and procedure output.

Submitting a Program to the Batch Queue

To submit your program to the batch queue, you can use the **batch**, **at**, **nohup**, or **cron** commands. For example, you could submit Weekly.rpt from your shell prompt as follows:

```
$ at 2am
sas weekly.rpt
<control-D>
warning: commands will be executed using /usr/bin/sh
job 8400.a at Wed Jun 11 02:00:00 2003
$
```

If you create a file that contains the SAS command necessary to run your program, for example CmdFile.sas, then you can enter the following command at your shell prompt:

```
at 2am < cmdfile.sas
```

SAS sends the output to a file that has the same name as the program and an extension of .lst, and the log goes to a file with an extension of .log. Both of these files are written to your current directory. Refer to the man pages for these commands for more information on submitting jobs to the batch queue. For more details on routing output, see Chapter 6, "Printing and Routing Output," on page 153.

Note: If your program contains statements that start an interactive procedure such as the FSEDIT procedure, you will need to run your program in the foreground. △

Writing Data from an External File Using Pipes

You can use a pipe to write data from an external file to a SAS program. For example, suppose that your data resides in the file MyData and your program MyProg.sas includes this statement:

```
INFILE STDIN;
```

Issue this command to have MyProg.sas read data from MyData:

```
cat mydata | sas myprog.sas
```

For details about using external files, see Chapter 5, "Using External Files and Devices," on page 131. See also "File Descriptors in the Bourne and Korn Shells" on page 140 for another way to have a SAS program read data from an external file.

Running SAS on a Remote Host in UNIX Environments

Introduction to Running SAS on a Remote Host

When you invoke SAS in an interactive mode, you can run SAS on your local host, or you can run SAS on a remote host and interact with the session through an X server

running on your workstation. The server provides the display services that are needed for the X Window System.

Most of the time, the server name is derived from the machine's name. For example, if your machine is named **green**, the name of the server is **green:0.0**. In most cases, the X server will already be running when you log in. If you need to start your server manually, consult the documentation that is provided with your X Window System software.

To run SAS on a remote host, you must tell SAS which display to use by either setting the DISPLAY environment variable or specifying the **-display** X command line option.

Steps for Running SAS on a Remote Host

To run SAS on a remote host, follow these steps:

1. Make sure that the clients running on the remote host have permission to connect to your server. Most systems control this by using the **xhost** client. Other systems control access through a session manager. To use the **xhost** client to permit all remote hosts to connect to your server, enter the following command at the system prompt on the system that is running your X server:

   ```
   xhost +
   ```

   To run this command automatically each time you log in, enter this command in a file named **.xhost**.

   If your system does not control access with the **xhost** client, consult your system documentation for information on allowing remote access.

2. Log in to the remote system, or use a remote shell.

3. Identify your server as the target display for X clients that are run on the remote host. You can do this in one of two ways:

   a. Set the DISPLAY environment variable. In the Bourne and Korn shells, you can set the DISPLAY variable as follows:

      ```
      DISPLAY=green:0.0
      export DISPLAY
      ```

      In the Korn shell, you can combine these two commands:

      ```
      export DISPLAY=green:0.0
      ```

      In the C shell, you must use the **setenv** command:

      ```
      setenv DISPLAY green:0.0
      ```

      The DISPLAY variable will be used by all Xclients on the system.

      Note: To determine the shell for your system, type **ps** at the command prompt or check the value of the SHELL environment variable. △

   b. Use the **-display** option. For example:

      ```
      sas -display green:0.0
      ```

      If you have trouble establishing a connection, you can try using an IP address instead of a display name, for example:

      ```
      -display 10.22.1.1:0
      ```

      Note: This option is a command line option for the X Window system, not for SAS. Specifying this option in a SAS configuration file or in the SASV9_OPTIONS environment variable might cause problems when you are running other interfaces. △

Preventing SAS from Attempting to Connect to the X Server

To prevent SAS from attempting to connect to the X server, unset the DISPLAY environment variable and use the **-noterminal** option on the command line.

Troubleshooting Connection Problems

If SAS cannot establish a connection to your display, it prints a message that indicates the nature of the problem and then terminates. An example of a message that you might receive is the following:

```
ERROR:  Cannot open Xdisplay.  Check display name/server access authorization.
```

Make sure that you have brought up the SAS session correctly. You might need to use the **xhost** client (enter **xhost +**) or some other method to change display permissions. You can also specify the NODMS system option when you invoke SAS to bring your session up in line mode.

If you are unable to invoke SAS, try running another application such as **xclock**. If you cannot run the application, you might need to contact your system administrator for assistance.

X Command Line Options

How to Specify X Window System Options

When you invoke some X clients, such as SAS, you can use command line options that are passed to the X Window System. In general, you should specify X Window System options after SAS options on the command line.

Supported X Command Line Options

The following list describes the X command line options that are available when you invoke a SAS session from the command prompt.

-display *host*:*server*.*screen*
specifies the name or IP address of the terminal on which you want to display the SAS session. For example, if your display node is wizard, you might enter

```
-display wizard:0.0
```

or

```
-display 10.22.1.1:0
```

-name *instance-name*
reads the resources in your SAS resource file that begin with *instance-name*. For example, **-name MYSAS** reads the resources that begin with **MYSAS**, such as

```
MYSAS.dmsfont: Cour14
MYSAS.defaultToolbox: True
```

-noterminal
specifies that you do not want to display the SAS session. You must specify this option to generate a graph in batch mode. For more information, see "Running SAS/GRAPH Programs" in *SAS/GRAPH Reference, Volumes 1 and 2*.

Note: To prevent SAS from attempting to connect to the X server, unset the DISPLAY environment variable and use the -noterminal option on the command line. △

-title *string*
specifies the title up to six characters long for your SAS session window. To use multiple words in the title, enclose the words in single or double quotation marks. For example, **`-title MYSAS`** produces **`MYSAS:Explorer`** in the title bar of the Explorer window.

-xrm *string*
specifies a resource to override any defaults. For example, the following resource turns off the Confirm dialog box when you exit SAS:

```
-xrm 'SAS.confirmSASExit: False'
```

Unsupported X Command Line Options

SAS does not support the following X command line options because their functionality is not applicable to SAS or is provided by SAS resources. Refer to "Overview of X Resources" on page 55 for more information on SAS resources.

-geometry
Window geometry is specified by the **`SAS.windowHeight`**, **`SAS.windowWidth`**, **`SAS.maxWindowHeight`**, and **`SAS.maxWindowWidth`** resources.

-background, -bg
These options are ignored.

-bordercolor, -bd
These options are ignored. Refer to "Defining Colors and Attributes for Window Elements (CPARMS)" on page 88 for a description of specifying the color of window borders.

-borderwidth, -bw
These options are ignored. The width of window borders is set by SAS.

-foreground, -fg
These options are ignored.

-font, -fn
SAS fonts are specified by the **`SAS.DMSFont`**, **`SAS.DMSboldFont`**, and **`SAS.DMSfontPattern`** resources.

-iconic
This option is ignored.

-reverse, -rv, +rv
These options are ignored. Refer to "Defining Colors and Attributes for Window Elements (CPARMS)" on page 88 for a description of specifying reverse video.

-selectionTimeout
Timeout length is specified by the **`SAS.selectTimeout`** resource.

-synchronous, +synchronous
The XSYNC command controls the X synchronization.

-xn1language
This option is ignored.

Executing Operating System Commands from Your SAS Session

Deciding Whether to Run an Asynchronous or Synchronous Task

You can execute UNIX commands from your SAS session either asynchronously or synchronously. When you run a command as an *asynchronous* task, the command executes independently of all other tasks that are currently running. To run a command asynchronously, you must use the SYSTASK statement. See "SYSTASK Statement" on page 305 for information about executing commands asynchronously.

When you execute one or more UNIX commands *synchronously*, then you must wait for those commands to finish executing before you can continue working in your SAS session. You can use the CALL SYSTEM routine, %SYSEXEC macro program statement, X statement, and X command to execute UNIX commands synchronously. The CALL SYSTEM routine can be executed with a DATA step. The %SYSEXEC macro statement can be used inside macro definitions, and the X statement can be used outside of DATA steps and macro definitions. You can enter the X command on any SAS command line. See "CALL SYSTEM Routine" on page 239 and "Macro Statements in UNIX Environments" on page 265 for more information.

Executing a Single UNIX Command

To execute only one UNIX command, you can enter the X command, X statement, CALL SYSTEM routine, or %SYSEXEC macro statement as follows:

X *command*

X *command*;

CALL SYSTEM ('*command*');

%SYSEXEC *command*;

Note: When you use the %SYSEXEC macro statement, if the UNIX command you specify includes a semicolon, you must enclose the UNIX command in a macro quoting function. Refer to *SAS Macro Language: Reference* for more information on quoting functions. △

Example 1: Executing a UNIX Command Using the X Statement

You can use the X statement to execute the **`ls`** UNIX command (in a child shell) as follows:

```
x ls -l;
```

Example 2: Executing a UNIX Command Using the CALL SYSTEM Routine

Inside a DATA step, you can use the CALL SYSTEM routine to execute a **cd** command, which will change the current directory of your SAS session:

```
data _null_;
call system ('cd /users/smith/report');
run;
```

The search for any relative (partial) filenames during the SAS session will now begin in the **/users/smith/report** directory. When you end the session, your current directory will be the directory in which you started your SAS session.

For more information about the CALL SYSTEM routine, see "CALL SYSTEM Routine" on page 239.

How SAS Processes a Single UNIX Command

When you specify only one command, SAS checks to see whether the command is **cd**, **pwd**, **setenv**, or **umask** and, if so, executes the SAS equivalent of these commands. The SAS **cd** and **pwd** commands are equivalent to their Bourne shell counterparts. The SAS **setenv** command is equivalent to its C shell namesake. The SAS **umask** command is equivalent to the numeric mode of the **umask** command supported by the Bourne, Korn, and C shells. These four commands are built into SAS because they affect the environment of the current SAS session. When executed by SAS software, they affect only the SAS environment and the environment of any shell programs started by the SAS session. They do not affect the environment of the shell program that began your SAS session.

If the command is not **cd**, **pwd**, or **setenv**, SAS starts a shell* in which it executes the command that you specified. If the command is **umask**, but you do not specify a *mask*, then SAS passes the command to the shell in which the current SAS session was started. For more information about the umask command, see "Changing the File Permissions for Your SAS Session" on page 15.

Executing Several UNIX Commands

You can also use the X command, X statement, CALL SYSTEM routine, and %SYSEXEC macro statement to execute several UNIX commands:

X '*command-1*;*...command-n*'

X '*command-1*;*...command-n*';

CALL SYSTEM ('*command-1*;*...command-n*');

%SYSEXEC *quoting-function*(*command-1*;*...command-n*);

Separate each UNIX command with a semicolon (;).

Note: When you use the %SYSEXEC macro statement to execute several UNIX commands, because the list of commands uses semicolons as separators, you must enclose the string of UNIX commands in a macro quoting function. Refer to *SAS Macro Language: Reference* for more information on quoting functions. △

Example: Executing Several Commands Using the %SYSEXEC Macro

The following code defines and executes a macro called **pwdls** that executes the **pwd** and **ls -l** UNIX commands:

* The shell used depends on the SHELL environment variable.

```
%macro pwdls;
%sysexec %str(pwd;ls -l);
%mend pwdls;
%pwdls;
```

This example uses **%str** as the macro quoting function.

How SAS Processes Several UNIX Commands

When you specify more than one UNIX command (that is, a list of commands separated by semicolons), SAS passes the entire list to the shell and does not check for the **cd**, **pwd**, **setenv**, or **umask** commands, as it does when a command is specified by itself (without semicolons).

For more information about how SAS processes the **cd**, **pwd**, **setenv**, or **umask** commands, see "How SAS Processes a Single UNIX Command" on page 14.

Starting a Shell

If you are not running in the SAS windowing environment, you can start a shell by not specifying any UNIX commands in the X statement:

X;

SAS responds with

```
Enter 'exit' to return to your SAS session.
```

SAS then starts a shell.

Enter any UNIX commands. When you are ready to return to the SAS session, enter the **exit** command.

Even if you changed directories while in the shell, you will be in the same directory as when you started the shell.

Changing the File Permissions for Your SAS Session

At invocation, a SAS session inherits the file permissions from the parent shell. Any file that you create will inherit these permissions. If you want to change the file permissions from within SAS, issue the **umask mask** command on the X statement. The **umask mask** command sets the permissions for any new file that you create. The value of **mask** can be either numeric or symbolic. For more information about this command, see the man page for umask)

When SAS executes the **umask mask** command, it changes the file permissions of the current SAS session, but it does not change the permissions in the parent shell. Any subsequent file that you create during this SAS session will inherit the permissions that you specified.

Executing X Statements in Batch Mode

If you run your SAS program in batch mode and if your operating system supports job control, the program will be suspended when an X statement within the program needs input from the terminal.

If you run your SAS program from the batch queue by submitting it with the **at** or **batch** commands, SAS processes any X statements as follows:

- If the X statement does not specify a command, SAS ignores the statement.

- If any UNIX command in the X statement attempts to get input, it receives an end-of-file (standard input is set to **/dev/null**).
- If any UNIX command in the X statement writes to standard output or standard error, the output is mailed to you unless it was previously redirected.

Customizing Your SAS Registry Files

SAS registry files store information about the SAS session. The SAS registry is the central storage area for configuration data for SAS. Some of the data stored in the registry includes

- the libraries and file shortcuts that SAS assigns at startup. These shortcuts could include secure information, such as your password.
- the printers that are defined for use and their print setup.
- configuration data for various SAS products.

The Sasuser registry file (called regstry.sas7bitm) contains your user defaults. These registry entries can be customized by using the SAS Registry Editor or by using PROC REGISTRY. For more information, see "The SAS Registry" in *SAS Language Reference: Concepts*.

CAUTION:
For experienced users only. Registry customization is generally performed by experienced SAS users and system administrators. △

Customizing Your SAS Session Using Configuration and Autoexec Files

You can customize your SAS environment in several ways. To customize your SAS environment at the point of invocation, you can use configuration and autoexec files. For information about how to customize a SAS session using the windowing environment, see Chapter 3, "Customizing the SAS Windowing Environment," on page 53.

Introduction to Configuration and Autoexec Files

You can customize your SAS session by defining configuration and/or autoexec files. You can use these files to specify system options and to execute SAS statements automatically whenever you start a SAS session. (SAS system options control many aspects of your SAS session, including output destinations, the efficiency of program execution, and the attributes of SAS files and data libraries. Refer to *SAS Language Reference: Dictionary* for a complete description of system options.)

The configuration file (for SAS 9.1) is typically named sasv9.cfg, and the autoexec file is named autoexec.sas. These files typically reside in the directory where SAS was installed. By default, this is the **!SASROOT** directory.

You can have customized configuration and autoexec files in your user home directory. If you do, then SAS will use the customizations specified in these files when you start a SAS session. For more information about the order of precedence SAS uses when processing configuration files, see "Order of Precedence for SAS Configuration Files" on page 17.

Differences between Configuration and Autoexec Files

The differences between configuration files and autoexec files are as follows:

- Configuration files can contain only SAS system option settings, while autoexec files can contain any valid SAS statement. For example, you might want to create an autoexec file that includes an OPTIONS statement to change the default values of various system options and LIBNAME and FILENAME statements for the SAS data libraries and external files that you use most often.
- Configuration files are processed *before* SAS initializes, while autoexec files are processed immediately *after* SAS initializes but before it processes any source statements. An OPTIONS statement in an autoexec file is equivalent to submitting an OPTIONS statement as the first statement of your SAS session.

Creating a Configuration File

To create a configuration file, follow these steps:

1. Use a text editor to write the SAS system options into a UNIX file. Save the file as either sasv9.cfg or .sasv9.cfg. (See "Order of Precedence for SAS Configuration Files" on page 17 for more information.)
2. Specify one or more system options on each line. Use the same syntax that you would use for specifying system options with the SAS command – except do not include the SAS command itself. For example, a configuration file might contain the following lines:

```
-nocenter
-verbose
-linesize 64
-work /users/myid/tmp
```

3. Close the configuration file.

Order of Precedence for SAS Configuration Files

SAS is shipped with a default configuration file in the **!SASROOT** directory. Your SAS Installation Representative can edit this configuration file so that it contains whichever options are appropriate to your site.

You can also create one or more of your own configuration files. SAS reads option settings from each of these files in the following order:*

1. sasv9.cfg in the **!SASROOT** directory. (See Appendix 1, "The !SASROOT Directory," on page 397.)
2. .sasv9.cfg in your home directory. (Notice the leading period.)
3. sasv9.cfg in your home directory.
4. sasv9.cfg in your current directory.
5. any restricted configuration files. Restricted configuration files contain system options that are set by the site administrator and cannot be changed by the user. Options can be restricted globally, by group, or by user. For more information about restricted configuration files, see *SAS System Configuration Guide for UNIX*.

SAS uses the last value it encounters for a system option. For example, if the WORKPERMS system option is specified in sasv9.cfg in the **!SASROOT** directory and in sasv9.cfg in your current directory, SAS will use the value specified in sasv9.cfg in your current directory.

* For future versions of SAS, the names of these files will change accordingly.

Specifying a Configuration File for SAS to Use

When you specify a configuration file for SAS to use, you bypass the search of the configuration files listed in "Order of Precedence for SAS Configuration Files" on page 17.

Note: SAS still processes any restricted configuration files that exist. The settings in these files take precedence over the settings in the configuration file that you specify. △

To specify a configuration file, complete one of the following steps:

- specify a configuration file with the CONFIG system option in the SAS command:

```
sas -config filename
```

- specify a configuration file in the SASV9_OPTIONS environment variable. See "Defining Environment Variables in UNIX Environments" on page 21. For example, in the Korn shell, you would use:

```
export SASV9_OPTIONS='-config filename'
```

- define the environment variable SASV9_CONFIG. See "Defining Environment Variables in UNIX Environments" on page 21. For example, in the Korn shell, you would use:

```
export SASV9_CONFIG=filename
```

filename is the name of a file containing SAS system options.

If you have specified a configuration file in the SASV9_OPTIONS or SASV9_CONFIG environment variables, you can prevent SAS from using that file by specifying NOCONFIG in the SAS command.

Customizing Your SAS Session Using System Options

You can customize your SAS environment in several ways. One way is through the use of SAS system options. For information about other ways to customize a SAS session, see Chapter 3, "Customizing the SAS Windowing Environment," on page 53.

Ways to Specify a SAS System Option

SAS options can be specified in one or more ways:

- in a configuration file
- in the SASV9_OPTIONS environment variable
- in the SAS command
- in an OPTIONS statement (either in a SAS program or an autoexec file)
- in the System Options window.

Table 17.3 on page 385 shows where each SAS system option can be specified.

Any options that do not affect the initialization of the SAS, such as CENTER and NOCENTER, can be specified and changed at any time.

Some options can be specified only in a configuration file, in the SASV9_OPTIONS variable, or in the SAS command. These options determine how SAS initializes its interfaces with the operating system and the hardware; they are often called configuration options. After you start a SAS session, these options cannot be changed. Usually, configuration files specify options that you would not change very often. In

those cases when you need to change an option just for one job, specify the change in the SAS command.

Overriding the Default Value for a System Option

The default values for SAS system options will be appropriate for many of your SAS programs. However, you can override a default setting using one or more of the following methods:

configuration file
: Modify your current configuration file (see "Order of Precedence for SAS Configuration Files" on page 17) or create a new configuration file. Specify SAS system options in the file by preceding each with a hyphen. For ON/OFF options, just list the keyword corresponding to the appropriate setting. For options that accept values, list the keyword identifying the option followed by the option value. All SAS system options can appear in a configuration file.

  For example, a configuration file might contain these option specifications:

```
-nocenter
-verbose
-linesize 64
```

SASV9_OPTIONS environment variable
: Specify SAS system options in the SASV9_OPTIONS environment variable before you invoke SAS. See "Defining Environment Variables in UNIX Environments" on page 21.

  Settings that you specify in the SASV9_OPTIONS environment variable affect SAS sessions that are started when the variable is defined.

  For example, in the Korn shell, you would use:

```
export SASV9_OPTIONS='-xwait -nodate'
```

SAS command
: Specify SAS system options in the SAS command. Precede each option with a hyphen:

```
sas -option1 -option2...
```

  For ON/OFF options, list the keyword corresponding to the appropriate setting. For options that accept values, list the keyword that identifies the option, followed by the option value. For example,

```
sas -nodate -work mywork
```

  Settings that you specify in the SAS command last for the duration of the SAS session; or, for those options that can be changed within the session, until you change them. All options can be specified in the SAS command.

OPTIONS statement within a SAS session
: Specify SAS system options in an OPTIONS statement at any point within a SAS session. The options are set for the duration of the SAS session or until you change them. When you specify an option in the OPTIONS statement, do not precede its name with a hyphen (-). If the option has an argument, use = after the option name.

  For example,

```
options nodate linesize=72;
options editcmd='/usr/bin/xterm -e vi';
```

Refer to *SAS Language Reference: Dictionary* for more information on the OPTIONS statement. Not all options can be specified in the OPTIONS statement. To find out about a specific option, look up its name in Table 17.3 on page 385.

OPTIONS statement in an autoexec file
Specify SAS system options in an OPTIONS statement in a autoexec file. For example, your autoexec file could contain the following statements:

```
options nodate pagesize=80;
filename rpt '/users/myid/data/report';
```

System Options window
Change the SAS system options from within the System Options window.

In general, use quotation marks to enclose filenames and pathnames specified in the OPTIONS statement or the System Options window. Do not use quotation marks otherwise. Any exceptions are discussed under the individual option. You can use the abbreviations listed in Table 4.6 on page 115 to shorten the filenames and pathnames you specify.

How SAS Processes System Options Set in One Place

If the same option is set more than once within the SAS command, a configuration file, or the SASV9_OPTIONS environment variable, only the last setting is used; the others are ignored. For example, the DMS option is ignored in the following SAS command:

```
sas -dms -nodms
```

The DMS option is also ignored in the following configuration file:

```
-dms
-linesize 80
-nodms
```

By default, if you specify the HELPLOC, MAPS, MSG, SAMPLOC, SASAUTOS, or SASHELP system options more than one time, the last value that is specified is the value that SAS uses. If you want to add additional pathnames to the pathnames already specified by one of these options, you must use the APPEND or INSERT system options. See “APPEND System Option” on page 315 and “INSERT System Option” on page 337 for more information.

How SAS Processes System Options Set in Multiple Places

When the same option is set in more than one place, the most recent specification is used. The following places are listed in order of precedence. For example, a setting made in the System Options window or OPTIONS statement will override any other setting, but if you set a system option using the SASV9_OPTIONS environment variable, then this will override only the setting for the same system option in your configuration file.

Precedence for Processing System Options

The precedence for processing system options is as follows:

1 System Options window or OPTIONS statement (from a SAS session or job).

2 autoexec file that contains an OPTIONS statement (after SAS initializes)

3 SAS command
4 SASV9_OPTIONS environment variable
5 configuration files (before SAS initializes). For more information, see "Order of Precedence for SAS Configuration Files" on page 17.

For example, if a configuration file specifies NOSTIMER, you can override the setting in the SAS command.

By default, if you specify the HELPLOC, MAPS, MSG, SAMPLOC, SASAUTOS, or SASHELP system option more than one time, the last value that is specified is the value that SAS uses. If you want to add additional pathnames to the pathnames already specified by one of these options, you must use the APPEND or INSERT system options to add the new pathname. See "APPEND System Option" on page 315 and "INSERT System Option" on page 337 for more information.

Defining Environment Variables in UNIX Environments

What Is an Environment Variable?

Environment variables are variables that apply to both the current shell and to any subshells it creates (for example, when you send a job to the background or execute a script). If you change the value of an environment variable, the change is passed forward to subsequent shells but not backward to the parent shell.

In a SAS session, you can use the SASV9_OPTIONS environment variable to specify system options and the SASV9_CONFIG environment variable to specify a configuration file. You can also use environment variables as filerefs and librefs in various statements and commands.

Note: A SAS/ACCESS product initializes the environment variables it needs when loading. Any changes that you make to an environment variable after initialization will not be recognized. For more information, see the documentation for your SAS/ACCESS product. △

How to Define an Environment Variable for Your Shell

The way in which you define an environment variable depends on the shell that you are running. (To determine which shell you are running, type **ps** at the command prompt or **echo $shell** to see the current value of the SHELL environment variable.)

Bourne and Korn Shells

In the Bourne shell and in the Korn shell, use the **export** command to export one or more variables to the environment. For example, these commands make the value of the variable **scname** available to all subsequent shell scripts:

```
$ scname=phonelist
$ export scname
```

In the Korn shell, you can combine these into one command:

```
$ export scname=phonelist
```

If you change the value of **scname**, the new value affects both the shell variable and the environment variable. If you do not export a variable, only the shell script in which you define has access to its value.

C Shell

In the C shell, you set (define and export) environment variables with the **setenv** (set environment) command. For example, this command is equivalent to the commands shown previously:

```
% setenv scname phonelist
```

Displaying the Value of an Environment Variable

To display the values of individual environment variables, use the **echo** command and parameter substitution. An example is: **echo $SHELL** which returns the current value of the SHELL environment variable. Use the **env** (or **printenv**) command to display all environment variables and their current values.

Determining the Completion Status of a SAS Job in UNIX Environments

The exit status for the completion of a SAS job is returned in **$status** for the C shell, and in **$?** for the Bourne and Korn shells. A value of 0 indicates normal termination. You can affect the exit status code by using the ABORT statement. The ABORT statement takes an optional integer argument, *n*, which can range from 0 to 255.

The following table summarizes the values of the exit status code.

Table 1.1 Exit Status Code Values

Condition	Exit Status Code
All steps terminated normally	0
SAS System issued warning(s)	1
SAS System issued error(s)	2
User issued ABORT statement	3
User issued ABORT RETURN statement	4
User issued ABORT ABEND statement	5
User issued ABORT RETURN *n* statement	*n*
User issued ABORT ABEND *n* statement	*n*

If you specify the ERRORABEND SAS system option on the command line, and the job has errors, the exit status code is set to 5.

UNIX exit status codes are in the range 0-255. Numbers greater than 255 may not print what the user expects because the code is interpreted as a signed byte.

Interrupting or Terminating Your SAS Session in UNIX Environments

Preferred Methods of Exiting SAS

The preferred methods of exiting a SAS session are the following:

- select

  File ► Exit

  if you are using SAS in the windowing environment
- use **endsas;**
- enter **BYE** in the command line
- use CTRL+D if you are using SAS in interactive line mode.

Additional Methods for Interrupting or Terminating SAS

In addition to the preferred methods, you can terminate SAS in the following ways:

- Press the interrupt or quit control key
- Use the session manager
- Enter the UNIX **kill** command.

Although you can terminate SAS using these techniques, you should try one of the four preferred techniques listed first.

Using Control Keys

Control keys enable you to interrupt or terminate your session by simply pressing the interrupt or quit key sequence. However, control keys can be used only when your SAS program is running in interactive line mode or in batch mode in the foreground. You cannot use control keys to stop a background job.

Note: You cannot use control keys to stop a batch job that has been submitted with the **batch**, **at**, **nohup**, or **cron** command. △

Because control keys vary from system to system, issue the UNIX **stty** command to determine which key sends which signal. The **stty** command varies considerably among UNIX operating environments, so check the **stty** UNIX man page before using it. Usually, one of these forms of the command will print all of the current terminal settings:

```
stty
stty -a
stty everything
```

The output should contain lines similar to these:

```
intr = ^C; quit = ^\; erase = ^H;
kill = ^U; eof = ^D; eol = ^@
```

The caret (^) stands for the CTRL key. In this example, CTRL+C is the interrupt key and CTRL+\ is the quit key.

Using the SAS Session Manager

If you invoke SAS in the windowing environment, you can use the session manager to interrupt or terminate your SAS session. The session manager is automatically iconified when you start SAS. To interrupt or terminate your SAS session, open the SAS Session Manager window and click Interrupt or Terminate.

Note: Clicking Interrupt is equivalent to specifying the **-SIGINT** option on the **kill** command. Clicking Terminate is equivalent to specifying the **-SIGTERM** option on the **kill** command. △

Using the UNIX kill Command

Note: Only use the **kill** command after you have tried all other methods to exit your SAS session. △

The **kill** command sends an interrupt or quit signal to SAS, depending on which signal you specify. You can use the **kill** command to interrupt or terminate a SAS session running in any mode. The **kill** command cannot be issued from within a SAS session. You must issue it from another terminal or from another window (if your terminal permits it).

The format of the **kill** command is

kill <-*signal-name*> *pid*

To send the interrupt signal, specify **-SIGINT**. To send the terminate signal, specify **-SIGTERM**. Use the **ps** command to determine the process identification number (*pid*) of the SAS session that you want to interrupt or terminate.

For example, suppose you want to stop a SAS job running in the background. First, issue the **ps** command to determine the PID of the SAS job.

```
> ps
    PID TTY      TIME COMMAND
   2103 ttyu0    0:00 motifxsa
   2111 ttyu0    0:01 sas
   2116 ttyu0    0:00 ps
   3856 ttyu2    0:03 ksh
```

Four PIDs appear, but only one is for a SAS program. (**motifxsa** is the SAS session manager. See "The SAS Session Manager (motifxsassm) in UNIX" on page 33 for more information.) Therefore, to send the interrupt signal to that SAS program, you would issue this command:

```
kill -SIGINT 2111
```

SAS replies with a prompt:

```
Press Y to cancel submitted statements,
N to continue.
```

For more information, refer to the UNIX man pages for the **ps** and **kill** commands.

Messages in the SAS Console Log

If SAS encounters an error or warning condition when the SAS log is not available, then any messages that SAS issues are written to the SAS console log. Normally, the SAS log is unavailable only early in SAS initialization and late in SAS termination.

Under UNIX, the STDOUT fileref specifies the location of the console log.

Ending a Process That Is Running as a SAS Server

If you need to end a process running as a SAS server, use one of the following methods:

- If you are using the SAS Metadata Server, use the SAS Management Console to end a process.
- If you are using another SAS server, use the UNIX scripts that shipped with the servers to stop the process. You can also use these scripts to start (or restart) a

server, as well as determine if the server is already running. For more information about these scripts, contact your site administrator.

Note: If the server does not respond to the UNIX script, then you can use the **kill** command to end the server process. For more information, see "Using the UNIX kill Command" on page 24. △

Ending a SAS Process on a Relational Database

How to Interrupt a SAS Process

CAUTION:

When you interrupt a SAS process, you might terminate the current query. If you are using the current query to create a new data set, then the data set is still created even if the query is terminated. If you are using the current query to overwrite a data set, the data set is not overwritten if the query is terminated. In either case, however, you do not receive a warning that the query did not complete. △

The method that you use to interrupt a SAS process depends on how you invoke SAS.

- If you are running SAS in interactive line mode or in batch mode using a foreground process, then you can use either of the following methods to interrupt SAS:
  - Press the control key sequence that is set to interrupt in the shell that invoked SAS. In most cases, this control key sequence is CTRL+C. See the man page for the stty command to determine the appropriate key sequence for your environment.

    Note: This control key sequence will not interrupt a SAS process if you issue it in a SAS window, such as the Program Editor or SAS log. △
  - Use the **-SIGINT** option in the **kill** command. For more information, see "Using the UNIX kill Command" on page 24
- If you are running the SAS windowing environment in the foreground, then click Interrupt in the SAS Session Manager.
- If you are running an interactive SAS process in the background, then you must click Interrupt in the SAS Session Manager. You cannot use a control key sequence to interrupt the SAS process.

The interrupt signal is sent to the DATA step or procedure that is currently executing for interpretation. The interrupt signal is interpreted differently by DATA steps and procedures. The actions that you can take appear in the interrupt menu. Since the options in the interrupt menu are dependent on what is currently executing, you might see a different interrupt menu for the following:

- each SAS procedure. The options available for PROC SORT differ from those available for PROC SQL.
- a DATA step. The options available when SAS is processing a DATA step are different from when SAS is processing a procedure.
- each SAS application. For example, SAS webAF has a different interrupt menu than the one for PROC SQL.

Note: Depending on the relational database, the interrupt signal might be handled differently. If the DATA step or procedure does not respond to the interrupt in a given amount of time, then SAS does not terminate the current DATA step or procedure. △

Example: Interrupt Menu for PROC SQL

The following is an example of the interrupt menu that you might see if you issue an interrupt signal while SAS is processing a PROC SQL statement:

```
Select:
      1. Cancel Submitted Statements
      2. Halt Datastep/Proc: SQL
      C. Cancel the dialog
      T. Terminate the SAS System
```

The following table explains each of these options:

Table 1.2 Description of Interrupt Menu Options for PROC SQL

Option	Description	What This Option Does
1	Cancel Submitted Statements	Selecting this option will end the current DATA step or procedure and the underlying DBMS process. In interactive mode, you will return to the command prompt.
2	Halt Datastep/Proc: SQL	If you select this option and SAS is currently executing an SQL procedure, then the following menu appears: **Press:** **C to continue** **Q to cancel the current query** **S to cancel the submitted statements** **X to exit SQL procedure** **?** □ If you select **C**, then the menu will disappear, but since the current query ended when you interrupted the SAS process, SAS will not return to the current query. Instead, SAS will begin processing the next line of code. □ If you select **Q**, then SAS cancels the current query even if it is on a relational database. SAS continues processing the next statement. □ If you select **S**, then all of the PROC SQL statements that you submitted are cancelled. □ If you select **X**, SAS exits the current SQL procedure and starts processing the next statement in the submit block.
C	Cancel the dialog	Selecting this option returns you to normal processing; however, the current query might have been interrupted. If you are running a long query and the control is on the DBMS server, then selecting **C** will end the current query. If you are running a short query and SAS has the control, then selecting **C** will cause the interrupt menu to disappear and the current query will continue. To determine whether or not the query was interrupted while reading or writing out the DBMS data, use PROC PRINT to view the partially created DBMS table or SAS data set.
T	Terminate the SAS System	Selecting this option ends your SAS session as well as the current query.

How to Terminate a SAS Process

The method that you use to terminate a SAS process depends on how you invoke SAS.

- If you are running in interactive line mode or in batch mode using a foreground process, use the **-SIGTERM** option in the **kill** command. For more information, see "Using the UNIX kill Command" on page 24:
- If you are running the SAS windowing environment in the foreground, then click Terminate in the SAS Session Manager.
- If you are running an interactive SAS process in the background, then you must click Terminate in the SAS Session Manager. You cannot use a control key sequence to terminate the SAS process.

If you click Terminate in the SAS Session Manager, then a dialog box appears confirming that you want to end the session. If you click OK, then both the SAS session and the current query are terminated. If you click Cancel, then you are returned to the SAS session.

What Happens When You Interrupt a SAS Process and the Underlying DBMS Process

CAUTION:

Interrupting a SAS process and the underlying DBMS process might kill all jobs that are running on your DBMS. Interrupting your SAS and DBMS processes should be an exception. Extensive care should be taken when you construct your queries. △

Note: In this section, SAS process refers to a series of events. It is not the process on the operating system. When you interrupt or terminate a SAS process, the process on the operating system might still be running. △

When you interrupt or terminate a query on a server, the following processes stop:

- processing of current extractions. For example, suppose you forgot to include a WHERE clause in your SQL query and are now extracting 1 billion rows into SAS. Issuing an interrupt stops the SAS process and the extract step in the DBMS.
- processing of queries that are in progress on the server. For example, you have a very complex extract query that runs for a long time before producing a result. Issuing an interrupt stops the SAS and DBMS processes. As a result, the complex query running on your DBMS server is interrupted and terminated.
- update, delete, and insert processing. For example, you are updating, deleting, or inserting many rows in your DBMS. An interrupt stops the SAS and DBMS processes.

CHAPTER 2

Working in the SAS Windowing Environment

Definition of the SAS Windowing Environment

The SAS windowing environment refers to the windows that open when you invoke SAS. These windows include: the Program Editor, Log, Output, Explorer, and Results. These windows appear when you start SAS from your workstation or through an emulator. For more information about these windows, see SAS Help and Documentation.

The SAS windowing environment supports the use of X-based graphical user interfaces (GUIs). In UNIX environments, SAS provides an X Window System interface that is based on the Motif style.

Many features of the SAS windowing environment are controlled by X resources. For example, colors, window sizes, the appearance of the Toolbox, and key definitions are all controlled through X resources. Chapter 3, "Customizing the SAS Windowing Environment," on page 53 provides general information about resources, such as how to specify resources, and describes all of the resources that you can use to customize the interface.

Description of SAS in the X Environment

Definition of X Window System

The X Window System is a networked windowing system. If several machines are on a network, you can run an X *server* that in turn serves X applications (as *clients*) to all the other machines in the network.

X Window Managers

In UNIX environments, SAS features an X Window System interface that is based on Motif. This interface uses the window manager on your system to manage the windows on your display. Any window manager that is compliant with the *Inter-Client Communication Conventions Manual* (ICCCM) can be used with the Motif interface to SAS. Vendors provide at least one window manager with the X Window System environment. A common window manager is the Common Desktop Environment (CDE). If you are using another window manager, such as Gnome, you should also read the documentation that is supplied by the vendor for that window manager.

All window managers perform the same basic functions, but they differ in their style and in their advanced functions. The appearance and function of the interface to SAS depends to some extent on your X window manager. Most window managers provide some kind of frame around a window. The window manager also governs the placement, sizing, stacking, and appearance of windows, as well as their interaction with the keyboard. The basics of interacting with SAS are the same for all window managers: opening pull-down and pop-up menus, moving windows, responding to dialog boxes, dragging text, and so on.

SAS Window Session ID

When you run SAS on an X workstation, SAS shares the display with other X applications, including other SAS sessions. To enable you to distinguish between different applications and SAS sessions, SAS generates a SAS window session ID for each session by appending a number to the application name, which by default is **`SAS`**. This session ID appears in the window title bar for each SAS window and in the window icon title. The SAS sessions are assigned sequentially. Your first SAS session is not assigned a number, so the session ID is **`SAS`**; your second SAS session is assigned the session ID **`SAS2`**, and so on. Although the default application name is **`SAS`**, you can use the **`-name`** X option to change the instance name. The instance name can be up to six characters long.

Workspace and Gravity in a SAS Session

When you use SAS on an X workstation, the display may be shared by many concurrent applications. When SAS windows from several different sessions and windows from other applications appear on the display, the display can become cluttered. To help alleviate this problem, the windows for a SAS session first appear within an *application workspace* (AWS). The AWS defines a rectangular region that represents a virtual display in which SAS windows are initially created. SAS attempts to position the AWS in relation to the upper-left corner of your display. In other words, the workspace gravitates toward a certain direction (*session gravity*) on the display.

Some window manager configurations might override the placement that SAS has chosen for a window.

If you issue windowing commands or execute SAS procedures that create new SAS windows, the same rules of initial position and size apply to these windows: they are initially placed in the SAS AWS. You can use the WSAVE command to save the current window positions (or geometry). See “Customizing Session Workspace, Session Gravity, and Window Sizes in UNIX Environments” on page 93 for details.

Window Types

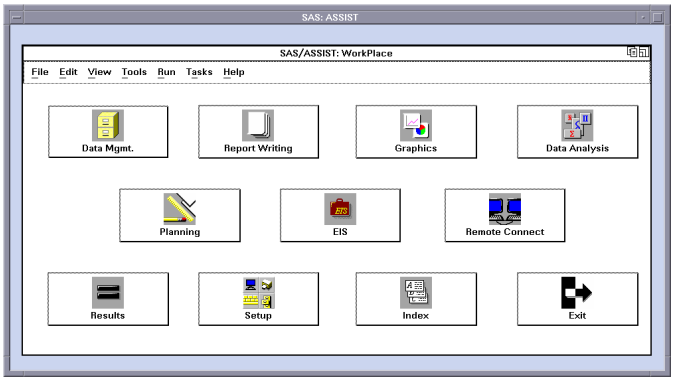
Top-Level Windows

SAS uses primary and interior windows. Some SAS applications consist of one or more primary windows controlled by the X window manager in addition to the interior windows controlled by SAS. The SAS windowing environment primary windows, as well as most SAS application windows, initially appear as *top-level windows*. Top-level windows interact directly with the X window manager. They have a full title bar along with other window manager decorations. You can manipulate them individually once they appear on the display.

Interior Windows

Interior windows behave differently than primary windows. SAS/ASSIST software is an application with interior windows. Interior windows are contained within *container windows*, which may or may not be primary windows. The following display shows an interior window in SAS/ASSIST software.

Display 2.1 Sample Interior Window



SAS provides some degree of window management for interior windows. Specifically, interior windows have the following sizing and movement capabilities:

- You can move interior windows by clicking the interior window title bar and dragging the window to the desired location. If the destination of the interior window is outside the bounds of the container window, the container window changes according to the value of the **`SAS.awsResizePolicy`** resource. (The space within the container window is the application workspace, which is described in “Workspace and Gravity in a SAS Session” on page 31.) See “Overview of X Resources” on page 55 for more information.

- Interior windows cannot be iconified individually. Clicking on the container window icon button iconifies the container window and its interior windows.
- A *push-to-back button* (the small overlapping squares in the upper right corner) is also available with interior windows. However, you cannot push an active window behind an inactive window.

The SAS Session Manager (motifxsassm) in UNIX

What Is the SAS Session Manager?

The SAS Session Manager for X (**motifxsassm**) is an X client that is run by SAS when you use the SAS windowing environment. The session manager appears as shown in the following display.

Display 2.2 SAS Session Manager Dialog Box

The session manager window describes the following:

- which SAS session it controls
- the host machine from which the SAS session was invoked
- the UNIX process identifier of the SAS session.

When SAS opens, the session manager window is automatically minimized (iconified).

Features of the SAS Session Manager

The session manager enables you to

- map and iconize all windows of the SAS session. The Restore and Minimize buttons restore and minimize all of the windows that are open in the SAS session that is controlled by that session manager. These functions are performed with standard X library calls and will work with most X window managers.
- interrupt the SAS session. The Interrupt button sends a UNIX signal to SAS. When SAS receives the signal, it displays a dialog box that asks for confirmation before it cancels the submitted statements.
- terminate the SAS session. Terminate displays a dialog box that asks you to confirm that you want to terminate the SAS session. If you select OK, the session manager sends a UNIX signal to the SAS session that forces the session to terminate.

CAUTION:
Terminating your SAS session might result in data loss or data corruption. Before terminating your session, you should attempt to end SAS using one of the methods described in "Preferred Methods of Exiting SAS" on page 22. △

- use your host editor from within your SAS session. When you issue the HOSTEDIT command, SAS passes the request to the session manager, which then invokes your host editor, so the session manager must be running in order for the HOSTEDIT command to take effect. When you issue the HOSTEDIT command, SAS creates a temporary file that contains the data from the active SAS window and passes this file to your host editor. (These temporary files are stored in the directory specified by the SASWORK option.) When you save your file in the host editor, the file is copied back into the SAS window if the window is writable, and the temporary files are deleted when the SAS session ends. See "Configuring SAS for Host Editor Support in UNIX Environments" on page 49 for more information.

Disabling the SAS Session Manager

You can disable the SAS Session Manager by performing one of the following steps:

- Select

  Tools ▶ Options ▶ Preferences

  On the **General** tab, deselect the **Start Session manager** check box.
- Specify the following X resource on the SAS command line at invocation:

```
sas -xrm 'SAS.startSessionManager: False'
```

  Specifying the **SAS.startSessionManager** X resource will deselect the **Start Session manager** check box in the Preferences dialog box.

Note: SAS saves the settings in the Preferences dialog box when it exits. If you have disabled the SAS Session Manager during your session, then the next time you invoke SAS, the SAS Session Manager will not run. To start the SAS Session Manager, select the **Start Session manager** check box in the Preferences dialog box or specify the following on the SAS command line at invocation:

```
sas -xrm 'SAS.startSessionManager: True'
```

△

Displaying Function Key Definitions in UNIX Environments

Benefits of Assigning Function Key Definitions

Function keys provide quick access to commands. They enable you to issue commands, insert text strings, and insert commands in programs. Function key definitions can be different on different terminals. These definitions are fully customizable.

How to Display Function Key Definitions

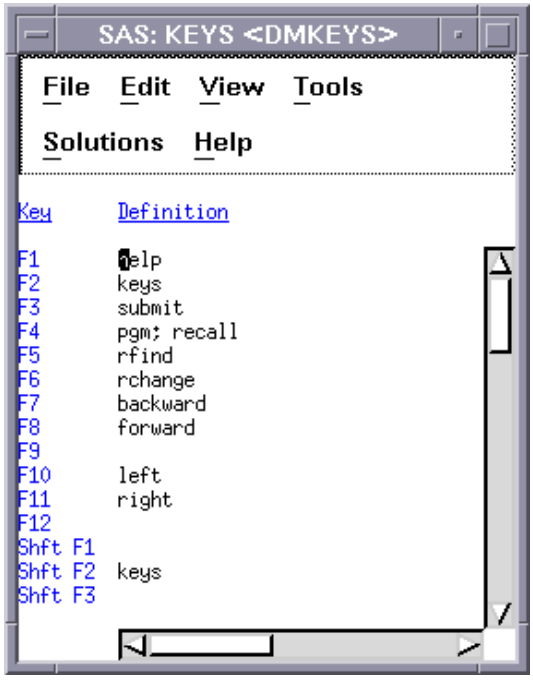
You can open the KEYS (DMKEYS) window to display all of the your function key definitions in one of the following ways:

- Press F2.
- Issue the KEYS command.

- Select

  Tools ▶ Options ▶ Keys

Display 2.3 KEYS (DMKEYS) Window



To view a single key definition without bringing up the KEYS window, use the KEYDEF command and specify the key definition that you want to view. For example, the following command displays the definition for key F4:

```
keydef f4
```

For information on customizing key definitions, see "Customizing Key Definitions in UNIX Environments" on page 73. Refer to *SAS Language Reference: Dictionary* for more information on the KEYS window and the KEYDEF command.

The SAS ToolBox in UNIX Environments

Introduction to the SAS ToolBox

The SAS ToolBox has two parts as illustrated in the following display:

- A command window that enables you to quickly enter any command in the active SAS window. For information about commands that are available under UNIX, see Chapter 9, "Commands under UNIX," on page 201 and the SAS commands section in the Base SAS Software section in SAS Help and Documentation.
- A toolbar that contains several tool icons. When you select a tool icon, SAS immediately executes the command that is associated with that icon. The toolbar and the tool icons are completely customizable. For more information, see "Using the Tool Editor" on page 68.

Display 2.4 SAS ToolBox

The name of the active window is displayed in the title bar of the SAS ToolBox. For example, if the Log window were active, the title bar would say SAS ToolBox: Log instead of SAS ToolBox: Program Editor.

Under UNIX, the default SAS ToolBox automatically appears at the bottom of the SAS windows stack by default. To control its configuration, you use the Preferences dialog box. (See "Modifying the ToolBox Settings" on page 60.)

Customizing the Default Toolbox

The default toolbox is automatically copied to your SASUSER.PROFILE.DMS.TOOLBOX regardless of whether you customize the toolbox. If you invoke an application that does not have an associated PMENU entry, the default toolbox is displayed for that application. If you then customize the toolbox for that application, the customized toolbox is stored in SASUSER.PROFILE.DEFAULT.TOOLBOX, where DEFAULT is the same entry name as the PMENU entry for the window or application.

You can customize the default toolbox, create multiple toolboxes and switch between them, and create application-specific toolboxes (such as with SAS/AF applications) that are automatically loaded when the application is loaded. Only one toolbox is displayed at a time, and the tools in the toolbox change as you move between applications. For more information about customizing your toolboxes, see "Customizing Toolboxes and Toolsets in UNIX Environments" on page 67.

Default Configuration for the Command Window and the Toolbar

By default, the toolbar and the command window are joined and are automatically displayed when SAS initializes unless

- you executed your SAS job in a nonwindowing environment mode.
- the **`SAS.defaultToolBox`** or **`SAS.defaultCommandWindow`** resource is set to **`False`**. The default value is **`True`**. For more information about the resources that control the toolbox, see "X Resources That Control Toolbox Behavior" on page 67.
- you deselect **`Display tools window`**, **`Display command window`**, or **`Combine windows`** from the Toolbox tab in the Preferences dialog box.

The following display shows the command window and the toolbar in their default configuration.

Display 2.5 Default Configuration for Command Window and Toolbar

Opening and Closing the Command Window and the Toolbar

The following table lists the steps that you can use to open and close the command window and toolbar.

Table 2.1 Steps for Opening and Closing the Command Window and the Toolbar

Window	How to Open	How to Close
Command Window and Toolbar	To open both windows, complete any of the following steps: □ Issue the COMMAND WINDOW command. □ Issue the TOOLLOAD command. See "TOOLLOAD Command" on page 217 for more information. □ Select Tools ▶ Options ▶ Toolbox	To close these windows, complete any of the following steps: □ Select **Close** from the ToolBox window menu. □ Enter the TOOLCLOSE command as described in "TOOLCLOSE Command" on page 216. □ Select Tools ▶ Options ▶ Toolbox so that ToolBox is deselected.

Window	How to Open	How to Close
Command Window	To open only the command window: **1** Deselect **Combine windows** in the ToolBox tab of the Preferences dialog box. **2** Complete any of the following steps: □ Select **Display command window** in the ToolBox tab of the Preferences dialog box. □ Issue the COMMAND WINDOW command.	To close only the command window: **1** Deselect **Display command window** in the ToolBox tab of the Preferences dialog box. **2** Select **Close** from the window menu.
Toolbar	To open only the toolbar: **1** Deselect **Combine windows** in the ToolBox tab of the Preferences dialog box. **2** Complete any of the following steps: □ Select **Display tools window** in the ToolBox tab of the Preferences dialog box. □ Issue the TOOLLOAD command. See "TOOLLOAD Command" on page 217 for more information. □ Select Tools ▶ Options ▶ Toolbox	To close only the toolbar: **1** Deselect **Combine windows** in the ToolBox tab of the Preferences dialog box. **2** Complete any of the following steps: □ Deselect **Display tools window** in the ToolBox tab of the Preferences dialog box. □ Issue the TOOLCLOSE command as described in "TOOLCLOSE Command" on page 216. □ Select Tools ▶ Options ▶ Toolbox so that ToolBox is deselected.

Executing Commands

You can execute commands from either the command window or the toolbar. The following table gives more details on how to execute commands.

Table 2.2 Executing Commands in the Command Window and the Toolbar

Window	Executing a Command
Command Window	To execute a command, complete the following steps: **1** Click in the command window. **2** Type in the command. **3** Press ENTER or click the check mark. The command is executed in the active SAS window. You can use the up and down arrow keys to scroll through previously entered commands, or you can select a previous command from the drop-down list. Use the left mouse button to select a command from the drop-down list. Use MB2 to select and execute a command from the list.
Toolbar	Clicking a tool icon in the toolbar executes the command or commands associated with that icon. If you place the cursor over an icon for the amount of time specified by the **`SAS.toolBoxTipDelay`** resource, a pop-up window displays text that describes the command for that icon.

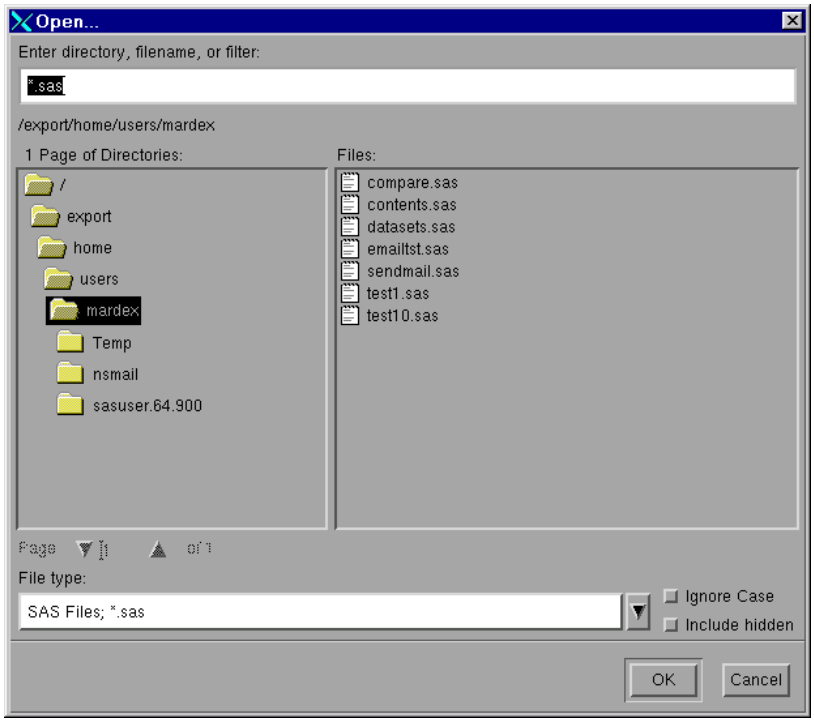
Opening Files in UNIX Environments

Opening the Open Dialog Box

The Open dialog box enables you to select files from the host file system. To open this dialog box, select

File ▶ Open

Display 2.6 Open Dialog Box



Description of the Open Dialog Box Options

The following table describes the options found on the Open Dialog Box.

Table 2.3 Options in the Open Dialog Box

Option	Description
Enter directory, filename, or filter	is where you can type in the name of the directory, file, or file filter (file type) that you want to open. The directory shown under the **Filter** field is the currently selected directory. You can change this directory either by selecting a name from the **Page of Directories** list or by typing the new name directly into the field. The dialog box displays nonreadable directories with a different icon. To display a list of all the files in a directory, enter the asterisk (*) wild card in the **Filter** field or select **All Files; *** as the file type.
Page of Directories	contains the names of the directories specified in the **Filter** and **Page** fields.
Files	contains the files in the selected directory that match the filter specified.
Page	enables you to change the directories that are listed in the **Page of Directories** list box. A new page is defined when the number of entries in the **Page of Directories** list exceeds twice the screen height. To change pages, use the right or left arrows next to the **Page** field.
File type	enables you to select the type or types of files to be shown in the **Files** list box. You can display a list of possible file filters by selecting the down arrow next to the field. Click on a file filter to select it.
Ignore Case	specifies that both uppercase and lowercase names be included in the display. (If you select **All Files; *** as the filter, both uppercase and lowercase names are displayed whether or not you select **Ignore Case**.)
Include hidden	includes or excludes hidden files and directories from the graphical display.

Specifying the Initial Filter and Directory Using SAS Resources

You can specify the initial filter in the **File type** field by assigning a value to the **SAS.pattern** resource. However, the Open dialog box retains its filter between invocations, so the **SAS.pattern** resource applies only to the first invocation of the Open dialog box. You can also use the **SAS.directory** resource to specify the directory that you want when you first invoke the Open dialog box.

For more information about specifying SAS resources, see “Overview of X Resources” on page 55.

Using Regular Expressions in Filenames

Everything that you enter into the Open dialog box is treated as a regular expression. When you are opening or saving a file and you want to use a regular

expression special character as part of the filename, precede the character with a backslash (\). For example, to write to a file named $Jan, enter \$Jan as the filename.

For more information on regular expressions, refer to man page 5 for regexp:

```
man 5 regexp
```

Changing Your Working Directory in UNIX Environments

What Is Your Working Directory?

The working directory is the operating system directory to which many SAS commands and actions apply. By default, SAS uses the current directory as the working directory when you begin your SAS session.

Changing Your Working Directory

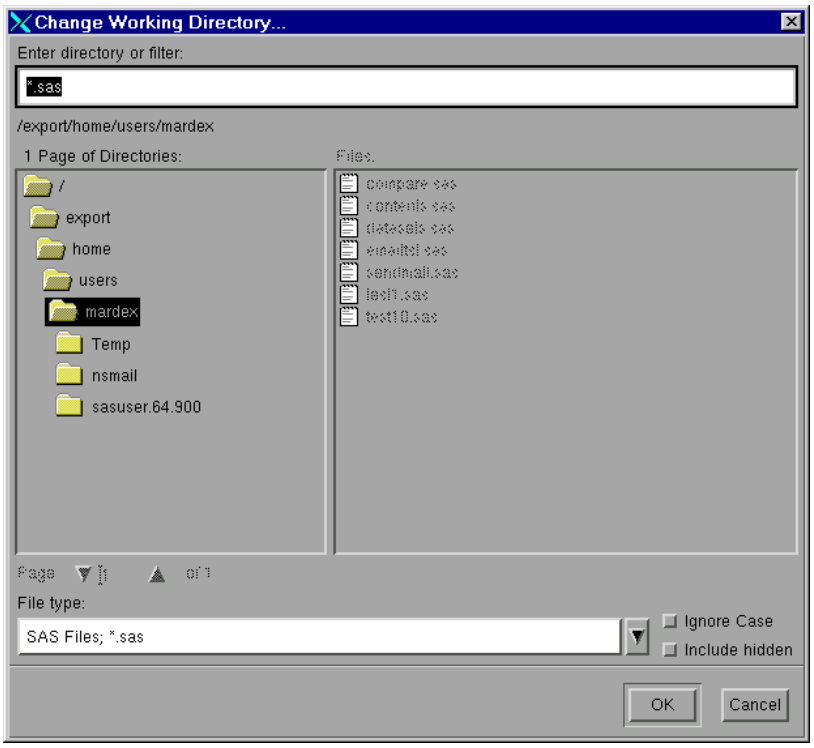
You can change the working directory during your SAS session. You can use the Change Working Directory dialog box to select a new directory, or you can use the X command, the X statement, the CALL SYSTEM routine, or the %SYSEXEC macro statement to issue the change directory (**cd**) command. For information on the X command and statement, the CALL SYSTEM routine, and the %SYSEXEC macro statement, see "Executing Operating System Commands from Your SAS Session" on page 13.

Change Working Directory Dialog Box

To open the Change Working Directory dialog box, issue the DLGCDIR command or select

Tools ► Options ► Change Directory

Display 2.7 Change Working Directory Dialog Box



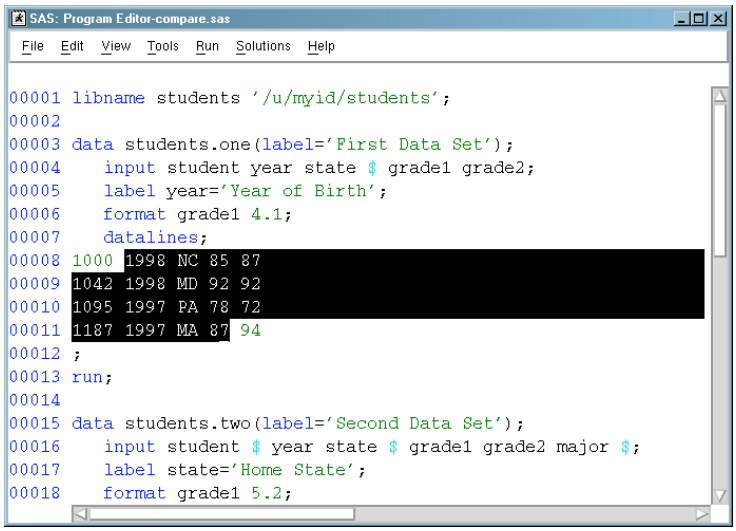
The Change Working Directory dialog box works exactly the same as the Open dialog box, except that you cannot select a file from the list. For an explanation of the options on the Change Working Directory dialog box, see “Description of the Open Dialog Box Options” on page 40.

Selecting (Marking) Text in UNIX Environments

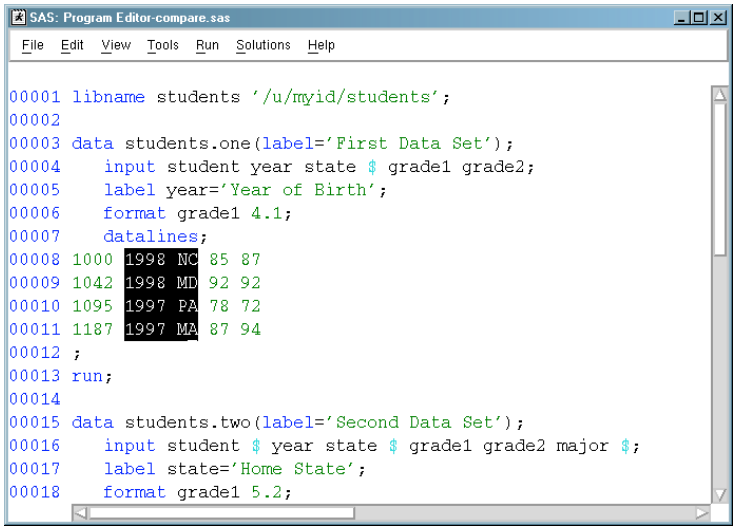
Difference between Marking Character Strings and Blocks

When you select text in a SAS window, you can select character strings or blocks. Character strings include the text in successive columns of one or more rows, as shown in the following display. Blocks are rectangular blocks that include the same columns from successive rows, as shown in Display 2.9 on page 43.

Display 2.8 Strings That Are Marked



Display 2.9 Blocks That Are Marked



Techniques for Selecting Text

When using one of the following techniques, the text that you select might not remain highlighted although it has been copied into the paste buffer.

Selecting Text with the Mouse

To select your text, complete the following steps:

1. Position the cursor at the beginning of the text that you want to mark.
2. Press and hold the left mouse button. If you want to select a block instead of a string, press and hold the CTRL key before you press the left mouse button.
3. Drag the mouse pointer over the text that you want to mark.

4 Press and hold down the Alt key (or Extend char key or Meta key, depending on your keyboard) while you release the mouse button. The marks that are generated by the mouse are called *drag marks*.

To extend an area of marked text, press and hold the Shift key, and use the left mouse button and the Alt key (and the CTRL key, if you are marking a block) to mark the new ending position. To unmark the selected text, press the mouse button anywhere in the window.

Selecting Text with the MARK Command

You can issue the MARK command from the command line, or you can assign it to a function key. With the MARK command, you can select more than one area of text in the same window at the same time. For more information about the MARK command, see SAS Help and Documentation.

To select your text, complete the following steps:

1 Position the cursor at the beginning of the text that you want to mark.
2 Issue the MARK command. If you want to select a block instead of a string, add the BLOCK argument to the MARK command.
3 Move the cursor to the end of the text that you want to mark.
4 Issue the MARK command a second time.

To unmark the selected text, issue the UNMARK command.

Selecting Text Using the Edit Menu

To select your text using the Edit menu, complete the following steps:

1 Position the cursor at the beginning of the text that you want to mark.
2 Select

Edit ▶ Select

3 Position the cursor to the end of the text that you want to mark.
4 Press the left mouse button.

To unmark the selected text, select

Edit ▶ Deselect

Copying or Cutting and Pasting Selected Text in UNIX Environments

Techniques for Copying or Cutting and Pasting Selected Text

After you have marked text, you can copy or cut the text and paste it in another location.

- To copy text, select the Copy icon from the toolbox, issue the STORE or WCOPY command, or select

  Edit ▶ Copy

- To cut text, select the Cut icon from the toolbox, issue the CUT or WCUT command, or select

  Edit ▶ Cut

- To paste the cut or copied text, select the Paste icon from the toolbox, issue the PASTE or WPASTE command, or select

  Edit ▶ Paste

For more information on the CUT, PASTE, and STORE commands, refer to *SAS Language Reference: Dictionary*.

How SAS Uses the Automatic Paste Buffer

When you end a drag mark by releasing the mouse button without holding down the Alt key, SAS performs an end-of-mark action that might automatically generate a STORE command to save the contents of the mark into a SAS paste buffer. If the STORE command is generated automatically, you do not have to explicitly copy the text before you paste it.

Disabling the Automatic Paste Buffer

You can disable the automatic paste buffer in the following ways:

- Set the **SAS.markPasteBuffer** resource.
- Deselect **Automatically store selection** on the Editing tab in the Preferences dialog box:

  Tools ▶ Options ▶ Preferences

For more information, see "Customizing Cut-and-Paste in UNIX Environments" on page 91.

Copying and Pasting Text between SAS and Other X Clients

You can cut or copy and paste text between X clients if you associate the default SAS paste buffer with a paste buffer specific to X. For example, if you associate the default SAS paste buffer with the XTERM paste buffer, you can copy and paste text between xterm windows and SAS windows. To associate the SAS buffer with an X buffer, specify the **SAS.defaultPasteBuffer** resource. For example:

```
SAS.defaultPasteBuffer:  XTERM
```

For more information about using paste buffers, see "Customizing Cut-and-Paste in UNIX Environments" on page 91.

Using Drag and Drop in UNIX Environments

Difference between Default and Nondefault Drag and Drop

The SAS windowing environment on UNIX offers two types of drag and drop: default and nondefault. Default drag and drop enables you to move text from one place to another. Nondefault drag and drop enables you to choose whether to move or copy the text, submit the text if you are dragging SAS code, or cancel the drag and drop operation. With default drag and drop, you can drag text between SAS windows in different SAS sessions and between SAS windows and other Motif applications, such as

Netscape, that support drag and drop. Nondefault drag and drop is available only between windows in the same SAS session.

Limitations of Drag and Drop in UNIX

Under UNIX, you cannot drag and drop files or RTF (Rich Text Format) text.

How to Drag and Drop Text

To drag and drop text, first mark the text in one of the ways described in "Selecting (Marking) Text in UNIX Environments" on page 42. To use default drag and drop, simply use the middle mouse button (MB2) to drag the text where you want it. To use nondefault drag and drop, press and hold the Alt (or Extend Char) key before you release MB2.

Searching For and Replacing Text Strings in UNIX Environments

What Are the Find and Replace Dialog Boxes?

The Find and Replace dialog boxes (Display 2.10 on page 47) enable you to search for and replace strings in SAS text editor windows such as the Program Editor, the SCL editor, or the Notepad.

Opening the Find Dialog Box

To search for a string, open the Find dialog box by issuing the DLGFIND command or by selecting

Edit ▶ Find

Description of the Find Dialog Box Options

The Find dialog box works like the Replace dialog box, except it does not have the **Replace** field or the Replace and Replace All buttons.

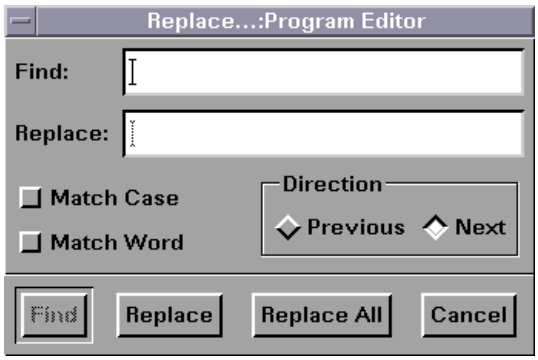
For a description of the options on the Find Dialog Box, see "Description of the Replace Dialog Box Options" on page 47.

Opening the Replace Dialog Box

To replace one text string with another, open the Replace dialog box by issuing the DLGREPLACE command or by selecting

Edit ▶ Replace

Display 2.10 Replace Dialog Box



Description of the Replace Dialog Box Options

To find a character string, type the string in the **Find** field, and select Find. To change a character string, type the string in the **Find** field, type its replacement in the **Replace** field, and select Replace. To change every occurrence of the string to its replacement string, select Replace All.

You can tailor your find or replace operation using the following buttons:

Match Case
tells the search to match the uppercase and lowercase characters exactly as you entered them.

Match Word
searches for the specified string delimited by space, end-of-line, or end-of-file characters.

Previous
searches from the current cursor position toward the beginning of the file.

Next
searches from the current cursor position toward the end of the file.

Sending Mail from within Your SAS Session in UNIX Environments

Default E-mail Protocol in SAS

By default, SAS uses SMTP (Simple Mail Transfer Protocol) to send e-mail from within your SAS session. You can use the EMAILSYS system option to specify which script or protocol you want to use for sending electronic mail. See "EMAILSYS System Option" on page 326 for more information.

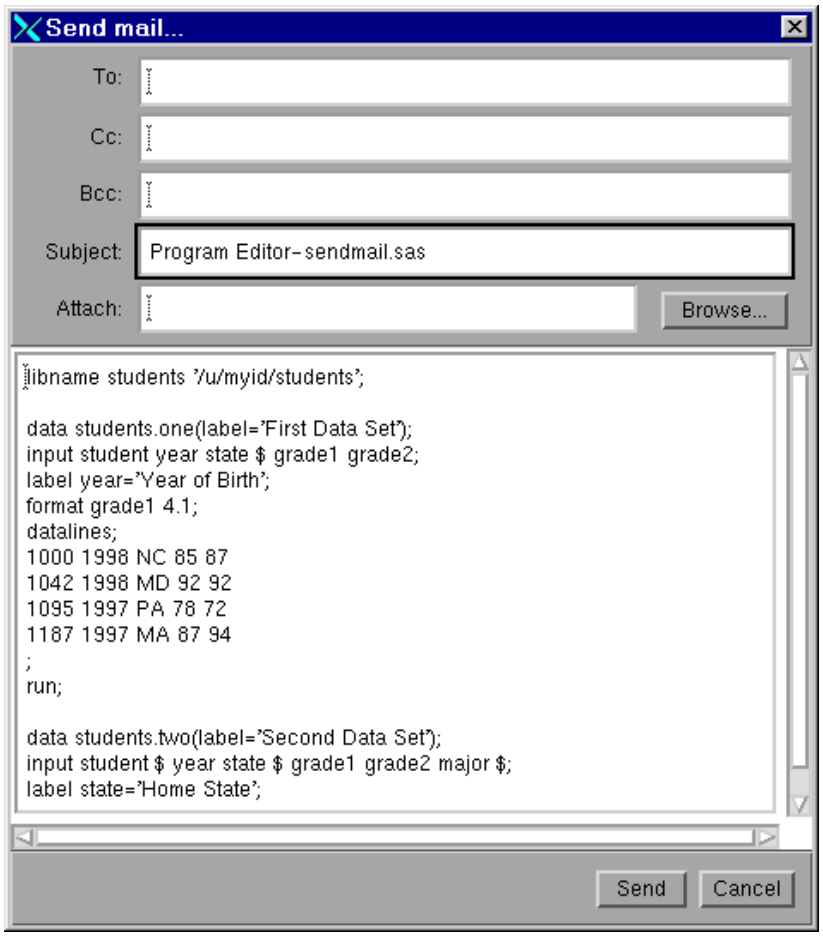
For more information about the SMTP e-mail interface, see *SAS Language Reference: Concepts*.

What Is the Send Mail Dialog Box?

The Send Mail dialog box (Display 2.11 on page 48) enables you to send e-mail without leaving your current SAS session. To invoke the dialog box, issue the DLGSMAIL command or select

File ▶ Send Mail

Display 2.11 Send Mail Dialog Box



Sending E-mail Using the Send Mail Dialog Box

To send e-mail, complete the following steps as needed:

- Enter the ID(s) of the e-mail recipients in the **To**, **CC**, and **BCC** fields. Separate multiple addresses with either blanks or commas.
- Edit the entry in the **Subject** field as needed.
- Enter the name of the file that you want to send in the **Attach** field. Separate multiple filenames with blanks. You can also use Browse to select a file.

  Note: Some external scripts do not support sending e-mail attachments. △

- Type your message in the message area or edit the contents grabbed from the active SAS text window.
- Click Send.

To cancel a message, click Cancel.

Sending the Contents of a Text Window

You can e-mail the contents of an active SAS text window (such as the Program Editor or the Log) by using the Send Mail dialog box. To open the Send Mail dialog box, select

File ▶ Send Mail

SAS automatically copies the contents of the active SAS window and includes the text in the body of your e-mail. You can change or add to the e-mail message in the Send Mail dialog box.

If you do not want to include the contents of the active SAS editor window in your message, select

Edit ▶ Clear All

before invoking the Send Mail dialog box.

Sending the Contents of a Non-Text Window

To send the contents of a non-text window (such as a graph generated by SAS/GRAPH or an image from your PROC REPORT output), select

File ▶ Send Mail

from the active SAS window. SAS automatically copies the image data to a temporary file and enters that filename into the **Attach** field of the Send Mail dialog box. To change the default file type for this temporary file, see "Changing the Default File Type" on page 49.

SAS only copies the portion of the image that is visible in the active window, along with the window frame and title. This behavior is similar to using the DLGSCRDUMP command. For more information, see "DLGSCRDUMP Command" on page 208.

If you do not want to attach this image to your e-mail, clear the contents of the **Attach** field.

Note: Some external scripts do not support sending e-mail attachments. △

Changing the Default File Type

You can change the default file type for the temporary file that SAS creates using the Preferences dialog box. To open the Preferences dialog box, select

Tools ▶ Options ▶ Preferences

On the **DMS** tab in the **Image type for Email attachments** box, select one of the following file types:

- Portable Network Graphics (.png)
- Graphics Interchange Format (.gif)
- Tagged Image File Format (.tif).

Configuring SAS for Host Editor Support in UNIX Environments

Requirements for Using a Host Editor

SAS supports the use of a host text editor with the Motif interface, so you can use an editor such as vi or EMACS with your SAS session. There is no host editor set as the default host editor, so you must specify one to use this feature. Host editor support requires the use of the **motifxsassm** client. (See "The SAS Session Manager (motifxsassm) in UNIX" on page 33 for more information.)

Invoking and Using Your Host Editor

How to Open and Use the Host Editor

To use your host text editor with SAS, complete the following steps:

1. Specify the command required to invoke your editor with the EDITCMD system option.
2. Invoke the editor as needed with the HOSTEDIT command.

The HOSTEDIT command passes data from a SAS window to the host editor. When you save in the host editor, the data is copied back into the SAS window if the window is writable.

After you return to the SAS text editor window, you can issue the UNDO command to undo all of the changes that you made with your host editor. You must issue the UNDO command a second time to return to the state of the window before the HOSTEDIT command was issued. If you issue the HOSTEDIT command in a read-only window, you can save your editing changes to an external file, but the SAS text editor window remains unchanged.

See "EDITCMD System Option" on page 325 and "HOSTEDIT Command" on page 213 for more information.

Example 1: Invoking SAS to Use xedit with the HOSTEDIT Command

Some systems have an X-based editor installed that is called xedit. If you want to use xedit with the HOSTEDIT command, you can invoke SAS with the following command:

```
sas -editcmd '/usr/local/bin/xedit'
```

Example 2: Invoking SAS to Use vi

The vi editor is a terminal-based editor that requires a terminal window. The xterm client's **-e** option runs a program when the xterm client is invoked. To use the EDITCMD option to display an xterm client in conjunction with vi, invoke SAS as follows:

```
sas -editcmd '/usr/bin/X11/xterm -e /usr/bin/vi'
```

Troubleshooting the Transfer of Text Attributes

Text attributes, such as color and highlighting, are not transferred between a host editor window and a SAS text editor window. Issue the HEATTR ON command to display a dialog box that will warn you if you are editing text with highlighting and color attributes that will be removed by the host editor. This dialog box prompts you to continue or abort the HOSTEDIT command. Specify HEATTR OFF to suppress this dialog box.

Getting Help in UNIX Environments

The **Help** menu is always available within your SAS session. Here are descriptions of the help topics available from the **Help** menu:

Using this Window
Help information that is relevant to the active window. This is the same as clicking the Help button or pressing the F1 key.

SAS Help and Documentation
tutorials and sample programs to help you learn how to use SAS, comprehensive documentation for all products installed at your site, and information about contacting SAS for additional support.

Getting Started with SAS Software
opens a tutorial that will help you get started with SAS.

Learning SAS Programming
opens the SAS Online Tutor, if it is installed, to help you develop your SAS programming skills. SAS Online Tutor is a separately licensed product.

SAS on the Web
provides links to useful areas on the SAS Institute Web site, including technical support, frequently asked questions, sending feedback to SAS, and the SAS homepage.

About SAS System
opens the About SAS System dialog box which provides information about SAS software, your operating environment, and Motif.

CHAPTER

3

Customizing the SAS Windowing Environment

Overview of Customizing SAS in X Environment

The SAS windowing environment supports the use of X-based graphical user interfaces (GUIs). In UNIX environments, SAS provides an X Window System interface that is based on the Motif style. For more information on SAS in the X environment, see "Description of SAS in the X Environment" on page 31.

You can customize your working environment by using X resources.

Overview of X Resources

Introduction to X Resources

X clients usually have characteristics that can be customized; these properties are known as *X resources*. Since SAS functions as an X windows client, many aspects of the appearance and behavior of the SAS windowing environment are controlled by X resources. For example, X resources can be used to define a font, a background color, or a window size. The resources for an application, such as SAS, are placed in a *resource database*.

SAS functions correctly without any modifications to the resource database. However, you may want to change the default behavior or appearance of the interface. There are several ways to specify your customizations. Some methods modify all SAS sessions displayed on a particular X server. Some methods affect all SAS sessions run on a particular host. Other methods affect only a single SAS session.

If you need more information on X Window System clients and X resources, refer to the documentation provided by your vendor.

Syntax for Specifying X Resources

A resource specification has the following format:

resource-string: *value*

The *resource string* usually contains two identifiers and a separator. The first identifier is the client or application name (**SAS**), the separator is a period (.) or asterisk (*) character, and the second identifier is the name of the specific resource. The *value* given may be a Boolean value (**True** or **False**), a number, or a character string, depending on the resource type.

The application name and resource name can both specify an *instance value* or a *class value*. A specification for a class applies to a larger scope than a single instance.

The following are sample resource specifications:

```
SAS.startSessionManager: True
SAS.maxWindowHeight: 100
SAS.awsResizePolicy: grow
```

Refer to your X Window System documentation for more information on resource specifications.

Methods for Customizing X Resources

The following list describes the methods that you can use to customize X resources.

- Use the Font dialog box, the Preferences dialog box, or the Resource Helper to customize your SAS session. All of these tools write X resource definitions out to a location that SAS will read the next time you start a SAS session. See "Modifying X Resources through the Preferences Dialog Box" on page 57, "Setting X Resources with the Resource Helper" on page 62, and "Customizing Fonts in UNIX Environments" on page 80 for more information on these tools.

  Note: The settings that you specify in the Preferences dialog box will override any command line settings. △

- Specify session-specific resources by using the -xrm option on the command line for each invocation of SAS. For example, the following command specifies that SAS will not display the Confirm dialog box when you exit your SAS session:

  ```
  sas -xrm 'SAS.confirmSASExit: False'
  ```

  You can specify the -xrm option as many times as needed. You must specify the -xrm option for each resource.

  Note: If you normally invoke SAS with a shell script, you should protect the quote characters from the shell with the backslash (\) character:

  ```
  sasscript -xrm \'SAS.confirmSASExit: False\'
  ```

  △

- Add resource definitions to a file in your home directory. If you place resources in a file that X Toolkit normally searches for when applications are invoked, these resources will be loaded when you invoke SAS. For information about where the X Toolkit searches for resources, refer to the documentation for the X Windows System.

  You can also add resources to the resource database after SAS has initialized by running xrdb. For example, the following command merges the definitions in the MyResources file into the resource database:

  ```
  xrdb -merge myresources
  ```

- Create a subdirectory for storing resource definitions. (This subdirectory is usually named **app-defaults**.) Set the XUSERFILESEARCHPATH environment variable to the pathname of this subdirectory. You can use **%N** to substitute an application class name for a file when specifying the XUSERFILESEARCHPATH environment variable. Specify the definition for this environment variable in the initialization file for your shell, for example, the $HOME/.login, $HOME/.cshrc, or $HOME/.profile files, to ensure that the XUSERFILESEARCHPATH variable is defined for each shell that is started.

  Create a file called **SAS** in the subdirectory identified by XUSERFILESEARCHPATH. Include your resource definitions in this file.

  Note: Alternatively, you could set the XAPPLRESDIR environment variable to the pathname of the subdirectory that stores your resource definitions. The XAPPLRESDIR and XUSERFILESEARCHPATH environment variables use a slightly different syntax to specify the location of your resource definitions. The location specified by the XUSERFILESEARCH environment variable takes precedence over the location specified by the XAPPLRESDIR variable. For more information, see the X man page. △

- If you want the customized resource definitions to be used for all users on a particular host, create a file called **SAS** to contain your resource definitions, and store this file in the system **app-defaults** directory.

For more information on X resources, refer to the X Window System documentation supplied by your vendor or to other documentation on the X Window System.

Modifying X Resources through the Preferences Dialog Box

What Is the Preferences Dialog Box?

The Preferences dialog box enables you to control the settings of certain X resources. Changes made through the Preferences dialog box (with the exception of those resources on the **General** tab) become effective immediately, and the settings are saved in the SasuserPrefs file in your Sasuser directory.

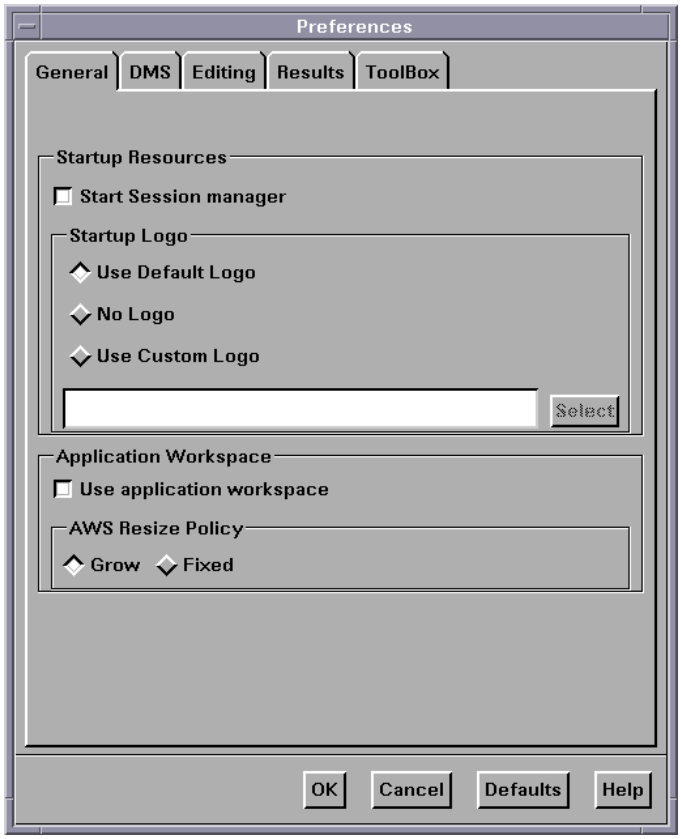
Note: The settings that you specify in the Preferences dialog box will override any command line settings. △

Opening the Preferences Dialog Box

You can invoke the Preferences dialog box by issuing the DLGPREF command or by selecting

Tools ▶ Options ▶ Preferences

Display 3.1 Preferences Dialog Box



Description of the Options on the Preferences Dialog Box

Modifying the General Settings

To modify the General settings, select the **General** tab in the Preferences dialog box.

Start Session manager
specifies whether you want the session manager to be started automatically when you start your SAS session. If you want to use your host editor in your SAS session, the session manager must be running. The session manager enables you to interrupt or terminate your SAS session and minimize and restore all of the windows in a SAS session. See "The SAS Session Manager (motifxsassm) in UNIX" on page 33 and "Configuring SAS for Host Editor Support in UNIX Environments" on page 49 for more information. This check box sets the **SAS.startSessionManager** resource.

Startup Logo
specifies whether you want SAS to display an XPM file while your SAS session is being initialized and, if so, which file.

If you select **Use Default Logo**, SAS uses the default file for your site. If you select **No Logo**, then no file is displayed. If you select **Use Custom Logo**, then you can either enter the XPM filename directly in the text field or press Select to open the File Selection dialog box. This check box sets the **SAS.startupLogo** resource.

Use application workspace
confines all windows displayed by an application to a single Application Work Space. This check box sets the **SAS.noAWS** resource. You must exit and reopen the windows for changes to this resource to take effect.

AWS Resize Policy
controls the policy for resizing AWS windows as interior windows are added and removed. (See "Workspace and Gravity in a SAS Session" on page 31 and "Window Types" on page 32.)

Grow
The AWS window will attempt to grow any time an interior window is grown or moved (to make all of its interior windows visible), but it will not shrink to remove unused areas.

Fixed
The AWS window will attempt to size itself to the size of the first interior window and will not attempt any further size changes.

This check box sets the **SAS.awsResizePolicy** resource.

Modifying the DMS Settings

To modify the DMS settings, select the **DMS** tab in the Preferences dialog box.

Use menu access keys
activates pull-down menu mnemonics. When mnemonics are turned on, you can select menu items by typing the single, underlined letter in the item. This check box sets the **SAS.usePmenuMnemonics** resource.

Confirm exit
displays the Exit dialog box when you exit your SAS session. This check box sets the **SAS.confirmSASExit** resource.

Save Settings on Exit
tells SAS to issue the WSAVE ALL command when you exit your SAS session. This command saves the global settings, such as window color and window position, that are in effect for all windows that are currently open. These settings are saved in your Sasuser.Profile catalog. This check box sets the **SAS.wsaveAllExit** resource.

Note: For the WSAVE command to work, your window manager must support explicit window placement. Consult the documentation for your window manager to determine how to configure your window manager. For example, if you are running Exceed, open the Screen Definition Settings dialog box and deselect **Cascade Windows**. △

Backup Documents
enables you to specify whether you want SAS to automatically save (at the interval specified by the **SAS.autoSaveInterval** resource) the documents that you currently have open. This check box sets the **SAS.autoSaveOn** resource.

Help & Documentation Browser -- Netscape Path
specifies the path name for the Web browser that you want to use to view the SAS Help and Documentation. This field sets the **SAS.helpBrowser** resource.

Image type for Email attachments
specifies the default file type for the temporary file that SAS creates when sending the contents of a non-text window via e-mail. Examples of non-text windows include a graph generated by SAS/GRAPH or an image from your PROC REPORT output. For more information, see "Sending the Contents of a Non-Text Window" on page 49.

Modifying the Editing Settings

To modify the Editing settings, select the **Editing** tab in the Preferences dialog box.

Default paste buffer
defines an alias for the default SAS buffer. The following list describes the paste buffer alias names and the X buffer with which each name is associated.

XPRIMARY
X primary selection (**PRIMARY**)

XSCNDARY
X secondary selection (**SECONDARY**)

XCLIPBRD
X clipboard (**CLIPBOARD**)

XTERM
exchange protocol used by the xterm client

XCUT*n*
X cut buffer where *n* is between 0 and 7, inclusive

This check box sets the **SAS.defaultPasteBuffer** resource. See "Controlling Pull-Down Menus in UNIX Environments" on page 91 for more information about cut-and-paste butters.

Automatically store selection
generates a STORE command every time you mark a region of text with the mouse. This check box sets the **SAS.markPasteBuffer** resource.

Cursor
controls the editing mode in SAS text editor windows. The **Insert** and **Overtype** check boxes set the **SAS.insertModeOn** resource to **True** and **False**, respectively.

Modifying the Results Settings

The items on the **Results** tab affect only output that is produced through the Output Delivery System (ODS). To modify these settings, select the **Results** tab in the Preferences dialog box.

For a complete description of ODS, refer to *SAS Output Delivery System: User's Guide*.

Create Listing
opens the ODS Listing destination, which produces monospace output. Selecting this check box is equivalent to entering the ODS LISTING SELECT ALL statement.

Create HTML
opens the ODS HTML destination, which produces output that is formatted in Hypertext Markup Language. Selecting this check box is equivalent to entering the ODS HTML SELECT ALL statement.

Folder
specifies a destination for HTML files. Specifying a directory in this field is equivalent to specifying a directory with the PATH option in the ODS HTML statement.

Use WORK Folder
tells the ODS to send all HTML files to your WORK directory. Selecting this check box is equivalent to specifying the pathname of your WORK directory with the PATH option in the ODS HTML statement.

Style
specifies the style definition to use for HTML output. The style definition controls such aspects as color, font name, and font size. Specifying a style in this field is equivalent to specifying a style with the STYLE option in the ODS HTML statement. You can specify any style that is defined in the \ODS\PREFERENCES\STYLES key in the SAS Registry. You can open the SAS Registry by issuing the REGEDIT command or by selecting

Solutions ▶ Accessories ▶ Registry Editor

View results as they are generated
tells SAS to automatically display results files when they are generated. If you select this check box, make sure that **Password protect HTML file browsing** is deselected.

Password protect HTML file browsing
tells SAS to prompt you for your password before sending HTML files to your browser. If you select this check box, make sure that **View results as they are generated** is deselected. This check box sets the **SAS.htmlUsePassword** resource.

Modifying the ToolBox Settings

The items on the **ToolBox** tab of the Preferences dialog box affect both the toolbar and the command window. To modify these settings, select the **ToolBox** tab in the Preferences dialog box.

Display tools window
determines whether to display the default toolbox. This check box sets the **SAS.defaultToolBox** resource.

Display command window
determines whether to display the command window. This check box sets the **SAS.defaultCommandWindow** resource.

Auto Complete Commands
specifies whether SAS automatically fills in the remaining letters of a command as you type a command in the command window that begins with the same letter as a command that you have entered previously. If both this check box and **Save Commands** are selected, then SAS can automatically fill in commands that were entered in previous sessions. This check box sets the **SAS.autoComplete** resource.

Save Commands
specifies whether SAS saves the commands that you enter in the command window and how many commands are saved. You can specify a number from 0 to 50. If you specify 0, no commands will be saved. If you specify 1 or more, that number of commands is saved in the file **commands.hist** in your Sasuser directory. If this check box is selected, then SAS will be able to automatically fill in (see **Auto Complete Commands**) commands that were entered in previous sessions. This field sets the **SAS.commandsSaved** resource.

Combine windows
combines the toolbox and command window into one window. The toolbox and command window are combined by default. This check box sets the **SAS.useCommandToolBoxCombo** resource.

Use arrow decorations
adds arrows to both ends of the combined toolbox/command window. This check box sets the **SAS.useShowHideDecorations** resource.

Always on top
keeps the toolbox or the combined toolbox/command window on top of the window stack. This check box is selected by default, which might cause problems with window managers and other applications that want to be on top of the window stack. If you have such a situation, turn off this feature. This check box sets the **SAS.toolBoxAlwaysOnTop** resource.

Toolbox Persistent
specifies whether the toolbox that is associated with the Program Editor stays open when you close the Program Editor. By default, the Program Editor toolbox stays open whenever you close the Program Editor. If you deselect this check box, then the toolbox will close if you close the Program Editor. This check box sets the **SAS.isToolBoxPersistent** resource.

The items in the Tools area affect the individual tools in the toolbox.

Use large tools
controls whether tool icons are displayed as 24x24 or 48x48 pixels. The default is 24x24. This check box sets the **SAS.useLargeToolBox** resource.

Use tip text
specifies whether tool tip text is displayed when you position your cursor over a tool in the toolbox. Some window managers may place the toolbox tip behind the toolbox. If this happens in your environment, deselect this check box. This check box sets the **SAS.useToolBoxTips** resource.

delay
controls the delay in milliseconds before popping up the toolbox tip. This check box sets the **SAS.toolBoxTipDelay** resource. You can enter a value directly into the field or use the arrows to the right of the field to change the value.

Setting X Resources with the Resource Helper

Introduction to the Resource Helper

With Resource Helper, you can customize the key definitions and the colors of the SAS interactive interface. Resource Helper creates SAS resource definitions and stores them in a location where the Resource Manager can find them. See "How the Resource Helper Searches for X Resources" on page 66 for a list of the locations that Resource Helper searches for resource definitions. Resource settings that are saved with Resource Helper will take effect the next time you start a SAS session.

You can start Resource Helper from within a SAS session or from your shell prompt.

How to Start the Resource Helper

Starting the Resource Helper from a SAS Session

Start the SAS Resource Helper from a SAS window by entering

```
reshelper
```

on the command line in the command window.

Display 3.2 Main Window for Resource Helper

Starting the Resource Helper from a Shell Prompt

Resource Helper is installed into the **/utilities/bin** subdirectory in the directory where SAS is installed (**!SASROOT**). The name of the executable module is **reshelper**. For example, if SAS is installed in **/usr/local/sas91**, you start Resource Helper by typing the following command:

```
/usr/local/sas91/utilities/bin/reshelper &
```

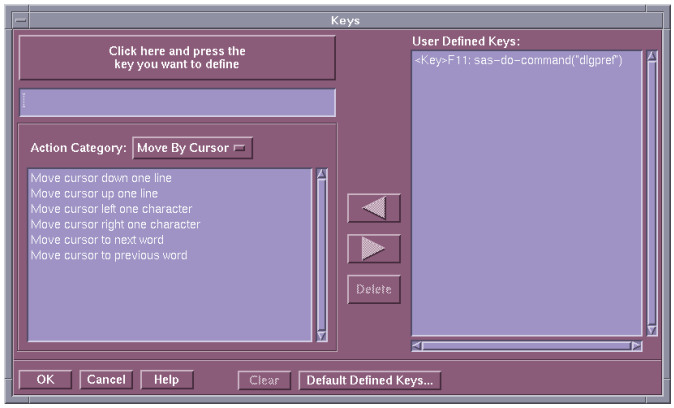
Defining Keys with the Resource Helper

How to Define a Key

To define a key, follow these steps:

1 Start the Resource Helper (see "How to Start the Resource Helper" on page 62) and select the Keys icon.

Display 3.3 Keys Window for Resource Helper



Key definitions are divided into several **`Action Categories`**:

- **`Move By Cursor`**
- **`Move By Field`**
- **`Edit`**
- **`Miscellaneous`**
- **`All Actions`**

2 Select Click here and press the keys you want to define.

3 Press the key or combination of keys that you want to assign an action to. For example, press **`F12`**. If a default SAS translation has already been assigned to the key combination, Resource Helper displays the default translation.

4 Select the action category menu button to open a list of action categories. Select the action category that you want. For example, if you want to define a key to delete the current field, select **`Edit`** as your **`Action Category`**. Resource Helper will display a list of actions in that category.

5 Select an action from the list. For example, **`Delete current field`**. Resource Helper can assign only one action to a translation. If the action that you select requires an argument (such as **`sas-action-routine`**), Resource Helper prompts you for the argument.

Resource Helper displays the key combination and its new definition:

```
None<Key>F12: sas-delete()
```

Note: If you select the sas-function-key action routine, then the key definition is automatically displayed in the KEYS window. If you choose another action routine and if you want the definition to appear in the KEYS window, you will need to define a window label for the key. See "Syntax of the SAS.keysWindowLabels Resource" on page 76 for information on defining labels in the KEYS window. △

6 Select the right arrow to add this key translation to the list of **User-Defined Keys**.

7 Select OK to exit the Keys window after you have finished defining key translations.

8 To save your translations permanently, from the Resource Helper pull-down menus, select

File ▶ Save Resources

To modify a key definition that is already in the **User-Defined Keys** list, select the definition, select the left arrow to remove the definition from the list, and edit the definition.

To delete a definition from **User-Defined Keys**, select it and select **Delete**.

Clear clears the key definition edit window.

Default Defined Keys displays the default key definitions for your system.

Troubleshooting Incorrect Key Definitions

In most cases, using Resource Helper is much easier and faster than defining the resources yourself. However, the X Window System searches for resources in several places, so it is possible for Resource Helper to pick up the wrong key symbol for the key you are trying to define. If you get unexpected results while using Resource Helper, you might need to define your key resources yourself. See “Defining Key Translations” on page 74 for more information.

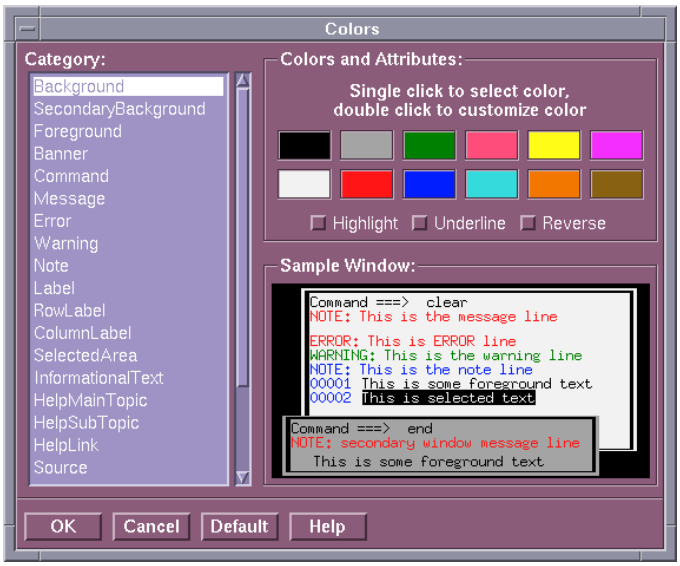
Modifying the Color of a SAS Window Using the Resource Helper

How to Use the Color Window

You can modify the color of part of a SAS window as follows:

1 Start Resource Helper and select the Colors icon.

Display 3.4 Colors Window for Resource Helper

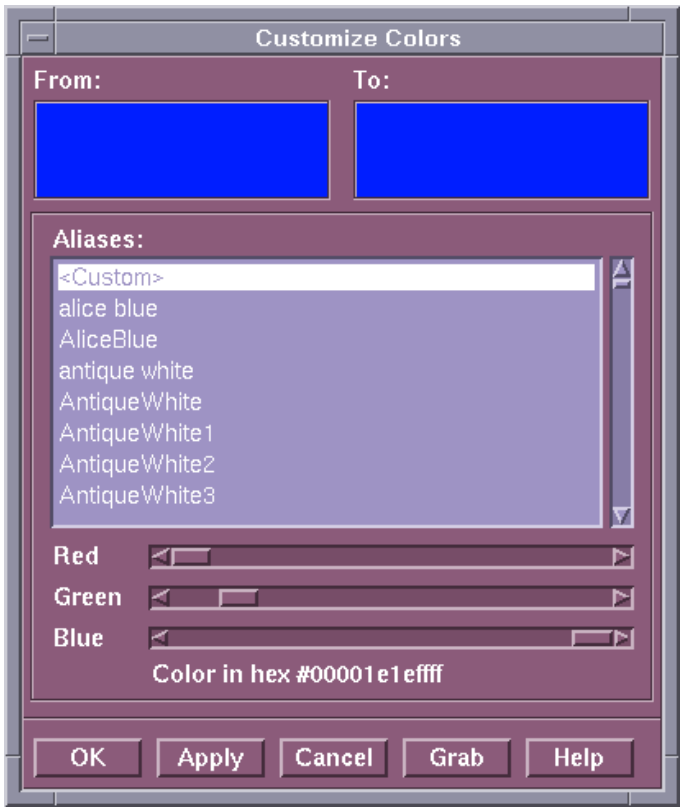


2 Select a category from the **Category** window.

3 Click a color and/or attribute in the Colors and Attributes window, or double-click a color to open the Customize Colors window, shown in the following display.

You can also change the attributes of some categories of SAS windows. The attributes options are **Highlight**, **Underline**, or **Reverse**.

Display 3.5 Customize Colors Window for Resource Helper



You can customize a color by

- selecting a new **Alias**
- moving the **Red, Blue,** or **Green** sliders
- selecting Grab and clicking a color anywhere on your screen.

4 Select OK to exit the Colors window after you have finished defining your color settings.

The result is displayed in the **Sample Window**. The hex value of the color is displayed at the bottom of the window.

Example: Changing the Color of a SAS Window

For example, double-click **Red** in the **Colors** window. The **From:** display shows the red currently used by SAS windowing environment. Click **Aquamarine** under **Aliases** and observe the change in the **To:** display. Move the **Red**, **Green**, and **Blue** sliders with your mouse button and note the changes in the color of the **To:** display. Click Apply and note the difference in the color displayed as **Red** in the **Colors** window. Select OK to save your changes.

Returning to the Default Settings

Select Defaults to restore your color settings to their default values.

Permanently Saving Your Color Settings

To save your color settings permanently, from the Resource Helper pull-down menus, select

File ▶ Save Resources

How the Resource Helper Searches for X Resources

The following list describes the locations where the Resource Helper searches for resource definitions and the order in which it searches these locations.

1. Resource Helper loads the resources in the file pointed to by the XENVIRONMENT environment variable. If XENVIRONMENT is not set, Resource Helper loads the resources in the **~/.Xdefaults-***hostname* file, where *hostname* is the name of the machine on which Resource Helper is running.
2. Resource Helper loads the resources defined in the RESOURCE_MANAGER property. If the RESOURCE_MANAGER property is the first location in which Resource Helper finds resources, the RESOURCE_MANAGER property will override any resources that you generate with Resource Helper.

   To determine if any resources have been defined in your RESOURCE_MANAGER property, issue the following command:

   ```
   xrdb -q | more
   ```

   If no listing is returned, the RESOURCE_MANAGER property does not exist. In this case, Resource Helper loads the resources defined in the **~/.Xdefaults** file.
3. Resource Helper loads the resources in the file pointed to by the XUSERFILESEARCHPATH environment variable.

   You can use **%N** to substitute an application class name for a file when specifying the XUSERFILESEARCHPATH environment variable. For example, to point to **/usr/local/resources** as the location of all the resources for any application, issue the following command in the Bourne or Korn shells:

   ```
   export XUSERFILESEARCHPATH=\
   /usr/local/resources/%N
   ```

   In the C shell, the command is

   ```
   setenv XUSERFILESEARCHPATH \
   /usr/local/resources/%N
   ```

   As a result, when SAS is invoked, the file pointed to by XUSERFILESEARCHPATH is

   ```
   /usr/local/resources/SAS
   ```

   SAS is the application class name for SAS.
4. Resource Helper loads the resources in the file specified by the XAPPLRESDIR environment variable. The application's class name is appended to the XAPPLRESDIR environment variable and the resulting string is used to search for resources. For example, if you issue the following command in the Bourne or Korn shells:

   ```
   export XAPPLRESDIR=/usr/local/app-defaults
   ```

at the next invocation of SAS, the application's class name is appended to the path:

```
/usr/local/app-defaults/SAS
```

In the C shell, the command is

```
setenv XAPPLRESDIR /usr/local/app-defaults
```

5 Resource Helper loads the resources in the file named **~/SAS**.

6 Resource Helper loads the resources in the file or substitution specified by the XFILESEARCHPATH environment variable.

Note: To determine if an environment variable has been set, you can issue the following command:

```
env|grep <environment_variable>
```

△

7 Resource Helper loads the resources defined in **/usr/lib/X11/app-defaults**. Resource Helper does not need to have write access to this file, but it must be able to read the file and add the SAS resources to a writable resource file. Resource Helper does not generate a warning message if the file is not present or if it cannot read the file.

8 Resource Helper loads the fallback resources that are defined in the SAS code.

Except for the **/usr/lib/X11/app-defaults** file, Resource Helper tries to write the new resources to the same directory and file where it first found SAS resources. This location must be a writable file in a writable directory. If Resource Helper cannot write to the file, the SAS resources in that file will remain in effect and any new or modified resources generated by Resource Helper will not take effect. If this happens, Resource Helper displays an error dialog box that contains the file or directory and suggests a way to fix the problem.

Customizing Toolboxes and Toolsets in UNIX Environments

Techniques for Customizing Toolboxes

You can customize toolboxes

- through the Preferences dialog box. The Preferences dialog box enables you to customize the appearance and behavior of toolboxes. See "Modifying X Resources through the Preferences Dialog Box" on page 57 and "Modifying the ToolBox Settings" on page 60 for information on using the Preferences dialog box.
- by specifying SAS resources in your resource file. "X Resources That Control Toolbox Behavior" on page 67 describes the SAS resources that affect toolboxes.
- through the Tool Editor. The Tool Editor enables you to customize the individual tools in a toolbox. See "Using the Tool Editor" on page 68 for more information.

X Resources That Control Toolbox Behavior

You can control the behavior of toolboxes with the following SAS resources:

SAS.autoComplete: True | False
specifies whether SAS automatically fills in the remaining letters of a command as you type a command in the command window that begins with the same letter as a command that you have entered previously. The default value is True.

SAS.commandsSaved : close up | *n*]
specifies whether SAS saves the commands that you enter in the command window and how many commands are saved. You can specify a number from 0 to 50. If you specify 0, no commands will be saved. If you specify 1 or more, that number of commands is saved in the file **commands.hist** in your Sasuser directory. If you specify 1 or more for this resource and **SAS.autoComplete** is True, then SAS will be able to automatically fill in commands that were entered in previous sessions. The default value is 25.

SAS.defaultToolBox: True | False
controls opening the default toolbox when SAS is invoked. The default is True.

SAS.isToolBoxPersistent: True | False
controls whether the toolbox that is associated with the Program Editor stays open when you close the Program Editor. The default value is True.

SAS.toolBoxAlwaysOnTop: True | False
controls whether the toolbox is always on top of the window stack. The default value of True might cause problems with window managers that are not Motif interface window managers or other applications that want to be on top of the window stack. If you have such a situation, set this resource to False.

SAS.toolBoxTipDelay: *delay-in-milliseconds*
sets the delay in milliseconds before displaying the toolbox tip. The default is 750.

SAS.useCommandToolBoxCombo: True | False
controls whether the command window and toolbox are joined or separated. The **SAS.defaultToolBox** and **SAS.defaultCommandWindow** resources control whether the toolbox and command window are displayed. If both are displayed, this resource controls whether they are joined or separated. The default value is True.

SAS.useLargeToolBox: True | False
controls whether tool icons in the toolbox are displayed as 24x24 pixels or 48x48 pixels. The default is False (24x24 pixels).

SAS.useShowHideDecorations: True | False
controls whether the combined command/toolbox window has arrows at the left and right. You can use these arrows to hide or show portions of the window as they are needed. The default value is False.

SAS.useToolBoxTips: True | False
determines if toolbox tip text is displayed. Some window managers might place the toolbox tip behind the toolbox. If this happens in your environment, set this resource to False. The default is True.

Using the Tool Editor

What Is a Toolset?

The Tool Editor enables you to create custom toolsets for your SAS applications. A *toolset* is a set of predefined tools that is associated with an application. Toolsets make it easier for individual users to customize their application toolboxes. If you create a toolset for an application, users can press the Actions button in the Tool Editor and choose the tools that they want to appear in their toolboxes. Users do not have to define the icons, commands, tip text, and IDs for those tools.

For example, you can define a default toolbox for your application that includes tools for opening files, cutting, copying, and pasting text, and saving files. You can define a toolset that includes those tools and tools for opening the Preferences dialog box, opening the Replace dialog box, and entering the RECALL command. These additional tools will not appear in the users' toolbox unless a user adds them to their toolbox with the Tool Editor. See "Changing an Existing Tool" on page 70 and "Creating or Customizing an Application- or Window-Specific Toolset" on page 72 for more information.

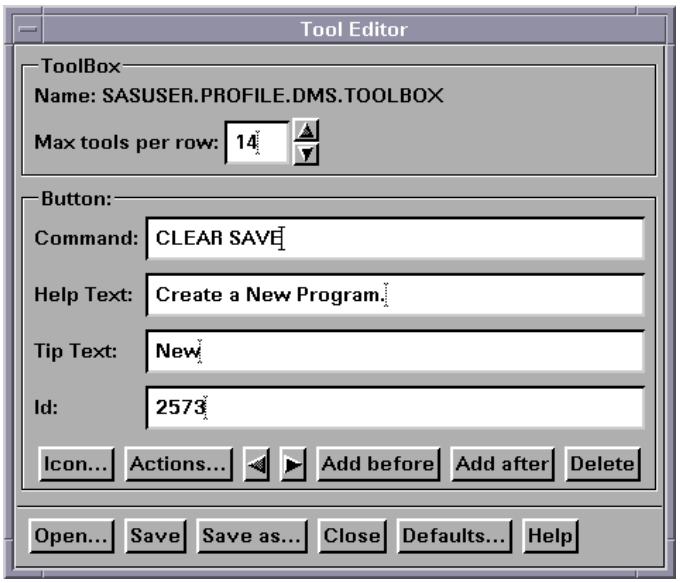
Invoking the Tool Editor

You can change the appearance and contents of a toolbox using the Tool Editor. To invoke the Tool Editor, select

Tools ► Options ► Edit Toolbox

Alternatively, you can issue the TOOLEDIT command as described in "TOOLEDIT Command" on page 216.

Display 3.6 Tool Editor Dialog Box



By default, the Tool Editor edits the current toolbox. To edit a different toolbox, click Open and specify the libref, catalog, and entry name for the toolbox you want to edit.

Once you invoke the Tool Editor, the toolbox goes into preview mode. In preview mode, clicking a tool icon makes that icon the current icon and displays its associated commands in the **Command** field. The current icon always appears pressed.

Changing the Appearance of the Entire Toolbox

The items in the area of the Tool Editor labeled "ToolBox" affect the entire toolbox:

Name
displays the catalog entry that you are editing. The default toolbox is named SASUSER.PROFILE.DMS.TOOLBOX.

Max tools per row
specifies how the icons in the toolbox are arranged. The default value creates a horizontal toolbox. One tool per row creates a vertical toolbox.

Changing an Existing Tool

When you open the Tool Editor, the first icon is the current icon, and information about the current icon appears in the Button area of the dialog box. To change an existing tool, you can select a tool from the toolset displayed by the Actions button or you can modify the fields individually.

Note: The Actions button will display a toolset only if a toolset is associated with (has the same entry name as) the toolbox that you are editing. See "Saving Changes to the Toolbox or Toolset" on page 71 for more information. △

To use Actions, select the tool that you want to change, and then select Actions. The Tool Editor displays the toolset associated with toolbox. If you select a tool from this toolset, the Tool Editor enters the appropriate information into the button fields for you.

To modify the fields individually:

1. Select the icon you want to change.
2. Click in and change the Button fields as appropriate.

   Command
   specifies the command or commands that you want executed when you click on the icon. You can use any windowing environment command available under UNIX. For information on commands that are valid in all operating environments, see *SAS Language Reference: Dictionary*. Separate commands with a semi-colon (;). For example, you could create an icon to open the Change Working Directory dialog box by using the DLGCDIR command.

   Help Text
   is used for applications that are designed to be run under Windows or OS/2. The help text is displayed in the AWS status line on Windows and OS/2 when a toolbox is ported to and loaded on those platforms.

   Tip Text
   specifies the text that is displayed when you position the cursor over the icon.

   Id
   is useful if you are creating toolboxes for SAS/AF applications. The ID is the identifier of the corresponding pull-down menu item in the application. This number is the value assigned to the item in the ID option of the ITEM statement in PROC PMENU. If you specify an ID, then the application can set the state of the PMENU item to match the state of the tool in the toolbox, and it can make the PMENU item active or inactive to match whether the PMENU item is active or inactive. If you do not specify an ID, the ID defaults to 0.
3. Change the icon if necessary.
   a. Select Icon or double-click an icon in the preview toolbox. The Tool Editor opens the Select A Pixmap dialog box, which displays the icons provided with SAS. These icons are divided into several categories such as SAS windows, data, analysis, numbers and symbols, files, folders, and reports, and so on. To change categories, select the arrow to the right of the **Icon Category** field and select a new category.
   b. Select the icon you want to use and then select OK.
4. Save your changes as described in "Saving Changes to the Toolbox or Toolset" on page 71.

Adding Tools to the Toolbox

To add a tool to the toolbox:

1. Select the icon next to where you want to add the new tool.
2. Select Add before or Add after. The Tool Editor adds a new icon to the toolbox and clears the Button fields.
3. Enter the appropriate information in the Button fields as described in "Changing an Existing Tool" on page 70.
4. Change the icon, if necessary, as described in "Changing an Existing Tool" on page 70.
5. Save your changes as described in "Saving Changes to the Toolbox or Toolset" on page 71.

Changing the Order of the Tools in the Toolbox

To change the position of a tool in the toolbox, select the tool icon, and then click on the left or right arrows to move the tool.

Deleting Tools from the Toolbox

To delete a tool from the toolbox:

1. Select the tool you want to delete.
2. Select Delete.
3. Save your changes as described in "Saving Changes to the Toolbox or Toolset" on page 71.

Returning to the Default Settings

To return all tools in the current Toolbox to their default settings, select Defaults. The Tool Editor asks you to verify your request. Select Yes, No, or Cancel.

Saving Changes to the Toolbox or Toolset

You can save the changes to the catalog entry shown in the **Name** field or create a new toolbox with a different name.

If you are customizing a window- or application-specific toolbox or toolset for your own personal use, you should save the customized toolbox or toolset in your Sasuser.Profile catalog using the same entry name as the PMENU entry for the window or application. SAS searches for toolboxes and toolsets first in the Sasuser.Profile catalog, and then in the application catalog.

If you are a SAS/AF application developer or site administrator and you are editing a window- or application-specific toolbox that you want to be accessible to all users, you must save the TOOLBOX entry with the same library, catalog, and entry name as the PMENU entry for the window or application. To associate a toolset with a specific toolbox, save the TOOLSET entry with the same library, catalog, and entry name as the TOOLBOX entry. You will need write permissions to the appropriate location. For example, to store a customized toolbox for the graphics editor, the site administrator needs to store the toolbox in SASHELP.GI.GEDIT.TOOLBOX.

Save saves the toolbox information to the catalog entry shown in the **Name** field. Save As prompts you to enter a different libref, catalog, and entry name. You can also choose to save the toolbox as a toolset. If you save the toolbox as a toolset, the entry type will be TOOLSET; otherwise, the entry type is always TOOLBOX. (Saving a set of tools as a TOOLSET does not change your TOOLBOX entry. See "Customizing Toolboxes and Toolsets in UNIX Environments" on page 67 and "Creating or Customizing an Application- or Window-Specific Toolset" on page 72 for information about toolsets.)

If you select Close or Open without first saving your changes, the Tool Editor prompts to save the changes to the current toolbox or toolset before continuing.

After you save the toolbox or toolset, the Tool Editor remains open for additional editing, and the **Name** field changes to the name of the new entry (if you entered a new name).

Creating a New Toolbox

To create an entirely new toolbox, you can:

- edit an existing toolbox using the Tool Editor and save it using Save as as described in "Saving Changes to the Toolbox or Toolset" on page 71.
- open the Sasuser.Profile catalog in the Explorer window and add a new toolbox by selecting

  File ▶ New ▶ Toolbox

Creating or Customizing an Application- or Window-Specific Toolbox

If you are an application developer and want to create or edit an existing application toolbox, you must:

1. Delete any existing TOOLBOX entry in your Sasuser.Profile for the window or application that you want to customize. Deleting the copy of the toolbox in your Sasuser.Profile allows you to pick up a copy of the toolbox supplied with SAS when you invoke the Tool Editor.
2. Create or edit the application toolbox as described in "Creating a New Toolbox" on page 72 or "Using the Tool Editor" on page 68.
3. Save the edited toolbox as described in "Saving Changes to the Toolbox or Toolset" on page 71.
4. Inform your users that you have changed the window or application toolbox. If they want to use the new toolbox, they must delete the corresponding TOOLBOX entry from their Sasuser.Profile. The new toolbox will then be automatically loaded when the window or application is invoked. If a user does not delete the corresponding TOOLBOX entry from their Sasuser.Profile, that copy of the toolbox will be loaded instead of the new toolbox.

The TOOLLOAD and TOOLCLOSE commands are most useful when you are developing SAS/AF applications. You can use the EXECMDI routine with these commands to enable your application to open and close the toolbox and to give users of your applications access to several toolboxes during the course of their work. See *SAS Component Language: Reference* for a description of the EXECMDI routine.

Creating or Customizing an Application- or Window-Specific Toolset

You define application- or window-specific toolsets in the same way that you create an application- or window-specific toolbox. There are only two differences:

- To create a new toolset, start by defining a toolbox as described in "Creating a New Toolbox" on page 72.
- After you have defined the toolbox, save it as a TOOLSET entry, not as a TOOLBOX entry.

Note: If you are an application developer, make sure that you delete any existing TOOLSET entry for your application as described in "Creating or Customizing an Application- or Window-Specific Toolbox" on page 72 before you modify your application's toolset. △

Customizing Key Definitions in UNIX Environments

Techniques for Customizing Your Key Definitions

There are four ways to customize your key definitions:

- through the KEYS window. To open the KEYS window, issue the KEYS command or select

  Tools ► Options ► Keys

  If you change any key definitions through the KEYS window for the primary SAS windowing environment windows, the definitions are stored in the Sasuser.Profile catalog in the entry DMKEYS.KEYS. Key definitions for other SAS windows are stored in catalog entries named BUILD.KEYS, FSEDIT.KEYS, and so on.

  Refer to SAS Help and Documentation for more information on the KEYS command and the KEYS window.

- with the KEYDEF command. The KEYDEF command enables you to redefine individual function keys:

  ```
  keydef keyname <command|~text-string>
  ```

  For example, if you specify **`keydef F8 dlgpref`**, then the F8 key will open the Preferences dialog box.

  For more information about the KEYDEF command, refer to the Base SAS section in SAS Help and Documentation.

- through the Resource Helper (reshelper). Resource Helper generates SAS resource specifications based on keys and functions that you select. You can use Resource Helper to change the function of any key listed in the KEYS window. See "Setting X Resources with the Resource Helper" on page 62 and "Defining Keys with the Resource Helper" on page 63 for more information about the Resource Helper.

  In most cases, Resource Helper is much easier and faster than defining the resources yourself. However, because the X Window System searches for resources in several places, it is possible for Resource Helper to pick up the wrong key symbol for the key you are trying to define. Also, unless the action routine that you assign to your keys is the **`sas-function-key`** routine, then Resource Helper does not provide a way to change the key labels in the KEYS window. In both of these cases, you will need to define your key resources yourself.

- by defining the **`SAS.keyboardTranslations`** and **`SAS.keysWindowLabels`** resources in your resources file as described in "Defining Key Translations" on page 74.

You can define most of the keys on your keyboard. However, a few keys have dedicated functions that are associated with them. For example, the mouse buttons are dedicated to the cursor and cut-and-paste operations and are not available for user customization.

Defining Key Translations

What Is a Key Translation?

Key customization for the X Window System consists of defining a key sequence and an action to be executed when that key sequence is typed on the keyboard. This is known as *binding* keys to *actions*; together they are referred to as a *translation*.

What Is the SAS.keyboardTranslations Resource?

The **SAS.keyboardTranslations** resource specifies the set of key bindings that SAS uses in all SAS windows. The default value for the **SAS.keyboardTranslations** resource is determined at run time based on the vendor identification string reported by the X server that you are using as the display. These defaults are listed in the files contained in **!SASROOT/X11/resource_files**. To modify the default bindings supplied by SAS, you must modify the keyboardTranslations resource.

Note: The X Toolkit Intrinsics translations specified in this resource apply to both the user area and the command line of all SAS windows that are affected by this resource. This resource does not affect windows that are controlled by Motif interface resources, such as the Command window, the Open or Import dialog boxes, and some other pull-down menu dialog boxes. △

Steps for Creating a Key Definition

To create a key definition, follow these steps:

1. Determine the keysyms for the keys that you want to define. *Keysyms* are the symbols recognized by the X Window System for each key on a keyboard. See "Determining Keysyms" on page 75 for more information.
2. Modify/add the **SAS.keyboardTranslations** resource in your resource file to include the definitions of the keys that you want to define. Use a keyboard action routine to define which action you want the key to perform. The definition in the right column in the KEYS window will no longer control the function of any keys that are defined with a keyboard action routine other than **sas-function-key**. The definitions of those keys in the KEYS window become labels that have no effect. See "Syntax of the SAS.keyboardTranslations Resource" on page 75 for more information.
3. Modify/add the **SAS.keysWindowLabels** resource in your resource file. The **SAS.keysWindowLabels** resource specifies the set of valid labels that will appear in the SAS KEYS window. Modify this resource only if you want to add new labels or modify existing labels in the left column in the KEYS window.

   The **SAS.keysWindowLabels** resource defines only the mnemonics used in the KEYS window. For a specific key to perform an action, you must specify a **SAS.keyboardTranslations** definition for the key. See "Syntax of the SAS.keysWindowLabels Resource" on page 76 for more information.
4. Start a SAS session and open the KEYS window.
5. In the right-hand column in the KEYS window, type a command name or other description of each key that you have defined.

See "Examples: Defining Keys Using SAS Resources" on page 80 for examples of key definitions.

Determining Keysyms

You can use the **xev** utility to determine the keysyms associated with the keys on your keyboard. **xev** is distributed with most UNIX operating systems, but if **xev** is not installed on your operating system, contact your system administrator for information about other methods that are available in your UNIX environment.

xev prints a message for each X event that occurs. The **KeyPress** event specifies the keysym for each key that is pressed.

1. Start **xev** on the X server for which you want to define keys. The **xev** client displays a small Event Tester window that lists the X events that occur. (The **xev** client generates a large amount of output, so you might want to save the output to a file for later review. You can issue the UNIX **script** command to save the output to a file.)
2. Give keyboard focus to the Event Tester window by clicking the mouse pointer on the window, if necessary.
3. Press the key that you want to define, and watch for the **KeyPress** event to be listed. The listing has a number of items that are separated by commas. One of the fields in the **KeyPress** event lists the keysym name that is associated with the key that was pressed.

   For example, when the 0 key on the keypad of an HP 9000/700 keyboard is pressed, it generates the following output:

```
KeyPress event, serial 14, synthetic NO,
  window 0x4400001,root 0x23, subw 0x4400002,
  time 507920400, (54,37),root:(67,66),
  state 0x0, keycode 30 (keysym 0xffb0, KP_0),
  same_screen YES,
  XLookupString gives 1 characters:  "0"
```

   In this example, the keysym name is **KP_0**.

Note: SAS defines a set of *virtual keysyms* with the **SAS.defaultVirtualBindings** resource. Virtual keysyms all begin with osf, such as osfPageDown, osfClear, and osfPrimaryPaste. If you remap these virtual bindings instead of using the defaults supplied by SAS, you might get unexpected results. If you specify a key translation that does not work, you might be trying to redefine a key that is bound to a virtual keysym. In this case, you must specify the virtual keysym in the **SAS.keyboardTranslations** resource instead of the keysym displayed by **xev**. To determine the virtual keysym that is bound to a key, you can start the Resource Helper, select Keys, and press the key or key combination that you want to define. Resource Helper will display the virtual keysym name. You can also refer to the key definition files in **/X11/resource_files** in the directory where SAS is installed (**!SASROOT**) and to the man pages for VirtualBinding or xmbind. △

Syntax of the SAS.keyboardTranslations Resource

Note: Most SAS documentation uses angle brackets (<>) to indicate optional syntax. However, in this topic, optional syntax is shown with square brackets ([]). The angle brackets that are shown in this topic are part of the syntax and should be entered exactly as shown. △

The syntax of the **SAS.keyboardTranslations** resource is

SAS.keyboardTranslations: #override \

[*modifier*] <Key>*keysym* : *action-routine* \n\

[*modifier*] <Key>*keysym* : *action-routine*

#override
indicates that this definition should override any existing bindings for the specific keys that you define without affecting any other keys. If you omit the **#override** directive, the new bindings replace all of the default bindings, and none of the other keys on the keyboard will be available.*

modifier
can be **Alt**, **Ctrl**, **Meta**, **Shift**, **Lock**, **Mod1**, **Mod2**, **Mod3**, **Mod4**, **Mod5**, **None**, or a blank space. The list of valid modifiers varies depending on your keyboard. To display a list of valid modifiers for your keyboard, enter the **xmodmap** UNIX command. Refer to the **man** page for **xmodmap** for more information.

<Key>
is required. It signals the beginning of the keysym.

keysym
is the key symbol recognized by X for the key that you are defining. See "Determining Keysyms" on page 75 for more information.

action-routine
is what you want the key to do. You can specify any action routine described in "SAS Keyboard Action Names" on page 77.

\n
allows the X translation manager to determine where one translation sequence ends and the next one begins. Do not enter \n after the end of the last translation.

\
prevents the newline character at the end of the line from being interpreted as part of the definition. This is a stylistic convention that allows each translation to be listed on a separate line. Do not enter a backslash after the end of the last translation.

Note: SAS does not prevent you from specifying invalid keys in the **SAS.keyboardTranslations** resource. In some cases, invalid keys will produce warnings in the shell window. △

Syntax of the SAS.keysWindowLabels Resource

Note: The square brackets ([]) in the following syntax indicate that the *(InternalKeyName)* is optional. △

The syntax of the **SAS.keysWindowLabels** resource is

SAS.keyWindowLabels: \

KeyWindowLabel [(*InternalKeyName*)] \n\

KeyWindowLabel [(*InternalKeyName*)]

KeyWindowLabel
is the label (1 to 8 characters) that you want to appear in the KEYS window.

* For information on the **#augment** and **#replace** directives, refer to the documentation for the X Window System.

InternalKeyName
is the character string that is passed to the **sas-function-key** action routine in the corresponding **SAS.keyboardTranslations** key binding. (*InternalKeyName* is used by SAS to correlate KEYS window entries to key definitions in the KEYS modules loaded from SAS catalogs or defined in the SAS KEYS window.) If the *InternalKeyName* is not specified, SAS uses the *KeyWindowLabel* as the *InternalKeyName*.

\n and \
serves the same purpose as in the **SAS.keyboardTranslations** resource. See "Syntax of the SAS.keyboardTranslations Resource" on page 75 for more information.

SAS Keyboard Action Names

Note: Most SAS documentation uses angle brackets (<>) to indicate optional syntax. However, in this topic optional syntax is shown with square brackets ([]). The angle brackets that are shown in this topic are part of the syntax and should be entered exactly as shown. △

SAS declares a set of keyboard actions during X initialization. You can think of these keyboard actions as simple functions. When the actions are executed, they act on the window that currently has keyboard input focus.

The following list of keyboard actions represents action routines registered by the Motif interface for use with X toolkit keyboard event translations.

sas-cursor-down()
moves the cursor down one line in the SAS window. The cursor does not wrap when it reaches the bottom of the SAS window interior.

sas-cursor-left()
moves the cursor left one character in the SAS window. The cursor does not wrap when it reaches the left side of the SAS window interior.

sas-cursor-right()
moves the cursor right one character in the SAS window. The cursor does not wrap when it reaches the right side of the SAS window interior.

sas-cursor-up()
moves the cursor up one line in the SAS window. The cursor does not wrap when it reaches the top of the SAS window interior.

sas-delete()
deletes all text in the current field.

sas-delete-begin()
deletes text from the current cursor position to the beginning of the current text field.

sas-delete-char()
deletes the character under the text cursor and leaves the cursor in place.

sas-delete-end()
deletes text from the current cursor position to the end of the current text field.

sas-delete-prev-chr()
deletes the character to the left of the text cursor and moves the cursor back one space.

sas-delete-prev-word()
deletes text to the start of the previous word from the current cursor position. If the cursor is in the interior of a word when the action is invoked, the text from the cursor position to the start of the word is deleted.

sas-delete-word()
deletes text from the current cursor position to the end of the current or next word.

sas-do-command()
accepts one or more text string parameters that are interpreted as SAS commands to be executed when the action is invoked. The action may be invoked with multiple parameters. The parameters are concatenated with semicolon delimiters supplied by the sas-do-command action between the parameters. The assembled SAS command string is then submitted for execution. For example, the following translation syntax can be used to define a HOME, SUBMIT key sequence for all SAS windowing environment windows:

```
<Key>KP_F3: sas-do-command(HOME;SUBMIT)
```

sas-function-key("*InternalKeyName*")
invokes the SAS commands associated with the function key identified by the *InternalKeyName* label. *InternalKeyName* is the character string (1 to 8 characters long) that is passed to the keysWindowLabels resource. Enclose *InternalKeyName* in quotes. Refer to "Defining Key Translations" on page 74 for a description of internal key names.

sas-home-cursor()
is the equivalent of the HOME command. It is provided for convenience so that the HOME action may be defined for all SAS windowing environment windows.

sas-insert-char(["*InsertionString*"])
inserts or overwrites the character typed into the input field under the text cursor. Insert or overstrike behavior is determined by the sas-toggle-insert action, which has a mode that is reflected by the text cursor style displayed; the block cursor indicates overstrike mode, and the underline cursor indicates insert mode. Normally, sas-insert-char translates the XKeyEvent into the appropriate character and inserts it at the SAS text cursor location. If you specify the parameter, the text string represented by this parameter is inserted at the SAS text cursor location. White space in the string is interpreted by the X Toolkit as a parameter delimiter unless you enclose the string in double quotation marks. Refer to your X Window System documentation for information on embedding quotes in the string parameter. To include an escaped quote, use the following syntax:

```
Shift<Key>KP_1:  sas-insert-char("One\\"1\\"")
```

This produces the text string **One"1"** at the SAS text cursor location.

sas-kp-application()
sets the workstation's numeric keypad to allow function key translations to be reinstated. This action only works for those keypad keys that are bound to sas-function-key() actions. Keypad bindings to other actions are not affected by this translation.

sas-kp-numeric()
sets the workstation's keypad to generate numeric characters instead of its previous function key assignment. This action only works for keypad keys that are bound to sas-function-key() actions. Keypad bindings to other actions are not affected by this translation.

sas-move-begin()
moves the cursor to the beginning of the current text field.

sas-move-end()
moves the cursor to the end of the current text field.

sas-new-line()
generates an end-of-line event when invoked. This is a context-sensitive action. If the action is typed on the SAS command line, the text entered will be submitted for execution. If invoked in the SAS application client area, the action depends on the attributes of the text area under the text cursor. In simplest terms, this action is the general line terminator for an input field.

sas-next-field()
advances the SAS application to the next field in the SAS window client area.

sas-next-word()
skips the text cursor forward to the beginning of the next word in the current text field. If sas-next-word does not find the beginning of a word in the current text field, it advances to the next SAS application field. If you are typing in the SAS command line area of the window, the cursor will not wrap into the SAS window client area.

sas-page-down()
scrolls the current window contents forward by one page.

sas-page-end()
moves the text cursor to the end of the current page.

sas-page-top()
moves the text cursor to the top of the current page.

sas-page-up()
scrolls the window contents backward by one page.

sas-prev-field()
returns the SAS application to the previous field in the SAS window client area.

sas-prev-word()
skips the text cursor backward to the beginning of the previous word in the current text field. If sas-prev-word does not find the beginning of a previous word in the current text field, it returns to the end of the previous SAS application field. If you are typing in the SAS command line area of the window, the cursor will not wrap into the SAS window client area.

sas-to-bottom()
Moves the text cursor to the absolute bottom of the window's text range.

sas-to-top()
Moves the text cursor to the absolute top of the window's text range.

sas-toggle-insert()
switches the associated window line-editing behavior between insert and overstrike modes. This only applies to the SAS command line and the SAS window client area. The current mode is indicated by the cursor style in use. The block cursor indicates overstrike mode, and the underline cursor indicates insert mode.

sas-xattr-key(<*KeyType*>[,<*KeyParam*>])
processes SAS extended attribute keys. The *KeyType* parameter must be one of the following values: XACOLOR, XAATTR, XACLEAR. For *KeyType* XACOLOR, the 12 DMS color names are valid parameters; for *KeyType* XAATTR, the valid values are HIGHLIGHT, REVERSE, BLINK, and UNDERLINE; for XACLEAR, no

parameter is required. The BLINK attribute is not supported in the Motif interface. However, if you specify the BLINK attribute, it will be displayed when the catalog is ported to other operating environments.

Examples: Defining Keys Using SAS Resources

Note: Most SAS documentation uses angle brackets (<>) to indicate optional syntax. However, in these examples, optional syntax is shown with square brackets ([]). The angle brackets that are shown in these examples are part of the syntax and should be entered exactly as shown. △

In the following example, the sas-do-command action routine specifies that the COMMAND command is to override any existing definition for **KP_0**.

```
SAS.keyboardTranslations: #override \n\
    None<Key>KP_0: sas-do-command(COMMAND)
```

All other keys retain their current definitions.

The following example binds the key sequence CTRL-K to the KEYS command and specifies that CTRL-D deletes the character under the cursor. Commands entered in the KEYS window for CTRL-K and CTRL-D will have no effect.

```
SAS.keyboardTranslations: #override\
    Ctrl<Key>k: sas-do-command(keys)\n\
    Ctrl<Key>d: sas-delete-char()
```

The following example specifies that the key associated with the keysym **hpClearLine** performs the command entered beside the **MyClrLn** label in the KEYS window.

```
SAS.keyboardTranslations: #override \
   <Key>hpClearLine : sas-function-key("ClearLn")
SAS.keysWindowLabels: MyClrLn(ClearLn)
```

The character string that appears inside the parentheses in the **SAS.keysWindowLabels** resource must match the string entered as the parameter to the sas-function-key routine. The label (**MyClrLn**) can be any character string, and the keysym **hpClearLine** must be a valid keysym for your keyboard.

Customizing Fonts in UNIX Environments

Difference between the System Font and Windowing Environment Fonts

SAS uses two main types of fonts:

- The system font is used in most dialog boxes and pull-down menus. SAS inherits the system font defined by the CDE ***.systemFont** resource. If this resource is not defined, SAS uses a Helvetica font.
- Windowing environment fonts are used in SAS windows. You can change the SAS windowing environment font either through the Fonts dialog box or by specifying the resources in your resources file. The windowing environment font must be a fixed font.

Note: It is best to change fonts before invoking any applications. Changing fonts while applications are running might result in unexpected behavior. △

How SAS Determines Which Windowing Environment Font To Use

SAS determines the normal (not bold) default windowing environment font as follows:

1. If you have saved a font in SASUSER.PROFILE.DMSFONT.UNXPREFS through the Font dialog box, this font is used as the default normal font.
2. If you have not saved a font through the Font dialog box, but you have set the **`SAS.DMSFont`** resource, SAS uses the font specified by this resource as the default font.
3. If you have not set the **`SAS.DMSFont`** resource, SAS uses any *Font resources that you have defined.
4. If you have not set the *Font resources, but you have set the **`SAS.DMSFontPattern`** resource, SAS uses this resource to determine which font to use. The **`SAS.DMSfontPattern`** resource will have no effect if a *Font resource is defined.
5. If no resources have been set, SAS chooses a font from the fonts that are available on your server.

If you have not specified a value for the **`SAS.DMSboldFont`** resource, SAS uses the default normal font to determine the default bold font. If the normal **`SAS.DMSFont`** has an XLFD name associated with it, then SAS selects the matching bold font and loads it. If SAS cannot automatically select or load a bold font, the normal font is also used for the bold font.

In many cases, font names are given aliases so that a shorter name can be used to refer to a font that has an XLFD name associated with it. The name used in determining a bold font is based on the XA_FONT font property for the normal font.

Customizing Fonts Using the Fonts Dialog Box

Introduction to the Fonts Dialog Box

The Fonts dialog box enables you to change the windowing environment font for the entire SAS session. If you change the font, the font that you select is stored in SASUSER.PROFILE.DMSFONT.UNXPREFS and will be used in future SAS sessions.

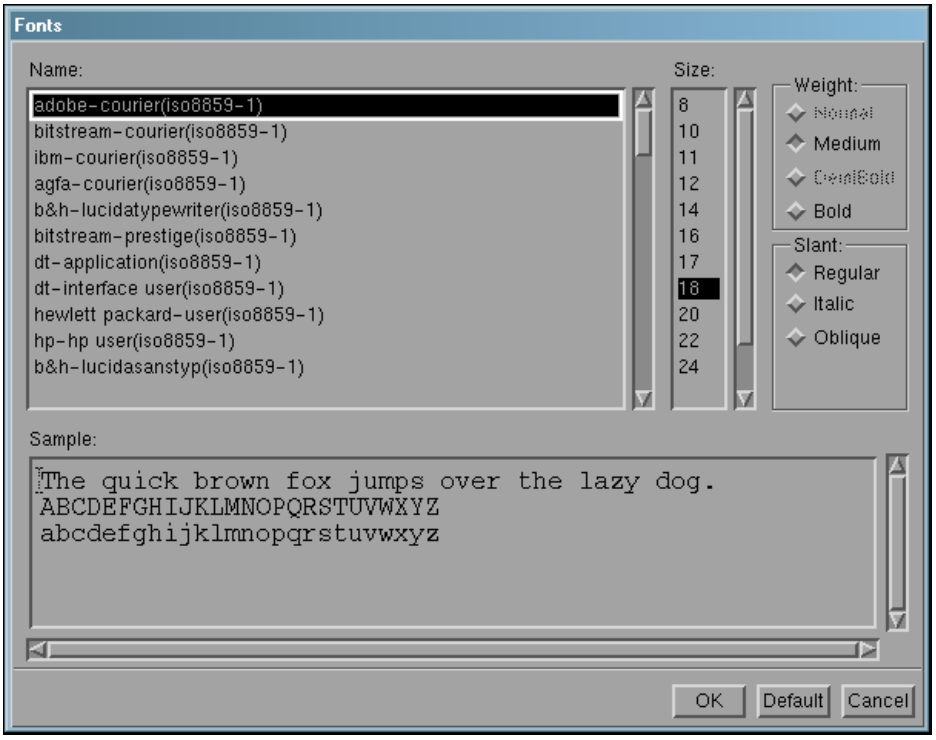
How to Change the Windowing Environment Font

To change the windowing environment font, complete the following steps:

1. To open the Fonts dialog box, use one of the following methods:
   - Issue the DLGFONT command in the command window.
   - Select

     Tools ▶ Options ▶ Fonts

Display 3.7 Fonts Dialog Box



- Select a font name and, if desired, a size, weight, and slant. (Not all fonts are available in all sizes, weights, or slants.) The **Sample** field shows what the selected font looks like.
- Click OK to change the existing font to the selected font.

To return to the default font, click Default.
To cancel any changes and exit the Fonts dialog box, click Cancel.

Specifying Font Resources

You can customize the fonts used in the SAS windowing environment with the following resources:

SAS.DMSFont: ***font-name***
specifies the font that you want to be used as the default normal font. The default normal font is Courier.

SAS.DMSboldFont: ***font-name***
specifies the font that you want to be used as the default bold font.

SAS.DMSDBfont: ***font-name***
specifies the multibyte normal character set font used by the SAS windowing system for operating environments that support multibyte character sets.

SAS.DMSDBboldFont: ***font-name***
specifies the multibyte bold character set font used by the SAS windowing system for operating environments that support multibyte character sets.

SAS.DMSfontPattern: ***XLFD-pattern***
specifies an X Logical Font Description, or XLFD pattern that you want SAS to use to determine the windowing environment font. Most fonts in the X Window

System are associated with an XLFD, which contains a number of different fields delimited by a dash (–) character. The fields in the XLFD indicate properties such as the font family name, weight, size, resolution, and whether the font is proportional or monospaced. Refer to your X Window documentation for more information on the XLFD and font names used with X.

The *XLFD-pattern* that you specify for **SAS.DMSfontPattern** must contain the same number of fields as an XLFD. An asterisk (*) character means that any value is acceptable for that particular field. For example, the following pattern matches any font that has a regular slant, is not bold, is monospaced, and is an iso8859 font:

```
SAS.DMSFontPattern: -*-*-*-r-*--*-*-*-*-m-*-iso8859-1
```

SAS uses the *XLFD-pattern* to choose a font as follows:

1. SAS queries the X server for the list of fonts that match the **SAS.DMSfontPattern** resource.
2. SAS excludes all fonts that have X and Y resolution values different from the current X display, all fonts that have variable character cell sizing (such as proportional fonts), and all fonts that have point sizes smaller than 8 points or larger than 15 points. If this step results in an empty list, SAS chooses a generic (and usually fixed) font.
3. The font with the largest point size is chosen from the remaining list.

SAS.fontPattern: ***XLFD-pattern***
specifies an XLFD font pattern that describes the candidate fonts used to resolve SAS graphics font requests. This allows the user to optimize or control the use of X fonts within the context of various SAS graphics applications. The default value of * usually does not affect performance to a significant degree. You might want to restrict the font search if you are running SAS on a server with an excessive number of fonts or that is operating in performance-limited environment.

SAS.systemFont: ***font-name***
specifies the system font. The SAS windowing environment font is used in SAS windows. The system font is used in most dialog boxes and menus. SAS typically inherits the system font from the font resources set by the X window environment, such as the Common Desktop Environment (CDE), or K Desktop Environment (KDE). If the ***.systemFont** resource, SAS uses a 12-point Helvetica font.

Specifying Font Aliases

If your server does not provide fonts to match all of those supplied by SAS, you can use font alias resources to substitute the fonts available on your system. (Ask your system administrator about the fonts that are available.) Use the following syntax to specify font aliases in your resource file:

```
SAS.supplied-fontAlias: substitute-family
```

where *supplied-font* is the name of the font supplied by SAS. *substitute-family* is the family name of the font that you want to substitute.

CAUTION:
Do not specify a SAS font as a font alias. There might be a conflict if you specify a font supplied by SAS as a font alias, as in the following example:

```
SAS.timesRomanAlias: symbol
```

Assigning this value to a font alias prevents the selection of any symbol fonts through the font selection dialog box, because they are specified as the Times Roman alias. △

The following table lists SAS font alias resource names.

Table 3.1 SAS Font Alias Resources

Resource Name	Class Name
SAS.timesRomanAlias	TimesRomanAlias
SAS.helveticaAlias	HelveticaAlias
SAS.courierAlias	CourierAlias
SAS.symbolAlias	SymbolAlias
SAS.avantGardeAlias	AvantGardeAlias
SAS.bookmanAlias	BookmanAlias
SAS.newCenturySchoolbookAlias	NewCenturySchoolbookAlias
SAS.palatinoAlias	PalatinoAlias
SAS.zapfChanceryAlias	ZapfChanceryAlias
SAS.zapfDingbatsAlias	ZapfDingbatsAlias

Example: Substituting the Lucida Font for Palatino

Suppose that your system does not have a Palatino font, but has the following Lucida font:

```
b&h-lucida-bold-r-normal-sans-
    10-100-75-75-p-66-iso8859-1
```

To substitute Lucida for Palatino, include the following line in your resource file:

```
SAS.palatinoAlias: lucida
```

Customizing Colors in UNIX Environments

Methods for Customizing the Color Settings in Your SAS Session

SAS ships to all sites a default set of colors and attribute settings for the elements of all SAS windows. You can customize the colors in your SAS session

- through Resource Helper (reshelper). Resource Helper enables you to customize any color. See “Setting X Resources with the Resource Helper” on page 62 and “Modifying the Color of a SAS Window Using the Resource Helper” on page 64 for more information.
- through the SASCOLOR window, as described in “Customizing Colors Using the SASCOLOR Window” on page 85. You can customize any window element for most SAS windows with the SASCOLOR window.
- with the COLOR command as described in “Syntax of the COLOR Command” on page 85. The COLOR command affects only the specified element of the active window. Changes made with the COLOR command override changes entered through any of the other methods described here.

- by entering the color resource specifications yourself. You can enter specific RGB values or color names for any of the X resources that control color. See "Defining Color Resources" on page 86 for more information.

Customizing Colors Using the SASCOLOR Window

You can use the SASCOLOR window to change the color and highlighting of specific elements of SAS windows. To open the SASCOLOR window, issue the SASCOLOR command or select

Tools ▶ Options ▶ Colors

Display 3.8 SASCOLOR Window

To change a color for a window element, select the element name, and then select color and attribute that you want assigned to the element.

The BLINK attribute is not supported. The HIGHLIGHT attribute causes text to be displayed in bold font.

When you select Save, your changes are saved to the catalog entry SASUSER.PROFILE.SAS.CPARMS.

Note: Close and reopen any active windows for new color settings to take effect. △

For more information about the SASCOLOR window, see the SAS Help and Documentation.

Syntax of the COLOR Command

You can use the COLOR command to set the color for specific elements of the active window:

```
color field-type <color|NEXT <highlight>>
```

field-type
specifies an area of the window such as background, banner, command, border, message, and so on.

color
specifies a color such as blue (which can be abbreviated B), red (R), green (G), cyan (C), pink (P), yellow (Y), white (W), black (K), magenta (M), gray (A), brown (B), or orange (O).

NEXT
changes the color to the next available color.

highlight
can be H (which causes text to be displayed in a bold font), U (underlined), or R (reverse video). The BLINK attribute is not supported.

To save your changes, issue the WSAVE command. The changes are saved to SASUSER.PROFILE.*window*.WSAVE.

Note: The WSAVE command is not available for all SAS windows. For example, with SAS/FSP or SAS/CALC software, changes are saved either through the EDPARMS or the PARMS window. (To determine whether WSAVE is available for a SAS window, refer to the product documentation.) △

For more information on the COLOR and WSAVE commands, see SAS Help and Documentation.

Defining Color Resources

Types of Color Resources

Color resources fall into two categories:

foreground and background definitions
These resources allow you to customize the RGB values that are used to define the 12 DMS colors. Since each color could be used as either a background or a foreground color, you can specify different RGB values or color names for each color for each usage. For example, you can specify that when blue is used as a foreground color, color #0046ED is used, and when blue is used as a background color, CornflowerBlue is used.

window element definitions
These resources, which are referred to as CPARMS resources, enable you to specify which of the 12 DMS colors you want to use for each window element. For example, you can specify that message text is displayed in magenta.

These two types of resources work together. The CPARMS color values use the current foreground and background definitions. For example, the following resources specify that the background of your primary windows will be CornflowerBlue:

```
SAS.blueBackgroundColor: CornflowerBlue
SAS.cparmBackground: DmBlue
```

Specifying RGB Values or Color Names for Foreground and Background Resources

SAS uses **SAS.systemBackground**, **SAS.systemForeground**, and the resources listed in the following table to determine the colors to be used in its windows.

SAS.systemForeground: ***color***
specifies the color for the foreground system color in the SASCOLOR window.

SAS.systemBackground: ***color***
specifies the color for the background system color in the SASCOLOR window.

SAS.systemSecondaryBackground: ***color***
sets the system secondary background color and specifies the color for the secondary background system color in the SASCOLOR window.

You can specify color names such as MediumVioletRed or RGB values such as #0000FF for all of the foreground and background resources. Refer to your X Window System documentation for information on RGB color values.

The following table lists all of the foreground and background color resources and their class names. All of these resources are of the type String.

Table 3.2 Foreground and Background Color Resources

Resource Name	Class Name
SAS.systemForeground	SystemForeground
SAS.systemBackground	SystemBackground
SAS.systemSecondaryBackground	Background
SAS.blackForegroundColor	BlackForegroundColor
SAS.blueForegroundColor	BlueForegroundColor
SAS.brownForegroundColor	BrownForegroundColor
SAS.cyanForegroundColor	CyanForegroundColor
SAS.grayForegroundColor	GrayForegroundColor
SAS.greenForegroundColor	GreenForegroundColor
SAS.magentaForegroundColor	MagentaForegroundColor
SAS.orangeForegroundColor	OrangeForegroundColor
SAS.pinkForegroundColor	PinkForegroundColor
SAS.redForegroundColor	RedForegroundColor
SAS.whiteForegroundColor	WhiteForegroundColor
SAS.yellowForegroundColor	YellowForegroundColor
SAS.blackBackgroundColor	BlackBackgroundColor
SAS.blueBackgroundColor	BlueBackgroundColor
SAS.brownBackgroundColor	BrownBackgroundColor
SAS.cyanBackgroundColor	CyanBackgroundColor
SAS.grayBackgroundColor	GrayBackgroundColor
SAS.greenBackgroundColor	GreenBackgroundColor
SAS.magentaBackgroundColor	MagentaBackgroundColor
SAS.orangeBackgroundColor	OrangeBackgroundColor
SAS.pinkBackgroundColor	PinkBackgroundColor
SAS.redBackgroundColor	RedBackgroundColor

Resource Name	Class Name
SAS.whiteBackgroundColor	WhiteBackgroundColor
SAS.yellowBackgroundColor	YellowBackgroundColor

Defining Colors and Attributes for Window Elements (CPARMS)

You can define the colors and attributes for specific window elements by assigning values to SAS resources known as CPARMS. Each CPARMS resource defines the color and attribute of a specific window element, such as the background in a secondary window or the border of a primary window.

You can specify multiple color and attribute names in the same resource definition, but only the final color and attribute will be used:

```
SAS.cparmResource: DmColorName|DmAttrName\
<+DmColorName|DmAttrName>
```

Resource can be any of the CPARMS resources listed in the following table. All of these resources are of type DmColor, and their default values are dynamic—that is, the default values are determined at run time.

Table 3.3 SAS CPARMS Resources

Resource Name	Specifies the color and attribute settings for . . .	Class Name	Default Color
SAS.cparmBackground	backgrounds within all primary windows displayed in a SAS session	CparmBackground	DmWhite
SAS.cparmBanner	a banner within a window	CparmForeground	DmBlack
SAS.cparmBorder	the border of a primary window	CparmBackground	DmBlack
SAS.cparmByline	BY lines written to the Output window	CparmForeground	DmBlue
SAS.cparmColumn	text labels for column information. You can use this resource within the SAS editor to identify editing lines and in spreadsheet windows to label spreadsheets.	CparmForeground	DmBlue/ Underline
SAS.cparmCommand	the command data entry field when menus are disabled.	CparmForeground	DmBlack
SAS.cparmData	general lines written to the Log window or the Output window	CparmForeground	DmBlack
SAS.cparmError	ERROR: lines that are written to the Log window or Output window	CparmForeground	DmRed

Resource Name	Specifies the color and attribute settings for . . .	Class Name	Default Color
SAS.cparmFootnote	FOOTNOTE lines written to the Output window	CparmForeground	DmBlue
SAS.cparmForeground	all text fields within a SAS windowing environment window that can be edited	CparmBackground	DmBlack
SAS.cparmHeader	HEADER lines written to the Output window	CparmForeground	DmBlue
SAS.cparmHelpLink	links to additional levels of information in the HELP system	CparmForeground	DmGreen/ Underline
SAS.cparmHelpMainTopic	topic words or phrases in the HELP system	CparmForeground	DmBlack
SAS.cparmHelpSubTopic	topic words or phrases in the HELP system	CparmForeground	DmBlack
SAS.cparmInfo	text that is displayed in a window as an aid to the user, for example: **Press Enter to continue**	CparmForeground	DmBlack
SAS.cparmLabel	text that precedes a widget. For example, the text **Name:** in the following example is a label: **Name:** _____	CparmForeground	DmBlack
SAS.cparmMark	areas that have been selected for operations such as FIND, CUT, and COPY	CparmForeground	DmBlack/ DmReverse
SAS.cparmMessage	the message field	CparmForeground	DmRed
SAS.cparmNote	NOTE: lines that are written to the Log window or the Output window	CparmForeground	DmBlue
SAS.cparmSecondaryBackground	backgrounds in secondary windows	CparmForeground	DmGray
SAS.cparmSecondaryBorder	the border of a secondary window	CparmForeground	DmBlack
SAS.cparmSource	SAS source lines that are written to the Log window	CparmForeground	DmBlack
SAS.cparmText	text labels for row information. You can use this resource within the SAS editor to identify editing lines and in spreadsheet windows to label spreadsheet rows.	CparmForeground	DmBlue

Resource Name	Specifies the color and attribute settings for . . .	Class Name	Default Color
SAS.cparmTitle	TITLE lines written to the Output window	CparmForeground	DmBlue
SAS.cparmWarning	WARNING lines written to the Log window or the Output window	CparmForeground	DmGreen

DmColorName can be any one of the following colors:

- DmBLUE
- DmRED
- DmPINK
- DmGREEN
- DmCYAN
- DmYELLOW
- DmWHITE
- DmORANGE
- DmBLACK
- DmMAGENTA
- DmGRAY
- DmBROWN

DmAttrName can be any one of the following attributes:

- DmHIGHLIGHT
- DmUNDERLINE
- DmREVERSE

For example, the following resources specify that all background colors are gray and all foreground colors are black:

```
SAS.cparmBackground: DmGRAY
SAS.cparmForeground: DmBLACK
```

These resources specify that errors should be displayed in red with reverse video, and warnings should be displayed in yellow with reverse video and a bold font:

```
SAS.cparmError: DmRED + DmREVERSE
SAS.cparmWarning: DmHIGHLIGHT + DmYELLOW + DmREVERSE
```

SAS looks for default CPARMS resources in two places:

- If your SAS Site Representative has entered color and attribute settings in the SASHELP.BASE.SAS.CPARMS catalog entry, then these settings become the default for your site.
- If you have saved settings in SASUSER.PROFILE.SAS.CPARMS, then these settings override the settings specified for your site.

Controlling Contrast

During interactive move/stretch operations, such as rubberbanding and dragging rectangles in SAS/INSIGHT software, you might find it hard to see the outline of the

graphics primitive because of the lack of contrast between the primitive and the background. The XCONTRAST command makes the primitive visible against the background. The rendering performance and the aesthetic appearance of the primitive is compromised for the sake of visibility. You can enter XCONTRAST to act as a toggle, or you can specify XCONTRAST ON or XCONTRAST OFF.

In some color combinations, text fields, push buttons, check boxes, and other foreground categories might not be visible. The **SAS.dmsContrastCheck** resource makes these categories legible.

SAS.dmsContrastCheck: True | False
controls whether contrast mapping is applied to nongraphic foreground colors in a SAS window. The default value is False. A value of True specifies that DMS foreground colors will be remapped if necessary to produce a contrast. Some color usage based on graphic operations are not affected by this resource.

Controlling Pull-Down Menus in UNIX Environments

Pull-down menus are controlled by the following resources:

SAS.pmenuOn: True | False
forces the global PMENU state on regardless of the information stored by the WSAVE command. The WSAVE state of an individual window takes precedence over the global state. The default is True. (You can also use the PMENU ON and PMENU OFF commands to turn pull-down menus on and off.)

SAS.usePmenuMnemonics: True | False
specifies whether mnemonics are attached to the pmenus for the current SAS session. The default is True.

Customizing Cut-and-Paste in UNIX Environments

Note: For instructions on cutting and pasting text, see "Selecting (Marking) Text in UNIX Environments" on page 42 and "Copying or Cutting and Pasting Selected Text in UNIX Environments" on page 44. △

Types of Paste Buffers

There are four SAS paste buffers. Each SAS paste buffer is associated with a X paste buffer:

XPRIMARY
is associated with X primary selection (PRIMARY).

XSCNDARY
is associated with the X secondary selection (SECONDARY).

XCLIPBRD
is associated with the X clipboard selection (CLIPBOARD). This paste buffer allows you to use the MIT X Consortium xclipboard client with SAS.

XTERM
is associated with the paste buffer used by the xterm client. XTERM is the default buffer. DEFAULT is an alias for XTERM. If you copy or cut text into the XTERM

buffer, the text is actually copied or cut into all four of the paste buffers. When you paste text from the XTERM buffer, the text is pasted from the XPRIMARY buffer.

XCUT*n*
is associated with X cut buffer*n* where 0 <= *n* <= 7.

Selecting a Paste Buffer

If you are not sure which X data exchange protocols your other X clients are using, you should use the XTERM paste buffer. You can specify your default paste buffer with the **SAS.defaultPasteBuffer** resource:

```
SAS.defaultPasteBuffer: XTERM
```

If you know that the X clients in your workstation environment all use the X PRIMARY selections to exchange data, you should use the XPRIMARY paste buffer:

```
SAS.defaultPasteBuffer: XPRIMARY
```

This specification uses both SAS and X resources more efficiently and provides for the on-demand transfer of data between clients.

Sun OpenWindows desktop clients use the CLIPBOARD selection as the basis for their copy-and-paste operations. If you use the SAS XCLIPBRD paste buffer, you can exchange text directly with these clients.

You can also use the SAS XCLIPBRD paste buffer to interact with Motif clients that use the Motif clipboard mechanism for text exchanges. This clipboard mechanism makes it unnecessary to have a dedicated client such as xclipboard. For example, you can use XCLIPBRD to exchange text directly with the Motif xmeditor application when you select the **Cut**, **Copy**, or **Paste** items from the xmeditor **Edit** pull-down menu.

The Motif quick-copy data exchange and Motif clipboard data exchange mechanisms are specific to the Motif interface toolkit and are not currently supported as SAS paste buffers. However some dialog boxes, such as the File Selection dialog box, use Motif interface text widgets. In these dialog boxes, the Motif quick copy and clipboard data exchange mechanisms are available.

Manipulating Text Using a Paste Buffer

If you want SAS to automatically copy selected text into your paste buffer every time you mark a region of text with the mouse, you should also specify your paste buffer name in the **SAS.markPasteBuffer** resource:

```
SAS.markPasteBuffer: XTERM
```

Alternatively, because DEFAULT is an alias for XTERM, you could specify

```
SAS.markPasteBuffer: DEFAULT
```

The **SAS.markPasteBuffer** definition causes SAS to automatically issue a STORE command whenever you select text.

The STORE command, as well as the CUT and PASTE commands, support a BUFFER= option that specifies which buffer to use. When these commands are issued from function keys or pull-down menus whose definitions do not include the BUFFER= option, if the **SAS.markPasteBuffer** resource is not defined, these commands use BUFFER=DEFAULT. If this resource is defined, these commands use BUFFER=*buffer-name*.

You can customize your normal cut, copy, or paste keys to issue any of these commands with the BUFFER= option. For example, you can override the

SAS.keyboardTranslations definition for the **osfCopy** and **osfPaste** keys with the following specifications:

```
SAS.keyboardTranslations: #override \
 <Key>osfCopy: sas-do-command(\"STORE BUFFER=XCLIPBRD\") \n\
 <Key>osfPaste: sas-do-command(\"PASTE BUFFER=XCLIPBRD\")
```

For more information on customizing keys, see "Customizing Key Definitions in UNIX Environments" on page 73.

Notes on Preserving Text and Attribute Information

When you cut or copy and paste text between SAS sessions using the XTERM, XPRIMARY, or XSCNDARY paste buffers, the color and attribute information is preserved. However, if you copy and paste the same text into an xterm window while using the vi editor, the color and attribute information is lost. If you change the definition for **SAS.defaultPasteBuffer** and **SAS.markPasteBuffer** to XCUT0, then you will not retain the text and color attributes when you copy and paste text between two SAS sessions.

When you use the xclipboard client, SAS text attributes are not preserved in exchanges made between SAS sessions. However, when you use the XCLIPBRD paste buffer without a clipboard manager such as the xclipboard client, SAS text attributes are preserved in exchanges between SAS sessions.

Customizing Session Workspace, Session Gravity, and Window Sizes in UNIX Environments

SAS uses the following resources to determine the size of the session workspace, the gravity of the workspace, and the size of the windows. The default values for these resources are listed in Table 3.4 on page 98.

SAS.awsResizePolicy: grow | fixed
controls the policy for resizing AWS windows as interior windows are added and removed. Possible values include the following:

grow the AWS window will attempt to grow any time an interior window is grown or moved, in order to show all interior windows, but it will not shrink to remove dead areas.

fixed the AWS window will attempt to size itself to the size of the first interior window and will not attempt any further size changes.

SAS.maxWindowHeight: ***units***
specifies the number of units for the maximum height of a window. The unit is specified by the **SAS.windowUnitType** resource.

SAS.maxWindowWidth: ***units***
specifies the number of units for the maximum width of a window. The unit is specified by the **SAS.windowUnitType** resource.

SAS.noAWS: True | False
controls whether each of your application's windows appears in its own native window rather than in an Application Work Space (AWS). The default is True; each application runs in its own native window.

SAS.scrollBarSize: ***pixels***
specifies the default size of the scroll bar in pixels.

SAS.sessionGravity: ***value***
controls the region of the screen where SAS will attempt to place its windows. This resource might be ignored by some window manager configurations. Possible values include the following:

CenterGravity

EastGravity

WestGravity

SouthGravity

NorthGravity

SouthEastGravity

NorthEastGravity

SouthWestGravity

NorthWestGravity

SAS.sessionGravityXOffset: ***offset***
specifies an x offset to be added when SAS attempts to place a window in the gravity region.

SAS.sessionGravityYOffset: ***offset***
specifies a y offset to be added when SAS attempts to place a window in the gravity region.

SAS.windowHeight: ***units***
specifies the number of units for the default height of a window. The unit is specified by the **SAS.windowUnitType** resource.

SAS.windowUnitType: character | pixel | percentage
specifies the unit type for **SAS.windowWidth**, **SAS.windowHeight**, **SAS.maxWindowWidth**, and **SAS.maxWindowHeight**. Possible values include the following:

character
units specify the number of rows and columns.

pixel
units specify the number of pixels.

percentage
units specify the percentage of the screen.

SAS.windowWidth: ***units***
specifies the number of units for the default width of a window. The unit is specified by the **SAS.windowUnitType** resource.

Specifying User-Defined Icons in UNIX Environments

Why Specify User-Defined Icons?

You can add your own icons to those icons that are supplied with SAS. For example, if you want to use your own color icons in the toolbox, define the **SAS.colorUiconPath**, **SAS.colorUiconCount**, and **SAS.sasUiconx** resources. Then, when you are defining

tools in the tool editor, the tool editor will include your icons in the display of icons that you can choose for each tool.

How SAS Locates a User-Defined Icon

The resource name that is used to locate the icon bitmap filename for user icon number *x* is **SAS.sasUiconx**. For example, the following resource defines the filename **myicon** for the user icon **1**:

```
SAS.sasUicon1: myicon
```

If the resource name is not defined, SAS generates a filename of the form **sasuinnn.xbm** or **sasuinnn.xpm**. The path elements from the **SAS.uiconPath** or **SAS.colorUiconpath** resource are searched in sequence until the icon file is found or until the search path is exhausted.

For example, the following set of X resources defines a collection of color icons.

```
SAS.colorUiconPath: /users/jackaroe/pixmaps/
SAS.colorUiconCount: 7
SAS.sasUicon1: adsetup
SAS.sasUicon2: adverse
SAS.sasUicon3: altmenu
SAS.sasUicon4: batch
SAS.sasUicon5: is
SAS.sasUicon6: patgrps
SAS.sasUicon7: pctchg
```

The Motif interface will search for icon **sasUicon1** in a file named **/users/jackaroe/pixmaps/adsetup.xpm**.

X Resources for Specifying User-Defined Icons

SAS uses the following resources to determine the number of user-defined icons that are available and their location.

SAS.colorUiconPath: ***search-path***
specifies the file search path for locating user-defined color icon files. This string resource specifies the directory paths to be searched for an icon file. These files should be in X Pixmap (xpm) format. Use a comma to separate individual directory pathnames. For example, the following string first searches for icon files in the **/usr/lib/X11/pixmaps** directory and then in the **/usr/lib/X11/pixmaps/SAS** directory:

```
SAS.colorUiconPath : /usr/lib/X11/pixmaps, \
/usr/lib/X11/pixmaps/SAS
```

SAS.colorUiconCount: ***num-icons***
specifies the number of user-defined color icons that are available for SAS to use.

SAS.uiconCount: ***num-icons***
specifies the number of user-defined icons that are available for use in the SAS session.

SAS.uiconPath: ***search-path***
specifies the file search path for locating user-defined icon bitmap files. This string resource specifies the directory paths to be searched for an icon file. These files

should be in X Bitmap (xbm) format. Use a comma to separate individual directory pathnames. For example, the following string will first search for bitmap files in the **/usr/lib/X11/bitmaps** directory and then in the **/usr/lib/X11/bitmaps/SAS** directory:

```
SAS.uiconPath : /usr/lib/X11/bitmaps,\
/usr/lib/X11/bitmaps/SAS
```

SAS.sasUicon*x*: *name*
associates a value with the filename of an X bitmap or pixmap file. *x* is a number assigned to the file. A file extension of **.xbm** or **.xpm** is automatically supplied.

Miscellaneous Resources in UNIX Environments

You can also customize the following resources:

SAS.altVisualId: *ID*
specifies a visual type ID.

SAS.autoSaveInterval: *minutes*
specifies how often (in number of minutes) that the data from the Program Editor window should be saved.

SAS.autoSaveOn: True | False
specifies that data from the Program Editor window should be saved to a file at intervals specified by the **SAS.autoSaveInterval** resource.

SAS.confirmSASExit: True | False
controls whether SAS displays the Exit dialog box when you enter the DLGENDR command or select

File ▶ Exit

The default is True.

SAS.defaultCommandWindow: True | False
specifies whether the command window is invoked when you start your SAS session. The default is True.

SAS.directory: *directory-pathname*
specifies the directory that you want when you first invoke the Open dialog box. By default, the Open dialog box uses the current directory.

SAS.helpBrowser: *pathname*
specifies the pathname of the World Wide Web browser to use for viewing the online help or when the WBROWSE command is issued. The default browser is Netscape.

SAS.htmlUsePassword: True | False
specifies whether SAS prompts you to enter your password before sending HTML files to your browser. The default value is True.

SAS.insertModeOn: True | False
controls the editing mode in SAS editor windows. The default is False (overtype).

SAS.noDoCommandRecall: True | False
controls whether or not SAS commands submitted through the sas-do-command() action routine are recorded in the command recall buffer. The default value of True causes commands to be omitted from the command recall buffer; a value of False causes them to be recorded.

SAS.pattern: ***default-pattern***
specifies the default pattern that you want to be used as the file filter when you first invoke the Open and Import Image dialog boxes. This pattern is displayed in the text field at the top of the dialog box. By default, the dialog box uses the first filter in the File type list. The pattern resource has no effect on the File type field.

SAS.selectTimeout: ***seconds***
specifies the X toolkit selection conversion timeout value in units of seconds. This determines the amount of time that SAS will wait for a request to convert an X toolkit selection to complete. The default value should be adequate in most cases.

SAS.startupLogo: ***xpm-filename*** **| None | ""**
specifies the XPM file that you want SAS to display when it is initialized. If the string is empty, SAS uses the default logo.

SAS.startSessionManager: True | False
specifies whether SAS automatically starts the session manager when a new SAS session is started. Using your own host editor with SAS requires that the session manager be running. The default is True.

SAS.suppressMenuIcons: True | False
specifies whether SAS displays any menu icons other than the check box and toggle button icons in cascade or popup menus. Suppressing the icons reduces memory usage and improves how quickly the menus display on slower X servers. The default is False.

SAS.suppressTutorialDialog: True | False
specifies whether SAS displays the Getting Started Tutorial dialog box at the start of your SAS session. True suppresses the dialog box. You might want to suppress this dialog box if you have previously used SAS. The default is False.

SAS.useNativeXmTextTranslations: True | False
specifies whether any XmText widget translations are inherited by all instances of the Text, Combo Box, and Spin Box widgets used by the SAS X Motif user interface. When the value is False, the SAS keys windows translations supercede any user or system-supplied XmText translations. The default value is False. See the XmText man page for more information about XmText resources.

SAS.wsaveAllExit: True | False
specifies whether SAS should issue the WSAVE ALL command when you end your session. This command saves the global settings, such as window color and window position, that are in effect for all windows that are currently open. The default is False.

Note: For the WSAVE command to work, your window manager must support explicit window placement. Consult the documentation for your window manager to determine how to configure your window manager. For example, if you are running Exceed, open the Screen Definition Settings dialog box and deselect **Cascade Windows**. △

Summary of X Resources for SAS in UNIX Environments

The following table lists the instance and class names, type, and default values for many of the SAS resources. See the following tables for additional resources of specific types:

- "SAS Font Alias Resources," Table 3.1 on page 84

- "Foreground and Background Color Resources," Table 3.2 on page 87
- "SAS CPARM Resources," Table 3.3 on page 88.

Table 3.4 SAS Resources

Resource Name	Class Name	Type	Default
SAS.altVisualId	AltVisualId	Integer	NULL
SAS.autoComplete	AutoComplete	Boolean	True
SAS.autoSaveInterval	AutoSaveInterval	Integer	10
SAS.autoSaveOn	AutoSaveOn	Boolean	True
SAS.awsResizePolicy	AWSResizePolicy	String	grow
SAS.colorUiconCount	UiconCount	Integer	0
SAS.colorUiconPath	UiconPath	String	NULL
SAS.commandsSaved	CommandsSaved	Integer	25
SAS.confirmSASExit	ConfirmSASExit	Boolean	True
SAS.defaultCommandWindow	DefaultCommandWindow	Boolean	True
SAS.defaultPasteBuffer	DefaultPasteBuffer	String	XTERM
SAS.defaultToolBox	DefaultToolBox	Boolean	True
SAS.directory	Directory	String	NULL
SAS.dmsContrastCheck	DmsContrastCheck	Boolean	False
SAS.DMSDBFont	Font	String	*dynamic*
SAS.DMSDBboldFont	Font	String	*dynamic*
SAS.DMSboldFont	Font	String	*dynamic*
SAS.DMSFont	Font	String	*dynamic*
SAS.DMSfontPattern	DMSFontPattern	String	-*-*-*-r-*–*-*-*-*-m-*-iso8859-1
SAS.fontPattern	FontPattern	String	*
SAS.helpBrowser	HelpBrowser	String	netscape
SAS.htmlUsePassword	HtmlUsePassword	Boolean	True
SAS.insertModeOn	InsertModeOn	Boolean	False
SAS.isToolBoxPersistent	IsToolBoxPersistent	Boolean	True
SAS.keyboardTranslations	KeyboardTranslations	Translation	*dynamic*
SAS.keysWindowLabels	KeysWindowLabels	String	*dynamic*
SAS.markPasteBuffer	MarkPasteBuffer	String	XTERM
SAS.maxWindowHeight	WindowHeight	Dimension	95
SAS.maxWindowWidth	WindowWidth	Dimension	95
SAS.noAWS	NoAWS	Boolean	True
SAS.noDoCommandRecall	NoDoCommandRecall	Boolean	True

Resource Name	Class Name	Type	Default
SAS.pattern	Pattern	String	NULL
SAS.pmenuOn	PmenuOn	Boolean	True
SAS.sasUicon	SasUicon	String	NULL
SAS.scrollBarSize	ScrollBarSize	Dimension	17
SAS.selectTimeout	SelectTimeout	Integer	60
SAS.sessionGravity	SASGravity	String	NorthWestGravity
SAS.sessionGravityXOffset	SASGravityOffset	Integer	0
SAS.sessionGravityYOffset	SASGravityOffset	Integer	0
SAS.startSessionManager	StartSessionManager	Boolean	True
SAS.startupLogo	StartUpLogo	String	NULL
SAS.suppressMenuIcons	SuppressMenuIcons	Boolean	False
SAS.suppressTutorialDialog	SuppressTutorialDialog	Boolean	False
SAS.systemFont	SystemFont	String	"-adobe-helvetica-medium-r-normal–12–*–*–*–*–*–*–*"
SAS.toolBoxAlwaysOnTop	ToolBoxAlwaysOnTop	Boolean	True
SAS.toolBoxTipDelay	ToolBoxTipDelay	Integer	750
SAS.uiconCount	UiconCount	Integer	0
SAS.uiconPath	UiconPath	String	NULL
SAS.useCommandToolBoxCombo	UseCommandToolBoxCombo	Boolean	True
SAS.useLargeToolBox	UseLargeToolBox	Boolean	False
SAS.useNativeXmTextTranslations	UseNativeXmTextTranslations	Boolean	False
SAS.usePmenuMnemonics	UsePmenuMnemonics	Boolean	True
SAS.useShowHideDecorations	UseShowHideDecorations	Boolean	False
SAS.useToolBoxTips	UseToolBoxTips	Boolean	True
SAS.wsaveAllExit	WsaveAllExit	Boolean	False
SAS.windowHeight	WindowHeight	Dimension	50
SAS.windowWidth	WindowWidth	Dimension	67
SAS.windowUnitType	WindowUnitType	String	percentage

CHAPTER

4

Using SAS Files

Introduction to SAS Files, Data Libraries, and Engines in UNIX Environments

What Is a SAS File?

Your data can reside in different types of files, including SAS files and files that are formatted by other software products, such as database management systems. Under UNIX, a SAS file is a specially structured UNIX file. Although the UNIX operating environment manages the file for SAS by storing it, the operating system cannot process it because of the structure built into the file by SAS. For example, you can list the filename with the **`ls`** command, but you cannot use the vi editor to edit the file. A SAS file can be permanent or temporary.

How SAS Filenames Appear in Your Operating Environment

In SAS, you can create filenames using lowercase, uppercase, or mixed case. For example, valid data set names include: test1, TEST1, and Test1. However, when you view these files in your operating environment, the names will appear only in lowercase. For example, all of the previous data set names will appear as test1 in your operating environment.

What Are Data Libraries?

SAS files are stored in *SAS data libraries*. A SAS data library is a collection of SAS files within a UNIX directory. Any UNIX directory can be used as a SAS data library. (The directory can also contain files called *external files* that are not managed by SAS. See Chapter 5, "Using External Files and Devices," on page 131 for how to access external files.) SAS stores temporary SAS files in a Work data library (see "Work Data Library" on page 120), which is automatically defined for you. You must specify a data library for each permanent SAS file.

What Is a Libref?

SAS data libraries can be identified with *librefs*. A libref is a name by which you reference the directory in your application. For more information about how to assign a libref, see "Referring to SAS Data Files Using Librefs in UNIX Environments" on page 111.

What Is an Engine?

SAS files and SAS data libraries are accessed through *engines*. An engine is set of routines that SAS must use to access the files in the data library. SAS can read from and, in some cases, write to the file by using the engine that is appropriate for that file

type. For some file types, you need to tell SAS which engine to use. For others, SAS automatically chooses the appropriate engine. The engine that is used to create a SAS data set determines the format of the file.

Additional Resources

For more information about SAS files, data libraries, and engines, see *SAS Language Reference: Concepts*.

Common Types of SAS Files in UNIX Environments

What Are Data Sets?

Data sets consist of descriptor information and data values organized as a table of rows and columns that can be processed by one of the engines. The descriptor information includes data set type, data set label, the names and labels of the columns in the data set, and so on. A SAS data set can also include *indexes* for one or more columns.

SAS data sets are implemented in two forms:

- If the data values and the data set's descriptor information are stored in one file, the SAS data set is called a *SAS data file*.
- If the file simply contains information about where to obtain a data set's data values and descriptor information, the SAS data set is called a *SAS data view*.

The default engine processes the data set as if the data file or data view and the indexes were a single entity.

For more information, see "SAS Data Files (Member Type DATA)" on page 104 and "SAS Data Views (Member Type VIEW)" on page 105.

SAS Data Files (Member Type DATA)

The SAS data file is probably the most frequently used type of SAS file. These files have have the extension **`.sas7bdat`**. SAS data files are created in the DATA step and by some SAS procedures. There are two types of data files:

- *Native data files* store data values and their descriptor information in files formatted by SAS. These are the traditional SAS data sets familiar from previous releases of SAS.

  Native SAS data files created by the default engine can be indexed. An *index* is an auxiliary file created in addition to the data file it indexes. The index provides fast access to observations within a SAS data file by a variable or key. Under UNIX, indexes are stored as separate files but are treated as integral parts of the SAS data file by SAS.

  CAUTION:
  Do not remove index files using UNIX commands. Removing the index file can damage your SAS data set. Also, do not change its name or move it to a different directory. Use the DATASETS procedure to manage indexes. △

- *Interface data files* store data in files that have been formatted by other software and that SAS can only read. See "Accessing BMDP, OSIRIS, or SPSS Files in UNIX Environments" on page 125 for more information.

SAS Data Views (Member Type VIEW)

A SAS data view contains only the information needed to derive the data values and the descriptor information. Depending on how the SAS data view is created, the actual data can be in other SAS data sets or in other vendors' files.

Views can be of two kinds:

- *Native SAS data views* contain information about data in one or more SAS data files or SAS data views. This type of view is created with the SQL procedure or DATA step.
- *Interface SAS data views* contain information about data formatted by other software products, for example, a database management system. The ACCESS procedure in SAS/ACCESS software, for example, creates such a view.

What Are Catalogs?

Catalogs are a special type of SAS file that can contain multiple entries. Many different types of entries can be kept in the same SAS catalog. For example, catalogs can contain entries created by SAS/AF and SAS/FSP software, windowing applications, key definitions, SAS/GRAPH graphs, and so on.

Catalogs have the SAS member type of CATALOG.

What Are Stored Program Files?

Stored program files are compiled DATA steps generated by the Stored Program Facility. For details on the Stored Program Facility, see *SAS Language Reference: Dictionary*.

Stored program files have the SAS member type of PROGRAM.

What Are Access Descriptor Files?

Access descriptor files describe the data formatted by other software products such as the Oracle or the SYBASE database management systems. Descriptor files created by the ACCESS procedure in SAS/ACCESS software have the SAS member type of ACCESS.

Filename Extensions and Member Types in UNIX Environments

Because SAS needs to distinguish between the different file types, it automatically assigns an extension to each file when it creates the file. Also, since each SAS file is a member of a data library, SAS assigns each file a member type.

The following table lists the file extensions and their corresponding SAS member types.

CAUTION:

Do not change the file extensions of SAS files. File extensions determine how SAS accesses files; changing them can cause unpredictable results. △

Table 4.1 File Extensions for SAS File Types

Version 6		Version 8, SAS System 9			
Random Access Files	**Sequential Access Files**	**Random Access Files**	**Sequential Access Files**	**SAS Member Type**	**Description**
.sas	.sas	.sas	.sas	.sas	SAS program
.lst	.lst	.lst	.lst	.lst	Procedure output
.log	.log	.log	.log	.log	SAS log file
.ssd*nn*[1]	.sdq*nn*	.sas7bdat	.sas7sdat	DATA	SAS data file
.snx*nn*	.siq*nn*	.sas7bndx	.sas7sndx	INDEX	Data file index; not treated by the SAS System as a separate file
.sct*nn*	.scq*nn*	.sas7bcat	.sas7scat	CATALOG	SAS catalog
.ssp*nn*	.ssq*nn*	.sas7bpgm	.sas7spgm	PROGRAM	Stored program (DATA step)
.ssv*nn*	.svq*nn*	.sas7bvew	.sas7svew	VIEW	SAS data view
.ssa*nn*	.saq*nn*	.sas7bacs	.sas7sacs	ACCESS	Access descriptor file
.sst*nn*	.stq*nn*	.sas7baud	.sas7saud	AUDIT	Audit file
.sfd*nn*	.sfq*nn*	.sas7bfdb	.sas7sfdb	FDB	Consolidation database
.ssm*nn*	.smq*nn*	.sas7bmdb	.sas7smdb	MDDB	Multi-dimensional database
.sds*nn*	.soq*nn*	.sas7bods	.sas7sods	SASODS	Output delivery system file
.snm*nn*	.sqn*nn*	.sas7bdmd	.sas7sdmd	DMDB	Data mining database
.sit*nn*	.srq*nn*	.sas7bitm	.sas7sitm	ITEMSTOR	Item store file
.sut*nn*	.suq*nn*	.sas7butl	.sas7sutl	UTILITY	Utility file
.spu*nn*	.spq*nn*	.sas7bput	.sas7sput	PUTILITY	Permanent utility file
.ssb*nn*	.sbq*nn*	.sas7bbak	.sas7sbak	BACKUP	Backup file

1 All Version 6 files end with a two-character code (*nn*) that identifies sets of compatible SAS files. See "Sharing Files in UNIX Environments" on page 124 for more information.

A UNIX directory can store a variety of files, but you might find it more practical to store files in separate directories according to their use. Also, you can keep libraries that are accessed by different engines in the same directory, but this is not recommended. See "Using Multiple Engines for a Library in UNIX Environments" on page 116 for more information.

Migrating 32-Bit SAS Files to 64-Bit in UNIX Environments

Compatibility of Existing SAS Files with SAS 9.1

Starting in SAS 9, SAS for the AIX, HP-UX, and Solaris operating environments is 64-bit only. Consequently, some SAS files (such as your SAS catalogs) that were created in 32-bit releases of SAS cannot be read by the V9 engine. SAS automatically tries to use Cross-Environment Data Access (CEDA) to process these files. The following table lists the supported processing for each SAS file.

Table 4.2 Supported Processing for SAS Files

File Type	Support
SAS data files	input processing [1], output processing [2]
MDDB file	input processing
PROC SQL view	input processing
SAS/ACCESS view for Oracle or SYBASE	input processing
SAS/ACCESS view other than for Oracle or SYBASE	no support
SAS catalog	no support
stored compiled DATA step program	no support
DATA step view	no support
item store	no support

1 To read data sets created in Version 6, use the V6 or V6TAPE read-only engines.
2 In SAS 9 if you create a new data file from the 32-bit file, the new file will be 64-bit.

For the supported SAS files, CEDA provides only read and write access. You cannot update these files. To update these files, you will need to convert them to 64-bit. You can use the MIGRATE procedure to convert all of your SAS files (both supported and unsupported) to 64-bit.

Note: See *SAS/CONNECT User's Guide* for information about accessing Version 6 SAS files if you use Remote Library Services to access SAS files on a server. △

Migrating Supported SAS Files

Benefits of Converting Supported SAS Files

If you need to access 32-bit SAS data sets, SAS/ACCESS views from Oracle or SYBASE, SQL views, or MDDB files from a 64-bit SAS session, then you can access these files using CEDA. (If you are trying to access Version 6 data sets, then you will need to use the V6 or V6TAPE read-only engines.) CEDA provides read and write access to these files. However, CEDA does not support update processing. CEDA also consumes additional resources each time that you read or write to these files.

Converting your data files enables you to

- have update access to these data files
- avoid the overhead of reading or writing to 32-bit data files in a 64-bit SAS session.

How to Convert Supported SAS Files

To convert these SAS files, use the MIGRATE procedure. Using PROC MIGRATE to convert your data files enables you to preserve any integrity constraints or indexes that are associated with the data file.

Note: You do not need access to a 32-bit release of Version 8 to convert these files. △

Migrating Unsupported SAS Files

Why You Need to Convert Your Unsupported SAS Files

Catalogs and other SAS files (not including SAS data sets) contain data structures that are only known to the application that created them. These files might contain data objects other than character or numeric objects, and therefore, these files cannot be shared between 64-bit SAS and earlier 32-bit releases of SAS. The MIGRATE procedure will automatically convert these files to the 64-bit version.

How to Convert Unsupported File Types

To convert an unsupported SAS file (see Table 4.2 on page 107 for a list of the supported file types), you can use the MIGRATE procedure. However, you must have access to a 32-bit release of Version 8 and a license for SAS/SHARE or SAS/CONNECT.

If you do not have SAS/SHARE or SAS/CONNECT, then you will need to use the CPORT and CIMPORT procedures and have access to a 32-bit release of Version 8 to convert your files. For more information, see "CPORT Procedure" on page 275 and "CIMPORT Procedure" on page 270.

Additional Resources

- For more information about the MIGRATE procedure, see the Migration Community at **`support.sas.com/rnd/migration`**.
- For more information about reading Version 6 data sets, see "Reading Version 6 Data Sets" on page 111.
- For more information about CEDA, see *SAS Language Reference: Concepts*.

Accessing SAS Files across Compatible Machine Types in UNIX Environments

Characteristics of Compatible Machine Types

Different computers store numeric binary data in different forms. Hewlett-Packard, Sun, and IBM store bytes in one order. Linux and Compaq Tru64 UNIX (formerly Digital UNIX) store bytes in a different order. SAS files can be transported between compatible machine types using various methods including NFS, FTP, and CD. For two machine types to be compatible, they must have the following characteristics in common:

- same word length. Word lengths can be either 32-bit or 64-bit.
- same ordering of bytes in memory. Machine types differ in whether the most significant byte (MSB) or the least significant byte (LSB) is loaded at the lower memory address. This is often referred to as "big endian" or "little endian."

Compatible Machine Types for Release 6.12 through Release 8.2

The tables in this section show the compatible machine types for Releases 6.12 through 8.2. After each table, a brief explanation is provided.

Table 4.3 Compatible Machine Types for Sharing Release 6.12 SAS Files

Bits	Compatible Machine Types
32	Intel ABI, Linux
32	HP-UX, Solaris, AIX, IRIX
64	Tru64 UNIX

You can move a Release 6.12 SAS data set that was created on a 32-bit HP-UX host to a 32-bit AIX host using NFS, FTP, or CD. Because HP-UX and AIX are compatible machine types, you can use the V6 or V6TAPE engine to read the HP-UX data set on the AIX host.

The same 32-bit HP-UX data set can be moved to a 32-bit Intel ABI host. However because these machine types are incompatible, you cannot use the V6 or V6TAPE engine to read the HP-UX data set.

For information about reading Version 6 data sets, see "Reading Version 6 Data Sets" on page 111.

Table 4.4 Compatible Machine Types for Sharing Version 8 SAS Data Sets

Bits	Compatible Machine Types
32	Intel ABI, Linux
32	HP-UX, Solaris, AIX, IRIX
64	AIX, Solaris, HP-UX PA Risc
64	Tru64 UNIX

You can move a Version 8 data set that was created on a 32-bit Solaris host to a 32-bit HP-UX host using methods such as NFS, FTP, or CD. Because these are compatible machine types, you will be able to read this data set in SAS.

The same 32-bit Solaris data set can be moved to a 64-bit HP-UX host. Because these machine types are incompatible, SAS will use CEDA to read this data set. For more information, see "Using CEDA to Read Data Sets" on page 111.

Determining Compatible Machine Types in SAS 9.1

In SAS 9.1, the **`Data Representation`** field of the PROC CONTENTS output shows the compatible machine types for a SAS file. The following is a portion of the PROC CONTENTS output.

Output 4.1 Portion of PROC CONTENTS Output Using the V9 Engine

```
                          The CONTENTS Procedure

Data Set Name        CLASSES.MAJORS                          Observations          5
Member Type          DATA                                    Variables             6
Engine               V9                                      Indexes               0
Created              Monday, February 24, 2003 14:30:19      Observation Length    48
Last Modified        Monday, February 24, 2003 14:30:19      Deleted Observations  0
Protection                                                   Compressed            NO
Data Set Type                                                Sorted                NO
Label                Second Data Set
Data Representation HP_UX_64, RS_6000_AIX_64, SOLARIS_64, HP_IA64
Encoding             latin1  Western (ISO)
```

In this example, the **Data Representation** field output shows that the compatible machine types for this data set are: HP-UX (**HP_UX_64**) AIX 64-bit (**RS_6000_AIX_64**), Solaris 64-bit (**Solaris_64**) and HP-UX Itanium (**HP_IA64**). Therefore, you can move a data set that was created on a 64-bit HP-UX host to a 64-bit AIX, Solaris, or HP-UX Itanium host using NFS, FTP, or CD. You could not move this data set to a 32-bit Linux host using these methods. However, you can read this data set using CEDA. For more information, see "Using CEDA to Read Data Sets" on page 111.

The following table lists the possible values for the **Data Representation** field for SAS 9.1 and the corresponding machine types.

Table 4.5 Data Representation Value for Each Machine Type in SAS 9.1

Data Representation Value	Corresponding Machine Type
ALPHA_TRU64	Compaq Tru64
HP_UX_64	HP-UX PA-Risc 64-bit
HP_IA64	HP-UX Itanium Processor Family
INTEL_ABI	Intel ABI Compliant Operating Systems
LINUX	Linux on Intel 32-bit Hardware
RS_6000_AIX_64	AIX 64-bit
SOLARIS_64	Solaris 64-bit

Note: The **Encoding** value might affect your ability to move SAS files between compatible machine types. It is important to note this value when you are transporting SAS files between hosts. For more information about encoding, see *SAS National Language Support (NLS): User's Guide*. △

Creating a SAS File to Use with an Earlier Release

The V9 and V9TAPE engines differ slightly from previous SAS engines. These engines support longer format and informat names than previous SAS engines. To ensure compatibility between releases, see *SAS Language Reference: Concepts*.

Reading SAS Data Sets from Previous Releases or from Other Hosts

Reading Version 6 Data Sets

Using the V6 and V6TAPE read-only engines, SAS can read Version 6 data sets that were created by compatible machine types. The following examples demonstrate how you can use the V6 engine.

- If you are running SAS 9.1 on Linux, you can use the V6 engine to read Version 6 data sets that were created with any Intel ABI release of SAS, such as SCO UNIX.
- If you are running SAS 9.1 on HP-UX, you can use the V6 engine to read Version 6 data sets that were created on HP-UX, Solaris, AIX, or IRIX.

For a list of the compatible machines types for V6, see "Compatible Machine Types for Release 6.12 through Release 8.2" on page 108. For more information about the compatibility of Version 6 files with SAS 9.1, see *SAS Language Reference: Concepts*.

Using CEDA to Read Data Sets

CEDA enables a SAS data set that was created in Version 7 or later in any directory-based operating environment (such as UNIX, Windows, and OpenVMS Alpha) to be read by a SAS session that is running in another directory-based environment. In SAS 9.1, if you try to access a data set that was created in a previous release, then SAS automatically uses CEDA to process the file. For example, if you are running SAS 9.1 on Linux, SAS will use CEDA to process a data set that was created in Version 8 on a 64-bit Solaris host. With CEDA, you have read and write access to these files; however, you will not be able to update the file.

For information about CEDA, see *SAS Language Reference: Concepts*.

Referring to SAS Data Files Using Librefs in UNIX Environments

Techniques for Referring to a SAS File

If you want to read or write to a permanent SAS file, you can refer to the SAS file in one of two ways:

- refer to the data file directly by using its pathname in the appropriate statements (such as DATA, SET, MERGE, UPDATE, OUTPUT, and PROC).
- assign a libref to the SAS data library (directory) that contains the data file and use the libref as the first level of a two-level filename.

What Is a Libref?

A libref is a nickname that you can use to refer to the data library during a SAS session or job. You will probably want to use a libref when:

- the data file pathname is long and must be specified several times within a program

- the pathname might change. If the pathname changes, you need to change only the statement assigning the libref, not every reference to the file.
- your application will be used on other platforms. Using librefs makes it easier to port an application to other operating environments.
- you need to concatenate libraries. See "Assigning a Libref to Several Directories (Concatenating Directories)" on page 115 for more information.

Librefs can be stored in the SAS registry. See "Customizing Your SAS Registry Files" on page 16 for more information.

Assigning Librefs

You can use any of the following methods to assign a SAS libref:

- the LIBNAME statement
- the LIBNAME function
- the LIBASSIGN command
- the LIBNAME window
- the SAS Explorer window .

A libref assignment remains in effect for the duration of the SAS job, session, or process unless you either clear the libref or use the same libref in another LIBNAME statement or LIBNAME function.

If you assign a libref from a SAS process, that libref is valid only within that SAS process. If you clear a libref from within a SAS process, that libref is not cleared from other SAS processes.

Using the LIBNAME Statement

The LIBNAME statement identifies a SAS data library to SAS, associates an engine with the library, enables you to specify options for the library, and assigns a libref to it. For details about LIBNAME statement syntax, see "LIBNAME Statement" on page 301.

Using the LIBNAME Function

The LIBNAME function takes the same arguments and options as the LIBNAME statement. For more information about the LIBNAME function, see "LIBNAME Function" on page 250.

Using the LIBASSIGN Command

Perform the following steps to assign a libref using the LIBASSIGN command:

1. Issue the LIBASSIGN command in the command window. The New Library dialog box opens.
2. Specify the libref in the **Name** field.
3. Specify an engine for the libref in the **Engine** field by selecting the default engine or another engine from the drop-down menu. Depending on the engine that you specify, the fields in the **Library Information** area might change.
4. Click the **Enable at startup** box to assign this libref when you invoke SAS.
5. Specify the necessary information for the desired SAS data library in the **Library Information** area. Depending on the engine selected, there may or may not be a **Path** field available for input.

6 Specify LIBNAME options in the **Options** field. These options can be specific to your host or engine, including options that are specific to a SAS engine that accesses another software vendor's relational database system.

7 Click OK.

Using the LIBNAME Window

Perform the following steps to assign a libref from the LIBNAME window

1 Issue the LIBNAME command in the command window. The LIBNAME window opens.

2 From the **File** pull-down menu, select **New**. The New Library dialog box opens.

3 Fill in the fields in the New Library dialog box, described in "Using the LIBASSIGN Command" on page 112.

4 Click OK.

Using the SAS Explorer Window

Perform the following steps to assign a libref from the SAS Explorer window:

1 From the **File** pull-down menu, select **New** when the Libraries node in the tree structure is active. The New dialog box opens.

2 Select **Library**, and then click OK. The New Library dialog box opens.

3 Fill in the fields in the New Library dialog box, described in "Using the LIBASSIGN Command" on page 112.

4 Click OK.

Permanently Assigning a Libref

You might want to save a libref so that it is valid between SAS sessions. You can assign a libref permanently by using one of the following methods:

- Specify the LIBNAME statement or LIBNAME function in an autoexec file. For more information, see "LIBNAME Function" on page 250 or "LIBNAME Statement" on page 301
- Select the **Enable at startup** box when you assign a libref using the LIBASSIGN command, LIBNAME window, or Explorer Window. This will save the libref in the SAS Registry. For more information about these methods, see "Assigning Librefs" on page 112.
- Use environment variables as librefs. Include these environment variables in your startup files so that these variables are set when SAS is invoked.

Accessing a Permanent SAS Data Library Using a Libref

After you have defined a libref, you can use the libref in one of two ways to access a permanent SAS data library:

- as the first level of a two-level SAS filename:

  libref.member-name

  where *libref* is the first-level name referring to the directory where the file is stored, and *member-name* specifies the name of the file being read or created.
- as the value of the USER= option. (See "Using One-Level Names To Access Permanent Files (User Data Library)" on page 120 for details.)

For example, these SAS statements access the data file Final.sas7bdat in the Sales library that is stored in the **/users/myid/mydir** directory :

```
libname sales '/users/myid/mydir';
data sales.final;
```

Specifying Pathnames in UNIX Environments

Rules for Specifying Directory and Pathnames

Whether you specify a data file name directly in the various SAS statements or specify the data library name in a LIBNAME statement and then refer to the libref, the same rules apply for specifying UNIX directory and file pathnames.

Specify directory and file pathnames in quotation marks. The level of specification depends on your current directory.

Example 1: Accessing a File That Is Not in the Current Directory

If **/u/1999/budgets** is not your current directory, then to access the data file named May, you must specify the entire pathname:

```
data '/u/1999/budgets/may';
```

If you wanted to use a libref, you would specify:

```
libname budgets '/u/1999/budgets';
data budgets.may;
```

Example 2: Accessing a File in the Current Directory

If **/u/1999/budgets** is your current directory, you could specify only the data file names:

```
data 'quarter1';
merge 'jan' 'feb' 'mar';
run;
```

Note: If you omit the quotation marks, then SAS assumes that these data sets are stored in the Saswork directory. △

If you wanted to use a libref, you would specify:

```
libname budgets '.';
data budgets.quarter1;
merge budgets.jan budgets.feb budgets.mar;
run;
```

Valid Character Substitutions in Pathnames

You can use the character substitutions shown in the following table to specify pathnames.

Table 4.6 Character Substitutions in Pathnames

Characters	Meaning
~/	$HOME/ Can be used only at the beginning of a pathname.
~*name*/	*name*'s home directory (taken from file **/etc/passwd**). Can be used only at the beginning of a pathname.
!sasroot	name of **sasroot** directory (see Appendix 1, "The !SASROOT Directory," on page 397). Specified only at the beginning of a pathname.
.	current working directory
..	parent of current working directory
$*VARIABLE*	environment variable *VARIABLE*

Assigning a Libref to Several Directories (Concatenating Directories)

Introduction to Concatenating Directories

You can use the LIBNAME statement to assign librefs and engines to one or more directories, including the Work directory.

If you have SAS data sets located in multiple directories, you can treat these directories as a single SAS data library by specifying a single libref and concatenating the directory locations, as in the following example:

```
libname income ('/u/2002/revenue', '/u/2002/costs');
```

This statement indicates that the two directories, **/u/2002/revenue** and **/u/2002/costs**, are to be treated as a single SAS data library.

If you have already assigned librefs to your SAS data libraries, you can use these librefs to indicate that you want to concatenate the data libraries, as in this example:

```
libname income ('/u/2002/corpsale', '/u/2002/retail');
libname costs ('/u/2002/salaries', '/u/2002/expenses');
libname profits (income, costs, '/u/2002/capgain');
```

This statement indicates that the five directories, **/u/2002/corpsale**, **/u/2002/retail**, **/u/2002/salaries**, **/u/2002/expenses**, and **/u/2002/capgain**, are to be treated as a single SAS data library.

How SAS Accesses Concatenated Data Libraries

When you concatenate SAS data libraries, SAS uses a protocol for accessing the libraries, which depends on whether you are accessing the libraries for read, write, or update. (A *protocol* is a set of rules.)

SAS uses the protocol shown in the following sections to determine which directory is accessed. (The protocol illustrated by these examples applies to all SAS statements and procedures that access SAS files, such as the DATA, UPDATE, and MODIFY statements in the DATA step, and the SQL and APPEND procedures.)

Accessing Files for Input and Update

When a SAS data set is accessed for input or update, the first SAS data set that is found by that name is the one that is accessed. For example, if you submit the following statements and the data set Old.Species exists in both directories, the one in the **mysasdir** directory is the one that is printed:

```
libname old ('mysasdir','saslib');
proc print data=old.species;
run;
```

The same would be true if you opened Old.Species for update with the FSEDIT procedure.

Accessing Files for Output

If the data set is accessed for output, it is always written to the first directory, provided that the directory exists. If the directory does not exist, an error message is displayed. For example, if you submit the following statements, SAS writes the Old.Species data set to the first directory (**mysasdir**) and replaces any existing data set with the same name:

```
libname old ('mysasdir','saslib');
data old.species;
x=1;
y=2;
run;
```

If a copy of the Old.Species data set exists in the second directory, it is not replaced.

Accessing Data Sets with the Same Name

If you use the DATA and SET statements to access data sets with the same name, the DATA statement uses the output rules and the SET statement uses the input rules. For example, suppose you submit the following statements and Test.Species originally exists only in the second directory, **mysasdir**:

```
libname test ('sas','mysasdir');
data test.species;
set test.species;
if value1='y' then
   value2=3;
run;
```

The DATA statement opens Test.Species for output according to the output rules; that is, SAS opens a data set in the first of the concatenated libraries (**sas**). The SET statement opens the existing Test.Species data set in the second (**mysasdir**) directory, according to the input rules. Therefore, the original Test.Species data set is not updated. After the data step executes, two Test.Species data sets exist, one in each directory.

Using Multiple Engines for a Library in UNIX Environments

You can assign multiple librefs to a single directory, and specify a different engine with each libref. For example, after the following statements are executed, data sets

that are referenced by One are created and accessed using the default engine, while data sets that are referenced by Two are created and accessed using the sequential engine:

```
libname one v9 '/users/myid/educ';
libname two tape '/users/myid/educ';
```

Note: Keeping different types of data libraries in one directory is not recommended because you must remember the appropriate engine for accessing each library. SAS cannot determine the right engine for accessing libraries in a directory that contains libraries of different types. See "Omitting Engine Names from the LIBNAME Statement" on page 304 for more information. △

Using Environment Variables as Librefs in UNIX Environments

An environment variable can also be used as a libref. The variable name must be in all uppercase characters, and the variable value must be the full pathname of the directory, that is, the name of the directory must begin with a slash.

Note: SAS on UNIX does not support the assignment of the User libref via the USER environment variable. △

Suppose you want to use the data library in **/users/mydir/educ**, and you want to refer to it with the EDUC environment variable. You can define the variable at two times:

- before you invoke SAS. See "Defining Environment Variables in UNIX Environments" on page 21. For example, in the Korn shell, you would use

  ```
  export EDUC=/users/mydir/educ
  ```

- after you invoke SAS you can use the X statement (see "Executing Operating System Commands from Your SAS Session" on page 13) and the SAS **setenv** command:

  ```
  x setenv EDUC /users/mydir/educ;
  ```

You cannot specify an engine when you define a libref as an environment variable, so SAS determines which engine to use as described in "Omitting Engine Names from the LIBNAME Statement" on page 304.

After the libref is defined, you can use it to access data sets stored in the library:

```
proc print data=educ.class;
run;
```

Note: If a variable and a libref have the same name but refer to different files, SAS uses the libref. △

Librefs Assigned by SAS in UNIX Environments

Automatically Defined Librefs

SAS automatically defines three librefs:

Sashelp
: contains a group of catalogs that contain information that is used to control various aspects of your SAS session. The Sashelp library is in the **!SASROOT** directory. See Appendix 1, "The !SASROOT Directory," on page 397.

Sasuser
: contains SAS catalogs that enable you to tailor features of SAS (such as window size, font settings, and printer entries) for your needs. If the defaults in the Sashelp library are not suitable for your applications, you can modify them and store your personalized defaults in your Sasuser library.

Work
: is the temporary, or scratch, library automatically defined by SAS at the beginning of each SAS session or job. The Work library stores two types of temporary files: those you create and those created internally by SAS as part of normal processing.

These librefs and the LIBRARY libref are reserved librefs. If your site also has SAS/GRAPH software or SAS/GIS software, the MAPS or GISMAPS librefs might also be automatically defined. All these libraries are described in *SAS Language Reference: Dictionary*. Sasuser and Work have operating system dependencies.

Sasuser Data Library

What Is the Sasuser Library?

The Sasuser library contains the customizations (such as window size and positioning, colors, fonts, and printer entries) that you specified for your SAS session. When you invoke SAS, it looks for the Sasuser directory to find these customizations. If this directory does not exist, SAS uses the SASUSER system option to create it. The default directory is set in the system configuration file (sasv9.cfg) and is usually similar to the following:

```
-sasuser ~/sasuser.v91
```

This specification tells SAS to create a directory for the Sasuser libref in your home directory. To determine the value of this directory for your system, use PROC OPTIONS or **libname sasuser LIST**.

You can permit read-only access to the Sasuser library by using the RSASUSER system option. See Chapter 17, "System Options under UNIX," on page 311 for details on the SASUSER and RSASUSER system options.

After the Sasuser library has been created, SAS automatically assigns the same Sasuser libref to it each time you start a SAS session. It cannot be cleared or reassigned during a SAS session. If you delete the library, SAS re-creates it the next time you start a session. Because SAS assigns the libref for you, you do not need to use a LIBNAME statement before referencing this library.

Contents of the Sasuser Library

Your customizations are stored in one of the following locations in the Sasuser library:

Sasuser.Profile catalog

The Sasuser.Profile catalog is the profile.sas7bcat file in your Sasuser library. If you change any function key definitions, window attributes (such as size, color, and position), or PMENU settings during your SAS session, SAS saves the changes in the Sasuser.Profile catalog.

If Sasuser.Profile does not exist, then at invocation SAS checks for the Sashelp.Profile catalog. (This catalog will only exist if you have copied your Sasuser.Profile catalog to the Sashelp library.) If the Sashelp.Profile catalog exists, then SAS will copy it to the Sasuser library, and this will become your new Sasuser.Profile catalog. If the Sashelp.Profile catalog does not exist, then SAS will create a Sasuser.Profile using the default settings for a SAS session.

If you invoke SAS and discover that your customizations have been lost, then your Sasuser.Profile is either corrupted or locked by another SAS session started with the same user ID. If either of these conditions are true, then the following messages will appear in the SAS log:

```
NOTE:  Unable to open SASUSER.PROFILE.  WORK.PROFILE will be opened instead.
NOTE:  All profile changes will be lost at the end of the session.
```

When this occurs, SAS creates a Work.Profile catalog (in the Work library) using the default settings for a SAS session. This Work.Profile catalog is used for the duration of the SAS session. Since the contents of the Work directory are temporary, any customizations that you save to the Work.Profile catalog will be lost at the end of the SAS session.

To resolve these problems with your Sasuser.Profile catalog, you can try one of the following options:

- If your Sasuser.Profile has been corrupted, then you can remove this catalog, and let SAS create a new one at the next invocation.
- If your Sasuser.Profile is being used by multiple SAS sessions, then you can specify the RSASUSER system option to permit read-only access to the Sasuser library. Since this permission is read-only, you will not be able to save any customizations to your Sasuser.Profile during that SAS session.

Sasuser.Registry catalog

The Sasuser.Registry catalog is the registry.sas7bitm file in your Sasuser library. If you change any Universal Printing entries or libref assignments during a SAS session, then SAS saves the changes in the Sasuser.Registry catalog.

At invocation, SAS looks in the Sasuser directory to see if it can write to the Sasuser.Registry catalog. If SAS cannot write to this catalog, then the following warning appears in the SAS log:

```
WARNING:  Unable to open SASUSER.REGISTRY.  WORK.REGISTRY will be used instead.
NOTE:  All registry changes will be lost at the end of the session.
```

If SAS can read the Sasuser.Registry, then SAS copies the Sasuser.Registry to create a Work.Registry catalog (in the Work library). This Work.Registry catalog will be used for the duration of the SAS session. Since the contents of the Work library are temporary, then any customizations that you save to the Work.Registry catalog will be lost at the end of the SAS session. However, the customizations saved in Sasuser.Registry will still exist.

If SAS cannot read the Sasuser.Registry, then SAS creates the Work.Registry catalog using the default settings for a SAS session. In this case, SAS issues an additional warning to the SAS log:

```
WARNING:  Unable to copy SASUSER.REGISTRY to WORK.REGISTRY.
```

Sasuser.Prefs file
The settings that you specify in the Preferences dialog box (with the exception of those resources on the **General** tab) are saved in the SasuserPrefs file. For more information about these resources, see "Modifying X Resources through the Preferences Dialog Box" on page 57.

These are three of the files that you can have in your Sasuser library. However, you can store other data sets and catalogs in the Sasuser library as well.

Work Data Library

The Work data library is the temporary library that is automatically defined by SAS at the beginning of each SAS session or job. The Work data library stores temporary SAS files that you create as well as files created internally by SAS.

To access files in the Work data library, simply specify a one-level name for the file. The libref Work is automatically assigned to these files unless you have assigned the User libref.

When you invoke SAS, it assigns the Work libref to a subdirectory of the directory specified in the WORK system option described in Chapter 17, "System Options under UNIX," on page 311. This subdirectory is usually named **SAS_workcode_nodename** where

code
is a 12-character code. The first four characters are randomly generated numbers. The next eight characters are based on the hexadecimal process ID of the SAS session

nodename
is the name of the UNIX box where the SAS process is running

This libref cannot be cleared or reassigned during a SAS session.

The WORKINIT and WORKTERM system options control the creation and deletion of the Work data library. See *SAS Language Reference: Dictionary* for details.

Note: If a SAS session is terminated improperly (for example, using the **kill -9** command), SAS will not delete the **SAS_workcode_nodename** directory. You might want to use the **cleanwork** command to delete these straggling directories (see Appendix 2, "Tools for the System Administrator," on page 399). △

Using One-Level Names To Access Permanent Files (User Data Library)

Introduction to One-Level Names

SAS data sets are referenced with a one- or two-level name. The two-level name is of the form *libref.member-name* where *libref* refers to the SAS data library in which the data set resides and *member-name* refers to the particular *member* within that library. The one-level name is of the form *member-name* (without a *libref*). In this case, SAS stores the files in the temporary Work data library. To override this action and have files with one-level names stored in a permanent library, first assign the User libref to an existing directory. To refer to temporary SAS files while User is assigned, use a two-level name with Work as the libref.

Techniques for Assigning the User Libref

You have three ways to assign the User libref:

- Assign the User libref directly using the LIBNAME statement:

```
libname user '/users/myid/mydir';
```

- Specify the USER= system option before you start the session. For example, you can assign the User libref when you invoke SAS:

```
sas -user /users/myid/mydir
```

- Specify the USER= system option after you start the session. First, assign a libref to the permanent library. Then use the USER= system option in an OPTIONS statement to equate that libref to User. For example, these statements assign the libref User to the directory with libref Mine:

```
libname mine '/users/myid/mydir';
options user=mine;
```

See Chapter 17, "System Options under UNIX," on page 311 for details on the USER= system option.

Note: SAS on UNIX does not support the assignment of the User libref via the USER environment variable. △

Accessing Disk-Format Data Libraries in UNIX Environments

You will probably create and access data libraries on disk more than any other type of library. The default engine and the compatibility engines allow read, write, and update access to SAS files on disk. They also support indexing and compression.

In the following example, the In libref is assigned to a directory that contains the Stats1 data set:

```
libname in '/users/myid/myappl';
proc print data=in.stats1;
run;
```

Remember, a *SAS-data-library* must already exist before SAS can read from or write to this directory. For example, if you want to create the SAS data set Orders in a directory, use the X statement to issue the **mkdir** UNIX command. Then you can use the LIBNAME statement to associate the libref with the directory.

```
x mkdir /users/publish/books;
libname books '/users/publish/books';
data books.orders;
  more SAS statements
run;
```

By default, the LIBNAME statement associates the V9 engine with the directory.

Accessing Sequential-Format Data Libraries in UNIX Environments

Benefits and Limitations of Sequential Engines

The sequential engines enable you to access data libraries in sequential format on tape or disk. The sequential engines do not support indexing and compression of observations.

Note: Before using sequential engines, read the information about sequential data libraries in *SAS Language Reference: Dictionary*. △

Reading and Writing SAS Files on Tape

Introduction to SAS Libraries on Tape

A SAS library on tape can contain one or more SAS data sets; however, only one SAS data set from a given library on tape can be accessed at any given point in a SAS job.

Benefit of Using a Staging Directory

You can write SAS files directly to tape using the TAPE engine; however, it is more efficient to use a staging directory so that the files can be processed directly from disk. You can use the UNIX **tar** command to move SAS data sets between the staging directory and tape. (Do not use the UNIX **cp** command.)

Syntax of the LIBNAME Statement

To access SAS 9.1 files on tape, you can specify the V9TAPE or TAPE engine in the LIBNAME statement:

LIBNAME *libref* V9TAPE '*tape-device-pathname*';

The *tape-device-pathname* must be a pathname for a tape device; it should be the name of the special file associated with the tape device. (Check with your system administrator for details.) The name must be enclosed in quotation marks. You cannot specify remote tape devices in the LIBNAME statement.

Example: Assigning a Libref to the Tape Device

The following LIBNAME statement assigns the libref Seq2 to the **/dev/tape2** tape device. Because the tape device is specified, the engine does not have to be specified.

```
libname seq2 '/dev/tape2';
```

Accessing Multi-Volume Tape Libraries

Multi-volume tape libraries are supported if you specify the TAPECLOSE=LEAVE system option when you start your SAS session.

Reading and Writing Transport Formats on Tape

Transport formats on tape are handled in a manner similar to external files. Read "Processing Files on TAPE in UNIX Environments" on page 149 before continuing with this topic.

Example: Transporting a File from Tape to the Work Library

The following SAS statements issue the UNIX **mt** command to rewind the tape and create a transport file using the xport engine and PROC CPORT:

```
x 'mt -t /dev/rmt/0mn rewind';
libname tranfile xport '/dev/rmt/0mn';
proc cport library=sasuser file=tranfile;
run;
```

The following statements import the transport file from tape into the Work data library:

```
x 'mt -t /dev/rmt/0mn rewind';
libname tranfile xport '/dev/rmt/0mn';
proc cimport infile=tranfile library=work;
run;
```

Writing Sequential Data Sets to Named Pipes

Why Use Named Pipes?

You can send output to and read input from the operating system by using named pipes. For example, you might want to compress a data set or send it to a tape management system without creating intermediate files.

Syntax of the LIBNAME Statement

You can read from and write to named pipes from within your SAS session by specifying the pipe name in the LIBNAME statement:

LIBNAME *libref* <TAPE> '*pipename*';

Since you cannot position a pipe file, SAS uses the TAPE engine to ensure sequential access. You do not have to specify the engine name; TAPE is assumed.

Example: Creating a SAS Data Set Using a Named Pipe

To create a SAS data set and compress the data set without creating an intermediate, uncompressed data set, create a named pipe (such as **mypipe**) and enter the **compress** command:

```
mknod mypipe p compress <mypipe >sasds.Z
```

In your SAS session, assign a libref to the pipe and begin writing to the data set:

```
libname x 'mypipe';
data x.a;
  ...more SAS statements...
output;
run;
```

The data is sent to **mypipe**, compressed, and written to the data set. When SAS closes the data set, the compress finishes, and you have a compressed, sequential data set in **sasds.Z**.

If you begin writing to a named pipe before the task on the other end (in this case, the **compress** command) begins reading, your SAS session will be suspended until the task begins to read.

Sharing Files in UNIX Environments

Sharing Files with the FILELOCKS System Option

If more than one user accesses a SAS file at the same time or if a single user has access to the same file from different SAS sessions, the results could be unpredictable. By default, the FILELOCKS system option is set to FAIL, which enables multiple SAS sessions to simultaneously read the same SAS file. (See "FILELOCKS System Option" on page 329.)

Conditions to Check When FILELOCKS=NONE

If FILELOCKS has been set to NONE, then you should do one of the following:

- Make sure that your **sasuser** directory is unique for each SAS session. Typically, the system administrator assigns this directory in the system configuration file. The specification in that file or in your personal configuration file helps ensure that the directory is unique as long as you run only one SAS session at a time.

  If you run two or more SAS sessions simultaneously, you can guarantee unique user files by specifying different **sasuser** directories for each session. In the first session, you can use

  ```
  -sasuser ~/sasuser
  ```

  In the *n*th session, you can use

  ```
  -sasuser ~/sasusern
  ```

  For details, see "Order of Precedence for SAS Configuration Files" on page 17 and "SASUSER System Option" on page 361. The RSASUSER option can be used to control modifications to the Sasuser library when it is shared by several users (see "RSASUSER System Option" on page 354).
- If you run two or more SAS sessions simultaneously (either by using the X statement or by invoking it from two different windows) and you use the same Sasuser.Profile catalog, do not do anything within the SAS session to change that catalog (for example, using the WSAVE command or changing key assignments) because both sessions can use the same one.
- If you and other people use the same data sets, avoid writing to them at the same time. However, multiple people can read these data sets at the same time.

Sharing Files in a Network

Introduction to Sharing Files Across Workstations

SAS can be licensed to run on one or more workstations in a network of similar machines. The license specifically lists the workstations that SAS can run on. Other workstations in the network may have access to the SAS executable files but not be able to run SAS.

If the licensed workstations are connected via NFS mounts so that they share a file system, they can all share a single copy of the SAS executables, although this is not necessary. They can also share SAS files. However, if a SAS session attempts to update a data set or catalog, it must obtain an exclusive file lock on that file to prevent other sessions from accessing that file.

If SAS is installed on workstations of different types that are connected via NFS, then each type of workstation must have its own copy of the SAS executables. For information about how to move catalogs and data sets between hosts, see "Accessing SAS Files across Compatible Machine Types in UNIX Environments" on page 108.

Accessing Files on Different Types of Workstations

If the data set or catalog you want to process exists on your network but cannot be accessed with the LIBNAME statement because it resides on a different type of workstation, you have several alternatives:

- You can log in to the remote machine and convert the file to SAS transport format using the CPORT procedure, copy the transport file to your machine or access it via NFS, and import it using the CIMPORT procedure to your type of machine format.
- You can FTP or RCP the file from the remote machine to your machine.
- You can log in to the remote machine and perform the work there. This alternative works best when the file is used rarely, or if the original file changes often.
- You can do part of your work on your machine and the other part on the remote machine. One example of this alternative would be to run a set of statements on a small test case on the local machine, and then submit the real work to be done on the remote machine. Similarly, you might want to subset a large data set on another machine, and then do local analysis on that subset. This can be accomplished with SAS/CONNECT software. For more information about Remote Library Services, see *SAS/CONNECT User's Guide*.

Troubleshooting Accessing Data Over NFS Mounts

SAS might hang when accessing data over NFS mounts if the FILELOCKS option is set to **FAIL** or **CONTINUE**. To alleviate the problem, make sure that all NFS filelocking daemons are running on both machines (usually **statd** and **lockd**). Your system administrator can assist with starting **statd** and **lockd**.

Note: To test whether there is a problem with file locking, you can set the FILELOCKS system option to **NONE** temporarily. If setting FILELOCKS to **NONE** resolves the problem, then you know that there probably is a problem with the **statd** and **lockd** daemons. It is recommended that you do not set FILELOCKS to **NONE** permanently as it might cause data corruption or unpredictable results. △

Accessing BMDP, OSIRIS, or SPSS Files in UNIX Environments

Introduction to the BMDP, OSIRIS, and SPSS Files

SAS includes three interface library engines, BMDP, OSIRIS and SPSS, that enable you to access external data directly from a SAS program. All these engines are read-only.

Because they are sequential, these engines cannot be used with the POINT= option on the SET statement or with the FSBROWSE, FSEDIT, or FSVIEW procedures. You can use PROC COPY, PROC DATASETS, or a DATA step to copy a BMDP or OSIRIS system file or a SPSS export file to a SAS data set and then perform these functions on the SAS data set. Also, some procedures (such as PROC PRINT) give a warning message about the engine being sequential.

With these engines, the physical filename associated with a libref is an actual filename, not a directory. This is an exception to the rules concerning librefs.

You can also use the CONVERT procedure to convert BMDP, OSIRIS and SPSS files to SAS data files. See "CONVERT Procedure" on page 272 for more information.

The BMDP Engine

The BMDP interface library engine enables you to read BMDP files from the BMDP statistical software package directly from a SAS program. The BMDP engine is a read-only engine. The following discussion assumes you are familiar with the BMDP save file terminology.*

Note: This engine is available for AIX, HP-UX, and Solaris. △

Syntax for Accessing BMDP Save Files

To read a BMDP save file, issue a LIBNAME statement that explicitly specifies the BMDP engine. In this case, the LIBNAME statement takes this form:

LIBNAME *libref* BMDP '*filename*';

In this form of the LIBNAME statement, *libref* is a SAS libref and *filename* is the BMDP physical filename. If the libref appears previously as a fileref, omit *filename* because the physical filename associated with the fileref is used. This engine can only read save files created under UNIX.

Because there can be multiple save files in a single physical file, you reference the CODE= value as the member name of the data set within the SAS language. For example, if the save file contains CODE=ABC and CODE=DEF and the libref is MyLib, you reference them as MyLib.ABC and MyLib.DEF. All CONTENT types are treated the same, so even if member DEF is CONTENT=CORR, it is treated as CONTENT=DATA.

If you know that you want to access the first save file in the physical file or if there is only one save file, refer to the member name as _FIRST_. This is convenient if you do not happen to know the CODE= value.

Example: BMDP Engine

Assume that the physical file MyBMDP.dat contains the save file ABC. The following SAS code associates the libref MyLib with the BMDP physical file and runs the CONTENTS and PRINT procedures on the save file:

```
libname mylib bmdp 'mybmdp.dat';
proc contents data=mylib.abc;
run;
proc print data=mylib.abc;
run;
```

The following example uses the LIBNAME statement to associate the libref MyLib2 with the BMDP physical file. Then it prints the data for the first save file in the physical file:

```
libname mylib2 bmdp 'mybmdp.dat';
proc print data=mylib2._first_;
run;
```

* See the documentation provided by BMDP Statistical Software Inc. for more information.

The OSIRIS Engine

The Inter-University Consortium on Policy and Social Research (ICPSR) uses the OSIRIS file format for distribution of its data files. SAS provides the OSIRIS interface library engine to support the many users of the ICPSR data and to be compatible with PROC CONVERT.

With the OSIRIS engine, you can read OSIRIS data and dictionary files directly from a SAS program. The following discussion assumes you are familiar with the OSIRIS file terminology and structure. If you are not, refer to the documentation provided by the ICPSR.

Notes on the OSIRIS Data Dictionary Files

Since the OSIRIS software does not run outside the MVS environment, the layout of an OSIRIS data dictionary is consistent across operating environments. However, the OSIRIS engine is designed to accept a data dictionary from any other operating environment on which SAS runs. It is important that the dictionary and data files not be converted from EBCDIC to ASCII; the engine expects EBCDIC data.

The dictionary file should consist of fixed-length records of length 80. The data file should contain records large enough to hold the data described in the dictionary.

Syntax for Accessing an OSIRIS File

To read an OSIRIS file, issue a LIBNAME statement that explicitly specifies the OSIRIS engine. The syntax of the LIBNAME statement in this case is

LIBNAME *libref* OSIRIS '*data-filename*' DICT='*dictionary-filename*';

libref
is a SAS libref.

'*data-filename*'
is the physical filename of the data file. If the libref appears also as a fileref, omit the data filename.

'*dictionary-filename*'
is the physical filename of the dictionary file. The dictionary filename can also be an environment variable or a fileref, but if it is either of those, do not enclose it in quotation marks. The DICT= option is required.

OSIRIS data files do not have member names. Therefore, use whatever member name you like.

To use the same dictionary file with different data files, code a separate LIBNAME statement for each one.

Example: OSIRIS Engine

In the following example, the data file is **/users/myid/osr/dat**, and the dictionary file is **/users/myid/osr/dic**. The example associates the libref MYLIB with the OSIRIS files and runs a PROC CONTENTS and PROC PRINT on the data.

```
libname mylib osiris '/users/myid/osr/dat'
   dict='/users/myid/osr/dic';
proc contents data=mylib._first_;
run;
proc print data=mylib._first_;
run;
```

The SPSS Engine

With the SPSS interface library engine, you can read only SPSS export files. This engine does not read SPSS-X native files.

Syntax for Accessing an SPSS Export File

To read an SPSS export file, issue a LIBNAME statement that explicitly specifies the SPSS engine. The syntax of the LIBNAME statement in this case is

LIBNAME *libref* SPSS '*filename*';

Libref is a SAS libref and *filename* is the physical filename. If the libref appears also as a fileref, omit *filename*; the physical filename associated with the fileref is used.

Export files must be created by the SPSS EXPORT command and can originate from any operating environment. Export files must be transported to and from your operating environment in ASCII format. If they are transported in binary format, other operating environments will not be able to read them.

Because SPSS-X files do not have internal names, refer to them by any member name you like. A common extension for export files is .por, but this extension is not required.

Example: SPSS Engine

The following example associates the libref MYLIB with the physical file **/users/myid/mydir/myspsssx.por** in order to run the CONTENTS and PRINT procedures on the export file:

```
libname mylib spss '/users/myid/mydir/myspsssx.por';
proc contents data=mylib._first_;
proc print data=mylib._first_;
run;
```

In the next example, the FILENAME statement associates the fileref MyLib2 with the **/users/myid/mydir/aspssx.por** SPSS physical file, and the LIBNAME statement associates the libref with the SPSS engine. The PRINT procedure prints the data from the save file.

```
filename mylib2 '/users/myid/mydir/aspssx.por';
libname mylib2 spss;
proc print data=mylib2._first_;
run;
```

Support for Links in UNIX Environments

SAS provides limited support for hard links and symbolic links in UNIX environments. You can create links that point to a SAS data set or SAS catalog. If you reference the link in a SAS program, SAS will follow the link to find the data set or catalog.

For example, you can create a symbolic link in the **/tmp** directory to the **/home/user/mydata.sas7bdat** data set by typing the following command at the UNIX prompt:

```
ln -s /home/user/mydata.sas7bdat /tmp/mydata.sas7bdat
```

The following SAS code uses the symbolic link in the **/tmp** directory to find the mydata.sas7bdat data set. This code does not change the symbolic link, but it does sort the data in the data set.

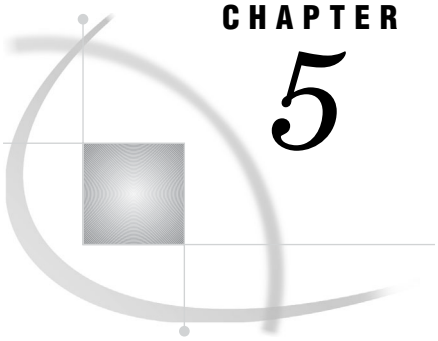
```
libname tmp '/tmp';

proc sort data=tmp.mydata;
   by myvariable;
run;
```

If you are running in the SAS windowing environment, you can use the Explorer window to view the symbolic links that are stored within a specific directory. Any symbolic link that points to a nonexistent SAS file will have a file size of **0.0KB** and a modified date of **31DEC59:19:00:00**.

Note: SAS does not support links for a data set that has any of the following:

- an index
- version data sets.

△



CHAPTER

5

Using External Files and Devices

Introduction to External Files and Devices in UNIX Environments

At times during a SAS session, you might want to use *external files*, that is, files that contain data or text or files in which you want to store data or text. These files are created and maintained by the operating system, not by SAS.

You can use external files in a SAS session to

- hold raw data to be read with the INPUT statements
- store printed reports created by a SAS procedure
- submit a file containing SAS statements for processing
- store data written with PUT statements.

For SAS, external files and devices can serve both as sources of input and as receivers of output. The input can be either raw data to be read in a DATA step or SAS statements to be processed by SAS. The output can be

- the SAS log, which contains notes and messages produced by the program
- the formatted output of SAS procedures
- data written with PUT statements in a DATA step.

You might also want to use peripheral devices such as a printer, plotter, or your own terminal. UNIX treats these I/O devices as if they were files. Each device is associated with a file, called a *special file*, which is treated as an ordinary disk file. When you write to a special file, the associated device is automatically activated. All special files reside in the **dev** directory or its subdirectories. Although there are some differences in how you use the various devices, the basic concept is the same for them all.

UNIX also enables you to use pipes to send data to and from operating system commands as if they were I/O devices.

If you need to access an external file containing a transport data library, refer to *Moving and Accessing SAS Files*.

Accessing an External File or Device in UNIX Environments

To access an external file or device, you will need to specify its pathname or fileref in the appropriate SAS statements:

FILE
specifies the current output file for PUT statements.

%INCLUDE
includes a file containing SAS source statements into the Program Editor.

INFILE
identifies an external file that you want to read with an INPUT statement.

In the SAS statement, refer to the file or device in one of two ways:

- specify the pathnames for the files. For more information, see "Specifying Pathnames in UNIX Environments" on page 133.
- assign a fileref to a device, one or more files, or a directory and use the fileref when you want to refer to the file or device.

In most cases, you will want to use a fileref.

What Is a Fileref?

A fileref is nickname that you assign to a file or device. You simply assign the fileref once, and then use it as needed. Filerefs are especially useful when

- the pathname is long and has to be specified several times within a program
- the pathname might change. If the pathname changes, you need to change only the statement that assigns the fileref, not every reference to the file.

You can assign filerefs in the File Shortcuts window of the Explorer, with the FILENAME statement, with the FILENAME function,* or by defining the fileref as an environment variable.

Specifying Pathnames in UNIX Environments

Rules for Specifying Pathnames

You can reference an external file directly by specifying its pathname in the FILE, INFILE, or %INCLUDE statements, or you can reference the file indirectly by specifying a fileref and a pathname in the FILENAME statement and then using the fileref in the FILE, INFILE, or %INCLUDE statements.

Whether you reference a file directly or indirectly, you will need to specify its pathname in the appropriate statement. In most cases, you must enclose the name in quotation marks. For example, the following INFILE statement refers to the file **/users/pat/cars**:

```
infile '/users/pat/cars';
```

* For a complete description of the FILENAME statement and the FILENAME function, see *SAS Language Reference: Dictionary*.

The following FILE statement directs output to the specified terminal:

```
file '/dev/ttyp1';
```

The level of specification depends on your current directory. You can use the character substitutions shown in Table 4.6 on page 115 to specify the pathname. You can also use wildcards as described in "Using Wildcards in Pathnames (Input Only)" on page 134.

Exceptions to Enclosing the Filename in Quotation Marks

You can omit the quotation marks on a filename if

- there is not already a fileref defined with that filename.
- the file has the filename extension expected by the statement that you are using to refer to the file. If you do not enclose a filename in quotation marks, the FILE and INFILE statements assume a file extension of .dat, and the %INCLUDE statement assumes a file extension of .sas.
- the file is in the current directory.
- the filename is all lower case characters.

For example, if the current directory is **/users/mkt/report** and it includes file Qtr.sas, you can reference Qtr.sas in any of the following statements:

```
%include '/users/mkt/report/qtr.sas';
%include 'qtr.sas';
file 'qtr.sas';
```

If there is no Qtr fileref already defined, you can omit the quotation marks and the filename extension on the %INCLUDE statement:

```
%include qtr;
```

Using Wildcards in Pathnames (Input Only)

Descriptions of the Valid Wildcards

You can use the *, ?, and [ ]wildcards to specify pathnames in the FILENAME (only if the fileref is to be used for input), INFILE, and %INCLUDE statements and the INCLUDE command.

*	matches one or more characters, except for the period at the beginning of filenames.
?	matches any single character.
[ ]	matches any single character from the set of characters defined within the brackets. You can specify a range of characters by specifying the starting character and ending character separated by a hyphen.

Wildcards are supported for input only. You cannot use wildcards in the FILE statement.

Example 1: Selecting Files By Including a Wildcard in a String

The following example reads input from every file in the current directory that begins with the string **wild** and ends with .dat:

```
filename wild 'wild*.dat';
data;
   infile wild;
   input;
run;
```

Example 2: Reading Each File in the Current Directory

The following example reads input from every file in every subdirectory of the current working directory:

```
filename subfiles '*/*';
data;
   infile subfiles;
   input;
run;
```

If new files are added to any of the subdirectories, they can be accessed with the Subfiles fileref without changing the FILENAME statement.

Example 3: Wildcards in Filenames When Using Aggregate Syntax

You can also use wildcards in filenames, but not in directory names, when you use aggregate syntax:

```
filename curdir ".";
data;
  infile curdir('wild*');
  input;
run;
```

In the example above, the period in the FILENAME statement refers to the current directory. See Table 4.6 on page 115 for information about other character substitutions available on UNIX.

Example 4: Associating a Fileref with Multiple Files

The following statement associates the fileref MyRef with all files that begin with alphabetic characters. Files beginning with numbers or other characters such as the period or tilde are excluded.

```
filename myref '[a-zA-Z]*.dat';
```

The following statement associates MyRef with any file beginning with Sales (in either uppercase, lowercase, or mixed case) and a year between 1990 and 1999:

```
filename myref '[Ss][Aa][Ll][Ee][Ss]199[0-9].dat';
```

Assigning Filerefs to External Files or Devices with the FILENAME Statement

Introduction to the FILENAME Statement

The most common way to assign a fileref to an external file or device is with the FILENAME statement. There are several forms of the FILENAME statement,

depending on the type of device you want to access. For more information, see "FILENAME Statement" on page 293.

Accessing DISK Files

The most common use of the FILENAME statement is to access DISK files. The FILENAME syntax for a DISK file is

FILENAME *fileref* <DISK> *'pathname'* <*options*>;

The following FILENAME statement associates the fileref MyFile with the external file **/users/mydir/myfile**, which is stored on a disk device:

```
filename myfile disk '/users/mydir/myfile';
```

The following FILENAME statement assigns a fileref of Prices to the file **/users/pat/cars**. The FILE statement then refers to the file using the fileref:

```
filename prices '/users/pat/cars';
data current.list;
   file prices;
   ...PUT statements...
run;
```

See "Concatenating Filenames in UNIX Environments" on page 138 for more information about using DISK files.

Debugging Code With DUMMY Devices

You can substitute the DUMMY device type for any of the other device types. This device type serves as a tool for debugging your SAS code without actually reading or writing to the device. After debugging is complete, replace the DUMMY device name with the proper device type, and your program will access the specified device type.

The FILENAME syntax for a DUMMY file is

FILENAME *fileref* DUMMY *'pathname'*<*options*>;

Output to DUMMY devices is discarded.

Sending Output to PRINTER Devices

The PRINTER device type enables you to send output directly to a printer. The FILENAME syntax to direct a file to a PRINTER is

FILENAME *fileref* PRINTER '<*printer*> <*printer-options*>' <*options*>;

For example, this SAS program sends the output file to the BLDG3 printer:

```
filename myfile printer 'bldg3';

data test;
   file myfile;
   put 'This will appear in bldg3 .';
run;
```

See "Using PRTFILE and PRINT with a Fileref" on page 160 and "Using the PRINTTO Procedure in UNIX Environments" on page 161 for more information.

Using Temporary Files (TEMP Device Type)

The TEMP device type associates a fileref with a temporary file stored in the same directory as the Work data library. (See "Work Data Library" on page 120.) Using the TEMP device type enables you to create a file that lasts only as long as the SAS session.

The FILENAME syntax for a TEMP file is

FILENAME *fileref* TEMP <*options*>;

For example, this FILENAME statement associates Tmp1 with a temporary file:

```
filename tmp1 temp;
```

Accessing TERMINAL Devices Directly

To access a terminal directly, use the TERMINAL device type. The FILENAME syntax to associate a file with a terminal is

FILENAME *fileref* TERMINAL <*'terminal-pathname'*> <*options*>;

The *terminal-pathname* must be a pathname of the special file associated with the terminal. Check with your system administrator for details. Enclose the name in quotation marks. If you omit the terminal pathname, the fileref is assigned to your terminal.

For example, this FILENAME statement associates the fileref Here with your terminal:

```
filename here terminal;
```

The following FILENAME statement associates the fileref ThatFile with another terminal:

```
filename thatfile terminal '/dev/tty3';
```

Assigning Filerefs to Files on Other Systems (FTP and SOCKET Access Types)

You can access files on other systems in your network by using the SOCKET and FTP access methods. The forms of the FILENAME statement are

FILENAME *fileref* FTP *'external-file'* <*ftp-options*>;

FILENAME *fileref* SOCKET *'external-file'* <*tcpip-options*>;

FILENAME *fileref* SOCKET *':portno'* SERVER <*tcpip-options*>;

These access methods are documented in *SAS Language Reference: Dictionary*. Under UNIX, the FTP access method supports an additional option:

MACH=*'machine'*
: identifies which entry in the **.netrc** file should be used to get the username and password. Consult the UNIX man page for more information about the **.netrc** file. You cannot specify the MACH option together with the HOST option in the FILENAME statement.

If you are transferring a file to UNIX from the z/OS operating environment and you want to use either the S370V or S370VB format to access that file, then the file must be of type RECFM=U and BLKSIZE=32760 before you transfer it.

CAUTION:
When you use the FTP access method to create a remote file, the UNIX permissions for that file are set to *-rw-rw-rw-*, which makes the file world-readable and world-writable. See the man page for **`chmod`** for information about changing file permissions. △

Concatenating Filenames in UNIX Environments

You can concatenate filenames in the FILENAME, %INCLUDE, and INFILE statements. Concatenating filenames enables you to read those files sequentially.

FILENAME *fileref* ("*pathname-1*" ... "*pathname-n*");

%INCLUDE '("*filename-1*" ... "*filename-n*")';

%INCLUDE "('*filename-1*' ... '*filename-n*')";

INFILE '("*filename-1*" ... "*filename-n*")';

INFILE "('*filename-1*' ... '*filename-n*')";

You can enclose the pathnames in single or double quotation marks and separate them with commas or blank spaces. You can use the characters shown in Table 4.6 on page 115 and the wildcards described in "Using Wildcards in Pathnames (Input Only)" on page 134 to specify the pathnames.

Assigning a Fileref to a Directory (Using Aggregate Syntax)

Introduction to Aggregate Syntax

Aggregate syntax enables you to assign a fileref to a directory and then work with any file in that directory by specifying its filename in parentheses after the fileref.

FILENAME *fileref directory-name*;

Aggregate syntax is especially useful when you have to refer to several files in one directory.

Example 1: Referring to a File Using Aggregate Syntax

To refer to a file in the directory, specify the fileref followed by the individual filename in parentheses. For example, you can refer to the file Cars.dat in the directory **`/users/pat`** as shown in this example:

```
filename prices '/users/pat';
data current.list;
   file prices(cars);
   ...other SAS statements...
run;
```

Example 2: Using Aggregate Syntax with Filerefs Defined by Environment Variables

You can also use aggregate syntax with filerefs that have been defined using environment variables (see "Using Environment Variables to Assign Filerefs in UNIX Environments" on page 139). For example:

```
x setenv PRICES /users/pat;
data current.list;
   file prices(cars);
   ...other SAS statements...
run;
```

Assigning a Fileref to Several Directories

In the FILENAME statement, you can concatenate directory names and use the fileref to refer to any file within those directories:

FILENAME *fileref* ("*directory-1*" ... "*directory-n*");

When you concatenate directory names, you can use aggregate syntax to refer to a file in one of the directories. For example, assume that the Report.sas file resides in the directory associated with the MYPROGS environment variable. When SAS executes the following code, it searches for Report.sas in the pathnames that are specified in the FILENAME statement and it executes the program.

```
filename progs ("$MYPROGS" "/users/mkt/progs");
%inc progs(report);
```

SAS searches the pathnames in the order specified in the FILENAME statement until

- it finds the first file with the specified name. Even if you use wildcards (see "Using Wildcards in Pathnames (Input Only)" on page 134) in the filename, SAS matches only one file.
- it encounters a filename in the list of pathnames that you specified in the FILENAME statement.

Using Environment Variables to Assign Filerefs in UNIX Environments

Requirements for Variable Names

An environment variable can also be used as a fileref to refer to DISK files. The variable name must be in all uppercase characters, and the variable value must be the full pathname of the external file; that is, the filename must begin with a slash.

Note: If a variable and a fileref have the same name but refer to different files, SAS uses the fileref. For example, the %INCLUDE statement below refers to file **/users/myid/this_one**. △

```
filename ABC '/users/myid/this_one';
x setenv ABC /users/myid/that_one;
%include ABC;
```

Reading a Data File

Suppose that you want to read the data file **/users/myid/educ.dat**, but you want to refer to it with the INED environment variable. You can define the variable at two times:

- before you invoke SAS. See "Defining Environment Variables in UNIX Environments" on page 21. For example, in the Korn shell, you use

```
export INED=/users/myid/educ.dat
```

- after you invoke SAS by using the X statement (see "Executing Operating System Commands from Your SAS Session" on page 13) and the SAS **setenv** command:

```
x setenv INED /users/myid/educ.dat;
```

After INED is associated with the file **/users/myid/educ.dat**, you can use INED as a fileref to refer to the file in the INFILE statement:

```
infile ined;
```

Writing to an External File

The same method applies if you want to write to an external file. For example, you can define OUTFILE before you invoke SAS:

```
OUTFILE=/users/myid/scores.dat
export OUTFILE
```

Then, use the environment variable name as a fileref to refer to the file:

```
file outfile;
```

Filerefs Assigned by SAS in UNIX Environments

Filerefs for Standard Input, Standard Output, and Standard Error

Often a command's arguments or options tell the command what to use for input and output, but in case they do not, the shell supplies you with three standard files: one for input (*standard input*), one for output (*standard output*), and one for error messages (*standard error*). By default, these files are all associated with your terminal: standard input with your keyboard, and both standard output and standard error with your terminal's display. When you invoke SAS, it assigns a fileref to each file that it opens, including the three standard files. SAS assigns the filerefs Stdin, Stdout, and Stderr to standard input, standard output, and standard error, respectively.

What Is a File Descriptor?

Each file has an internal *file descriptor* assigned to it. By default, 0 is the file descriptor for standard input, 1 is the file descriptor for standard output, and 2 is the file descriptor for standard error. As other files are opened, they get other file descriptors. In the Bourne shell and in the Korn shell, you can specify that data be written to or be read from a file using the file descriptor as described in "File Descriptors in the Bourne and Korn Shells" on page 140.

File Descriptors in the Bourne and Korn Shells

If you are using the Bourne shell or the Korn shell, SAS assigns filerefs of the following form to files that have a file descriptor (see "Filerefs Assigned by SAS in UNIX Environments" on page 140) larger than 2.

```
FILDES number
```

number is a two-digit representation of the file descriptor. You can use these filerefs in your SAS applications.

For example, if you invoke SAS with the following command, then the operating environment opens the file Sales_Data and assigns file descriptor 4 to it:

```
sas salespgm 4< sales_data
```

SAS assigns the fileref FILDES04 to the file and executes the application **salespgm**. When the application reads input from FILDES04, it reads the file Sales_Data. Using file descriptors as filerefs enables you to use the same application to process data from different files without changing the application to refer to each file. In the command that you use to invoke the application, you simply assign the appropriate file descriptor to the file to be processed.

Reserved Filerefs in UNIX Environments

The following filerefs are reserved.

DATALINES fileref in the INFILE statement
: specifies that input data immediately follow a DATALINES statement. You need to use INFILE DATALINES only when you want to specify options in the INFILE statement to read instream data.

LOG fileref in the FILE statement
: specifies that output lines produced by PUT statements be written to the SAS log. LOG is the default destination for output lines.

PRINT fileref in the FILE statement
: specifies that output lines produced by PUT statements be written to the same print file as output produced by SAS procedures.

Reading from and Writing to UNIX Commands (PIPE)

What Are Pipes?

Pipes enable your SAS application to receive input from any UNIX command that writes to standard output and to route output to any UNIX command that reads from standard input. In UNIX commands, the pipe is represented by a vertical bar (|). For example, to find the number of files in your directory, you could redirect the output of the **ls** command through a pipe to the **wc** (word count) command by entering

```
ls | wc -w
```

Syntax of the FILENAME Statement to Assign a Fileref to a Pipe

Under UNIX, you can use the FILENAME statement to assign filerefs not only to external files and I/O devices, but also to a pipe. The syntax of the FILENAME statement is

FILENAME *fileref* PIPE *'UNIX-command'* <*options*>;

fileref
is the name by which you reference the pipe from SAS.

PIPE
identifies the device-type as a UNIX pipe.

'*UNIX-command*'
is the name of a UNIX command, executable program, or shell script to which you want to route output or from which you want to read input. The command(s) must be enclosed in either double or single quotation marks.

options
control how the external file is processed. See "FILENAME Statement" on page 293 for an explanation of these options.

Whether you are using the command as input or output depends on whether you use the *fileref* in a reading or writing operation. For example, if the fileref is used in an INFILE statement, then SAS assumes that the input comes from a UNIX command; if the fileref is used in a FILE statement, then SAS assumes that the output goes to a UNIX command.

Using the Fileref for Reading

When the fileref is used for reading, the specified UNIX command executes, and any output sent to its standard output or standard error is read through the fileref. In this case, the standard input of the command is connected to **/dev/null**.

Example 1: Sending the Output of the Process Command to a SAS DATA Step

The following SAS program uses the PIPE device-type keyword to send the output of the **ps** (process) command to a SAS DATA step. The resulting SAS data set contains data about every process currently running SAS:

```
filename ps_list pipe "ps -e|grep 'sas'";
data sasjobs;
   infile ps_list;
   length process $ 80;
   input process $ char80.;
run;
proc print data=sasjobs;
run;
```

The **ps -e** command produces a listing of all active processes on the system, including the name of the command that started the task. In BSD-based UNIX systems, you use the **ps -ax** command.

The operating environment uses pipes to send the output from **ps** to the **grep** command, which searches for every occurrence of the string **'sas'**. The FILENAME statement connects the output of the **grep** command to the fileref Ps_List. The DATA step then creates a data set named SasJobs from the INFILE statement that points to the input source. The INPUT statement reads the first 80 characters on each input line.

Example 2: Using the Stdin Fileref to Read Input

In the next example, the Stdin fileref is used to read input through a pipe into the SAS command which in turn executes the SAS program. By placing the piping

operation outside the SAS program, the program becomes more general. The program in the previous example has been changed and stored in file Ps.sas:

```
data sasjobs;
   infile stdin;
   length process $ 80;
   input process $ char80.;
run;
proc print data=sasjobs;
run;
```

To run the program, use pipes to send the output of **ps** to **grep** and from **grep** into the SAS command:

```
ps -e|grep 'sas'|sas ps.sas &
```

The output will be stored in Ps.lst; the log in Ps.log as described in "The Default Routings for the SAS Log and Procedure Output in UNIX Environments" on page 155.

Using the Fileref for Writing

When the fileref is used for writing, the output from SAS is read in by the specified UNIX command, which then executes.

Example 1: Sending Mail Using Pipes

In this example, any data sent to the Mail fileref are piped to the **mail** command and sent to user PAT:

```
filename mail pipe 'mail pat';
```

Example 2: Starting a Remote Shell and Printing Output

Consider this FILENAME statement:

```
filename letterq pipe 'remsh alpha lp -dbldga3';
```

Any data sent to the LetterQ fileref are passed to the UNIX command, which starts a remote shell on the machine named Alpha.* The shell then prints the LetterQ output on the printer identified by the destination BldgA3. Any messages produced by the **lp** command are sent to the SAS log.

Sending Electronic Mail Using the FILENAME Statement (EMAIL)

Advantages of Sending Electronic Mail from within SAS

SAS lets you send electronic mail using SAS functions in a DATA step or in SCL. Sending e-mail from within SAS enables you to

- use the logic of the DATA step or SCL to subset e-mail distribution based on a large data set of e-mail addresses.
- send e-mail automatically upon completion of a SAS program that you submitted for batch processing.

* The form of the command that starts a remote shell varies among the various UNIX operating systems.

- direct output through e-mail based on the results of processing.
- send e-mail messages from within a SAS/AF frame application, customizing the user interface.

Initializing Electronic Mail

By default, SAS uses SMTP (Simple Mail Transfer Protocol) to send e-mail. SMTP, unlike some external scripts, supports attachments. This default is specified by the EMAILSYS system option. For information about how to change the e-mail protocol, see "EMAILSYS System Option" on page 326.

Before you can send e-mail from within SAS, your system administrator might need to set the EMAILHOST system option to point to the SMTP server. For more information about the EMAILHOST system option, see *SAS Language Reference: Dictionary*.

Components of the DATA Step or SCL Code Use to Send E-Mail

In general, a DATA step or SCL code that sends electronic mail has the following components:

- a FILENAME statement with the EMAIL device-type keyword
- options specified on the FILENAME or FILE statements indicating the e-mail recipients, subject, and any attached files
- PUT statements that contain the body of the message
- PUT statements that contain special e-mail directives (of the form !EM_*directive*!) that can override the e-mail attributes (TO, CC, BCC, SUBJECT, ATTACH) or perform actions (such as SEND, ABORT, and start a NEWMSG).

Syntax of the FILENAME Statement for Electronic Mail

To send electronic mail from a DATA step or SCL, issue a FILENAME statement of the following form:

FILENAME *fileref* EMAIL '*address*' <*email-options*>;

The FILENAME statement accepts the following *email-options*:

fileref
is a valid fileref.

'*address*'
is the destination e-mail address of the user to which you want to send e-mail. You must specify an address here, but you can override its value with the TO e-mail option.

email-options
can be any of the following:

TO=*to-address*
specifies the primary recipients of the electronic mail. If an address contains more than one word, enclose it in quotation marks. To specify more than one address, enclose the group of addresses in parentheses, enclose each address in quotation marks, and separate each address with a space. For example, **`to='joe@someplace.org'`** and **`to=("joe@smplc.org" "jane@diffplc.org")`** are valid TO values.

Note: You can send an e-mail without specifying a recipient in the TO= option as long as you specify a recipient in either the CC= or BCC= option. △

CC=*cc-address*
specifies the recipients you want to receive a copy of the electronic mail. If an address contains more than one word, enclose it in quotation marks. To specify more than one address, enclose the group of addresses in parentheses, enclose each address in quotation marks, and separate each address with a space. For example, **cc='joe@someplace.org'** and **cc=("joe@smplc.org" "jane@diffplc.org")** are valid CC values.

BCC=*bcc-address*
specifies the recipients you want to receive a blind copy of the electronic mail. Individuals listed in the **bcc** field will receive a copy of the e-mail. The BCC field does not appear in the e-mail header, so that these e-mail addresses cannot be viewed by other recipients.

If a BCC address contains more than one word, enclose it in quotation marks. To specify more than one address, enclose the group of addresses in parentheses, enclose each address in quotation marks, and separate each address with a space. For example, **bcc='joe@someplace.org'** and **bcc=("joe@smplc.org" "jane@diffplc.org")** are valid BCC values.

SUBJECT='*subject*'
specifies the subject of the message. If the subject text is longer than one word (that is, it contains at least one blank space), you must enclose it in quotation marks. You also must use quotation marks if the subject contains any special characters. For example, **subject=Sales** and **subject='June Report'** are valid subjects. Any subject not enclosed in quotation marks is converted to upper case.

ATTACH='*filename.ext*' | ATTACH = ('*filename.ext*' <*attachment-options*>)
specifies the physical name of the file(s) to be attached to the message and any options to modify attachment specifications. Enclose *filename.ext* in quotation marks. To attach more than one file, enclose the group of filenames in parentheses. For example, **attach='/u/userid/opinion.txt'** and **attach=("june98.txt" "july98.txt")** are valid file attachments.

By default, SMTP e-mail attachments are truncated at 256 characters. To send longer attachments, you can specify the LRECL= and RECFM= options from the FILENAME statement as the *attachment-options*. For more information about the LRECL= and RECFM= options, see "FILENAME Statement" on page 293.

For more information about the options that are valid when you are using SMTP, see "FILENAME Statement, EMAIL (SMTP) Access Method" in *SAS Language Reference: Dictionary*.

Specifying E-mail Options in the FILE Statement

You can also specify the *email-options* in the FILE statement inside the DATA step. Options that you specify in the FILE statement override any corresponding options that you specified in the FILENAME statement.

Defining the Body of the Message

In your DATA step, after using the FILE statement to define your e-mail fileref as the output destination, use PUT statements to define the body of the message.

Specifying E-mail Directives in the PUT Statement

You can also use PUT statements to specify e-mail directives that change the attributes of your electronic message or perform actions with it. Specify only one directive in each PUT statement; each PUT statement can contain only the text associated with the directive it specifies.

The following are the directives that change the attributes of your message:

!EM_TO! *addresses*
Replace the current primary recipient addresses with *addresses*. In the PUT statement, specify *addresses* without single quotation marks.

!EM_CC! *addresses*
Replace the current copied recipient addresses with *addresses*. In the PUT statement, specify *addresses* without single quotation marks.

!EM_BCC! *addresses*
Replace the current blind copied recipient addresses with *addresses*. In the PUT statement, specify *addresses* without single quotation marks.

!EM_SUBJECT! *subject*
Replace the current subject of the message with *subject*.

!EM_ATTACH! *pathname*
Replace the names of any attached files with *pathname*.

The following are the directives that perform actions:

!EM_SEND!
Sends the message with the current attributes. By default, SAS sends a message when the fileref is closed. The fileref closes when the next FILE statement is encountered or the DATA step ends. If you use this directive, SAS sends the message when it encounters the directive, *and* again at the end of the DATA step.

!EM_ABORT!
Aborts the current message. You can use this directive to stop SAS from automatically sending the message at the end of the DATA step.

!EM_NEWMSG!
Clears all attributes of the current message, including TO, CC, SUBJECT, ATTACH, and the message body.

Example: Sending E-mail from the DATA Step

Suppose that you want to share a copy of your Config.sas file with your coworker Jim, whose user ID is JBrown. If your e-mail program handles alias names and attachments, you could send it by submitting the following DATA step:

```
filename mymail email 'JBrown'
         subject='My CONFIG.SAS file'
         attach='config.sas';

data _null_;
  file mymail;
  put 'Jim,';
  put 'This is my CONFIG.SAS file.';
  put 'I think you might like the
       new options I added.';
run;
```

The following example sends a message and two attached files to multiple recipients. It specifies the e-mail options in the FILE statement instead of the FILENAME statement:

```
filename outbox email 'ron@acme.com';

data _null_;
  file outbox

        /* Overrides value in filename statement */
     to=('ron@acme.com' 'lisa@acme.com')
     cc=('margaret@yourcomp.com'
         'lenny@laverne.abc.com')
     subject='My SAS output'
     attach=('results.out' 'code.sas')
     ;
  put 'Folks,';
  put 'Attached is my output from the
       SAS program I ran last night.';
  put 'It worked great!';
run;
```

You can use conditional logic in the DATA step to send multiple messages and control which recipients get which message. For example, suppose you want to send customized reports to members of two different departments. If your e-mail program handles alias names and attachments, your DATA step might look like the following:

```
filename reports email 'Jim';

data _null_;
  file reports;
  infile cards eof=lastobs;
  length name dept $ 21;
  input name dept;

     /* Assign the TO attribute      */
  put '!EM_TO!' name;

     /* Assign the SUBJECT attribute */
  put '!EM_SUBJECT! Report for ' dept;

  put name ',';
  put 'Here is the latest report for ' dept '.';

     /* ATTACH the appropriate report */
  if dept='marketing' then
      put '!EM_ATTACH! mktrept.txt';
  else

    put '!EM_ATTACH! devrept.txt';

     /* Send the message */
  put '!EM_SEND!';

    /* Clear the message attributes */
  put '!EM_NEWMSG!';
```

```
  return;

   /* Abort the message before the */
   /*   RUN statement causes it to */
   /*   be sent again.            */
lastobs: put '!EM_ABORT!';

  datalines;
Susan          marketing
Jim            marketing
Rita           development
Herb           development
;
run;
```

The resulting e-mail message and its attachments are dependent on the department to which the recipient belongs.

Note: You must use the !EM_NEWMSG! directive to clear the message attributes between recipients. The !EM_ABORT! directive prevents the message from being automatically sent at the end of the DATA step. △

Example: Sending E-mail Using SCL Code

The following example is the SCL code behind a frame entry design for e-mail. The frame entry includes several text entry fields that let the user enter information:

mailto	the user ID to send mail to
copyto	the user ID to copy (CC) the mail to
attach	the name of a file to attach
subject	the subject of the mail
line1	the text of the message

The frame entry also contains a pushbutton called SEND that causes this SCL code (marked by the **`send:`** label) to execute.

```
send:

   /* set up a fileref */
rc = filename('mailit','userid','email');

   /* if the fileref was successfully set up
      open the file to write to */
if rc = 0 then do;
      fid = fopen('mailit','o');
      if fid > 0 then do;

         /* fput statements are used to
            implement writing the
            mail and the components such as
            subject, who to mail to, etc. */
      fputrc1  = fput(fid,line1);
      rc = fwrite(fid);
```

```
      fputrc2  = fput(fid,'!EM_TO! '||mailto);
      rc = fwrite(fid);
      fputrc3  = fput(fid,'!EM_CC! '||copyto);
      rc = fwrite(fid);

      fputrc4  = fput(fid,'!EM_ATTACH! '||attach);
      rc = fwrite(fid);
      fputrc5  = fput(fid,'!EM_SUBJECT! '||subject);
      rc = fwrite(fid);

         closerc  = fclose(fid);
      end;
   end;
return;

cancel:
   call execcmd('end');
return;
```

Processing Files on TAPE in UNIX Environments

Introduction to Processing Tape Files

Tape devices are inherently slow and should be used on a regular basis only for archiving files or for transferring data from one system to another.

There are four UNIX commands that are frequently used to process tape files on UNIX:

mt	positions the tape (winds forward and rewinds). On AIX, this command is **tctl**.
dd	converts, reblocks, translates, and copies files.
cat	concatenates, copies, and prints files.
tar	saves and restores archive files.
remsh	connects to the specified host and executes the specified command.

For a complete description of these commands, refer to the man pages.

In addition, you will almost always need to use a no-rewind device and the SAS system option TAPECLOSE=LEAVE to get the results you want.

You can use either the TAPE device type or the PIPE device type to process tape files.

Using the TAPE Device Type

To use the TAPE device type, enter the FILENAME statement as follows:

FILENAME *fileref* TAPE *'tape-device-pathname'* <*options*>;

The *tape-device-pathname* is the pathname of the special file associated with the tape device. Check with your system administrator for details. Enclose the name in quotation marks.

For example, this FILENAME statement associates YR1999 with a file stored on a tape that is mounted on device **/dev/tp0**:

```
filename  yr1999 tape '/dev/tp0';
```

Using the PIPE Device Type

You can also use the PIPE device type together with UNIX **dd** command to process the tape:

FILENAME *fileref* **PIPE** '*UNIX-commands*';

UNIX-commands are the commands needed to process the tape.

Using the PIPE device type and the **dd** command can process the tape more efficiently than the TAPE device type, and it allows you to use remote tape drives. However, using UNIX commands in your application means that the application will have to be modified if it is ported to a non-UNIX environment.

For example, the following DATA step writes an external file to tape:

```
options tapeclose=leave;
x 'mt -t /dev/rmt/0mn rewind';
filename outtape pipe 'dd of=/dev/rmt/0mn 2> /dev/null';
data _null_;
   file outtape;
   put '1 one';
   put '2 two';
   put '3 three';
   put '4 four';
   put '5 five';
run;
```

The following DATA step reads the file from tape:

```
options tapeclose=leave;
x 'mt -t /dev/rmt/0mn rewind';
filename intape pipe 'dd if=/dev/rmt/0mn 2> /dev/null';
data numbers;
   infile intape pad;
   input digit word $8.;
run;
```

If the tape drive that you want to access is a remote tape drive, you can access the remote tape drive by adding **remsh *machine-name*** to the X and FILENAME statements. For example, if the remote machine name is **wizard**, then you could read and write tape files on **wizard** by modifying the X and FILENAME statements as follows:

```
x 'remsh wizard mt -t /dev/rmt/0mn rewind';
filename intape pipe 'remsh wizard \
                dd if=/dev/rmt/0mn 2> /dev/null';
```

Working with External Files Created on the Mainframe

There are three main points to remember when dealing with tapes on UNIX that were created on a mainframe:

- UNIX does not support IBM standard label tapes. IBM standard label tapes contain user data files and labels, which themselves are files on the tape. To process the user data files on these tapes, use a no-rewind device (such as **/dev/rmt/0mn**) and the **mt** command with the **fsf *count*** subcommand to position the tape to the desired user data file. The formula for calculating ***count*** is

  ```
  count = (3 x user_data_file_number) - 2
  ```

- UNIX does not support multivolume tapes. To process multivolume tapes on UNIX, the contents of each tape must be copied to disk using the **dd** command. After all of the tapes have been unloaded, you can use the **cat** command to concatenate all of the pieces in the correct order. You can then use SAS to process the concatenated file on disk.

- You must know the DCB characteristics of the file. The records in files that are created on a mainframe are not delimited with end-of-line characters, so you must specify the original DCB parameters on the INFILE or FILENAME statement. In the INFILE statement, specify the record length, record format, and block size with the LRECL, RECFM, and BLKSIZE host options. In the FILENAME statement, if you use the PIPE device-type and the **dd** command, you must also specify the block size with the **ibs** subcommand. For more information about host options on the INFILE statement, see "INFILE Statement" on page 299. For more information about the **ibs** subcommand, refer to the man page for the **dd** command.

Example: Multivolume, Standard Label Tapes

This example assumes the use of a no-rewind device and TAPECLOSE=LEAVE.

Suppose that you are given a two-reel, multivolume, standard label tape set containing a mainframe external file and told that the record length is 7 and the record format is fixed. You will need to unload the data portion of each tape into disk files, concatenate the two disk files, and process the resultant file.

Make sure that the first tape is in the tape drive, then use the **mt** command to rewind the tape, skip over the label file, and position the tape at the beginning of the user data file. In this case, the user data file that you want to access is the first (and only) user data file on the tape. To skip over the label and position the tape at the beginning of the user data file, use the **fsf *count*** subcommand. Using the formula in "Working with External Files Created on the Mainframe" on page 150, the **fsf** count value is 1.

```
mt -t /dev/rmt/0mn rewind
mt -t /dev/rmt/0mn fsf 1
dd if=/dev/rmt/0mn of=/tmp/tape1 ibs=7
```

Repeat this process with the second tape, then concatenate the two disk files into one file.

```
mt -t /dev/rmt/0mn rewind
mt -t /dev/rmt/0mn fsf 1
dd if=/dev/rmt/0mn of=/tmp/tape2 ibs=7

cat /tmp/file1 /tmp/file2 > /tmp/ebcdic.numbers
```

You can then use the following DATA step to refer to the concatenated file (**/tmp/ebcdic.numbers**) and to convert the data using the appropriate EBCDIC informats:

```
filename ibmfile '/tmp/ebcdic.numbers';
data numbers;
   infile ibmfile lrecl=7 recfm=f;
   length digit 8 temp $ 1 word $ 6;
```

```
   input temp $ebcdic1. word $ebcdic6.;
   digit=input(temp,8.);
   drop temp;
run;
```

CHAPTER

6

Printing and Routing Output

Overview of Printing Output in UNIX Environments

When you print text or graphics, SAS needs to know where the output should go, how it should be written, and how the output should look. Universal Printing is the default printing mechanism in UNIX. Universal Printing generates both PostScript and PCL files in the SAS windowing environment. For more detailed information about Universal Printing, see *SAS Language Reference: Concepts*.

Forms printing is an older method of text printing available from SAS. It involves using the FORM subsystem, which consists of the Form window. For detailed information, see FORMS printing in *SAS Language Reference: Dictionary*.

If you are printing graphics, the output is controlled by native SAS/GRAPH drivers. Refer to the online help for SAS/GRAPH for detailed information about native SAS/GRAPH drivers.

Previewing Output in UNIX Environments

Previewing Output Using Universal Printing

With Universal Printing, you can preview your output before you send it to a printer, plotter, or external file. To preview your output, you first need to define a previewer for your system. For more information about Universal Printing, see *SAS Language Reference: Concepts*.

Previewing Output from within SAS/AF Applications

To preview output from within a SAS/AF application, use the DMPRTMODE and DMPRTPREVIEW commands to turn on preview mode, print the output, open the Print Preview dialog box, and then turn preview mode off. For example, the following code prints the GRAPH1 object using the host drivers and displays it in the Preview dialog box:

```
/* Turn on preview mode. */
CALL EXECCMDI ("DMPRTMODE PREVIEW");

/* Print the graph */
GRAPH1._PRINT_();

/* Open the Preview dialog box */
CALL EXECCMDI ("DMPRTPREVIEW");

/* Turn off preview mode */
CALL EXECCMDI ("DMPRTMODE NORMAL");
```

The Default Routings for the SAS Log and Procedure Output in UNIX Environments

For each SAS job or session, SAS automatically creates two types of output:

SAS log
contains information about the processing of SAS statements. As each program step executes, notes are written to the SAS log along with any applicable error or warning messages.

SAS output
is also called the *procedure output file* or *print file*. Whenever a SAS program executes a PROC step or a DATA step that produces printed output, SAS sends the output to the SAS output file.

Table 6.1 on page 155 shows the default routings of the SAS log and output files.

Table 6.1 Default Routings of the SAS Log and Output Files

Processing Mode	SAS Log File	SAS Output File
batch	***filename*.log**	***filename*.lst**
windowing environment	Log window	Output window
interactive line	terminal	terminal

By default, both the log file and the output file are written to your current directory. Your system administrator might have changed these default routings.

Changing the Default Routings in UNIX Environments

Techniques for Routing Output

There are four primary methods for routing your output:

- using the Print dialog box. The Print dialog box is available when you are using the SAS windowing environment.
- issuing windowing environment commands. The PRTFILE, PRINT, and FILE commands can be issued from any command line and can be used to send output to external files or to other devices defined with the FILENAME statement.
- using the PRINTTO procedure. You can use the PRINTTO procedure in any mode. Using the FILENAME statement with the PRINTTO procedure is the most flexible way of routing your output.
- using SAS system options, such as PRINT, LOG, ALTPRINT, or ALTLOG, to specify alternate destinations.

Determining Which Technique to Use When Changing the Routing

Use the following table to help you decide which method you should choose to change the routing.

Table 6.2 Decision Table: Changing the Default Destination

To route your SAS log or output to...	Using this mode of processing...	Use this method...	See...
a printer	any mode	FILENAME statement (UPRINTER or PRINTER device type) and PRINTTO procedure	"Using the PRINTTO Procedure in UNIX Environments" on page 161
	windowing environment	DMPRINT command	"Using the Print Dialog Box in UNIX Environments" on page 157
		Print dialog box	"Using the Print Dialog Box in UNIX Environments" on page 157
		FILENAME statement and PRTFILE, PRINT, and FILE commands	"Using PRTFILE and PRINT with a Fileref" on page 160
an external file	any mode	PRINTTO procedure and FILENAME statement	"Using the PRINTTO Procedure in UNIX Environments" on page 161
	windowing environment	Print dialog box	"Using the Print Dialog Box in UNIX Environments" on page 157
		FILENAME statement and PRTFILE, PRINT, and FILE commands	"Using PRTFILE and PRINT with a Fileref" on page 160
	batch	LOG and PRINT system options	"Using SAS System Options to Route Output" on page 163
a UNIX command (pipe)	any mode	FILENAME statement and PRINTTO procedure	"Using the PRINTTO Procedure in UNIX Environments" on page 161
	windowing environment	FILENAME statement and PRTFILE and PRINT commands	"Using PRTFILE and PRINT with a Fileref" on page 160
its usual location *and* to an external file	any mode	ALTLOG and ALTPRINT system options	"Using SAS System Options to Route Output" on page 163
	windowing environment	FILE command	"Using the FILE Command" on page 161

To route your SAS log or output to...	Using this mode of processing...	Use this method...	See...
		Print dialog box	"Using the Print Dialog Box in UNIX Environments" on page 157
a terminal	batch	FILENAME statement and PRINTTO procedure	"Routing Output to a Terminal" on page 163

Using the Print Dialog Box in UNIX Environments

Printing from Text Windows

Opening the Print Dialog Box

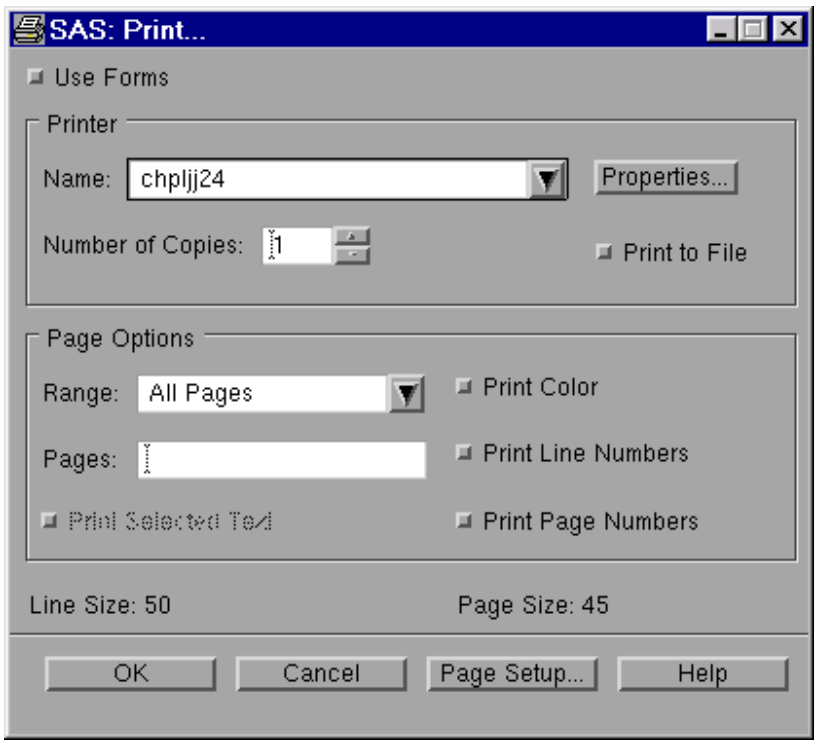
To print part or all of the contents of a window, complete the following steps:

1. Click in the window to make it the active window. If you want to mark and print only selected lines of text, mark the text before you open the Print dialog box.
2. Issue the DMPRINT command or select

   File ▶ Print

   to open the Print dialog box.

Display 6.1 Print Dialog Box



Default Printing Mode

In UNIX, the default printing mode is Universal Printing. For more information about how to use Universal Printing, click Help on the Print dialog box.

Specifics for Forms Printing

To use forms for printing, select **Use forms**. SAS prompts you to enter a spool command and the name of your system printer. When you click OK, SAS prints the contents of the active window using the command and printer name that you specified and additional information from your default form. See *SAS Language Reference: Dictionary* for more information about forms printing.

Troubleshooting Print Server Errors

After clicking OK, if SAS displays a clock icon for a long time and you are sending output to a network printer, your printer server might be down. If so, you will eventually see a message in the shell where you invoked your SAS session that indicates that the server is down.

Printing from GRAPH Windows

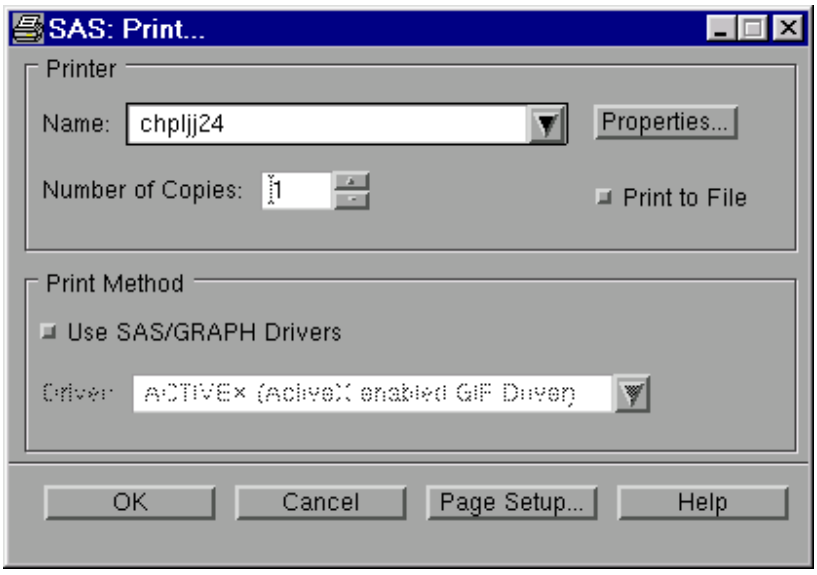
Opening the Print Dialog Box

With Universal Printing, you can use the Print dialog box to print the contents of a GRAPH window. Click in the window to make it the active window, and then issue the DMPRINT command or select

File ▶ Print

to open the Print dialog box.

Display 6.2 Print Dialog Box for Graphs



Note: Fonts set through the Print dialog box have no effect when you print from GRAPH windows. Make sure that you specify the correct options on a GOPTIONS statement. △

Specifics for SAS/GRAPH Drivers

To print output using a SAS/GRAPH driver, select **Use SAS/GRAPH Drivers**. Select the down arrow beside the **Driver** field to display the available drivers. Make sure that your printer destination has been set inside the device using the GDEVICE procedure or the GOPTIONS statement. For complete information about printing from GRAPH windows, refer to *SAS/GRAPH Reference, Volumes 1 and 2* and the online help for SAS/GRAPH.

Troubleshooting Print Server Errors

After clicking OK, if SAS displays a clock icon for a long time and you are sending output to a network printer, your printer server might be down. If so, you will eventually see a message in the shell where you invoked your SAS session that indicates that the server is down.

Using Commands to Print in UNIX Environments

Differences between the PRTFILE, PRINT, and FILE Commands

In the SAS windowing environment, you can use the PRTFILE, PRINT, and FILE commands to send the contents of the active window to an output device.

The following table lists the results of each of these commands.

Table 6.3 Routing Output Commands

Command	Action Performed
PRTFILE	specifies the filename or fileref for your output.
FILE	sends the contents of the active window to the filename or fileref you specify.
PRINT	sends the contents of the active window either: □ to your default printer when issued from the command line of the window □ to the location specified with the PRTFILE command.

Sending Output to a UNIX Command

If you want to send your output to a UNIX command, you can use the FILENAME statement. The FILENAME statement enables you to create filerefs that point to printers, plotters, or external files or filerefs that pipe to a UNIX command. For more information, see "FILENAME Statement" on page 293.

Specifying the Print File

When you issue the PRINT command, SAS sends your output to your default printer unless you specify a print file. You can specify a print file in two ways:

□ entering the PRTFILE command:

PRTFILE *file-spec* **CLEAR**|**APPEND**|**REPLACE**

The *file-spec* can be either a fileref or a filename.

- selecting

  File ► Print Utilities ► Set Print File

  and entering the name of the print file if you are using forms. This option is only available when Universal Printing is turned off.

Using PRTFILE and PRINT with a Fileref

You can use the PRTFILE command, followed by the PRINT command, to print the contents of windows. PRTFILE establishes the destination, and PRINT sends the contents of the window to that destination. If you do not specify a destination with the PRTFILE command, PRINT automatically sends the window contents to your default printer.

Steps for Sending Output Directly to a Printer

If you want to send output directly to a printer, you must first submit the FILENAME statement to assign a fileref to the PRINTER or PIPE device. For example, to print the contents of your OUTPUT window, complete the following steps:

Table 6.4 Printing the Contents of Your Output Window

Step	Action	Example
1	Submit a FILENAME statement or FILENAME function to associate a fileref with a system printer (PRINTER device type) or a UNIX command (PIPE device type). Enclose the printer name or UNIX command in either single or double quotation marks.	**filename myrpt printer 'bldga2';** or **filename ascout pipe 'lp -dmyljet';** For more information, see "Examples of FILENAME Statements Using PRINTER and PIPE" on page 160.
2	Issue the PRTFILE command as described in "Specifying the Print File" on page 159. Specify the fileref from your FILENAME statement or FILENAME function.	**prtfile myrpt**
3	Issue the PRINT command from the command line of the windows whose contents you want to print. If you are sending output to a system printer or if you are using forms-based printing, then you can print the contents of more than one window.	
4	Enter **A** in the requestor window that appears to warn you that the destination file already exists. The **A** value tells SAS to append the window contents to the destination file.	
5	Submit a FILENAME statement or FILENAME function to clear (deassign) the fileref.	**filename myrpt clear;**

To clear the print file setting, issue the PRTFILE CLEAR command.

Examples of FILENAME Statements Using PRINTER and PIPE

The following statement associates MyRpt with the system printer named BldgA2 and specifies two copies of every printout:

```
filename myrpt printer 'bldga2 -n2';
```

(See the documentation for your print command for information about other options that you can specify.)

The following statement enables you to print output using the **lp** command on the printer named myljet:

```
filename ascout pipe 'lp -dmyljet';
```

The following statement sends output to the **lp** command and redirects any error messages produced by this command to the LpError file in your home directory:

```
filename myrpt pipe 'lp 2>$HOME/lperror';
```

Note: Redirecting standard error is allowed only in the Bourne and Korn shells. △

If you frequently use the same print command and destination, you can add the appropriate FILENAME statement to your autoexec file. See "Customizing Your SAS Session Using System Options" on page 18 for more information.

Using the FILE Command

You can use the FILE command to copy the contents of many different windows to external files. Issue the FILE command on the command line of the window whose contents you want to copy. For example, to copy the contents of the LOG window to **/u/myid/log/app1**, issue the following command on the command line of the LOG window:

```
file '/u/myid/log/app1'
```

If the file does not exist, SAS creates it. If the file already exists, a requestor window asks you whether you want to replace it or to append data to the existing data.

If you have already associated a fileref with your external file, you can use the fileref instead of the filename:

```
file myref
```

Note: If you use the FILE command to save your output, carriage-control information is not saved (that is, page breaks are removed from the output). You might want to use the PRINT command with the FILE option instead:

PRINT FILE=*fileref* | *'pathname*'

△

Using the PRINTTO Procedure in UNIX Environments

Important Note about the PRINTTO Procedure

Any time you use PROC PRINTTO to route output, you must close the output device before PROC PRINTTO will release the output or log and send it to the destination you have specified. To close the output device, issue PROC PRINTTO without any parameters:

```
proc printto;
run;
```

Issuing PROC PRINTTO without any parameters closes the output device, generates output, and reroutes the log and procedure output to their default destinations. See Table 6.1 on page 155 for a list of the default destinations.

For more information, see "PRINTTO Procedure" on page 281 and *Base SAS Procedures Guide*.

Using the LOG= and PRINT= Options

When you use the PRINTTO procedure with its LOG= and PRINT= options, you can route the SAS log or SAS procedure output to an external file or a fileref from any mode. Specify the external file or the fileref in the PROC PRINTTO statement. The following example routes procedure output to **/u/myid/output/prog1**:

```
proc printto print='/u/myid/output/prog1' new;
run;
```

The NEW option causes any existing information in the file to be cleared. If you omit the NEW option from the PROC PRINTTO statement, the SAS log or procedure output is appended to the existing file.

If you plan to specify the same destination several times in your SAS program, you can assign a fileref to the file using a FILENAME statement. (See "Assigning Filerefs to External Files or Devices with the FILENAME Statement" on page 135 for details and examples.)

Routing Output to a Universal Printer

You can direct output directly to your Universal Printer by using the UPRINTER device type:

```
filename myoutput uprinter;
proc printto print=myoutput;
run;
```

Output will be sent to your default Universal Printer. This output will be in PostScript or PCL format.

Routing Output to a Printer

You can direct output directly to your system printer by using the PRINTER device type:

```
filename myoutput printer;
proc printto print=myoutput;
run;
```

Output will be sent to your default system printer or, if you have specified the SYSPRINT system option, to the printer specified with that option. This method will produce output in ASCII format.

Piping Output to a UNIX Command

You can also use the PIPE device type to send output to a UNIX command. When you specify the print command, you might also want to specify a destination for (redirect) any error messages produced by the print command. Enclose the UNIX

command in either single or double quotation marks. The following example associates the fileref MyOutput with the print command **lp**, which will send output to the printer named myljet:

```
filename myoutput pipe 'lp -dmyljet';
proc printto print=myoutput;
run;
```

You can send the SAS log to the same printer by using the LOG= option:

```
filename mylog pipe 'lp -dmyljet';
proc printto log=mylog;
run;
```

The log and procedure output continue to be routed to the designated external file until another PROC PRINTTO statement reroutes them.

Routing Output to a Terminal

In batch mode, you can direct output to a terminal by associating a fileref with a terminal and then using PROC PRINTTO to send output to that fileref. In the FILENAME statement, specify the TERMINAL device-type and the special file associated with the terminal. For example, the following statements send the SAS log to the terminal that is associated with the **/dev/tty3** special file:

```
filename term terminal '/dev/tty3';
proc printto log=term;
run;
```

Using SAS System Options to Route Output

Changing the Output Destination Using the LOG, PRINT, ALTLOG, and ALTPRINT System Options

You can use SAS system options to change the destination of the SAS log and procedure output. The options that you use depend on which task you want to accomplish:

- To route your SAS log or procedure output to an external file *instead of* to their default destinations, use the LOG and PRINT system options.
- To route the log or output to an external file *in addition* to their default destinations, use the ALTLOG and ALTPRINT system options. This method works in all modes of running SAS.

LOG and PRINT are normally used in batch and interactive line modes. These system options have no effect in the windowing environment. If you are running in the windowing environment, use the ALTLOG and ALTPRINT system options.

You can specify these options in following locations:

- the SAS command
- a configuration file
- the SASV9_OPTIONS environment variable.

For example, you could specify these options in the SAS command as follows:

```
sas -log '/u/myid/log' -print '/u/myid/prt'
sas -altlog '/u/myid/log' -altprint '/u/myid/prt'
```

See "Ways to Specify a SAS System Option" on page 18 for more information.

Creating Postscript Output with the PRINTCMD and SYSPRINT System Options

You can use the **pstext** UNIX command, the PRINTCMD and SYSPRINT system options, and the PRINT command to create PostScript output. The PRINTCMD option sets the UNIX print command that SAS will use, and the SYSPRINT option specifies a destination.

You can use the **pstext** command as your print command and redirect or pipe the output of that command. For example, the following options send your output through the **pstext** command and then redirect the output of that command to the file named **/tmp/file.ps**:

```
options printcmd='pstext';
options sysprint='>/tmp/file.ps';
```

When you issue the PRINT command, SAS creates the file **/tmp/file.ps**.

The following options send your output through the **pstext** command and then pipe the output of that command to the **lp** command:

```
options printcmd='pstext';
options sysprint='| lp -dmylaserjet';
```

When you issue the PRINT command from within SAS, the PostScript output is sent to the printer named mylaserjet.

Printing Large Files with the PIPE Device Type in UNIX Environments

When you print a file with the **lp** command, a symbolic link is created from the file to the **/usr/spool** directory. When you pipe output to the **lp** command, the output is copied under the **/usr/spool** directory.

If you experience problems printing large files using the PIPE device type, you can circumvent the problem in either of the following ways:

- save the print file to a disk file and then print it with the **lp** command. Issue the PRINT command from the output or log window, for example:

  ```
  print file='bigfile'
  ```

  Exit your SAS session and print the file, or use the SAS X command to print the file from within your SAS session, for example:

  ```
  x 'lp -dmylsrjt bigfile'
  ```

- create a fileref using the PIPE device type that can handle large files. For example, the following fileref saves the print file to disk, prints the saved file, and then removes the file:

  ```
  filename myfile pipe 'cat >bigfile;lp -dmylsrlt bigfile;rm bigfile;';
  ```

Changing the Default Print Destination in UNIX Environments

When you print a file, SAS looks in the following locations to determine where to send output. The locations are listed in order of precedence:

1. the destination specified in Universal Printing or the form printer device that you are using. See Universal Printing or forms printing in *SAS Language Reference: Dictionary* for more information.
2. the value specified in the SYSPRINT system option. You can use the SYSPRINT option to set your default print destination. Use the SYSPRINT system option to specify the destination option that is used with your print command. For example, if your print command is **lp**, you can set the default destination to the printer named myljet by entering the following OPTIONS statement:

   ```
   options sysprint='-dmyljet';
   ```

3. the value of the $LPDEST environment variable. See "Defining Environment Variables in UNIX Environments" on page 21 for more information.

SAS uses the first destination that it finds. If you specify a destination in all three locations, SAS uses the destination specified by Universal Printing.

Changing the Default Print Command in UNIX Environments

UNIX uses **lp** as the default print command. You can use the PRINTCMD system option to specify a different print command. For example, you can change your default print command to **lpr** by entering the following at SAS invocation:

```
sas -printcmd ''lpr''
```

You can also customize your default print command in your SAS configuration file. If you use this method, then you will not have to change the default print command every time you invoke SAS. For more information, see "PRINTCMD System Option" on page 352.

Controlling the Content and Appearance of Output in UNIX Environments

Some of the attributes of the SAS log and procedure output depend on the destination to which they are being sent. For example, if the log and output are being sent to your display, the default line and page size are derived from your display. If one or both of these files are sent to the system printer or written to a file, the default line size and page size depend on your printer and page setup. The line size and page size for your current settings can be seen in the Print dialog box.

Some of the attributes of the SAS log and procedure output depend on the mode in which you are running. For example, if you are running in interactive line mode, SAS source statements are not echoed to the SAS log. If you are using the SAS windowing environment all source statements are written to the log as they are submitted. In batch mode, the log and procedure output are formatted for a standard system printer.

See "Customizing Your SAS Session Using System Options" on page 18 for information about specifying system options.

SAS Log Options

Use the following options to control the contents of the log. See Chapter 17, "System Options under UNIX," on page 311 for details on specifying options.

FULLSTIMER
NOFULLSTIMER
: controls whether a list of resources (such as I/O performed, page faults, elapsed time, and CPU time) used for each PROC or DATA step is written to the log. NOFULLSTIMER is the default.

LINESIZE=*width*
: controls the line length used. *Width* can be any value from 64 to 256.

NEWS
NONEWS
: controls whether messages are written to the SAS log. NEWS is the default.

NOTES
NONOTES
: controls printing of NOTES on the log. NOTES is the default setting for all execution modes. Specify NOTES unless your SAS program is completely debugged.

PAGESIZE=*n*
: controls the number of lines that are printed on each page. *N* can be any number from 15 to 32767.

SOURCE
NOSOURCE
: controls whether SAS source statements are written to the log. NOSOURCE is the default setting in interactive line mode; otherwise, SOURCE is the default.

SOURCE2
NOSOURCE2
: controls whether SAS statements that are included with %INCLUDE statements are written to the log. NOSOURCE2 is the default setting for all execution modes.

STIMER
NOSTIMER
: controls whether user CPU time and elapsed time are written to the log. STIMER is the default.

Procedure Output Options

Use these system options to control the contents of the procedure output:

CENTER
NOCENTER
: controls whether the printed results are centered or left-aligned on the procedure output page. CENTER is the default.

DATE
NODATE
: controls whether the date is written at the top of each procedure output page. DATE is the default.

LINESIZE=*width*
: controls the line length used. *Width* can be any value from 64 to 256.

NUMBER
NONUMBER
controls whether the output page number is written on each procedure output page. NUMBER is the default.

PAGENO=*n*
resets the current page number in the print file. The default page number at the beginning of the SAS session is 1. The pages are numbered sequentially throughout the SAS session unless the PAGENO option is specified in an OPTIONS statement during the session.

PAGESIZE=*n*
controls the number of lines that are printed on each page. *N* can be any number from 15 to 32,767.

CHAPTER 7

Accessing Shared Executable Libraries from SAS

Overview of Shared Libraries in SAS

What Is a Shared Library?

Shared libraries in UNIX are libraries that contain executable programs that are written in any of several programming languages. In UNIX, the names of these programs typically end with a .so or .sl extension. However, they are not constrained to this naming convention.

Shared libraries are a mechanism for storing useful routines that might be needed by multiple applications. When an application needs a routine that resides in an external shared library, it loads the shared library, invokes the routine, and unloads the shared library upon completion.

Invoking Shared Libraries from within SAS

SAS provides routines and functions that let you invoke these external routines from within SAS. You can access the shared library routines from the DATA step, the IML procedure, and SCL code. You use the MODULE family of SAS call routines and functions (including MODULE, MODULEN, MODULEC, MODULEI, MODULEIN, and MODULEIC) to invoke a routine that resides in a shared library. This documentation refers to the MODULE family of call routines and functions generically as the MODULE function.

Steps for Accessing an External Shared Library

The following are steps for accessing an external shared library routine:

1. Create a text file that describes the shared library routine you want to access, including the arguments it expects and the values it returns (if any). This attribute file must be in a special format, as described in "The SASCBTBL Attribute Table" on page 170.
2. Use the FILENAME statement to assign the SASCBTBL fileref to the attribute file you created.
3. In a DATA step or SCL code, use a call routine or function (MODULE, MODULEN, or MODULEC) to invoke the shared library routine. The specific function you use depends on the type of expected return value (none, numeric, or character). (You can also use MODULEI, MODULEIN, or MODULEIC within a PROC IML step.) The MODULE functions are described in "MODULE Function" on page 251.

CAUTION:

Only experienced programmers should access external routines in shared libraries. By accessing a function in a shared library, you transfer processing control to the external function. If done improperly, or if the external function is not reliable, you might lose data, get unreliable results, or receive severe errors. △

The SASCBTBL Attribute Table

Introduction to the SASCBTBL Attribute Table

Because the MODULE function invokes an external routine that SAS knows nothing about, you must supply information about the routine's arguments so that the

MODULE function can validate them and convert them, if necessary. For example, suppose you want to invoke a routine that requires an integer as an argument. Because SAS uses floating point values for all of its numeric arguments, the floating point value must be converted to an integer before you invoke the external routine. The MODULE function looks for this attribute information in an attribute table that is referred to by the SASCBTBL fileref.

What Is the SASCBTBL Attribute Table?

The attribute table is a sequential text file that contains descriptions of the routines you can invoke with the MODULE function. The table defines how the MODULE function should interpret supplied arguments when it builds a parameter list to pass to the called routine.

The MODULE function locates the table by opening the file that is referenced by the SASCBTBL fileref. If you do not define this fileref, the MODULE function simply calls the requested shared library routine without altering the arguments.

CAUTION:
Using the MODULE function without defining an attribute table can cause SAS to crash, produce unexpected results, or result in severe errors. You need to use an attribute table for all external functions that you want to invoke. △

Syntax of the Attribute Table

The attribute table should contain the following:

- a description in a ROUTINE statement for each shared library routine you intend to call
- descriptions in ARG statements for each argument associated with that routine.

At any point in the attribute table file, you can create a comment using an asterisk (*) as the first nonblank character of a line or after the end of a statement (following the semicolon). You must end the comment with a semicolon.

ROUTINE Statement

The syntax of the ROUTINE statement is

ROUTINE *name* MINARG=*minarg* MAXARG=*maxarg*
<CALLSEQ=BYVALUE|BYADDR>
<TRANSPOSE=YES|NO> <MODULE=*shared-library-name*>
<RETURNS=SHORT|USHORT|LONG|ULONG
|DOUBLE|DBLPTR|CHAR<*n*>>

The following are descriptions of the ROUTINE statement attributes:

ROUTINE *name*
starts the ROUTINE statement. You need a ROUTINE statement for every shared library function you intend to call. The value for *name* must match the routine name or ordinal you specified as part of the '*module*' argument in the MODULE function, where *module* is the name of the shared library (if not specified by the MODULE attribute) and the routine name or ordinal. For example, in order to specify **`libc,getcwd`** in the MODULE function call, the ROUTINE *name* should be **`getcwd`**.

The *name* argument is case sensitive, and is required for the ROUTINE statement.

MINARG=*minarg*
specifies the minimum number of arguments to expect for the shared library routine. In most cases, this value will be the same as MAXARG; but some routines do allow a varying number of arguments. This is a required attribute.

MAXARG=*maxarg*
specifies the maximum number of arguments to expect for the shared library routine. This is a required attribute.

CALLSEQ=BYVALUE | BYADDR
indicates the calling sequence method used by the shared library routine. Specify BYVALUE for call-by-value and BYADDR for call-by-address. The default value is BYADDR.

FORTRAN and COBOL are call-by-address languages. C is usually call-by-value, although a specific routine might be implemented as call-by-address.

The MODULE function does not require that all arguments use the same calling method. You can identify any exceptions by using the BYVALUE and BYADDR options in the ARG statement.

TRANSPOSE=YES | NO
specifies whether SAS transposes matrices that have both more than one row and more than one column before it calls the shared library routine. This attribute applies only to routines called from within PROC IML with MODULEI, MODULEIC, and MODULEIN.

TRANSPOSE=YES is necessary when you are calling a routine that is written in a language that does not use row-major order to store matrices. (For example, FORTRAN uses column-major order.)

For example, consider this matrix with three columns and two rows:

```
            columns
            1   2   3
          --------------
 rows   1 | 10  11  12
        2 | 13  14  15
```

PROC IML stores this matrix in memory sequentially as 10, 11, 12, 13, 14, 15. However, FORTRAN routines will expect this matrix as 10, 13, 11, 14, 12, 15.

The default value is NO.

MODULE=*shared-library-name*
names the executable module (the shared library) in which the routine resides. You do not need to specify this attribute if the name of the shared library is the same name as the routine. If you specify the MODULE attribute here in the ROUTINE statement, then you do not need to include the module name in the *module* argument of the MODULE function (unless the shared library routine name you are calling is not unique in the attribute table). The MODULE function is described in “MODULE Function” on page 251.

You can have multiple ROUTINE statements that use the same MODULE name. You can also have duplicate routine names that reside in different shared libraries.

The MODULE function searches the directories that are defined in each operating system’s library path environment variable when it attempts to load the shared library argument provided in the MODULE attribute. The following table lists this environment variable for each UNIX operating system that SAS supports.

Table 7.1 Shared Library Environment Variable Name

Operating System	Environment Variable Name
Solaris	$LD_LIBRARY_PATH
AIX/R	$LIBPATH
HP-UX	$LD_LIBRARY_PATH or $SHLIB_PATH
Linux	$LD_LIBRARY_PATH

Note: For more information about these environment variables, see the man pages for your operating system. △

You can also use the PATH system option to point to the directory that contains the shared library specified in the MODULE= option. Using the PATH system option overrides your system's environment variable when you load the shared library. For more information, see "PATH System Option" on page 351.

RETURNS=SHORT | USHORT | LONG | ULONG | DOUBLE | DBLPTR | CHAR<*n*>

specifies the type of value that the shared library routine returns. This value will be converted as appropriate, depending on whether you use MODULEC (which returns a character) or MODULEN (which returns a number). The following are the possible return value types:

SHORT
: short integer.

USHORT
: unsigned short integer.

LONG
: long integer.

ULONG
: unsigned long integer.

DOUBLE
: double-precision floating point number.

DBLPTR
: pointer to a double-precision floating point number (instead of using a floating point register). Consult the documentation for your shared library routine to determine how it handles double-precision floating point values.

CHAR*n*
: pointer to a character string up to *n* bytes long. The string is expected to be null-terminated and will be blank-padded or truncated as appropriate. If you do not specify *n*, the MODULE function uses the maximum length of the receiving SAS character variable.

If you do not specify the RETURNS attribute, you should invoke the routine with only the MODULE and MODULEI call routines. You will get unpredictable values if you omit the RETURNS attribute and invoke the routine using the MODULEN/MODULEIN or MODULEC/MODULEIC functions.

ARG Statement

The ROUTINE statement must be followed by as many ARG statements as you specified in the MAXARG= option. The ARG statements must appear in the order that the arguments will be specified within the MODULE function.

The syntax for each ARG statement is

ARG *argnum* NUM|CHAR <INPUT|OUTPUT|UPDATE>
<NOTREQD|REQUIRED> <BYADDR|BYVALUE> <FDSTART>
<FORMAT=*format*>;

Here are the descriptions of the ARG statement attributes:

ARG *argnum*
defines the argument number. This a required attribute. Define the arguments in ascending order, starting with the first routine argument (ARG 1).

NUM | CHAR
defines the argument as numeric or character. This is a required attribute.

If you specify NUM here but pass the routine a character argument, the argument is converted using the standard numeric informat. If you specify CHAR here but pass the routine a numeric argument, the argument is converted using the BEST12 informat.

INPUT | OUTPUT | UPDATE
indicates the argument is either input to the routine, an output argument, or both. If you specify INPUT, the argument is converted and passed to the shared library routine. If you specify OUTPUT, the argument is *not* converted, but is updated with an outgoing value from the shared library routine. If you specify UPDATE, the argument is converted, passed to the shared library routine *and* updated with an outgoing value from the routine.

You can specify OUTPUT and UPDATE only with variable arguments (that is, no constants or expressions are allowed).

NOTREQD | REQUIRED
indicates whether the argument is required. If you specify NOTREQD, then the MODULE function can omit the argument. If other arguments follow the omitted argument, identify the omitted argument by including an extra comma as a placeholder. For example, to omit the second argument to routine XYZ, you would specify:

```
call module('XYZ',1,,3);
```

CAUTION:
Be careful when using NOTREQD; the shared library routine must not attempt to access the argument if it is not supplied in the call to MODULE. If the routine does attempt to access it, you might receive unexpected results or severe errors. △

The REQUIRED attribute indicates that the argument is required and cannot be omitted. REQUIRED is the default value.

BYADDR | BYVALUE
indicates whether the argument is passed by reference or by value.

BYADDR is the default value unless CALLSEQ=BYVALUE was specified in the ROUTINE statement, in which case BYVALUE is the default. Specify BYADDR when you are using a call-by-value routine that also has arguments to be passed by address.

FDSTART
indicates that the argument begins a block of values that are grouped into a structure whose pointer is passed as a single argument. Note that all subsequent arguments are treated as part of that structure until the MODULE function encounters another FDSTART argument.

FORMAT=*format*
names the format that presents the argument to the shared library routine. Any SAS supplied formats, PROC FORMAT style formats, or SAS/TOOLKIT formats are valid. Note that this format must have a corresponding valid informat if you specified the UPDATE or OUTPUT attribute for the argument.

The FORMAT= attribute is not required, but is recommended, since format specification is the primary purpose of the ARG statements in the attribute table.

CAUTION:
Using an incorrect format can produce invalid results, cause SAS to crash, or result in serious errors. △

The Importance of the Attribute Table

The MODULE function relies heavily on the accuracy of the information in the attribute table. If this information is incorrect, unpredictable results can occur (including a system crash).

Consider an example routine **xyz** that expects two arguments: an integer and a pointer. The integer is a code indicating what action takes place. For example, action 1 means that a 20-byte character string is written into the area that is pointed to by the second argument, the pointer.

Now suppose you call **xyz** using the MODULE function, but you indicate in the attribute table that the receiving character argument is only 10 characters long:

```
routine xyz minarg=2 maxarg=2;
arg 1 input num byvalue format=ib4.;
arg 2 output char format=$char10.;
```

Regardless of the value given by the LENGTH statement for the second argument to MODULE, MODULE passes a pointer to a 10-byte area to the **xyz** routine. If **xyz** writes 20 bytes at that location, the 10 bytes of memory following the string provided by MODULE are overwritten, causing unpredictable results:

```
data _null_;
    length x $20;
    call module('xyz',1,x);
run;
```

The call might work fine, depending on which 10 bytes were overwritten. However, this might also cause you to lose data or cause your system to crash.

Also, note that the PEEKLONG and PEEKCLONG functions rely on the validity of the pointers you supply. If the pointers are invalid, it is possible that severe errors will result. For example, this code would cause an error:

```
data _null_;
   length c $10;
     /* trying to copy from address 0!!!*/
   c = peekclong(0,10);
run;
```

Special Considerations When Using Shared Libraries

32-Bit and 64-Bit Considerations

Compatibility between Your Shared Libraries and SAS

Starting in SAS System 9, SAS is a 64-bit application that runs on all supported UNIX environments that are 64-bit enabled. The only exception is the Linux version of SAS which is a 32-bit application. When you call external routines in shared libraries, the shared library needs to be compatible with SAS.

For example, suppose you are running a 64-bit version of SAS on Solaris. You need to call a routine that is located in libc.so. In order for this shared library to be compatible with SAS, it needs to be a 64-bit shared library.

To determine whether a vendor supplied library is 32-bit or 64-bit, you can use the **file** command. The following output shows the results of using this command on Solaris for a 32-bit and 64-bit library.

```
$ file libc.so
libc.so:  ELF 32--bit MSB dynamic lib SPARC Version 1, dynamically linked,
not stripped

$ file ./libc.so
./libc.so:  ELF 64--bit MSB dynamic lib SPARCV9 Version 1, dynamically linked,
not stripped
```

Memory Storage Allocated by the Shared Library

When specifying your SAS format and informat for each routine argument in the FORMAT attribute of the ARG statement, you need to consider the amount of memory the shared library allocates for the parameters that it receives and returns. To determine how much storage is being reserved for the input and return parameters of the routine in the external shared library, you can use the *sizeof()* C function.

The following table lists the typical memory allocations for C data types for 32-bit and 64-bit systems:

Table 7.2 Memory Allocations for C Data Types

Type	32-Bit System Size (Bytes)	32-Bit System Size (Bits)	64-Bit System Size (Bytes)	64-Bit System Size (Bits)
char	1	8	1	8
short	2	16	2	16
int	4	32	4	32
long	4	32	8	64
long long	8	64	8	64
float	4	32	4	32

Type	32-Bit System Size (Bytes)	32-Bit System Size (Bits)	64-Bit System Size (Bytes)	64-Bit System Size (Bits)
double	8	64	8	64
pointer	4	32	8	64

For information about the SAS formats to use for your data types, see "Specifying Formats and Informats to Use with MODULE Arguments" on page 181.

Naming Considerations When Using Shared Libraries

SAS loads external shared libraries that meet the following naming constraints:

- the name is eight characters or less
- the name does not contain a period.

If the name of your external shared library is greater than eight characters or contains a period, then you can create a symbolic link to point to the destination of the shared library. Once the link is created, you can add the name of the symbolic link to the MODULE statement in the SASCBTBL attribute table. When you are ready to execute your SAS program, use the PATH system option to point to the directory that contains the symbolic link.

Example of Creating a Symbolic Link

The Hewlett Packard shared library libc.sl that is installed in the **/usr/lib/pa20_64** directory contains a period in the name. Before SAS will load this shared library, you need to create a symbolic link that meets the naming convention of eight characters or less and no period. The symbolic link shown in the following example points to the target location of libc.sl:

```
$ ln -s /usr/lib/pa20_64/libc.sl /tmp/libclnk
```

After the symbolic link is created, you can then update the MODULE= option in the SASCBTBL attribute table, as shown in the following code:

```
routine name minarg=2 maxarg=2 returns=short module=libclnk;
arg 1 char output byaddr fdstart format=$cstr9.;
arg 2 char output format=$cstr9.;
```

To load the shared library during your invocation of SAS, type the following command:

```
/usr/local/sasv91/sas -path /tmp module.sas
```

Using PEEKLONG Functions to Access Character String Arguments

Since the SAS language does not provide pointers as data types, you can use the SAS PEEKLONG functions to access the data stored at these address values.

For example, the following program demonstrates how the address of a pointer is supplied and how it can set the pointer to the address of a static table containing the contiguous integers 1, 2, and 3. It also calls the **useptr** routine in the **useptr** shared library on a 64-bit operating system.

```
static struct MYTABLE {
int value1;
int value2;
```

```
int value3;
} mytable = {1,2,3};

useptr(toset)
char **toset;
{
   *toset = (char *)&mytable
}
```

The following would be the SASCBTBL attribute table entry:

```
routine useptr minarg=1 maxarg=1;
arg 1 char update format=$char20.;
```

The following would be the SAS code:

```
data _null_;
   length ptrval $20 thedata $12;
   call module('*i','useptr',ptrval);
   thedata=peekclong(ptrval,12);

   /* Converts hexadecimal data to character data */
   put thedata=$hex24.;

   /* Converts hexadecimal positive binary values to fixed or floating point value */
   ptrval=hex40.;
run;
```

The following would be the SAS log output:

Output 7.1 Log Output for Accessing Character Strings with the PEEKCLONG Function

```
thedata=000000010000000200000003 ptrval=800003FFFF0C
```

In this example, the PEEKCLONG function is given two arguments, a pointer via a numeric variable and a length in bytes. PEEKCLONG returns a character string of the specified length containing the characters at the pointer location.

For more information about the PEEKLONG functions, see "PEEKLONG Function" on page 254.

Accessing Shared Libraries Efficiently

The MODULE function reads the attribute table that is referenced by the SASCBTBL fileref once per step (DATA step, PROC IML step, or SCL step). It parses the table and stores the attribute information for future use during the step. When you use the MODULE function, SAS searches the stored attribute information for the matching routine and module names. The first time you access a shared library during a step, SAS loads the shared library and determines the address of the requested routine. Each shared library you invoke stays loaded for the duration of the step, and is not reloaded in subsequent calls. All modules and routines are unloaded at the end of the step. For example, suppose the attribute table had the following basic form:

```
* routines XYZ and BBB in FIRST.Shared Library;
routine XYZ minarg=1 maxarg=1 module=FIRST;
```

```
arg 1 num input;
routine BBB minarg=1 maxarg=1 module=FIRST;
arg 1 num input;
* routines ABC and DDD in SECOND.Shared Library;
routine ABC minarg=1 maxarg=1 module=SECOND;
arg 1 num input;
routine DDD minarg=1 maxarg=1 module=SECOND;
arg 1 num input;
```

and the DATA step looked like the following:

```
filename sascbtbl 'myattr.tbl';
data _null_;
   do i=1 to 50;
      /* FIRST.Shared Library is loaded only once */
      value = modulen('XYZ',i);
      /* SECOND.Shared Library is loaded only once */
      value2 = modulen('ABC',value);
      put i= value= value2=;
   end;
run;
```

In this example, MODULEN parses the attribute table during DATA step compilation. In the first loop iteration (i=1), FIRST.Shared Library is loaded and the XYZ routine is accessed when MODULEN calls for it. Next, SECOND.Shared Library is loaded and the ABC routine is accessed. For subsequent loop iterations (starting when i=2), FIRST.Shared Library and SECOND.Shared Library remain loaded, so the MODULEN function simply accesses the XYZ and ABC routines. SAS unloads both shared libraries at the end of the DATA step.

Note that the attribute table can contain any number of descriptions for routines that are not accessed for a given step. This does not cause any additional overhead (apart from a few bytes of internal memory to hold the attribute descriptions). In the above example, BBB and DDD are in the attribute table but are not accessed by the DATA step.

Grouping SAS Variables as Structure Arguments

A common need when calling external routines is to pass a pointer to a structure. Some parts of the structure might be used as input to the routine, while other parts might be replaced or filled in by the routine. Even though SAS does not have structures in its language, you can indicate to the MODULE function that you want a particular set of arguments grouped into a single structure. You indicate this by using the FDSTART option of the ARG statement to flag the argument that begins the structure in the attribute table. SAS gathers that argument and all that follow (until encountering another FDSTART option) into a single contiguous block, and passes a pointer to the block as an argument to the shared library routine.

Example: Grouping Your System Information as Structure Arguments

This example uses the **uname** routine, which is part of the **/usr/lib/pa20_64/libc.sl** shared library in the HP/UX operating environment. This routine returns the following information about your computer system:

- the nodename on which you are executing SAS
- the version of the operating system
- the vendor of the operating system

- the machine identification number
- model type of your machine
- the unique identification number of your class of hardware. This value could be a serial number.

The following is the C prototype for this routine:

```
int uname(struct utsname *name);
```

In C, the **utsname** structure is defined with the following members:

```
#define UTSLEN 9
#define SNLEN 15

char sysname[UTSLEN];
char nodename[UTSLEN];
char release[UTSLEN];
char version[UTSLEN];
char machine[UTSLEN];
char idnumber[SNLEN];
```

Each of the above structure members are null terminated strings.

To call this routine using the MODULE function, you would use the following attribute table entries:

```
* attribute table entry;
routine uname minarg=6 maxarg=6 returns=short module=libc;
arg 1 char output byaddr fdstart format=$cstr9.;
arg 2 char output                format=$cstr9.;
arg 3 char output                format=$cstr9.;
arg 4 char output                format=$cstr9.;
arg 5 char output                format=$cstr9.;
arg 6 char output                format=$cstr15.;
```

The following would be the SAS source code to call the **uname** routine from within the DATA step:

```
x 'if [ ! -L ./libc ]; then ln -s /usr/lib/pa20_64/libc.sl ./libc ; fi' ;
x 'setenv LD_LIBRARY_PATH .:/usr/lib:/lib:/usr/lib/pa20_64'

data _null_;
   length sysname $9 nodename $9 release $9 version $9 machine $9 idnumber $15.
   retain sysname nodename release version machine idnumber '' '';
   rc=modulen('uname', sysname, nodename, release, version, machine, idnumber)
   put rc = ;
   put sysname = ;
   put nodename = ;
   put release  = ;
   put version  = ;
   put machine  = ;
   put idnumber = ;
run;
```

The following would be the SAS log output:

Output 7.2 Grouping SAS Variables as a Structure

```
rc=0
sysname=HP-UX
nodename=garage
release=B.11.00
version=u
machine=9000/800
idnumber=103901537
```

Using Constants and Expressions as Arguments to MODULE

You can pass any kind of expression as an argument to the MODULE function. The attribute table indicates whether the argument is for input, output, or update.

You can specify input arguments as constants and arithmetic expressions. However, because output and update arguments must be able to be modified and returned, you can pass only a variable for them. If you specify a constant or expression where a value that can be updated is expected, SAS issues a warning message pointing out the error. Processing continues, but the MODULE function cannot perform the update (meaning that the value of the argument you wanted to update will be lost).

Consider these examples. Here is the attribute table:

```
* attribute table entry for ABC;
routine abc minarg=2 maxarg=2;
arg 1 input format=ib4.;
arg 2 output format=ib4.;
```

Here is the DATA step with the MODULE calls:

```
data _null_;
  x=5;
  /* passing a variable as the    */
  /*   second argument - OK       */
  call module('abc',1,x);

  /* passing a constant as the    */
  /*   second argument - INVALID  */
  call module('abc',1,2);

  /* passing an expression as the */
  /*   second argument - INVALID  */
  call module('abc',1,x+1);
run;
```

In the above example, the first call to MODULE is correct because **x** is updated by the value that the **abc** routine returns for the second argument. The second call to MODULE is not correct because a constant is passed. MODULE issues a warning indicating you have passed a constant, and passes a temporary area instead. The third call to MODULE is not correct because an arithmetic expression is passed, which causes a temporary location from the DATA step to be used, and the returned value to be lost.

Specifying Formats and Informats to Use with MODULE Arguments

You specify the SAS format and informat for each shared library routine argument by specifying the FORMAT attribute in the ARG statement. The format indicates how

numeric and character values should be passed to the shared library routine and how they should be read back upon completion of the routine.

Usually, the format you use corresponds to a variable type for a given programming language. The following sections describe the proper formats that correspond to different variable types in various programming languages.

C Language Formats

C Type	SAS Format/Informat for 32-Bit Systems	SAS Format/Informat for 64-Bit Systems
double	RB8.	RB8.
float	FLOAT4.	FLOAT4.
signed int	IB4.	IB4.
signed short	IB2.	IB2.
signed long	IB4.	IB8.
char *	IB4.	IB8.
unsigned int	PIB4.	PIB4.
unsigned short	PIB2.	PIB2.
unsigned long	PIB4.	PIB8.
char[*w*]	$CHAR*w*. or $CSTR*w*. (see "$CSTR*w*. Format" on page 184)	$CHAR*w*. or $CSTR*w*. (see "$CSTR*w*. Format" on page 184)

Note: For information about passing character data other than as pointers to character strings, see "$BYVAL*w*. Format" on page 184. △

FORTRAN Language Formats

FORTRAN Type	SAS Format/Informat
integer*2	IB2.
integer*4	IB4.
real*4	RB4.
real*8	RB8.
character**w*	$CHAR*w*.

The MODULE function can support FORTRAN character arguments only if they are not expected to be passed by a descriptor.

PL/I Language Formats

PL/I Type	SAS Format/Informat
FIXED BIN(15)	IB2.
FIXED BIN(31)	IB4.
FLOAT BIN(21)	RB4.
FLOAT BIN(31)	RB8.
CHARACTER(*w*)	$CHAR*w*.

The PL/I descriptions are added here for completeness. This does not guarantee that you will be able to invoke PL/I routines.

COBOL Language Formats

COBOL Format	SAS Format/Informat	Description
PIC S*xxxx* BINARY	IB*w*.	integer binary
COMP-2	RB8.	double-precision floating point
COMP-1	RB4.	single-precision floating point
PIC *xxxx* or S*xxxx*	F*w*.	printable numeric
PIC *yyyy*	$CHAR*w*.	character

The following COBOL specifications might not match properly with the SAS supplied formats because zoned and packed decimal are not truly defined for systems based on Intel architecture.

COBOL Format	SAS Format/Informat	Description
PIC S*xxxx* DISPLAY	ZD*w*.	zoned decimal
PIC S*xxxx* PACKED-DECIMAL	PD*w*.	packed decimal

The following COBOL specifications do not have true native equivalents and are only usable in conjunction with the corresponding S370F*xxx* informat and format, which enables IBM mainframe-style representations to be read and written in the UNIX environment.

COBOL Format	SAS Format/Informat	Description
PIC *xxxx* DISPLAY	S370FZDU*w*.	zoned decimal unsigned
PIC S*xxxx* DISPLAY SIGN LEADING	S370FZDL*w*.	zoned decimal leading sign
PIC S*xxxx* DISPLAY SIGN LEADING SEPARATE	S370FZDS*w*.	zoned decimal leading sign separate
PIC S*xxxx* DISPLAY SIGN TRAILING SEPARATE	S370FZDT*w*.	zoned decimal trailing sign separate
PIC *xxxx* BINARY	S370FIBU*w*.	integer binary unsigned
PIC *xxxx* PACKED-DECIMAL	S370FPDU*w*.	packed decimal unsigned

$CSTR*w*. Format

If you pass a character argument as a null-terminated string, use the $CSTR*w*. format. This format looks for the last nonblank character of your character argument and passes a copy of the string with a null terminator after the last nonblank character. For example, given the following attribute table entry:

```
* attribute table entry;
routine abc minarg=1 maxarg=1;
arg 1 input char format=$cstr10.;
```

you can use the following DATA step:

```
data _null_;
     rc = module('abc','my string');
     run;
```

The $CSTR format adds a null terminator to the character string **my string** before passing it to the **abc** routine. This is equivalent to the following attribute entry:

```
* attribute table entry;
routine abc minarg=1 maxarg=1;
arg 1 input char format=$char10.;
```

with the following DATA step:

```
data _null_;
     rc = module('abc','my string'||'00'x);
run;
```

The first example is easier to understand and easier to use when using variable or expression arguments.

The $CSTR informat converts a null-terminated string into a blank-padded string of the specified length. If the shared library routine is supposed to update a character argument, use the $CSTR informat in the argument attribute.

$BYVAL*w*. Format

When you use a MODULE function to pass a single character by value, the argument is automatically promoted to an integer. If you want to use a character expression in the MODULE call, you must use the special format/informat called $BYVAL*w*. The $BYVAL*w*. format/informat expects a single character and will produce a numeric

value, the size of which depends on *w*. $BYVAL2. produces a short, $BYVAL4. produces a long, and $BYVAL8. produces a double. Consider this example using the C language:

```
long xyz(a,b)
  long a; double b;
  {
  static char c = 'Y';
  if (a == 'X')
     return(1);
  else if (b == c)
     return(2);
  else return(3);
  }
```

In this example, the **xyz** routine expects two arguments, a long and a double. If the long is an **X**, the actual value of the long is 88 in decimal. This is because an ASCII **X** is stored as hex 58, and this is promoted to a long, represented as 0x00000058 (or 88 decimal). If the value of **a** is **X**, or 88, then a 1 is returned. If the second argument, a double, is **Y** (which is interpreted as 89), then 2 is returned.

Now suppose that you want to pass characters as the arguments to **xyz**. In C, you would invoke them as follows:

```
x = xyz('X',(double)'Z');
y = xyz('Q',(double)'Y');
```

This is because the **X** and **Q** values are automatically promoted to integers (which are the same as longs for the sake of this example), and the integer values corresponding to **Z** and **Y** are cast to doubles.

To call **xyz** using the MODULEN function, your attribute table must reflect the fact that you want to pass characters:

```
routine xyz minarg=2 maxarg=2 returns=long;
arg 1 input char byvalue format=$byval4.;
arg 2 input char byvalue format=$byval8.;
```

Note that it is important that the BYVALUE option appears in the ARG statement as well. Otherwise, MODULEN assumes that you want to pass a pointer to the routine, instead of a value.

Here is the DATA step that invokes MODULEN and passes it characters:

```
data _null_;
     x = modulen('xyz','X','Z');
     put x= ' (should be 1)';
     y = modulen('xyz','Q','Y');
     put y= ' (should be 2)';
run;
```

Understanding MODULE Log Messages

If you specify **i** in the control string parameter to MODULE, SAS prints several informational messages to the log. You can use these messages to determine whether you have passed incorrect arguments or coded the attribute table incorrectly.

Consider this example that uses MODULEIN from within the IML procedure. It uses the MODULEIN function to invoke the **changi** routine (which is stored in theoretical TRYMOD.so). In the example, MODULEIN passes the constant 6 and the matrix x2, which is a 4x5 matrix to be converted to an integer matrix. The attribute table for **changi** is as follows:

```
routine changi module=trymod returns=long;
arg 1 input num format=ib4. byvalue;
arg 2 update num format=ib4.;
```

The following IML step invokes MODULEIN:

```
proc iml;
   x1 = J(4,5,0);
   do i=1 to 4;
      do j=1 to 5;
         x1[i,j] = i*10+j+3;
      end;
   end;
   y1= x1;
          x2 = x1;
                   y2 = y1;
   rc = modulein('*i','changi',6,x2);
   ....
```

The `'*i'` control string causes the lines shown in the following output to be printed in the log.

Output 7.3 MODULEIN Log Output

```
---PARM LIST FOR MODULEIN ROUTINE---  CHR PARM 1 885E0AA8 2A69 (*i)
CHR PARM 2 885E0AD0 6368616E6769 (changi)
NUM PARM 3 885E0AE0 0000000000001840
NUM PARM 4 885E07F0
0000000000002C400000000000002E4000000000000030400000000000003140000000000003240
0000000000003840000000000003940000000000003A400000000000003B400000000000003C40
0000000000041400000000080414000000000
---ROUTINE changi LOADED AT ADDRESS 886119B8 (PARMLIST AT 886033A0)--- PARM 1 06000000     <CALL-BY-VALUE>
PARM 2 88604720
0E0000000F000000100000001100000012000000180000001900000001A0000001B0000001C000000
220000002300000024000000250000002600000002C0000002D0000002E0000002F00000030000000
---VALUES UPON RETURN FROM changi ROUTINE---   PARM 1 06000000     <CALL-BY-VALUE>
PARM 2 88604720
140000001F0000002A000000350000004000000082000000 8D000000980000 0A3000000AE000000
F0000000FB000000060100001101000001C0100005E0100006901000074010000 7F0100008A010000
---VALUES UPON RETURN FROM MODULEIN ROUTINE---  NUM PARM 3 885E0AE0 0000000000001840
NUM PARM 4 885E07F0
0000000000034400000000000003F4000000000000045400000000000804A400000000000005040
000000000040604000000000A0614000000000000063400000000006064400000000000C06540
0000000000006E400000000000606F4000000000
```

The output is divided into four sections.

- The first section describes the arguments passed to MODULEIN.

  The 'CHR PARM *n*' portion indicates that character parameter *n* was passed. In the example, 885E0AA8 is the actual address of the first character parameter to MODULEIN. The value at the address is hex 2A69, and the ASCII representation of that value ('*i') is in parentheses after the hex value. The second parameter is printed similarly. Only these first two arguments have their ASCII equivalents printed. This is because other arguments might contain unreadable binary data.

  The remaining parameters appear with only hex representations of their values (NUM PARM 3 and NUM PARM 4 in the example).

  The third parameter to MODULEIN is numeric, and it is at address 885E0AE0. The hex representation of the floating point number 6 is shown. The fourth parameter is at address 885E07F0, which points to an area containing all the

values for the 4x5 matrix. The ***i** option prints the entire argument. Be careful if you use this option with large matrices, because the log might become quite large.

- The second section of the log lists the arguments that are to be passed to the requested routine and, in this case, changed. This section is important for determining whether the arguments are being passed to the routine correctly. The first line of this section contains the name of the routine and its address in memory. It also contains the address of the location of the parameter block that MODULEIN created.

  The log contains the status of each argument as it is passed. For example, the first parameter in the example is call-by-value (as indicated in the log). The second parameter is the address of the matrix. The log shows the address, along with the data to which it points.

  Note that all the values in the first parameter and in the matrix are long integers because the attribute table states that the format is IB4.
- In the third section, the log contains the argument values upon return from **changi**. The call-by-value argument is unchanged, but the other argument (the matrix) contains different values.
- The last section of the log output contains the values of the arguments as they are returned to the MODULEIN calling routine.

Examples of Accessing Shared Executable Libraries from SAS

Example 1: Updating a Character String Argument

This example uses the **tmpnam** routine in the Solaris supplied shared library libc.so installed in the **/usr/lib/sparcv9** directory. The **tmpnam** routine generates a unique filename that can be used safely as a temporary filename. The temporary filename is typically placed in the **/var/tmp** directory.

The C prototype for this routine is:

```
char * tmpnam(char *s);
```

The attribute table for this would be:

```
routine tmpnam minarg=1 maxarg=1 returns=char255. module=libc;
arg 1 char output byaddr format=$cstr255;
```

The SAS source code would be:

```
x 'if [ ! -L ./libc ] ; then ln -s /usr/lib/sparcv9/libc.so.1 ./libc ; fi' ;
x 'setenv LD_LIBRARY_PATH .:/usr/lib/sparcv9:/usr/lib:/lib';

data _null_;
   length tempname $255 tname $255;
   retain tempname tname '' '';
   tname = modulec ('tmpnam', tempname);
   put tempname = ;
   put tname = ;
run;
```

The SAS log output would be:

Output 7.4 Updating a Character String Argument

```
tempname=/var/tmp/aaaKraydG
tname=/var/tmp/aaaKraydG
```

The POSIX standard for the maximum number of characters in a pathname is defined in **/usr/include/limits.h** to be 255 characters, so this example uses 254 as the length of the generated filename (**tempname**) with one character reserved for the null terminator. The $CSTR255. informat ensures that the null-terminator and all subsequent characters are replaced by trailing blanks when control returns to the DATA step.

Example 2: Passing Arguments by Value

This example calls the **access** routine that is supplied by most UNIX vendors. This particular **access** routine is in the Hewlett Packard shared library libc.sl installed under the **/usr/lib/pa20_64** directory.

The C prototype for this routine is:

```
int access(char *path, int amode);
```

The **access** routine checks the file that is referenced by the accessibility path according to the bit pattern contained in **amode**. You can use the following integer values for **amode** that correspond to the type of permission for which you are testing:

```
4   Read access
2   Write access
1   Execute (search) access
0   Check existence of file
```

A return value of 0 indicates a successful completion and the requested access is permitted. A return value of –1 indicates a failure and the requested access is not permitted.

Because the **amode** argument is a pass by value, this example includes the BYVALUE specification for the arg 2 in the attribute table. If both arguments were pass by value, one could use the CALLSEQ=BYVALUE attribute in the ROUTINE statement and it would not be necessary to specify the BYVALUE option in arg 2.

The attribute table would be:

```
routine access minarg=2 maxarg=2 returns=short module=libc;
arg 1 char input byaddr format=$cstr200.;
```

```
arg 2 num input byvalue format=ib4.;
```

The SAS source code would be:

```
x 'if [ ! -L ./libc ] ; then ln -s /usr/lib/pa20_64/libc.so ; fi' ;
x 'setenv LD_LIBRARY_PATH .:/usr/lib/pa20_64:/usr/lib:/lib' ;

data _null_;
  length path $200.;
  path='/dev';

  /* A non-root user is testing for write permission in the /dev directory */
  rc = modulen("*ie",'access',path,2);
  put rc = ;
run;
```

The SAS log output would be:

Output 7.5 Log Output If Request Access Is Permitted

```
rc=-1
```

If you changed the SAS source code to check for a write permission in the user's $HOME directory, the output would be different.

```
data _null_;
   length homedir $200.;
   homedir=sysget('HOME');

   /* A user is testing for write permissions in their $HOME directory */
   rc = modulen(''*ie'','access',homedir,2);
   put rc = ;
run;
```

In this case, the SAS log output would be:

Output 7.6 Log Output for Successful Completion (Access Permitted)

```
rc=0
```

Example 3: Using PEEKCLONG to Access a Returned Pointer

This example uses the **strcat** routine which is part of the Red Hat Linux shared library libc-2.2.3.so. This library is typically installed under the **/lib/i686** directory. This routine concatenates two strings together and returns a pointer to the newly concatenated string.

The C prototype for this routine is:

```
char *strcat(char, *dest, const char *src);
```

The proper SASCBTBL attribute table would be:

```
routine strcat minarg=2 maxarg=2 returns=ulong module=libc;
arg 1 char input format=$cstr200.;
```

```
arg 2 char input format=$cstr200.;
```

The SAS code would be:

```
x 'if [ ! -L ./libc ]; then ln -s /lib/i686/libc-2.2.3.so ./libc ; fi' ;
x 'setenv LD_LIBRARY_PATH .:/lib/i686:/usr/lib:/lib';

data _null_;
   length string1 string2 newstring $200;
   string1 = 'This is string one and';
   string2 = 'this is string two.' ;
   ptraddr = modulen('strcat',string1,string2);
   newstring = peekclong(ptraddr,200);
   put newstring = ;
run;
```

The SAS log output would be:

Output 7.7 Log Output for Using PEEKCLONG to Access a Returned Pointer

```
newstring=This is string one and this is string two.
```

The PEEKCLONG function was used here because the Red Hat Linux shared library **/lib/i686/libc-2.2.3.so** is a 32-bit library. The following output demonstrates this:

```
$pwd
/lib/i686

$file ./libc-2.2.3.so
libc-2.2.3.so:  ELF 32-bit LSB shared object, Intel 80386, version 1, not stripped
```

For more information about the PEEKLONG functions, see "PEEKLONG Function" on page 254.

Example 4: Using Structures

"Grouping SAS Variables as Structure Arguments" on page 179 describes how to use the FDSTART attribute to pass several arguments as one structure argument to a shared library routine. This is another example of using structures with another routine in an external shared library.

The **statvfs** routine that is available under most UNIX operating systems retrieves file system information. This example uses the **statvfs** routine that is in the Solaris libc.so.1 shared library and typically installed under the **/usr/lib/sparcv9** directory.

The C prototype for this routine is:

```
int statvfs(const char *path, struct statvfs *buf);
```

The **statvfs** routine will return a 0 if the routine completes successfully and –1 if there is a failure.

The **statvfs** structure is defined with the following members:

```
unsigned long f_bsize;          /* preferred file system block size */
unsigned long f_frsize;         /* fundamental file system block */
unsigned long f_blocks;         /* total number of lbocks on file system in units */
unsigned long f_bfree;          /* total number of free blocks */
unsigned long f_bavail;         /* number of free blocks available to non-superuser */
```

```
unsigned long f_files;          /* total number of file nodes (inodes) */
unsigned long f_ffree;          /* total number of free file nodes */
unsigned long f_favail;         /* number of inodes available to non-superuser */
unsigned long f_fsid;           /* file system id (dev for now) */
char          f_basetype[16];   /* target fs type name, null-terminated */
unsigned long f_flag;           /* bit mask of flags */
unsigned long g f_namemax;      /* maximum filename length */
char          f_fstr[32];       /* file system specific string */
```

The SASCBTBL attribute table would be:

```
routine statvfs
   minarg=14
   maxarg=14
   returns=short
   module=libc;
arg 1  char input  byaddr         format=$char256.;
arg 2  num  output byaddr fdstart format=pib8.;
arg 3  num  output                format=pib8.;
arg 4  num  output                format=pib8.;
arg 5  num  output                format=pib8.;
arg 6  num  output                format=pib8.;
arg 7  num  output                format=pib8.;
arg 8  num  output                format=pib8.;
arg 9  num  output                format=pib8.;
arg 10 num  output                format=pib8.;
arg 11 char output                format=$cstr16.;
arg 12 num  output                format=pib8.;
arg 13 num  output                format=pib8.;
arg 14 char output                format=$cstr32.;
```

The SAS source code to call the **statvfs** routine from within the DATA step would be:

```
x 'if [ ! -L ./libc ]; then ln -s /usr/lib/sparcv9/libc.so.1 ./libc ; fi' ;
x 'setenv LD_LIBRARY_PATH .:/usr/lib/sparcv9:/usr/lib:/lib';

data _null_;
   length f_basetype $16. f_fstr $32.;
   retain f_bsize f_frsize f_blocks f_bfree f_bavail f_files f_ffree f_favail
          f_fsid f_flag f_namemax 0;
   retain f_basetype f_fstr ' ';
   rc=modulen ('statvfs' , '/tmp', f_bsize, f_frsize, f_blocks, f_bfree, f_bavail,
               f_files, f_ffree, f_favail, f_fsid, f_basetype, f_flag,
               f_namemax, f_fstr);
   put rc = ;
   put f_bsize = ;
   put f_frsize = ;
   put f_blocks = ;
   put f_bfree = ;
   put f_bavail = ;
   put f_files = ;
   put f_ffree = ;
   put f_favail = ;
   put f_fsid = ;
   put f_basetype = ;
   put f_flag = ;
```

```
   put f_namemax = ;
   /* Determining the total bytes available in the file system and then dividing the
    total number of bytes by the number of bytes in a gigabyte */
   gigsfree = ((f_bavail * f_bsize)/1073741824);
   put 'The total amount of space available in /tmp is 'gigsfree 4.2' Gigabytes.';
run;
```

The SAS log output would be:

Output 7.8 Log Output for Using Structures

```
rc=0
f_bsize=8192
f_frsize=8192
f_blocks=196608
f_bfree=173020
f_bavail=173020
f_files=884732
f_ffree=877184
f_favail=877184
f_fsid=2
f_basetype=tmpfs
f_flag=4
f_namemax=255

The total amount of space available in /tmp is 1.32 Gigabytes.
```

Example 5: Invoking a Shared Library Routine from PROC IML

This example shows how to pass a matrix as an argument within PROC IML. The example creates a 4x5 matrix. Each cell is set to $10x+y+3$, where x is the row number and y is the column number. For example, the cell at row 1 column 2 is set to (10*1)+2+3, or 15.

The example invokes several routines from the theoretical TRYMOD shared library. It uses the **changd** routine to add $100x+10y$ to each element, where x is the C row number (0 through 3) and y is the C column number (0 through 4). The first argument to **changd** specifies the extra amount to sum. The **changdx** routine works just like **changd**, except that it expects a transposed matrix. The **changi** routine works like **changd** except that it expects a matrix of integers. The **changix** routine works like **changdx** except that integers are expected.

Note: A maximum of three arguments can be sent when invoking a shared library routine from PROC IML. △

In this example, all four matrices x1, x2, y1, and y2 should become set to the same values after their respective MODULEIN calls. Here are the attribute table entries:

```
routine changd module=trymod returns=long;
arg 1 input num format=rb8. byvalue;
arg 2 update num format=rb8.;
routine changdx module=trymod returns=long
   transpose=yes;
arg 1 input num format=rb8. byvalue;
arg 2 update num format=rb8.;
routine changi module=trymod returns=long;
arg 1 input num format=ib4. byvalue;
arg 2 update num format=ib4.;
routine changix module=trymod returns=long
```

```
   transpose=yes;
arg 1 input num format=ib4. byvalue;
arg 2 update num format=ib4.;
```

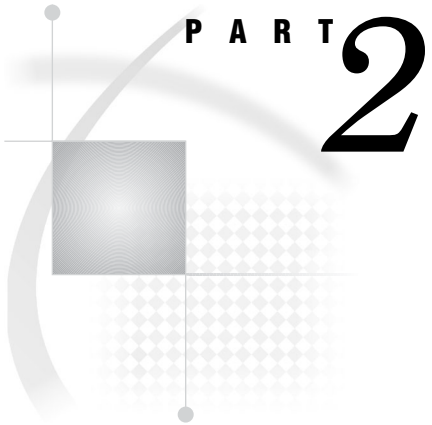
Here is the PROC IML step:

```
proc iml;
   x1 = J(4,5,0);
   do i=1 to 4;
      do j=1 to 5;
         x1[i,j] = i*10+j+3;
         end;
   end;
   y1= x1; x2 = x1; y2 = y1;
   rc = modulein('changd',6,x1);
   rc = modulein('changdx',6,x2);
   rc = modulein('changi',6,y1);
   rc = modulein('changix',6,y2);
   print x1 x2 y1 y2;
run;
```

The following are the results of the PRINT statement:

Output 7.9 Invoking a Shared Library Routine from PROC IML

```
X1
 20        31        42        53        64
130       141       152       163       174
240       251       262       273       284
350       361       372       383       394
 X2
 20        31        42        53        64
130       141       152       163       174
240       251       262       273       284
350       361       372       383       394
 Y1
 20        31        42        53        64
130       141       152       163       174
240       251       262       273       284
350       361       372       383       394
 Y2
 20        31        42        53        64
130       141       152       163       174
240       251       262       273       284
350       361       372       383       394
```



PART 2

Application Considerations

CHAPTER

8

Data Representation

Numeric Variable Length and Precision in UNIX Environments

The default length of numeric variables in SAS data sets is 8 bytes. (You can control the length of SAS numeric variables with the LENGTH statement in the DATA step.)

The issue of numeric precision affects the return values of almost all SAS math functions and many numeric values returned from SAS procedures. Numeric values in SAS for UNIX are represented as IEEE double-precision floating-point numbers. The decimal precision of a full 8-byte number is effectively 15 decimal digits.

The following table specifies the significant digits and largest integer that can be stored exactly in SAS numeric variables.

Table 8.1 Significant Digits and Largest Integer by Length for SAS Variables under UNIX

Length in Bytes	Significant Digits Retained	Largest Integer Represented Exactly
3	3	8,192
4	6	2,097,152
5	8	536,870,912
6	11	137,438,953,472
7	13	35,184,372,088,832
8	15	9,007,199,254,740,992

When you are specifying variable lengths, keep in mind that variable length affects both the amount of memory used and the time required for I/O and arithmetic operations. See *SAS Language Reference: Dictionary* for more information about specifying variable lengths.

If you know that the value of a numeric variable will be between 0 and 100, you can use a length of 3 to store the number and thus save space in your data set. For example:

```
data mydata;
   length num 3;
   ...more SAS statements...
run;
```

Numeric *dummy variables* (those whose only purpose is to hold 0 or 1) can be stored in a variable whose length is 3 bytes.

CAUTION:
Use 3 bytes only for those variables with small values, preferably integers. If the value of a variable becomes large or has many significant digits, you might lose precision in arithmetic calculations when the length of a variable is less than 8 bytes. △

For more information about specifying variable lengths and optimizing system performance, refer to *SAS Language Reference: Concepts*.

Missing Values in UNIX Environments

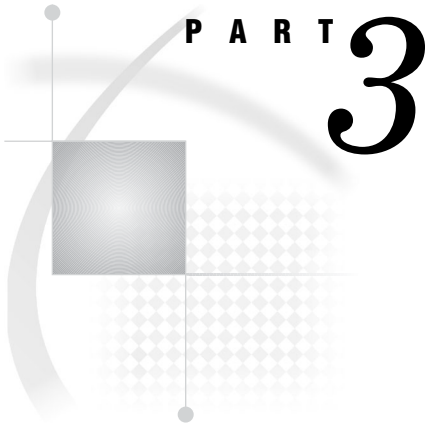
In SAS on UNIX, missing values are represented by IEEE Not-a-Number values. An IEEE Not-a-Number value is an IEEE floating-point bit pattern that represents something other than a valid numeric value. These numbers are not computationally derivable.

Reading and Writing Binary Data in UNIX Environments

Different computers store numeric binary data in different forms. For more information about compatible machine types, see "Accessing SAS Files across Compatible Machine Types in UNIX Environments" on page 108. If you try to move binary data in flat files across systems that are incompatible, problems will occur. A safer way to move data is by using SAS data sets.

SAS provides several sets of informats and formats for handling binary data. Some of these informats and formats are host dependent. For example, the IB*w.d*, PD*w.d*, PIB*w.d*, and RB*w.d*. informats and formats read and write data in native mode. That is, they use the byte-ordering system that is standard for the machine. If you create a file using the IB*w.d* format on a 64-bit HP-UX host and then use the IB*w.d* informat to read the same file on a 32-bit Linux host, you will get unpredictable results.

For more information about all of the informats and formats, refer to *SAS Language Reference: Dictionary*.



PART 3

Host-Specific Features of the SAS Language

CHAPTER

9

Commands under UNIX

SAS Commands under UNIX

This section describes commands that you can enter on the command line in the windowing environment of SAS. The commands that are described here have behavior or syntax that is specific to UNIX environments. Each command description includes a brief "UNIX specifics" section that explains which aspect of the command is specific to UNIX. If the information under the "UNIX specifics" says "all," then the command is described only in this document. Otherwise, the command is described in both this documentation and in *SAS Language Reference: Dictionary*.

The following commands are not supported in UNIX environments:

CASCADE

DCALC

ICON

PCLEAR

RESIZE

SCROLLBAR

SMARK

TILE

WGROW

WMOVE

WSHRINK

ZOOM

AUTOSCROLL Command

Controls the display of lines in the Log and Output windows

UNIX specifics: valid arguments and default values

Syntax

AUTOSCROLL <*n*>

n specifies the number of lines that the window should scroll when it receives a line of data that cannot fit.

Details

The AUTOSCROLL command controls the scrolling of lines as they are written to the Log and Output windows. The default value for AUTOSCROLL in the Log and Output windows is `1`. Processing is slower when AUTOSCROLL displays one line at a time. To expedite processing, you can specify a greater AUTOSCROLL value in your autoexec.sas file. Specifying a value of 0 optimizes processing and results in the fastest scrolling (similar to jump scrolling in xterm windows).

CAPS Command

Causes characters to be translated to uppercase when you move the cursor off of the line or press ENTER

UNIX specifics: all

Syntax

CAPS

COLOR Command

Changes the color and highlighting of selected portions of a window

UNIX specifics: valid field types and attributes

Syntax

COLOR *field-type color* | NEXT <*highlight*>

Details

Under UNIX, you cannot use the COLOR command to change the colors in these field types: BORDER, MENU, MENUBORDER, SCROLLBAR, or TITLE. Also, the H (highlight) and B (blink) attributes are not supported. For more information about the COLOR command, refer to the online help for the Program Editor window.

See Also

- Online help for the Program Editor window
- "Syntax of the COLOR Command" on page 85

DLGABOUT Command

Displays the About SAS dialog box

UNIX specifics: all

Syntax

DLGABOUT

Details

The About SAS dialog box displays information such as the version of SAS that you are running, your site number, the operating system, the version of Motif that you are using, and the color information from your terminal.

To access this dialog box from the pull-down menus, select

Help ▶ About SAS System

DLGCDIR Command

Invokes the Change Working Directory dialog box

UNIX specifics: all

Syntax

DLGCDIR

Details

The Change Working Directory dialog box enables you to select a new working directory. To access this dialog box from the pull-down menus, select

Tools ▶ Options ▶ Change Directory

DLGENDR Command

Displays the Exit dialog box

UNIX specifics: all

Syntax

DLGENDR

Details

The Exit dialog box prompts you to confirm that you want to exit SAS. If you choose OK, the SAS session ends. If you have set the **`SAS.confirmSASExit`** resource to **`False`**, this command becomes equivalent to the BYE command. To access this dialog box from the pull-down menus, select

File ▶ Exit

See Also

- "Miscellaneous Resources in UNIX Environments" on page 96

DLGFIND Command

Invokes the Find dialog box

UNIX specifics: all

Syntax

DLGFIND

Details

The Find dialog box enables you to search for text strings. To access this dialog box from the pull-down menus, select

Edit ▶ Find

See Also

- "DLGREPLACE Command" on page 207

DLGFONT Command

Invokes the Font dialog box

UNIX specifics: all

Syntax

DLGFONT

Details

The Font dialog box enables you to dynamically change the SAS windowing environment font. To access this dialog box from the pull-down menus, select

Tools ▶ Options ▶ Fonts

See Also

- "Customizing Fonts in UNIX Environments" on page 80
- "SETDMSFONT Command" on page 215

DLGOPEN Command

Invokes the Open or Import Image dialog box

UNIX specifics: all

Syntax

DLGOPEN <FILTERS='*filters*' <IMPORT> <SUBMIT|NOSUBMIT> <VERIFY>>

FILTERS='*filters*'
specifies one or more file filters to use as search criteria when displaying files. For example

```
DLGOPEN FILTERS="*.sas *.txt"
```

displays all files in the current directory that have a **.sas** extension and adds ***.txt** to the **File type** box in the dialog box. You can specify multiple filters; they all appear in the box. If you do not specify any filters, the dialog box displays a default list. See the description of the **SAS.pattern** resource in "Miscellaneous Resources in UNIX Environments" on page 96 for information about specifying a default file pattern.

IMPORT
invokes the Import Image dialog box, which enables you to import graphic files to SAS/GRAPH applications.

SUBMIT|NOSUBMIT
specifies whether the SUBMIT command is pushed after the file is opened.

VERIFY
checks whether the DLGOPEN command is appropriate for the active window.

Details

The Open and Import Image dialog boxes enable you to select a file to read into the active window. If the active window is a SAS/GRAPH window, then the Import Image dialog box is displayed; otherwise, the Open dialog box is displayed. To access these dialog boxes from the pull-down menus, select

File ▶ Open

or

File ▶ Import Image

See Also

- Information about image extensions in online documentation for SAS/GRAPH

DLGPREF Command

Invokes the Preferences dialog box

UNIX specifics: all

Syntax

DLGPREF

Details

The Preferences dialog box enables you to dynamically change certain resource settings. To access this dialog box from the pull-down menus, select

Tools ▶ Options ▶ Preferences

See Also

- "Modifying X Resources through the Preferences Dialog Box" on page 57

DLGREPLACE Command

Invokes the Change dialog box

UNIX specifics: all

Syntax

DLGREPLACE

Details

The Change dialog box enables you to search for and replace text strings. To access this dialog box from the pull-down menus, select

Edit ▶ Replace

See Also

- "DLGFIND Command" on page 205

DLGSAVE Command

Invokes the Save As or Export as Image dialog box

UNIX specifics: all

Syntax

DLGSAVE <FILTERS='*filters*' <EXPORT> <VERIFY>>

FILTERS='*filters*'
specifies one or more file filters to use as search criteria when displaying files. For example, the following command displays all files in the current directory that have a **.sas** extension and adds ***.txt** to the **File type** box in the dialog box:

```
DLGSAVE FILTERS="*.sas *.txt"
```

You can specify multiple filters; they all appear in the combo box. If you do not specify any filters, the dialog box displays a default list.

EXPORT
invokes the Export as Image dialog box, enabling you to export graphic files in your SAS session.

VERIFY
checks whether the DLGSAVE command is appropriate for the active window.

Details

To access this dialog box from the pull-down menus, select

File ▶ Save as

or

File ▶ Export as Image

See Also

- Information about image extensions in online documentation for SAS/GRAPH

DLGSCRDUMP Command

Saves the active GRAPH window as an image file using the filename and file type that you specify

UNIX specifics: all

Syntax

DLGSCRDUMP <'*filename.ext*' 'FORMAT=*file-type*'>

Details

DLGSCRDUMP saves the active GRAPH window as an image file by using the filename and file type that you specify. If you do not specify arguments, DLGSCRDUMP opens the Export dialog box and enables you to choose a filename and file type. You can save screen captures in any image format supported by SAS/GRAPH with image extensions. If your site has not licensed SAS/GRAPH with image extensions, then screen captures can be saved only as XPM files.

See Also

- Information about image extensions in online documentation for SAS/GRAPH

DLGSMAIL Command

Invokes the Send Mail dialog box

UNIX specifics: all

Syntax

DLGSMAIL

Details

The Send Mail dialog box lets you send electronic mail while working in SAS. To access this dialog box from the pull-down menus, select

File ▶ Send mail

See Also

- "Sending Electronic Mail Using the FILENAME Statement (EMAIL)" on page 143 and "Sending Mail from within Your SAS Session in UNIX Environments" on page 47
- "EMAILSYS System Option" on page 326

FILE Command

Writes the contents of the current window to an external file

UNIX specifics: valid values for *encoding-value* and *host-options*

Syntax

FILE <*file-specification*> <ENCODING='*encoding-value*'><*portable-options*>
<*host-options*>

file-specification
can be any of the following:

- a single filename. SAS writes the file in the current directory. If you enclose the filename in quotation marks, SAS uses the filename exactly as you specify it. If you do not enclose the filename in quotation marks and if you do not specify a filename extension, SAS uses .sas, .log, or .lst, depending on whether you issue the command from the Program Editor, Log, or Output window.
- an entire pathname. SAS does not assume any filename extensions, even if you do not enclose the pathname in quotation marks.
- a fileref.

ENCODING='*encoding-value*'
specifies the encoding to use when writing to the output file. The value for ENCODING= indicates that the output file has a different encoding from the current session encoding.

When you write data to the output file, SAS transcodes the data from the session encoding to the specified encoding.

For valid encoding values, see "Encoding Values in SAS Language Elements" in *SAS National Language Support (NLS): User's Guide*.

portable-options
are options for the FILE command that are valid in all operating environments. See the *SAS Language Reference: Dictionary* for information about these options.

host-options
are specific to UNIX environments. These options can be any of the following:

BLKSIZE=
BLK=
specifies the number of bytes that are physically written in one I/O operation. The default is 8K. The maximum is 1G-1.

LRECL=
specifies the logical record length. Its meaning depends on the record format in effect (RECFM). The default is 256. The maximum length is 1G.

- If RECFM=F, then the value for the LRECL= option determines the length of each output record. The output record is truncated or padded with blanks to fit the specified size.
- If RECFM=N, then the value for the LRECL= option must be at least 256.
- If RECFM=V, then the value for the LRECL= option determines the maximum record length. Records that are longer than the specified length are truncated.

NEW | OLD
indicates that a new file is to be opened for output. If the file already exists, then it is deleted and re-created. This is the default action.

RECFM=
specifies the record format. Values for the RECFM= option are

D	default format (same as variable).
F	fixed format. That is, each record has the same length. Do not use RECFM=F for external files that contain carriage-control characters.
N	binary format. The file consists of a stream of bytes with no record boundaries.
P	print format. SAS writes carriage-control characters.
V	variable format. Each record ends with a newline character.
S370V	variable S370 record format (V).
S370VB	variable block S370 record format (VB).
S370VBS	variable block with spanned records S370 record format (VBS).

UNBUF
tells SAS not to perform buffered writes to the file on any subsequent FILE statement. This option applies especially when you are writing to a data collection device.

Details

If you do not enter a file specification, then SAS uses the filename from the previous FILE or INCLUDE command. In this case, SAS first asks if you want to overwrite the file. If you have not issued any FILE or INCLUDE commands, then you receive an error message that indicates that no default file exists.

FILL Command

Specifies the fill character and fills the current field

UNIX specifics: default character

Syntax

FILL <*fill-character*>

Details

Under UNIX, the default fill character is an underscore (_).

FONTLIST Command

Lists all of the fonts that are available in your operating environment

UNIX specifics: all

Syntax

FONTLIST

Details

The FONTLIST command opens windows that list all of the software fonts that are available in your operating environment. This feature is useful if you want to choose a font to use in a SAS program, typically with a FONT= or FTEXT= option.

Issuing the FONTLIST command from the SAS command line opens the Select Font window, which contains two buttons, Copy and System. Selecting System opens the Fonts window, from which you can select and preview all available system fonts. After you select the desired font and font attributes, select OK. The Select font window reopens with your selected font name displayed. Selecting Copy places the font name in the copy buffer so that you can paste the selected font name into your SAS program.

GSUBMIT Command

Submits the specified SAS statements or the SAS code stored in a paste buffer

UNIX specifics: valid buffer names

Syntax

GSUBMIT BUF=*buffername*|"*statement1*;*statementN*...;"

buffername
can be XPRIMARY, XSCNDARY, XCLIPBRD, XTERM, or XCUT*n* where 0<=*n*<=7. See "Customizing Cut-and-Paste in UNIX Environments" on page 91 for more information.

statementN
can be any SAS statement.

HOME Command

Toggles cursor position between current position and command line

UNIX specifics: keyboard equivalent

Syntax

HOME

Details

Keyboards vary among the different UNIX operating environments. To determine which key is assigned to the HOME command, look in the KEYS window. To open the KEYS window, issue the KEYS command.

See Also

- Online help for the Program Editor window
- "Customizing Key Definitions in UNIX Environments" on page 73

HOSTEDIT Command

Invokes the host editor on the contents of the current window

UNIX specifics: all

Syntax

HOSTEDIT

Details

When you issue the HOSTEDIT command from a SAS text editor window, the contents of the buffer for that window are written to a temporary file in the **/tmp** directory. A command invoking the host editor that was specified in the EDITCMD system option is passed to the SAS Session Manager. The session manager issues the command to the operating environment to invoke the editor for the temporary file.

The X display used with the HOSTEDIT command is the same one used with your SAS session.

HED is an alias for the HOSTEDIT command.

See Also

- "Configuring SAS for Host Editor Support in UNIX Environments" on page 49
- "EDITCMD System Option" on page 325

INCLUDE Command

Copies the entire contents of an external file into the current window

UNIX specifics: valid values for *encoding-value* and *portable-options*

Syntax

INCLUDE <*file-specification*> <ENCODING='*encoding-value*'> <*portable-options*> <*host-options*>

file-specification
can be any of the following:

- a single filename. SAS searches for the file in the current directory. If you enclose the filename in quotation marks, then SAS uses the filename exactly as you specify it. If you do not enclose the filename in quotation marks and if you do not specify a filename extension, then SAS searches for *file-specification*.sas.
- an entire pathname. SAS does not assume any filename extensions, even if you do not enclose the pathname in quotation marks.
- a fileref.

ENCODING='*encoding-value*'
specifies the encoding to use when reading from the external file. The value for ENCODING= indicates that the external file has a different encoding from the current session encoding.

When you read data from an external file, SAS transcodes the data from the specified encoding to the session encoding.

For valid encoding values, see "Encoding Values in SAS Language Elements" in *SAS National Language Support (NLS): User's Guide*.

portable-options
are options for the INCLUDE command that are valid in all operating environments. See the *SAS Language Reference: Dictionary* for information about these options.

host-options
are specific to UNIX environments. These options can be any of the following:

BLKSIZE=
BLK=
specifies the number of bytes that are physically read in one I/O operation. The default is 8K. The maximum is 1G-1.

LRECL=
specifies the logical record length. Its meaning depends on the record format in effect (RECFM). The default is 256. The maximum length is 1G.

- If RECFM=F, then the value for the LRECL= option determines the number of bytes to be read as one record.
- If RECFM=N, then the value for the LRECL= option must be at least 256.
- If RECFM=V, then the value for the LRECL= option determines the maximum record length. Records that are longer than the specified length are truncated.

RECFM=
specifies the record format. Values for the RECFM= option are

D	default format (same as variable).
F	fixed format. That is, each record has the same length.
N	binary format. The file consists of a stream of bytes with no record boundaries.
P	print format.
V	variable format. Each record ends with a newline character.

Details

If you do not enter a file specification, then SAS uses the filename from the previous FILE or INCLUDE command. In this case, SAS first asks if you want to overwrite the file. If you have not issued any FILE or INCLUDE commands, then you receive an error message to indicate that no default file exists.

SETAUTOSAVE Command

Turns autosave on and off

UNIX specifics: all

Syntax

SETAUTOSAVE <ON|OFF>

Details

The SETAUTOSAVE command turns autosave on or off for the Program Editor. However, the value set for autosave on the Preferences dialog box has precedence. To open the Preferences dialog box, select

Tools ► Options ► Preferences

Autosave is controlled by the **Backup Documents** check box on the **DMS** tab. On this tab, there is also a field in which you can specify the interval for these backups.

If you turn autosave on using the SETAUTOSAVE command and the **Backup Documents** check box is selected, then SAS automatically saves the contents of the Program Editor into a file named **pgm.asv** in your current directory at the interval specified on the **DMS** tab.

If you issue this command but do not specify ON or OFF, SAS displays the current autosave setting.

See Also

- "Modifying the DMS Settings" on page 58
- "Miscellaneous Resources in UNIX Environments" on page 96

SETDMSFONT Command

Specifies a windowing environment font for the current session

UNIX specifics: all

Syntax

SETDMSFONT "*font-specification*"

font-specification
specifies an XLFD (X Logical Font Description) pattern that you want SAS to use in order to determine the windowing environment font.

Details

Most fonts in the X Window System are associated with an XLFD, which contains a number of different fields that are delimited by a dash (-) character. The fields in the XLFD indicate properties such as the font family name, weight, size, resolution, and whether the font is proportional or monospaced. Refer to your X Window documentation for more information about the XLFD and font names that are used with X.

See Also

- "DLGFONT Command" on page 205

TOOLCLOSE Command

Closes the toolbox

UNIX specifics: all

Syntax

TOOLCLOSE

Details

The TOOLCLOSE command closes the toolbox.

See Also

- "TOOLLOAD Command" on page 217

TOOLEDIT Command

Invokes the Tool Editor on the specified toolbox

UNIX specifics: all

Syntax

TOOLEDIT <*library.catalog.entry*>

Details

If you do not specify an entry name, the Tool Editor edits the toolbox for the active window.

TOOLLARGE Command

Sets the size of the buttons in the SAS ToolBox

UNIX specifics: all

Syntax

TOOLLARGE <ON|OFF>

ON
sets the size of the icons in the SAS ToolBox to 48x48.

OFF
sets the size of the icons in the SAS ToolBox to 24x24.

Details

If you do not specify ON or OFF, the TOOLLARGE command changes the size of the SAS ToolBox. The size of the SAS ToolBox changes for your current session only; the new size is not saved.

You can also use the pull-down menus to change the size of the SAS ToolBox through the Preferences dialog box.

Tools ▶ Options ▶ Preferences

Select the ToolBox tab, and then select **`Use large tools`**. If you change the size of the SAS ToolBox through the Preferences dialog box, the new size is saved, and SAS will display the large toolbox in subsequent sessions.

TOOLLOAD Command

Loads the specified toolbox

UNIX specifics: all

Syntax

TOOLLOAD <*library.catalog.entry*>

Details

If you do not specify an entry name, TOOLLOAD loads the toolbox for the active window.

See Also

□ “TOOLCLOSE Command” on page 216

TOOLTIPS Command

Turns tool tips on and off

UNIX specifics: all

Syntax

TOOLTIPS <ON|OFF>

ON
specifies that tool tip text is displayed when you move the cursor over an icon in the toolbox.

OFF
specifies that tool tip text is not displayed.

Details

If you do not specify ON or OFF, the TOOLTIPS command turns the tip text on or off, depending on the current setting.

You can also use the Preferences dialog box to specify whether tip text is displayed.

Tools ▶ Options ▶ Preferences

Select the ToolBox tab, then select **`Use tip text`**.

See Also

□ “Changing an Existing Tool” on page 70

WBROWSE Command

Invokes your World Wide Web (WWW) browser

UNIX specifics: all

Syntax

WBROWSE <“*url*”>

Details

WBROWSE invokes the Web browser that is specified by the resource **`SAS.helpBrowser`**. If you specify a URL, the document that the URL identifies is automatically displayed. If you do not specify a URL, the SAS home page is displayed.

See Also

- "Miscellaneous Resources in UNIX Environments" on page 96

WCOPY Command

Copies the marked contents of the active window to your default buffer

UNIX specifics: all

Syntax

WCOPY

Details

In Base SAS windows, this command executes the STORE command.

See Also

- SAS Help and Documentation for information about the STORE command

WCUT Command

Moves the marked contents of the active window to your default buffer

UNIX specifics: all

Syntax

WCUT

Details

In Base SAS windows, this command executes the CUT command.

This command is valid only when the active window is a text editor window, such as Program Editor or Notepad.

See Also

- SAS Help and Documentation for information about the CUT and WCUT commands

WDEF Command

Redefines the active window

UNIX specifics: behavior is controlled by the **SAS.awsResizePolicy** resource

Syntax

WDEF *starting-row starting-col nrows ncols*

Details

The WDEF command operates in the application workspace assigned to the SAS session. The WDEF command does not operate in the AWS container window, except when the container window needs to be enlarged so that you can view a SAS window contained in it. AWS resize behavior is controlled by the **SAS.awsResizePolicy** resource.

See Also

- The description of the **SAS.awsResizePolicy** resource in "Miscellaneous Resources in UNIX Environments" on page 96
- "X Window Managers" on page 31 or your window manager documentation for information about moving and resizing windows in the X environment

WPASTE Command

Pastes the contents of your default buffer into the active window

UNIX specifics: all

Syntax

WPASTE

Details

In Base SAS windows, this command executes the PASTE command.

See Also

- SAS Help and Documentation for information about the PASTE and WPASTE commands

WUNDO Command

Undoes one line of text entry

UNIX specifics: all

Syntax

WUNDO

Details

In Base SAS windows, this command executes the UNDO command. In SAS/GRAPH windows, WUNDO is invalid.

X Command

Enables you to enter UNIX commands without ending your SAS session

UNIX specifics: all

Syntax

X *UNIX-command*

X *'cmd1;cmd2....<;cmd-n>'*

Details

When you enter the X command, SAS starts a shell to execute the commands that you specified. The commands that you enter are processed differently, depending on whether you enter one command or more than one command.

See Also

- "Executing Operating System Commands from Your SAS Session" on page 13

XSYNC Command

Changes X synchronization during a SAS session

UNIX specifics: all

Syntax

XSYNC <ON|OFF>

Details

This command turns off the buffering that is normally done by the X Window System. X synchronization is off by default. Turning it on is useful when you are debugging applications, although it drastically reduces performance.

If you do not specify ON or OFF, XSYNC toggles the synchronization. The XSYNC command is valid from any SAS window.

CHAPTER
10

Data Set Options under UNIX

SAS Data Set Options under UNIX

This section describes SAS data set options that have behavior or syntax that is specific to UNIX environments. Each data set option description includes a brief "UNIX specifics" section that explains which aspect of the data set option is specific to UNIX. Each data set option is described in both this documentation and in *SAS Language Reference: Dictionary*.

Specify data set options following the data set name in SAS statements as follows:

...data-set-name(*option-1=value-1 option-2=value-2*...)

A few data set options are also SAS system options (for example, BUFSIZE=). If the same option is specified both as a system option and as a data set option, SAS uses the value given with the data set option. See "Customizing Your SAS Session Using System Options" on page 18 and Chapter 17, "System Options under UNIX," on page 311 for more information about SAS system options.

See "Summary of SAS Data Set Options in UNIX Environments" on page 227 for a table of all of the data set options available under UNIX.

Dictionary

ALTER= Data Set Option

Assigns an alter password to a SAS file and enables access to a password-protected SAS file

Default: none

Valid in: DATA step and PROC steps

Category: Data Set Control

Engines: V9, V8, V7, V6

UNIX specifics: TAPE engines ignore the *alter-password*

See: ALTER= Data Set Option in *SAS Language Reference: Dictionary*

Syntax

ALTER=*alter-password*

alter-password
must be a valid SAS name. See "Rules for Words and Names in the SAS Language" in *SAS Language Reference: Concepts*.

Details

The ALTER= option applies to all types of SAS files except catalogs. You can use this option to assign an *alter-password* to a SAS file or to access a read-protected, write-protected, or alter-protected SAS file.

Note: Under UNIX, TAPE engines ignore the *alter-password*. △

BUFNO= Data Set Option

Specifies the number of buffers to be allocated for processing a SAS data set

Default: 1

Valid in: DATA step and PROC steps

Category: Data Set Control

Engines: V9, V8, V7, V6, V9TAPE, V8TAPE, V7TAPE, V6TAPE

UNIX specifics: default value

See: BUFNO= Data Set Option in *SAS Language Reference: Dictionary*

Syntax

BUFNO=*n* | *n*K | *hex*X | MIN | MAX

***n* | *n*K**
specifies the number of buffers in multiples of 1 (bytes); 1,024 (kilobytes). For example, a value of **8** specifies 8 buffers, and a value of **1k** specifies 1024 buffers.

***hex*X**
specifies the number of buffers as a hexadecimal value. You must specify the value beginning with a number (0–9), followed by hex digits (0–9, A–F), and then followed by an X. For example the value **2dx** specifies 45 buffers.

MIN
sets the minimum number of buffers to 0, which causes SAS to use the minimum optimal value for the operating environment.

MAX
sets the number of buffers to the maximum possible number in your operating environment, up to the largest four-byte signed integer, which is 2^{31}-1, or approximately 2 billion.

Details

The buffer number is not a permanent attribute of the data set; it is valid only for the current SAS session or job. BUFNO= applies to SAS data sets that are opened for input, output, or update.

See Also

- "BUFSIZE= Data Set Option" on page 225
- "BUFNO System Option" on page 318

BUFSIZE= Data Set Option

Specifies the permanent buffer page size for an output SAS data set

Default: 0

Valid in: DATA step and PROC steps

Category: Data Set Control

Engines: V9, V8, V7, V9TAPE, V8TAPE, V7TAPE, V6TAPE

UNIX specifics: valid range

See: BUFSIZE= Data Set Option in
SAS Language Reference: Dictionary

Syntax

BUFSIZE=*n* | *n*K | *n*M | *n*G | *hex*X | MAX

***n* | *n*K | *n*M | *n*G**
specifies the buffer size in multiples of 1 (bytes); 1,024 (kilobytes); 1,048,576 (megabytes); or 1,073,741,824 (gigabytes). For example, a value of **8** specifies 8 bytes, and a value of **3m** specifies 3,145,728 bytes.

The buffer size can range from 1K to 2G-1. For values greater than 1G, use the *n*M option.

***hex*X**
specifies the page size as a hexadecimal value. You must specify the value beginning with a number (0–9), followed by hex digits (0–9, A–F), and then followed by an X. For example, **2dx** sets the page size to 45 bytes.

MAX
sets the buffer page size to the maximum possible number in your operating environment, up to the largest four-byte, signed integer, which is 2^{31}-1, or approximately 2 billion bytes.

Details

The BUFSIZE= data set option specifies the buffer size for data sets you are creating. This option is valid only for output data sets.

If you use the default value (0) when you create a SAS data set, the engine calculates a buffer size to optimize CPU and I/O use. This size is the smallest multiple of 8K that can hold 80 observations but is not larger than 64K.

If you specify a nonzero value when you create a SAS data set, the engine uses that value. If that value cannot hold at least one observation or is not a valid buffer size, the engine rounds the value up to a multiple of 1K.

See Also

□ "BUFSIZE System Option" on page 319

FILECLOSE= Data Set Option

Specifies how a tape is positioned when a SAS file on the tape is closed

Default: REREAD
Valid in: DATA step and PROC steps
Category: Miscellaneous
Engines: V9TAPE, V8TAPE, V7TAPE, V6TAPE
UNIX specifics: list of valid values
See: FILECLOSE= Data Set Option in *SAS Language Reference: Dictionary*

Syntax

FILECLOSE= FREE | LEAVE | REREAD | REWIND

FREE
rewinds and dismounts the tape. If the device cannot dismount the tape, it will only be rewound.

LEAVE
positions the tape at the end of the file that was just processed. Use FILECLOSE=LEAVE if you are not repeatedly accessing the same files in a SAS program but you are accessing one or more subsequent SAS files on the same tape.

REREAD
positions the tape volume at the beginning of the file that was just processed. Use FILECLOSE=REREAD if you are accessing the same SAS data set on tape several times in a SAS program.

REWIND
rewinds the tape volume to the beginning. Use FILECLOSE=REWIND if you are accessing one or more previous SAS files on the same tape but you are not repeatedly accessing the same files in a SAS program.

PW= Data Set Option

Assigns a read, write, or alter password to a SAS file and enables access to password-protected SAS file

Default: none

Valid in: DATA step and PROC steps

Category: Data Set Control

Engines: V9, V8, V7, V6, V9TAPE, V8TAPE, V7TAPE, V6TAPE

UNIX specifics: TAPE engines ignore the *alter-password*

See: PW= Data Set Option in
SAS Language Reference: Dictionary

Syntax

PW=*password*

password
must be a valid SAS name. See "Rules for Words and Names in the SAS Language" in *SAS Language Reference: Concepts*.

Details

The PW= option applies to all types of SAS files except catalogs. You can use this option to assign a password to a SAS file or to access a password-protected SAS file.

Note: Under UNIX, TAPE engines ignore the *alter-password*. △

Summary of SAS Data Set Options in UNIX Environments

SAS data set options are listed in the following table. The table gives the name of each option; a brief description; whether the option can be used for a data set opened for input, output, or update; and a list of engines with which the option is valid. The See column tells you where to look for more detailed information about an option. Use the following legend to see where to find more information about an option.

COMP	See the description of the data set option in this section.
LR	See *SAS Language Reference: Dictionary*.
NLS	See *SAS National Language Support (NLS): User's Guide*.

Table 10.1 Summary of SAS Data Set Options

Option Name	Description	When Used	Engines	See
ALTER=	assigns an alter password to a SAS file	output, update	V9, V8, V7, V6	LR, COMP
BUFNO=	specifies the number of buffers to be allocated for processing a SAS data set	input, output, update	V9, V8, V7, V6, V9TAPE, V8TAPE, V7TAPE, V6TAPE	LR, COMP
BUFSIZE=	specifies the permanent buffer page size for an output SAS data set	output	V9, V8, V7, V9TAPE, V8TAPE, V7TAPE, V6TAPE	LR, COMP
CNTLLEV=	specifies the level of shared access to SAS data sets	input, update	V9, V8, V7	LR
COMPRESS=	compresses observations in an output SAS data set.	output	V9, V8, V7, V6, V9TAPE, V8TAPE, V7TAPE	LR
DLDMGACTION=	specifies which action to take when a SAS catalog in a SAS data library is detected as damaged.	input, output, update	V9, V8, V7	LR
DROP=	excludes variables from processing or from output SAS data sets.	input, output, update	all	LR
ENCODING=	Specifies the character-set encoding to use for processing a particular input or output SAS data set	input, output	V9, V8, V7, V9TAPE, V8TAPE, V7TAPE	NLS
ENCRYPT=	encrypts SAS data files	output	all	LR
FILECLOSE=	specifies how a tape is positioned when a SAS file on the tape is closed	input, output	V9TAPE, V8TAPE, V7TAPE, V6TAPE	LR, COMP
FIRSTOBS=	causes processing to begin at a specified observation	input, update	all	LR
GENMAX=	requests generations for a data set and specifies the maximum number of versions	output, update	V9, V8, V7	LR
GENNUM=	references a specific generation of a data set	input, output, update	V9, V8, V7	LR
IDXNAME=	directs SAS to use a specific index to satisfy the conditions of a WHERE expression	input, update	V9, V8, V7, V6	LR

Option Name	Description	When Used	Engines	See
IDXWHERE=	overrides the SAS decision whether to use an index to satisfy the conditions of a WHERE expression	input, update	V9, V8, V7, V6	LR
IN=	creates a variable that indicates whether the data set contributed data to the current observation	input, update	all	LR
INDEX=	defines indexes when creating a data set	output	V9, V8, V7, V6, V9TAPE, V8TAPE, V7TAPE	LR
KEEP=	specifies variables for processing or writing to output SAS data sets	input, output, update	all	LR
LABEL=	specifies a label for the data set	input, output, update	all	LR
OBS=	specifies the last observation of the data set to process	input, update	all	LR
OBSBUF=	determines the size of the view buffer for processing a DATA step view	input	V9, V8, V7	LR
OUTREP=	specifies the data representation for the output SAS data set	output	V9, V8, V7, V9TAPE, V8TAPE, V7TAPE	LR
POINTOBS=	controls whether a compressed data set may be processed with random access (by observation number) rather than sequential access only	output	V9, V8, V7	LR
PW=	assigns a read, write, and alter password to a SAS file	input, output, update	V9, V8, V7, V6, V9TAPE, V8TAPE, V7TAPE, V6TAPE	LR, COMP
PWREQ=	controls the pop up of a requestor window for a data set password	input, output, update	V9, V8, V7, V6 V9TAPE, V8TAPE, V7TAPE, V6TAPE	LR
READ=	assigns a read password to a SAS file	input, output, update	V9, V8, V7, V6, V9TAPE, V8TAPE, V7TAPE, V6TAPE	LR

Option Name	Description	When Used	Engines	See
RENAME=	changes the name of a variable	input, output, update	all	LR
REPEMPTY=	controls replacement of like-named temporary or permanent SAS data sets when the new one is empty	output	V9, V8, V7	LR
REPLACE=	controls replacement of like-named temporary or permanent SAS data sets	output	all	LR
REUSE=	specifies reuse of space when observations are added to a compressed data set	output	V9, V8, V7, V6	LR
SORTEDBY=	specifies how the data set is currently sorted	input, output, update	V9, V8, V7, V6, V9TAPE, V8TAPE, V7TAPE, V6TAPE	LR
SORTSEQ=	specifies the collating sequence to be used by the SORT procedure	input, output, update	V9, V8, V7	NLS
SPILL=	specifies whether to create a spill file for non-sequential processing of a DATA step view	output	V9, V8, V7	LR
TOBSNO=[1]	specifies the number of observations to be transmitted in each multi-observation exchange with a SAS server	input, output, update	V9, V8, V7	LR
TYPE=	specifies the data set type for data that is used by some SAS/STAT procedures	input, output, update	all	LR
WHERE=	selects observations that meet the specified condition	input, output, update	all	LR
WHEREUP=	specifies whether to evaluate added observations and modified observations against a WHERE clause	output, update	V9, V8, V7, V6	LR
WRITE=	assigns a write password to a SAS file	output, update	V9, V8, V7, V6 V9TAPE, V8TAPE, V7TAPE, V6TAPE	LR

1 The TOBSNO= option is valid only for data sets that are accessed through a SAS server via the REMOTE engine.

CHAPTER
11

Formats under UNIX

SAS Formats under UNIX

This section describes SAS formats that have behavior or syntax that is specific to UNIX environments. Each format description includes a brief "UNIX specifics" section that explains which aspect of the data set option is specific to UNIX. Each format is described in both this documentation and in *SAS Language Reference: Dictionary*.

Dictionary

HEX*w.* Format

Converts real binary (floating-point) numbers to hexadecimal representation

Category: Numeric
Width range: 1 to 16
Default width: 8
Alignment: left
UNIX specifics: floating-point representation
See: HEXw. format in *SAS Language Reference: Dictionary*

Details

The HEX*w.* format converts a real (floating-point) binary number to its hexadecimal representation. When you specify a width value of 1 through 15, the real binary number

is truncated to a fixed-point integer before being converted to hex. When you specify 16 for the width, SAS writes the floating-point value of the number but does not truncate it.

Note: UNIX systems vary widely in their floating-point representation. See "Reading and Writing Binary Data in UNIX Environments" on page 198 for more information. △

$HEX*w.* Format

Converts character values to hexadecimal representation

Category: Character

Width range: 1 to 32767

Default width: 4

Alignment: left

UNIX specifics: produces ASCII codes

See: $HEXw. format in *SAS Language Reference: Dictionary*

Details

Under UNIX, the $HEX*w.* format produces hexadecimal representations of ASCII codes for characters, with each byte requiring two columns. Therefore, you need twice as many columns to output a value with the $HEX*w.* format.

IB*w.d* Format

Writes integer binary values

Category: Numeric

Width range: 1 to 8

Default width: 4

Decimal Range: 0–10

Alignment: left

UNIX specifics: byte order

See: IBw.d format in *SAS Language Reference: Dictionary*

Details

The IB*w.d* format writes integer binary (fixed-point) values. Integers are stored in integer-binary, or fixed-point, form. For example, the number 2 is stored as 00000002. If the format includes a *d* value, the data value is multiplied by 10^d.

For more details, see "Reading and Writing Binary Data in UNIX Environments" on page 198.

PD*w.d* Format

Writes packed decimal data

Category: Numeric
Width range: 1 to 16
Default width: 1
Decimal Range: 0–31
Alignment: left
UNIX specifics: data representation
See: PDw.d format in *SAS Language Reference: Dictionary*

Details

The PD*w.d* format writes values in packed decimal format. In packed decimal data, each byte contains two digits. The *w* value represents the number of bytes, not the number of digits. The value's sign is the first byte. Because the entire first byte is used for the sign, you should specify at least a width of 2.

For more details, see "Reading and Writing Binary Data in UNIX Environments" on page 198.

PIB*w.d* Format

Writes positive integer binary values

Category: Numeric
Width range: 1 to 8
Default width: 1
Decimal Range: 0–10
Alignment: left
UNIX specifics: byte order
See: PIBw.d format in *SAS Language Reference: Dictionary*

Details

The PIB*w.d* format writes fixed-point binary values, treating all values as positive. Thus, the high-order bit is part of the value, rather than the value's sign. If a *d* value is specified, the data value is multiplied by 10^d.

For more details, see "Reading and Writing Binary Data in UNIX Environments" on page 198.

RB*w.d* Format

Writes real binary (floating-point) data

Category: Numeric
Width range: 2 to 8
Default width: 4
Decimal Range: 0–10
Alignment: left
UNIX specifics: floating-point representation
See: RBw.d format in *SAS Language Reference: Dictionary*

Details

The RB*w.d* format writes numeric data in real binary (floating-point) notation. SAS stores all numeric values in floating-point.

Real binary is the most efficient format for representing numeric values because SAS already represents numbers this way and no conversion is needed.

For more details, see "RB*w.d* Informat" on page 260 and "Reading and Writing Binary Data in UNIX Environments" on page 198.

ZD*w.d* Format

Writes zoned decimal data

Category: Numeric
Width range: 1 to 32
Default width: 1
Alignment: left
UNIX specifics: data representation
See: ZDw.d format in *SAS Language Reference: Dictionary*

Details

The ZD*w.d* format writes zoned decimal data. This format is also known as overprint trailing numeric format. Under UNIX, the last byte of the field includes the sign along with the last digit. The conversion table for the last byte is as follows:

Digit	ASCII Character	Digit	ASCII Character
0	{	-0	}
1	A	-1	J
2	B	-2	K
3	C	-3	L

4	D	-4	M
5	E	-5	N
6	F	-6	O
7	G	-7	P
8	H	-8	Q
9	I	-9	R

For more details, see “ZD*w.d* Informat” on page 261 and “Reading and Writing Binary Data in UNIX Environments” on page 198.

CHAPTER

12

Functions and CALL Routines under UNIX

SAS Functions and CALL Routines under UNIX

This section describes SAS functions and CALL routines whose behavior is specific to UNIX environments. Each function and CALL routine description includes a brief "UNIX specifics" section that explains which aspect of the function and CALL routine is specific to UNIX. For more information about all of these CALL routines and functions, except SYSGET, see *SAS Language Reference: Dictionary*.

Dictionary

BYTE Function

Returns one character in the ASCII collating sequence

Category: Character

UNIX specifics: Uses the ASCII collating sequence

See: BYTE Function in *SAS Language Reference: Dictionary*

Syntax

BYTE(*n*)

n
specifies an integer that represents a specific ASCII character. The value of *n* can range from 0 to 255.

Details

If the BYTE function returns a value to a variable that has not yet been assigned a length, by default the variable is assigned a length of 1.

CALL SLEEP Routine

Suspends the execution of a program that invokes this CALL routine for a specified period of time

Category: Special

UNIX specifics: All

See: CALL SLEEP Routine in *SAS Language Reference: Dictionary*

Syntax

CALL SLEEP(*n*<,*unit*>);

n
is a numeric constant that specifies the number of units of time for which you want to suspend execution of a program.

unit
specifies the unit of time, as a power of 10, which is applied to *n*. For example, 1 corresponds to a second, and .001 corresponds to a millisecond. The default is .001.

Details

CALL SLEEP puts the DATA step in which it is invoked into a non-active wait state, using no CPU time and performing no input or output. If you are running multiple SAS processes, each process can execute CALL SLEEP independently without affecting the other processes.

Note: Extended sleep periods can trigger automatic host session termination based on timeout values set at your site. Contact your host system administrator as necessary to determine the timeout values used at your site. △

CALL SYSTEM Routine

Submits an operating system command for execution

Category: Special

UNIX specifics: *Command* must evaluate to a valid UNIX command

See: CALL SYSTEM Routine in *SAS Language Reference: Dictionary*

Syntax

CALL SYSTEM(*command*);

command
specifies any of the following: a UNIX command enclosed in quotation marks, an expression whose value is a UNIX command, or the name of a character variable whose value is a UNIX command.

Details

The CALL SYSTEM routine issues operating system commands. The output of the command appears in the window from which you invoked SAS.

The value of the XSYNC system option affects how the CALL SYSTEM routine works.

Note: The CALL SYSTEM routine can be executed within a DATA step. However, neither the X statement nor the %SYSEXEC macro program statement is intended for use during the execution of a DATA step. △

In the following example, for each record in **answer.week**, if the **resp** variable is **y**, the CALL SYSTEM routine will mail a message.

```
data _null_;
   set answer.week;
   if resp='y' then
      do;
         call system('mail mgr < $HOME/msg');
      end;
   run;
```

See Also

□ "Executing Operating System Commands from Your SAS Session" on page 13

COLLATE Function

Returns an ASCII collating sequence character string

Category: Character

UNIX specifics: Uses ASCII collating sequence

See: COLLATE Function in *SAS Language Reference: Dictionary*

Syntax

COLLATE(*start-position* <,*end-position*>) | (*start-position*<,,*length*>)

start-position
specifies the numeric position in the collating sequence of the first character to be returned.

end-position
specifies the numeric position in the collating sequence of the last character to be returned.

length
specifies the number of characters in the collating sequence.

Details

The COLLATE function returns a string of ASCII characters. The ASCII collating sequence contains 256 positions, referenced with the numbers 0 through 255. Characters above 127 correspond to characters used in European languages as defined in the ISO 8859 character set.

Unless you assign the return value of the COLLATE function to a variable with a defined length less than 200, the ASCII collating sequence string is padded with blanks to a length of 200. If the ASCII collating sequence is greater than 200 characters, you must specify the length for the return string in a LENGTH statement; otherwise, the returned string will be truncated to a length of 200 characters. For more information, see the following examples.

Examples: How SAS Determines the Length of the Return String

Example 1: Truncating the Variable Length to 200 Characters Since the following code does not include a LENGTH statement, the length attribute for the Address variable is truncated to 200 characters.

```
data sales;
   Address=collate(1,241);
run;

proc contents;
```

```
run;
```

Output 12.1 Portion of PROC CONTENTS Output

```
Alphabetic List of Variables and Attributes
#      Variable      Type      Len

1      Address       Char      200
```

Since length for Address is limited to 200 characters, the returned string from the COLLATE function will be limited to 200 characters.

Example 2: Specifying a Length Greater than 200 Characters To specify a length greater than 200 characters for a specific variable, you can use the LENGTH statement. In the following code, the length of Address is specified as 240 characters.

```
data sales;
   length Address $240;
   Address=collate(1,241);
run;

proc contents;
run;
```

Output 12.2 Portion of PROC CONTENTS Output

```
Alphabetic List of Variables and Attributes
#      Variable      Type      Len

1      Address       Char      240
```

Since the length of Address is set to 240 characters, the returned string from the COLLATE function will contain 240 characters.

See Also

- "LENGTH Statement" on page 300

DINFO Function

Returns information about a directory

Category: External Files

UNIX specifics: Directory pathname is the only information available

See: DINFO Function in *SAS Language Reference: Dictionary*

Syntax

DINFO(*directory-id*, *info-item*)

directory-id
specifies the identifier that was assigned when the directory was opened, generally by the DOPEN function.

info-item
specifies the information item to be retrieved. DINFO returns a blank if the value of *info-item* is invalid.

Details

Directories that are opened with the DOPEN function are identified by a *directory-id*. Use DOPTNAME to determine the names of the available system-dependent information items. Use DOPTNUM to determine the number of directory information items available.

Under UNIX, the only *info-item* available is Directory, which is the pathname of *directory-id*. If *directory-id* points to a list of concatenated directories, then Directory is the list of concatenated directory names.

See Also

- "DOPEN Function" on page 242
- "DOPTNAME Function" on page 242
- "DOPTNUM Function" on page 243

DOPEN Function

Opens a directory and returns a directory identifier value

Category: External Files
UNIX specifics: *fileref* can be assigned with an environment variable
See: DOPEN Function in *SAS Language Reference: Dictionary*

Syntax

DOPEN(*fileref*)

fileref
specifies the fileref assigned to the directory.

Details

DOPEN opens a directory and returns a directory identifier value (a number greater than 0) that is used to identify the open directory in other SAS external file access functions. If the directory could not be opened, DOPEN returns 0. The directory to be opened must be identified by a fileref.

DOPTNAME Function

Returns the name of a directory information item

Category: External Files

UNIX specifics: Directory is the only item available

See: DOPTNAME Function in *SAS Language Reference: Dictionary*

Syntax

DOPTNAME(*directory-id*, *nval*)

directory-id
specifies the identifier that was assigned when the directory was opened, generally by the DOPEN function.

nval
specifies the sequence number of the information item.

Details

Under UNIX, the only directory information item available is Directory, which is the pathname of the *directory-id*. The *nval*, or sequence number, of Directory is 1. If *directory-id* points to a list of concatenated directories, then Directory is the list of concatenated directory names.

DOPTNUM Function

Returns the number of information items that are available for a directory

Category: External Files

UNIX specifics: Directory is the only item available

See: DOPTNUM Function in *SAS Language Reference: Dictionary*

Syntax

DOPTNUM(*directory-id*)

directory-id
specifies the identifier that was assigned when the directory was opened, generally by the DOPEN function.

Details

Under UNIX, only one information item is available for a directory. The name of the item is Directory; its value is the pathname or list of pathnames for *directory-id*, and its sequence number is 1. Since only one information item is available for a directory, this function will return a value of 1.

FDELETE Function

Deletes an external file or an empty directory

Category: External Files
UNIX specifics: *fileref* can be assigned with an environment variable
See: FDELETE Function in *SAS Language Reference: Dictionary*

Syntax

FDELETE("*fileref*")

fileref
specifies the fileref that is assigned to the external file or directory. The fileref cannot be associated with a list of concatenated filenames or directories. If the fileref is associated with a directory, the directory must be empty. You must have permission to delete the file. Refer to the UNIX man page for **`chmod`** for more information about permissions.

Under UNIX, *fileref* can also be an environment variable. The *fileref* must be enclosed in double quotation marks.

Details

FDELETE returns 0 if the operation was successful, or a non-zero number if it was not successful.

FEXIST Function

Verifies the existence of an external file by its fileref

Category: External Files
UNIX specifics: *fileref* can be assigned with an environment variable
See: FEXIST Function in *SAS Language Reference: Dictionary*

Syntax

FEXIST("*fileref*")

fileref
specifies the fileref assigned to the external file or directory. Under UNIX, *fileref* can also be an environment variable. The *fileref* or the environment variable you specify must be enclosed in double quotation marks.

Details

The FEXIST function returns a value of 1 if the external file that is associated with *fileref* exists, and a value of 0 if the file does not exist.

FILEEXIST Function

Verifies the existence of an external file by its physical name

Category: External Files

UNIX specifics: *filename* can be assigned with an environment variable

See: FILEEXIST Function in *SAS Language Reference: Dictionary*

Syntax

FILEEXIST("*filename*")

filename
specifies a fully qualified physical filename of the external file. In a DATA step, *filename* can be a character expression, a string in quotation marks, or a DATA step variable. In a macro, *filename* can be any expression.

Under UNIX, *filename* can also be an environment variable. The *filename* or the environment variable you specify must be enclosed in double quotation marks.

Details

FILEEXIST returns 1 if the external file exists and 0 if the external file does not exist.

FILENAME Function

Assigns or deassigns a fileref for an external file, directory, or output device

Category: External Files

UNIX specifics: *fileref* can be assigned with an environment variable; valid values of *device-type* and *host-options*

See: FILENAME Function in *SAS Language Reference: Dictionary*

Syntax

FILENAME("*fileref*", "*filename*" <,*device-type*<,*host-options*<,*dir-ref*>>>)

fileref
in a DATA step, specifies the *fileref* to assign to an external file. In a macro (for example, in the %SYSFUNC function), *fileref* is the name of a macro variable (without an ampersand) whose value contains the fileref to assign to the external file. (For details, see the FILENAME function in *SAS Language Reference: Dictionary*.)

Under UNIX, the *fileref* can also be a UNIX environment variable. The *fileref* or the environment variable you specify must be enclosed in double quotation marks.

filename
specifies the external file. Specifying a blank filename (" ")deassigns a *fileref* that was previously assigned.

Under UNIX, the filename differs according to the device type. Table 16.1 on page 296 shows the information appropriate to each device. Remember that UNIX filenames are case-sensitive. The *filename* you specify must be enclosed in double quotation marks.

device-type
specifies the type of device or the access method that is used if the fileref points to an input or output device or location that is not a physical file. It can be any one of the devices listed in Table 16.1 on page 296. DISK is the default device type.

host-options
are options that are specific to UNIX. You can use any of the options that are available on the FILENAME statement. See "FILENAME Statement" on page 293 for a description of the host options.

dir-ref
specifies the fileref that is assigned to the directory in which the external file resides.

Details

FILENAME returns a 0 if the operation is successful, and a non-zero number if it was not successful.

FILEREF Function

Verifies that a fileref has been assigned for the current SAS session

Category: External Files

UNIX specifics: *fileref* can be assigned with an environment variable

See: FILEREF Function in *SAS Language Reference: Dictionary*

Syntax

FILEREF("*fileref*")

fileref
specifies the fileref assigned to be validated.

Under UNIX, *fileref* can also be a UNIX environment variable. The *fileref* or the environment variable you specify must be enclosed in double quotation marks.

Details

A negative return code indicates that the fileref exists but the physical file associated with the fileref does not exist. A positive value indicates that the fileref is not assigned. A value of zero indicates that the fileref and external file both exist.

See "FILENAME Function" on page 245 for more information.

FINFO Function

Returns the value of a file information item for an external file

Category: External Files

UNIX specifics: *info-item*s available

See: FINFO Function in *SAS Language Reference: Dictionary*

Syntax

FINFO(*file-id*, *info-item*)

file-id
specifies the identifier that was assigned when the file was opened, generally by the FOPEN function.

info-item
specifies the name of the file information item to be retrieved. This is a character value. *Info-item* is either a variable containing a valid value or the valid value in quotation marks.

Under UNIX, *info-item* for disk files can have one of the following values:

- File Name
- Owner Name
- Group Name
- Access Permission
- File Size (bytes)

If you concatenate filenames, then an additional *info-item* is available: File List.

If you are using pipe files, then the only valid value for *info-item* is Pipe Command.

Details

The FINFO function returns the value of a system-dependent information item for an external file that was previously opened and assigned a file-id by the FOPEN function. FINFO returns a blank if the value given for *info-item* is invalid.

For an example of how to use the FINFO function, see "Example: File Attributes When Using the Pipe Device Type" on page 248.

See Also

- "FOPEN Function" in *SAS Language Reference: Dictionary*

FOPTNAME Function

Returns the name of an information item for an external file

Category: External Files

UNIX specifics: Information items available

See: FOPTNAME Function in *SAS Language Reference: Dictionary*
SAS Language Reference: Dictionary

Syntax

FOPTNAME(*file-id*, *nval*)

file-id
specifies the identifier that was assigned when the file was opened, generally by the FOPEN function.

nval
specifies the number of the file information item to be retrieved. The following table shows the values that *nval* can have in UNIX operating environments for single, pipe, and concatenated files.

	Information Items Available For...		
nval	Single File	Pipe Files	Concatenated Files
1	File Name	Pipe Command	File Name
2	Owner Name		File List
3	Group Name		Owner Name
4	Access Permission		Group Name
5	File Size (bytes)		Access Permission
6			File Size (bytes)

Details

FOPTNAME returns a blank if an error occurs.

Example: File Attributes When Using the Pipe Device Type

The following example creates a data set that contains the name and value attributes returned by the FOPTNAME function when you are using pipes:

```
data fileatt;
   length name $ 20 value $ 40;
   drop fid j infonum;
   filename mypipe pipe 'UNIX-command';
   fid=fopen("mypipe","s");
   infonum=foptnum(fid);
   do j=1 to infonum;
      name=foptname(fid,j);
      value=finfo(fid,name);
      put 'File attribute' name 'has a value of ' value;
      output;
   end;
run;
```

The following statement should appear in the SAS log:

```
File attribute Pipe Command has a value of UNIX-command
```

UNIX-command is the UNIX-command or program where you are piping your output or where you are reading your input. This command or program must be either fully qualified or defined in your PATH environment variable.

See Also

- "FINFO Function" on page 247
- "FOPTNUM Function" on page 249
- "FOPEN Function" in *SAS Language Reference: Dictionary*

FOPTNUM Function

Returns the number of information items that are available for an external file

Category: External Files
UNIX specifics: Information items available
See: FOPTNUM Function in *SAS Language Reference: Dictionary*

Syntax

FOPTNUM(*file-id*)

file-id
specifies the identifier that was assigned when the file was opened, generally by the FOPEN function.

Details

Under UNIX, five information items are available for all types of files:

- File Name
- Owner Name
- Group Name
- Access Permission
- File Size (bytes).

If you concatenate filenames, then an additional information item is available: File List. If you are using pipe files, then the only information item available is Pipe Command.

The *open-mode* specified in the FOPEN function determines the value that FOPTNUM returns.

Open Mode	FOPTNUM Value	Information Items Available
Append Input Update	6 for concatenated files 5 for single files	All information items available.
Output	5 for concatenated files 4 for single files	Since the file is open for output, the File Size information type is unavailable.
Sequential (using Pipe Device Type)	1	The only information item available is Pipe Command.

For an example of how to use the FOPTNUM function, see “Example: File Attributes When Using the Pipe Device Type” on page 248.

See Also

- “FINFO Function” on page 247
- “FOPTNAME Function” on page 247
- “FOPEN Function” in *SAS Language Reference: Dictionary*

LIBNAME Function

Assigns or deassigns a libref for a SAS data library

Category: SAS File I/O

UNIX specifics: Behavior of the ' ' libref (with a space between the quotation marks)

See: LIBNAME Function in *SAS Language Reference: Dictionary*

Syntax

LIBNAME(’*libref*’<,’*SAS-data-library*’<,*engine*<,*options*>>>)

libref
specifies the libref that is assigned to a SAS data library. Under UNIX, the value of *libref* can be an environment variable. *libref* must be enclosed in single or double quotation marks.

SAS-data-library
specifies the physical name of the SAS data library that is associated with the libref. The value of *SAS-data-library* must be enclosed in single or double quotation marks.

engine
specifies the engine that is used to access SAS files opened in the data library.

options
names one or more options honored by the specified engine, delimited with blanks.

Details

If the LIBNAME function returns a 0, then the function was successful. However, you could receive a non-zero value, even if the function was successful. A non-zero value is returned if an error, warning, or note is produced. To determine if the function was successful, look through the SAS log and use the following guidelines:

- If a warning or note was generated, then the function was successful.
- If an error was generated, then the function was not successful.

Under UNIX, if you specify a *SAS-data-library* of ' '(with a space between the quotation marks), SAS deassigns the *libref*.

MODULE Function

Calls a specific routine or module that resides in a shared executable library

Category: External Files
UNIX specifics: All

Syntax

CALL MODULE(<*cntl*>,*module*,*arg-1*,*arg-2*...,*arg-n*);

num=**MODULEN**(<*cntl*>,*module*,*arg-1*,*arg-2*…,*arg-n*);

char=**MODULEC**(<*cntl*>,*module*,*arg-1*…,*arg-2*,*arg-n*);

Note: The following functions permit vector and matrix arguments; you can use them only within the IML procedure. △

CALL MODULEI (<*cntl*>,*modulearg-1*,*arg-2*...,*arg-n*);

num=**MODULEIN**(<*cntl*>,*module*,*arg-1*,*arg-2*...,*arg-n*)

char=**MODULEIC**(<*cntl*>,*module*,*arg-1*,*arg-2*...,*arg-n*);

cntl
is an optional control string whose first character must be an asterisk (*), followed by any combination of the following characters:

I	prints the hexadecimal representations of all arguments to the MODULE function and to the requested shared library routine before and after the shared library routine is called. You can use this option to help diagnose problems that are caused by incorrect arguments or attribute tables. If you specify the I option, the E option is implied.
E	prints detailed error messages. Without the E option (or the I option, which supersedes it), the only error message that the MODULE function generates is "Invalid argument to function," which is usually not enough information to determine the cause of the error.
S*x*	uses *x* as a separator character to separate field definitions. You can then specify *x* in the argument list as its own character

argument to serve as a delimiter for a list of arguments that you want to group together as a single structure. Use this option only if you do not supply an entry in the SASCBTBL attribute table. If you do supply an entry for this module in the SASCBTBL attribute table, you should use the FDSTART option in the ARG statement in the table to separate structures.

H provides brief help information about the syntax of the MODULE routines, the attribute file format, and the suggested SAS formats and informats.

For example, the control string '*IS/' specifies that parameter lists be printed and that the string '/' is to be treated as a separator character in the argument list.

module

is the name of the external module to use. The *module* can be specified as a shared library and the routine name or ordinal value, separated by a comma. You do not need to specify the shared library name if you specified the MODULE attribute for the routine in the SASCBTBL attribute table, as long as the routine name is unique (that is, no other routines have the same name in the attribute file).

The module must reside in a shared library, and it must be externally callable. Note that while the shared library name is not case sensitive, the routine name is based on the restraints of the routine's implementation language, so the routine name is case sensitive.

If the shared library supports ordinal-value naming, you can provide the shared library name followed by a decimal number, such as 'XYZ,30'.

You can specify *module* as a SAS character expression instead of as a constant; most often, though, you will pass it as a constant.

arg-1, arg-2, ...arg-n

are the arguments to pass to the requested routine. Use the proper attributes for the arguments (that is, numeric arguments for numeric attributes and character arguments for character attributes).

CAUTION:

Be sure to use the correct arguments and attributes. If you use incorrect arguments or attributes for a shared library function, you can cause SAS to crash, or you will see unexpected results. △

Details

The MODULE functions execute a routine *module* that resides in an external (outside SAS) shared library with the specified arguments *arg-1* through *arg-n*.

The MODULE call routine does not return a value, while the MODULEN and MODULEC functions return a number *num* or a character *char*, respectively. Which routine you use depends on the expected return value of the shared library function that you want to execute.

MODULEI, MODULEIC, and MODULEIN are special versions of the MODULE functions that permit vector and matrix arguments. Their return values are still scalar. You can invoke these functions only from PROC IML.

Other than this name difference, the syntax for all six routines is the same.

The MODULE function builds a parameter list by using the information in *arg-1* to *arg-n* and by using a routine description and argument attribute table that you define in a separate file. Before you invoke the MODULE routine, you must define the fileref of SASCBTBL to point to this external file. You can name the file whatever you want when you create it.

This way, you can use SAS variables and formats as arguments to the MODULE function and ensure that these arguments are properly converted before being passed to the shared library routine.

CAUTION:
Using the MODULE function without defining an attribute table can cause SAS to crash, produce unexpected results, or result in severe errors. You need to use an attribute table for all external functions that you want to invoke. △

See Also

- "The SASCBTBL Attribute Table" on page 170
- "PEEKLONG Function" on page 254

MOPEN Function

Opens a file by directory ID and by member name, and returns either the file identifier or a 0

Category: External Files
UNIX specifics: Open-modes
See: MOPEN Function in *SAS Language Reference: Dictionary*

Syntax

MOPEN(*directory-id*,*member-name*<,*open-mode*<,*record-length*<,*record-format*>>>)

Note: This is a simplified version of the MOPEN function syntax. For the complete syntax and its explanation, see the MOPEN function in *SAS Language Reference: Dictionary*. △

open-mode
specifies the type of access to the file:

A	APPEND mode allows writing new records after the current end of the file.
I	INPUT mode allows reading only (default).
O	OUTPUT mode defaults to the OPEN mode specified in the host option in the FILENAME statement or function. If no host option is specified, it allows writing new records at the beginning of the file.
S	Sequential input mode is used for pipes and other sequential devices such as hardware ports.
U	UPDATE mode allows both reading and writing.
W	Sequential update mode is used for pipes and other sequential devices such as ports.

Details

MOPEN returns the identifier for the file, or 0 if the file could not be opened.

PATHNAME Function

Returns the physical name of a SAS data library or of an external file, or returns a blank

Category: SAS File I/O

UNIX specifics: *fileref* or *libref* argument can also specify a UNIX environment variable

See: PATHNAME Function in *SAS Language Reference: Dictionary*

Syntax

PATHNAME((“*fileref*” | “*libref*”)<*search-ref*>)

fileref
specifies the fileref assigned to the external file. The value of the *fileref* can be a UNIX environment variable.

libref
specifies the libref assigned to a SAS library. The value of the *libref* can be a UNIX environment variable.

search-ref
specifies whether to search for a fileref or a libref.

F	specifies a search for a fileref.
L	specifies a search for a libref.

Details

PATHNAME returns the physical name of an external file or SAS library, or a blank if *filref* or *libref* is invalid.

For more information about using an UNIX environment variable for *fileref* or *libref*, see “FILENAME Function” on page 245.

PEEKLONG Function

Stores the contents of a memory address in a numeric variable on 32-bit and 64-bit platforms

Category: Special

UNIX specifics: All

See: PEEKLONG Function in *SAS Language Reference: Dictionary*

Syntax

PEEKCLONG(*address*,*length*);

PEEKLONG(*address*,*length*);

address
specifies the character string that is the memory address.

length
specifies the data length.

Details

CAUTION:
Use the PEEKLONG functions only to access information returned by one of the MODULE functions. △

The PEEKLONG function returns a value of length *length* that contains the data that starts at memory address *address*.

The variations of the PEEKLONG functions are:

PEEKCLONG accesses character strings.

PEEKLONG accesses numeric values.

Usually, when you need to use one of the PEEKLONG functions, you will use PEEKCLONG to access a character string. The PEEKLONG function is mentioned here for completeness.

RANK Function

Returns the position of a character in the ASCII collating sequence

Category: Character

UNIX specifics: Uses ASCII collating sequence

See: RANK Function in *SAS Language Reference: Dictionary*

Syntax

RANK(*x*)

x
is a character expression (or character string) that contains a character in the ASCII collating sequence. If the length of *x* is greater than 1, you receive the rank of the first character in the string.

Details

Because UNIX uses the ASCII character set, the RANK function returns an integer that represents the position of a character in the ASCII collating sequence.

SYSGET Function

Returns the value of the specified environment variable

Category: Special
UNIX specifics: *environment-variable* is a UNIX environment variable
See: SYSGET Function in *SAS Language Reference: Dictionary*

Syntax

SYSGET('*environment-variable*')

environment-variable
is the name of a UNIX environment variable.

Details

The SYSGET function returns the value of an environment variable as a character string. For example, this statement returns the value of the HOME environment variable:

```
here=sysget('HOME');
```

TRANSLATE Function

Replaces specific characters in a character expression

Category: Character
UNIX specifics: *to* and *from* arguments are required
See: TRANSLATE Function in *SAS Language Reference: Dictionary*

Syntax

TRANSLATE(*source*,*to-1*,*from-1*<,...*to-n*,*from-n*>)

Note: This is a simplified version of the TRANSLATE function syntax. For the complete syntax and its explanation, see the TRANSLATE function in *SAS Language Reference: Dictionary*. △

source
specifies the SAS expression that contains the original character value.

to
specifies the characters that you want TRANSLATE to use as substitutes.

from
specifies the characters that you want TRANSLATE to replace.

Details

Under UNIX, you must specify pairs of *to* and *from* arguments, and you can use a comma as a place holder.

CHAPTER
13

Informats under UNIX

SAS Informats under UNIX

This section describes SAS informats that have behavior or syntax that is specific to UNIX environments. Each informat description includes a brief "UNIX specifics" section that explains which aspect of the informat is specific to UNIX. All of these informats are described in both this documentation and in *SAS Language Reference: Dictionary*.

Dictionary

HEX*w.* Informat

Converts hexadecimal positive binary values to fixed- or floating-point binary values

Category: Numeric
Width range: 1 to 16
Default width: 8
UNIX specifics: floating-point representation
See: HEXw. informat in *SAS Language Reference: Dictionary*

Details

The HEX*w*. informat converts the hexadecimal representation of positive binary numbers to real floating-point binary values. The width value of the HEXw. informat

determines whether the input represents an integer (fixed-point) or real (floating-point) binary number. When you specify a width of 1 through 15, the informat interprets the input hexadecimal as an integer binary number. When your specify 16 for the width value, the informat interprets the input hexadecimal as a floating-point value.

For more details, see "Reading and Writing Binary Data in UNIX Environments" on page 198.

$HEX*w.* Informat

Converts hexadecimal data to character data

Category: Character

Width range: 1 to 32,767

Default width: 2

UNIX specifics: values are interpreted as ASCII values

See: $HEXw. informat in *SAS Language Reference: Dictionary*

Details

The $HEX*w.* informat converts every two digits of hexadecimal data into one byte of character data. Use the $HEX*w.* informat to encode hexadecimal values into a character variable when your input data is limited to printable characters. SAS under UNIX interprets values that are read with this informat as ASCII values.

IB*w.d* Informat

Reads integer binary (fixed-point) values

Category: Numeric

Width range: 1 to 8

Default width: 4

Decimal range: 0 to 10

UNIX specifics: byte values

See: IBw.d informat in *SAS Language Reference: Dictionary*

Details

The IB*w.d* informat reads fixed-point binary values. For integer binary data, the high-order bit is the value's sign: 0 for positive values, 1 for negative. Negative values are represented in two's-complement notation. If the informat includes a *d* value, the data value is divided by 10^d.

For more details, see "Reading and Writing Binary Data in UNIX Environments" on page 198.

PD*w.d* Informat

Reads packed decimal data

Category: Numeric
Width range: 1 to 16
Default width: 1
Decimal range: 0 to 31
UNIX specifics: data representation
See: PDw.d informat in *SAS Language Reference: Dictionary*

Details

The PD*w.d* informat reads packed decimal data. Although it is usually impossible to type in packed decimal data directly from a console, many programs write packed decimal data.

Each byte contains two digits in packed decimal data. The value's sign is the first byte. Because the entire first byte is used for the sign, you should specify at least a width of 2.

For more details, see "Reading and Writing Binary Data in UNIX Environments" on page 198.

PIB*w.d* Informat

Reads positive integer binary (fixed-point) values

Category: Numeric
Width range: 1 to 8
Default width: 1
Decimal range: 0 to 10
UNIX specifics: byte order
See: PIBw.d informat in *SAS Language Reference: Dictionary*

Details

The PIB*w.d* informat reads integer binary (fixed-point) values. Positive integer binary values are the same as integer binary (see "IB*w.d* Informat" on page 258), except that all values are treated as positive. Thus, the high-order bit is part of the value rather than the value's sign. If the informat includes a *d* value, the data value is divided by 10^d.

For more details, see "Reading and Writing Binary Data in UNIX Environments" on page 198.

RB*w.d* Informat

Reads real binary (floating-point) data

Category: Numeric

Width range: 2 to 8

Default width: 4

Decimal range: 0 to 10

UNIX specifics: floating-point representation; supports single-precision numbers only for those applications that truncate numeric data

See: RBw.d informat in *SAS Language Reference: Dictionary*

Details

The RB*w.d* informat reads numeric data that is stored in real binary (floating-point) notation. SAS stores all numeric values in floating-point.

It is usually impossible to type in floating-point binary data directly from a console, but many programs write floating-point binary data. Use caution if you are using the RB*w.d* informat to read floating-point data created by programs other than SAS because the RB*w.d* informat is designed to read only double-precision data.

All UNIX systems that are currently supported by SAS use the IEEE standard for floating-point representation. This representation supports both single-precision and double-precision floating-point numbers. Double-precision representation has more bytes of precision, and the data within the representation is interpreted differently. For example, for single-precision, the value of 1 in hexadecimal representation is **3F800000**. For double-precision, the hexadecimal representation of 1 is **3FF0000000000000**.

The RB*w.d* informat is designed to read only double-precision data. It supports widths less than 8 only for applications that truncate numeric data for space-saving purposes. RB4. does *not* expect a single-precision floating-point number; it expects a double-precision number truncated to four bytes. Using the example of 1 above, RB4. expects **3FF00000** to be the hexadecimal representation of the four bytes of data to be interpreted as 1. If given **3F800000**, the single-precision value of 1, a different number results.

External programs such as those written in C and FORTRAN can only produce single- or double-precision floating-point numbers. No length other than four or eight bytes is allowed. RB*w.d* allows a length of 3 through 8, depending on the storage you need to save.

The FLOAT4. informat has been created to read a single-precision floating-point number. If you read 3F800000 with FLOAT4., the result is a value of 1.

To read data created by a C or FORTRAN program, you need to decide on the proper informat to use. If the floating-point numbers require an eight-byte width, you should use the RB8. informat. If the floating point numbers require a four-byte width, you should use FLOAT4.

Consider this C example:

```
#include <stdio.h>

main() {

FILE *fp;
float x[3];
```

```
fp = fopen("test.dat","wb");
x[0] = 1; x[1] = 2; x[2] = 3;

fwrite((char *)x,sizeof(float),3,fp);
fclose(fp);
}
```

The file Test.dat contains **3f800000400000040400000** in hexadecimal representation.

The following statements read Test.dat correctly:

```
data _null_;
   infile 'test.dat';
   input (x y z) (float4.);
run;
```

Also available is the IEEE*w.d* informat, which reads IEEE floating-point data. On UNIX systems, IEEE8. is equivalent to RB8., and IEEE4. is equivalent to FLOAT4. IEEE*w.d* can be used on any platform, as long as the original IEEE binary data originated on a platform that uses the IEEE representation.

For more details, see "Reading and Writing Binary Data in UNIX Environments" on page 198.

ZD*w.d* Informat

Reads zoned decimal data

Category: Numeric

Width range: 1 to 32

Default width: 1

UNIX specifics: last byte includes the sign; data representation

See: ZDw.d informat in *SAS Language Reference: Dictionary*

Details

The ZD*w.d* informat reads zoned decimal data; it is also known as overprint trailing numeric format. Under UNIX, the last byte of the field includes the sign along with the last digit. The conversion table for the last byte is as follows:

Digit	ASCII Character	Digit	ASCII Character
0	{	-0	}
1	A	-1	J
2	B	-2	K
3	C	-3	L
4	D	-4	M
5	E	-5	N
6	F	-6	O
7	G	-7	P

8	H	-8	Q
9	I	-9	R

For more details, see “ZD*w.d* Format” on page 234 and “Reading and Writing Binary Data in UNIX Environments” on page 198.

CHAPTER
14
Macro Facility under UNIX

About the Macro Facility under UNIX

Most features of the SAS macro facility are valid in all operating environments. This documentation discusses only those components of the macro facility that depend on the UNIX environment. For more information, refer to

- *SAS Macro Language: Reference*
- *SAS Macro Facility Tips and Techniques*
- the online help for the macro facility.

Automatic Macro Variables in UNIX Environments

The following automatic macro variables are valid in all operating environments, but their values are determined by the operating environment:

SYSCC
contains the current SAS condition code. Upon exit, SAS translates this condition code to a return code that has a meaningful value for the operating environment.

Note: The value of SYSCC might or might not match the return code returned by the operating system. △

Under UNIX, the following codes can be returned:

0	Normal completion
1	SAS issued warning(s)
2	SAS issued error(s)
3	ABORT;

4 ABORT RETURN *n*;

5 ABORT ABEND *n*;

6 Internal error

Note: When ERRORCHECK=NORMAL, then the return code will be 0 even if an error exists in a LIBNAME or FILENAME statement, or in a LOCK statement in SAS/SHARE software. Also, the SAS job or session will not abort when the %INCLUDE statement fails due to a nonexistent file. For more information, see the "ERRORCHECK= System Option" in *SAS Language Reference: Dictionary*. △

SYSDEVIC
contains the name of the current graphics device. The current graphics device is determined by the DEVICE system option. Contact your SAS support representative to determine which graphics devices are available at your site. (See "DEVICE System Option" on page 324 and *SAS Language Reference: Dictionary* for information about the DEVICE system option.)

SYSENV
reports whether SAS is running interactively. Values for SYSENV are **FORE** when the TERMINAL system option is in effect and **BACK** when the NOTERMINAL system option is in effect.

SYSJOBID
lists the process identification number (PID) of the process that is executing SAS, for example, 00024.

SYSMAXLONG
returns the maximum long integer value allowed under UNIX, which is 9,007,199,254,740,992. On 32-bit systems, the maximum is 2,147,483,647.

SYSRC
holds the decimal value of the exit status code that is returned by the last UNIX command executed from your SAS session. The following output shows an interactive line mode SAS session that shows two sample SYSRC values.

Output 14.1 Sample SYSRC Values

```
1? x 'data';
/bin/ksh: data: not found
2? %put UNIX exit status code is &sysrc;
UNIX exit status code is 256
3? x 'date';
Tue Mar 19 09:41:27 CST 2003
4? %put UNIX exit status code is now &sysrc;
UNIX exit status code is now 0
```

SYSSCP
returns the abbreviation for your operating environment, such as **HP 800**, **SUN 4**, or **RS6000**.

SYSSCPL
returns the name of the specific UNIX environment that you are using, such as **HP-UX**, **SunOS**, or **AIX**. This variable returns the same value that is returned by the UNIX command **uname**.

Macro Statements in UNIX Environments

The arguments that can be entered with the following statements depend on the operating environment:

%SYSEXEC
executes UNIX commands. It is similar to the X statement described in "Executing Operating System Commands from Your SAS Session" on page 13. The %SYSEXEC statement enables you to execute operating environment commands immediately and, if necessary, determine whether they executed successfully by examining the value of the automatic macro variable SYSRC. You can use the %SYSEXEC statement inside a macro or in open code. The form of the %SYSEXEC statement is as follows, where *command* can be any UNIX command:

%SYSEXEC <*command*>;

For example, the following code writes the status of the default printer to your UNIX shell:

```
%sysexec lpstat;
```

Entering %SYSEXEC without a UNIX command starts a new shell, except under the X Interface to SAS. See "Executing Operating System Commands from Your SAS Session" on page 13 for details.

Macro Functions in UNIX Environments

The following functions have operating environment dependencies:

%SCAN
searches for a word that is specified by its position in a string. The form of the %SCAN function is

%SCAN(*argument*,*n*,<*delimiters*>);

On ASCII systems, the default delimiters are

```
blank . < ( + & ! $ * ) ; ^ - / , % |
```

%SYSGET
returns the character string that is the value of the environment variable passed as the argument. Both UNIX and SAS environment variables can be translated using the %SYSGET function. A warning message is printed if the global variable does not exist. The form of the %SYSGET function is

%SYSGET(*environment-variable*);

For example, the following code writes the value of the HOME environment variable to the SAS log:

```
%let var1=%sysget(HOME);
%put &var1;
```

SAS System Options Used by the Macro Facility in UNIX Environments

The following system options have operating environment dependencies:

MSYMTABMAX
specifies the maximum amount of memory available to all symbol tables (global and local combined). Under UNIX, the default value for this option is 4M. See *SAS Language Reference: Dictionary* for more information.

MVARSIZE
specifies the maximum number of bytes for any macro variable stored in memory. Under UNIX, the default value for this option is 32K. See *SAS Language Reference: Dictionary* for more information.

SASAUTOS
specifies the AUTOCALL library. See "The SASAUTOS System Option" on page 266 for more information.

Using Autocall Libraries in UNIX Environments

What Is an Autocall Library?

An autocall library contains files that define SAS macros. The following sections discuss aspects of autocall libraries that are dependent on the operating environment. For more information, see *SAS Macro Language: Reference*.

Available Autocall Macros

There are two types of autocall macros, those provided by SAS and those you define yourself. To use the autocall facility, you must have the MAUTOSOURCE system option set.

When SAS is installed, the SASAUTOS system option is defined in the configuration file to refer to the location of the default macros supplied by SAS. The products licensed at your site determine the autocall macros you have available. You can also define your own autocall macros and store them in one or more directories.

Guidelines for Naming Macro Files

If you store autocall macros in a UNIX directory, the file extension must be .sas. Each macro file in the directory must contain a macro definition with a macro name that matches the filename. For example, a file named PrtData.sas should define a macro named PRTDATA.

The SASAUTOS System Option

To use your own autocall macros in your SAS program, specify their directories with the SASAUTOS system option. For more information, see "SASAUTOS System Option" on page 358.

Note: The SASAUTOS system option under UNIX does not recognize filenames that are in uppercase or mixed case. △

You can set the SASAUTOS system option when you start SAS, or you can use it in an OPTIONS statement during your SAS session. However, autocall libraries specified with the OPTIONS statement override any previous specification.

If you use the CONFIG system option to specify a configuration file, add your autocall library to the library concatenation supplied by SAS. If you use the default configuration files (sasv9.cfg) simply specify your autocall library there.

Autocall libraries are searched in the order in which you specify them.

Example: Setting Up and Testing a Macro in an Autocall Library

This example shows how to set up and test a macro in an autocall library.

The following output shows the results of executing two UNIX (**cat**) commands to display the contents of two files and a SAS command to run the Autocall.sas program.

Output 14.2 AUTOCALL Library Example

```
$ cat maclib/testauto.sas
%macro testauto;
x echo 'Autocall library is working.';
%mend testauto;
$ cat source/autocall.sas
filename sysautos ('!SASROOT/sasautos' '$HOME/test/sasautos');
options mautosource sasautos=(sysautos '$HOME/macros/maclib');
%testauto
$ sas source/autocall.sas
Autocall library is working.
```

CHAPTER
15

Procedures under UNIX

SAS Procedures under UNIX

This section describes SAS procedures that have behavior or syntax that is specific to UNIX environments. Each procedure description includes a brief "UNIX specifics" section that explains which aspect of the procedure is specific to UNIX. Each procedure is described in both this documentation and in *Base SAS Procedures Guide*.

Dictionary

CATALOG Procedure

Manages entries in SAS catalogs

UNIX specifics: FILE= option in the CONTENTS statement

See: CATALOG Procedure in *Base SAS Procedures Guide*

Syntax

PROC CATALOG CATALOG=<*libref.*>*catalog* <ENTRYTYPE=*etype*> <KILL>;
CONTENTS <OUT=*SAS-data-set*> <FILE=*fileref*>;

Note: This is a simplified version of the CATALOG procedure syntax. For the complete syntax and its explanation, see the CATALOG procedure in *Base SAS Procedures Guide*. △

fileref
names a file specification that is specific to the UNIX operating environment.

Details

The FILE= option in the CONTENTS statement of the CATALOG procedure accepts a fileref. If the name specified does not correspond to a fileref, a file with that name and an extension of .lst is created in the current directory. For example, if MyFile is not a fileref, the following code creates the file MyFile.lst in your current directory:

```
proc catalog catalog=sasuser.profile;
   contents file=myfile;
run;
```

Note: The filename that is created is always in lowercase, even if you specified it in uppercase. △

CIMPORT Procedure

Restores a transport file created by the CPORT procedure

UNIX specifics: name and location of transport file

See: CIMPORT Procedure in *Base SAS Procedures Guide*

Syntax

PROC CIMPORT *destination=libref|<libref.>member-name <option(s)>*;

Note: This is a simplified version of the CIMPORT procedure syntax. For the complete syntax and its explanation, see the CIMPORT procedure in *Base SAS Procedures Guide*. △

destination
identifies the file(s) in the transport file as a single SAS data set, single SAS catalog, or multiple members of a SAS data library.

libref | <libref.>member-name
specifies the name of the SAS data set, catalog, or library to be created from the transport file.

Details

Note: Starting in SAS 9.1, you can use the MIGRATE procedure to convert your SAS files. For more information, see “Migrating 32-Bit SAS Files to 64-Bit in UNIX Environments” on page 106. △

The CIMPORT procedure *imports* a transfer file that was created (*exported*) by the CPORT procedure. The transport file can contain a SAS data set, a SAS catalog, or an entire SAS library.

Typically the INFILE= option is used to designate the source of the transport file. If this option is omitted, CIMPORT uses the default file Sascat.dat in the current directory as the transport file.

Note: CIMPORT works only with transport files created by the CPORT procedure. If the transport file was created using the XPORT engine with the COPY procedure, then another PROC COPY must be used to restore the transport file. For more information about PROC COPY, see *Base SAS Procedures Guide*. △

Example

For this example, a SAS data library that contains multiple SAS data sets was exported to a file (called transport-file) using the CPORT procedure on a foreign host. The transport file is then moved by a binary transfer to the receiving host.

The following code extracts all of the SAS data sets and catalogs stored within the transport file and restores them to their original state in the new library, called SAS-data-library.

```
libname newlib 'SAS-data-library';
filename tranfile 'transport-file';

proc cimport lib=newlib infile=tranfile;
run;
```

See Also

- "CPORT Procedure" on page 275
- "Migrating 32-Bit SAS Files to 64-Bit in UNIX Environments" on page 106
- The MIGRATE Procedure at **support.sas.com/rnd/migration**
- *Moving and Accessing SAS Files*

CONTENTS Procedure

Prints descriptions of one or more files from a SAS data library

UNIX specifics: information displayed in the SAS output

See: CONTENTS Procedure in *Base SAS Procedures Guide*

Syntax

PROC CONTENTS<*option(s)*>;

Details

The CONTENTS procedure produces the same information as the CONTENTS statement in the DATASETS procedure. See "DATASETS Procedure" on page 276 for sample output.

CONVERT Procedure

Converts BMDP and OSIRIS system files, and SPSS export files to SAS data sets

UNIX specifics: all

Syntax

PROC CONVERT *product-specification* <*option-list*>;

Details

The CONVERT procedure converts BMDP and OSIRIS system files, and SPSS export files to SAS data sets. The procedure is supplied for compatibility. The procedure invokes the appropriate engine to convert files.

PROC CONVERT produces one output data set, but no printed output. The new data set contains the same information as the input system file; exceptions are noted in "How Missing Values Are Handled" on page 273.

The procedure converts system files from these three products:

- BMDP saves files up to and including the most recent release of BMDP (available for AIX, HP-UX, and Solaris only).
- SPSS saves files in a portable file format. An SPSS portable file can have any file extension. Two common extensions are .por and .exp.
- OSIRIS saves files through and including OSIRIS IV. (Hierarchical file structures are not supported.)

Because the BMDP, OSIRIS, and SPSS products are maintained by other organizations, changes might be made that make new files incompatible with the current version of PROC CONVERT. SAS upgrades PROC CONVERT to support changes to these products only when a new version of SAS is released.

In the PROC CONVERT statement, *product-specification* is required and can be one of the following:

BMDP=*fileref* <(CODE=*code* CONTENT=*content-type*)>
converts the first member of a BMDP save file created under UNIX (AIX) into a SAS data set. Here is an example:

```
filename save '/usr/mydir/bmdp.dat';
proc convert bmdp=save;
run;
```

If you have more than one save file in the BMDP file referenced by the *fileref* argument, you can use two options in parentheses after *fileref*. The CODE= option specifies the code of the save file that you want, and the CONTENT= option specifies the content of the save file. For example, if a file with CODE=JUDGES has a content of DATA, you can use the following statements:

```
filename save '/usr/mydir/bmdp.dat';
proc convert bmdp=save(code=judges
                       content=data);
run;
```

OSIRIS=*fileref*|*libref*
specifies a fileref or libref for the OSIRIS file to be converted into a SAS data set. You must also include the DICT= option.

SPSS=*fileref*|*libref*
specifies a fileref or libref for the SPSS export file that is to be converted into a SAS data set. The SPSS file must be created by using the SPSS EXPORT command, but it can be from any operating system.

The *option-list* can be one or more of the following:

DICT=*fileref*|*libref*
specifies a fileref or libref of the dictionary file for the OSIRIS file. DICT= is valid only when used with the OSIRIS product specification.

FIRSTOBS=*n*
gives the number of the observation where the conversion is to begin, so that you can skip observations at the beginning of the BMDP, OSIRIS, or SPSS file.

OBS=*n*
specifies the number of the last observation to be converted. This enables you to exclude observations at the end of the file.

OUT=*SAS-data-set*
names the SAS data set that will hold the converted data. If OUT= is omitted, SAS still creates a Work data set and automatically names it DATA*n*, just as if you had omitted a data set name in a DATA statement. See Chapter 4, "Using SAS Files," on page 101 for more information.

How Missing Values Are Handled

If a numeric variable in the input data set has no value or a system missing value, CONVERT assigns it a missing value.

How Variable Names Are Assigned

The following sections explain how names are assigned to the SAS variables created by the CONVERT procedure.

CAUTION:
Be sure that the translated names will be unique. Variable names are translated as indicated in the following sections. △

Variable Names in BMDP Output Variable names from the BMDP save file are used in the SAS data set, but nontrailing blanks and all special characters are converted to underscores in the SAS variable names. The subscript in BMDP variable names, such as x(1), becomes part of the SAS variable name with the parentheses omitted: X1. Alphabetic BMDP variables become SAS character variables of corresponding length. Category records from BMDP are not accepted.

Variable Names in OSIRIS Output For single-response variables, the V1 through V9999 name becomes the SAS variable name. For multiple-response variables, the suffix R*n* is added to the variable name where *n* is the response. For example, V25R1 would be the first response of the multiple-response V25. If the variable after V1000 has 100 or more responses, responses above 99 are eliminated. Numeric variables that OSIRIS stores in character, fixed-point binary, or floating-point binary mode become SAS numeric variables. Alphabetic variables become SAS character variables; any alphabetic variable of length greater than 200 is truncated to 200. The OSIRIS variable description becomes a SAS variable label, and OSIRIS print format information becomes a SAS format.

Variable Names in SPSS Output SPSS variable names and variable labels become variable names and labels without change. SPSS alphabetic variables become SAS character variables of the same length. SPSS blank values are converted to SAS missing values. SPSS print formats become SAS formats, and the SPSS default precision of no decimal places becomes part of the variables' formats. The SPSS DOCUMENT data is copied so that the CONTENTS procedure can display it. SPSS value labels are not copied.

File Conversion Examples

These three examples show how to convert BMDP, OSIRIS, and SPSS files to SAS data sets.

Converting a BMDP save file
The following statements convert a BMDP save file and produce the temporary SAS data set Temp, which contains the converted data:

```
filename bmdpfile 'bmdp.savefile';
proc convert bmdp=bmdpfile out=temp;
run;
```

Converting an OSIRIS file
The following statements convert an OSIRIS file and produce the temporary SAS data set Temp, which contains the converted data:

```
filename osirfile 'osirdata';
filename dictfile 'osirdict';
proc convert osiris=osirfile dict=dictfile
             out=temp;
run;
```

Converting an SPSS file
The following statements convert an SPSS Release 9 file and produce the temporary SAS data set Temp, which contains the converted data:

```
filename spssfile 'spssfile.num1';
proc convert spss=spssfile out=temp;
run;
```

Comparison with Interface Library Engines

The CONVERT procedure is closely related to the interface library engines BMDP, OSIRIS, and SPSS. (In fact, the CONVERT procedure uses these engines.) For example, the following two sections of code provide identical results:

```
filename myfile 'mybmdp.dat';
proc convert bmdp=myfile out=temp;
run;
```

```
libname myfile bmdp 'mybmdp.dat';
data temp;
   set myfile._first_;
run;
```

However, the BMDP, OSIRIS, and SPSS engines provide more extensive capability than PROC CONVERT. For example, PROC CONVERT converts only the first BMDP member in a save file. The BMDP engine, in conjunction with the COPY procedure, copies all members.

See Also

- "Accessing BMDP, OSIRIS, or SPSS Files in UNIX Environments" on page 125

CPORT Procedure

Writes SAS data sets and catalogs into a special format in a transport file that can be moved between different hosts

UNIX specifics: name and location of transport file

See: CPORT Procedure in *Base SAS Procedures Guide*

Syntax

PROC CPORT *source-type=libref|<libref.>member-name <option(s)>*;

Note: This is a simplified version of the CPORT procedure syntax. For the complete syntax and its explanation, see the "CPORT Procedure" in *Base SAS Procedures Guide*. △

source-type
identifies the file(s) to export as either a single SAS data set, single SAS catalog, or multiple members of a SAS data library.

libref | <libref.> member-name
specifies the name of the SAS data set, catalog, or library to be exported.

Details

Note: Starting in SAS 9.1, you can use the MIGRATE procedure to convert your SAS files. For more information, see "Migrating 32-Bit SAS Files to 64-Bit in UNIX Environments" on page 106. △

The CPORT procedure creates a transport file to later be restored (*imported*) by the CIMPORT procedure. The transport file can contain a SAS data set, SAS catalog, or an entire SAS library.

Typically the FILE= option is used to specify the path of the transport file. The value of the FILE= option can be a fileref defined in a FILENAME statement or an environment variable. If this option is omitted, CPORT creates the default file Sascat.dat in the current directory as the transport file.

Examples

In this example, a SAS data library (called Oldlib) that contains multiple SAS data sets is being exported to the file, called transport-file.

```
libname oldlib 'SAS-data-library';
filename tranfile 'transport-file';

proc cport lib=oldlib file=tranfile;
run;
```

This transport file is then typically moved by binary transfer to a different host where the CIMPORT procedure will be used to restore the SAS data library.

See Also

- "CIMPORT Procedure" on page 270
- "Migrating 32-Bit SAS Files to 64-Bit in UNIX Environments" on page 106
- The MIGRATE Procedure at **support.sas.com/rnd/migration**
- *Moving and Accessing SAS Files*

DATASETS Procedure

Lists, copies, renames, and deletes SAS files, and also manages indexes for and appends SAS data sets in a SAS data library

UNIX specifics: Directory information, CONTENT statement output

See: DATASETS Procedure in *Base SAS Procedures Guide*

Syntax

PROC DATASETS< *option(s)*>;
CONTENTS <*option(s)*;>

Note: This is a simplified version of the DATASETS procedure syntax. For the complete syntax and its explanation, see the DATASETS procedure in *Base SAS Procedures Guide*. △

CONTENTS *option(s)*
the value for *option(s)* can be the following:

DIRECTORY
prints a list of information specific to the UNIX operating environment.

Details

The output from the DATASETS procedure shows you the libref, engine, and physical name that are associated with the library, as well as the names and other properties of the SAS files that are contained in the library. Some of the SAS data library information, such as the filenames and access permissions, that is displayed in the SAS log by the DATASETS procedure depends on the operating environment and the engine. The information generated by the CONTENTS statement also varies according to the device type or access method associated with the data set.

If you specify the DIRECTORY option in the CONTENTS statement, the directory information is displayed in both the log and output windows.

The CONTENTS statement in the DATASETS procedure generates the same Engine/Host Dependent information as the CONTENTS procedure.

Example

The following SAS code creates two data sets, Grades.sas7bdat and Majors.sas7bdat, and runs PROC DATASETS on Majors.sas7bdat.

```
options nodate pageno=1;
libname classes '.';

data classes.grades (label='First Data Set');
   input student year state $ grade1 grade2;
   label year='Year of Birth';
   format grade1 4.1;
   datalines;
1000 1980 NC 85 87
1042 1981 MD 92 92
1095 1979 PA 78 72
1187 1980 MA 87 94
;


data classes.majors(label='Second Data Set');
   input student $ year state $ grade1 grade2 major $;
   label state='Home State';
   format grade1 5.2;
   datalines;
1000 1980 NC 84 87 Math
1042 1981 MD 92 92 History
1095 1979 PA 79 73 Physics
1187 1980 MA 87 74 Dance
1204 1981 NC 82 96 French
;


proc datasets library=classes;
   contents data=majors directory;
run;
```

The output of this example is shown in Output 15.1. The first page of output from this example SAS code is produced by the DIRECTORY option in the CONTENTS statement. This information also appears on the SAS log. Pages 2 and 3 in this output describe the data set Classes.Majors.sas7bdat and appear only on the SAS output.

Output 15.1 PROC DATASETS Example

```
                              The SAS System

                          The DATASETS Procedure

                                Directory

                  Libref             CLASSES
                  Engine             V9
                  Physical Name      /remote/u/yourid
                  File Name          /remote/u/yourid
                  Inode Number       1058605
                  Access Permission  rwxrwxrwx
                  Owner Name         yourid
                  File Size (bytes)  1024


                             Member    File
                  #  Name    Type      Size   Last modified
                  1  GRADES  DATA      16384   12MAY2003:14:30:19
                  2  MAJORS  DATA      16384   12MAY2003:14:31:20
```

```
                              The SAS System

                          The DATASETS Procedure

Data Set Name       CLASSES.MAJORS                  Observations          5
Member Type         DATA                            Variables             6
Engine              V9                              Indexes               0
Created             Monday, May 12, 2003 14:31:20   Observation Length    48
Last Modified       Monday, May 12, 2003 14:31:20   Deleted Observations  0
Protection                                          Compressed            NO
Data Set Type                                       Sorted                NO
Label               Second Data Set
Data Representation HP_UX_64, RS_6000_AIX_64, SOLARIS_64, HP_IA64
Encoding            latin1  Western (ISO)


                       Engine/Host Dependent Information

     Data Set Page Size          8192
     Number of Data Set Pages    1
     First Data Page             1
     Max Obs per Page            169
     Obs in First Data Page      5
     Number of Data Set Repairs  0
     File Name                   /remote/u/yourid/majors.sas7bdat
     Release Created             9.0101B0
     Host Created                SunOS
     Inode Number                1059264
     Access Permission           rw-r--r--
     Owner Name                  yourid
     File Size (bytes)           16384
```

```
                              The SAS System

                          The DATASETS Procedure

                 Alphabetic List of Variables and Attributes

#    Variable    Type    Len    Format    Label

4    grade1      Num       8     5.2
5    grade2      Num       8
6    major       Char      8
3    state       Char      8              Home State
1    student     Char      8
2    year        Num       8
```

See Also

- "CONTENTS Procedure" in *Base SAS Procedures Guide*

OPTIONS Procedure

Lists the current values of all SAS system options

UNIX specifics: options available only under UNIX

See: OPTIONS Procedure in *Base SAS Procedures Guide*

Syntax

PROC OPTIONS=<*option(s)*>

Note: This is a simplified version of the OPTIONS procedure syntax. For the complete syntax and its explanation, see the OPTIONS procedure in *Base SAS Procedures Guide*. △

option(s)

HOST | NOHOST
displays only host options (HOST) or only portable options (NOHOST). PORTABLE is an alias for NOHOST.

RESTRICT
displays the system options that have been restricted by your site administrator. These options cannot be changed by the user. For each option that is restricted, the RESTRICT option displays the option's value, scope, and how it was set.

If your site administrator has not restricted any options, then the following message will appear in the SAS log:

```
Your Site Administrator has not restricted any options.
```

Details

PROC OPTIONS lists the current values of the system options that are available in all operating environments and, if you specify the HOST option in the PROC OPTIONS statement, it lists those options that are available only under UNIX (host options). The option values displayed by PROC OPTIONS depend on the default values shipped with SAS, the default values specified by your site administrator, the default values in your own configuration file, any changes made in your current session through the System Options window or OPTIONS statement, and possibly, the device on which you are running SAS.

For more information about a specific option, refer to Chapter 17, "System Options under UNIX," on page 311.

See Also

- For more information about restricted options, see "Order of Precedence for SAS Configuration Files" on page 17.

PMENU Procedure

Defines PMENU facilities for windows created with SAS software

UNIX specifics: ATTR= and COLOR= options in the TEXT statement have no effect; ACCELERATE= and MNEMONIC= options in the ITEM statement are ignored

See: PMENU Procedure in *Base SAS Procedures Guide*

Syntax

PROC PMENU <CATALOG=<*libref*.>*catalog*>
<DESC '*entry-description*'>;

Note: This is a simplified version of the PMENU procedure syntax. For the complete syntax and its explanation, see the PMENU procedure in *Base SAS Procedures Guide*. △

CATALOG=<*libref*.>*catalog*
specifies the catalog in which you want to store PMENU entries. If you omit *libref*, the PMENU entries are stored in a catalog in the Sasuser data library. If you omit CATALOG=, the entries are stored in the Sasuser.Profile catalog.

DESC '*entry-description*'
provides a description of the PMENU catalog entries created in the step.

Details

The PMENU procedure defines PMENU facilities for windows created by using the WINDOW statement in Base SAS software, the %WINDOW macro statement, the BUILD procedure of SAS/AF software, or the SAS Component Language (SCL) PMENU function with SAS/AF, SAS/CALC, and SAS/FSP software.

Under UNIX, the following options are ignored:

- ATTR= and COLOR= options in the TEXT statement. The colors and attributes for text and input fields are controlled by the CPARMS colors specified in the

SASCOLOR window. See “Customizing Colors in UNIX Environments” on page 84 for more information.

- ACCELERATE= and the MNEMONIC= options in the ITEM statement.

PRINTTO Procedure

Defines destinations for SAS procedure output and the SAS log

UNIX specifics: Valid values of *file specification*

See: PRINTTO Procedure in *Base SAS Procedures Guide*

Syntax

PROC PRINTTO <*option(s)*>

Note: This is a simplified version of the PRINTTO procedure syntax. For the complete syntax and its explanation, see the PRINTTO procedure in *Base SAS Procedures Guide*. △

LOG=*file-specification*
specifies a fully qualified pathname (in quotation marks), an environment variable, a fileref, or a file in the current directory (without extension).

PRINT=*file-specification*
specifies a fully qualified pathname (in quotation marks), an environment variable, a fileref, or a file in the current directory (without extension). If you specify a fileref that is defined with the PRINTER device-type keyword, output is sent directly to the printer.

Examples

The following statements send any SAS log entries that are generated after the RUN statement to the external file that is associated with the fileref MyFile:

```
filename myfile '/users/myid/mydir/mylog';
proc printto log=myfile;
run;
```

If MyFile has not been defined as a fileref, PROC PRINTTO will create the file MyFile.log in the current directory.

The following statements send any procedure output that is generated after the RUN statement to the file **/users/myid/mydir/myout**:

```
proc printto print='/users/myid/mydir/myout';
run;
```

The following statements send the procedure output from the CONTENTS procedure directly to the system printer:

```
filename myfile printer;
proc printto print=myfile;
run;

proc contents data=oranges;
run;
```

To redirect the SAS log and procedure output to their original default destinations, run PROC PRINTTO without any options:

```
proc printto;
run;
```

If MYPRINT and MYLOG have not been defined as filerefs, then the following statements send any SAS procedure output to MyPrint.lst and any log output to MyLog.log in the current directory:

```
proc printto print=myprint log=mylog;
run;
```

If filerefs MyPrint and MyLog had been defined, the output would have gone to the files associated with these filerefs.

See Also

□ Chapter 6, “Printing and Routing Output,” on page 153

SORT Procedure

Sorts observations in a SAS data set by one or more variables, then stores the resulting sorted observations in a new SAS data set or replaces the original data set

UNIX specifics: sort utilities available

See: SORT Procedure in *Base SAS Procedures Guide*

Syntax

PROC SORT<*option(s)*><*collating-sequence-option*>

Note: This is a simplified version of the SORT procedure syntax. For the complete syntax and its explanation, see the SORT procedure in *Base SAS Procedures Guide*. △

option(s)

SORTSIZE=*memory-specification*
specifies the maximum amount of memory available to the SORT procedure. For further explanation of the SORTSIZE= option, see the following Details section.

TAGSORT
stores only the BY variables and the observation number in temporary files. For further explanation of the TAGSORT option, see the following Details section.

Note: The TAGSORT option is ignored when used with a host sort. △

Details

The SORT procedure sorts observations in a SAS data set by one or more character or numeric variables, either replacing the original data set or creating a new, sorted data set. By default under UNIX, the SORT procedure uses the ASCII collating sequence.

The SORT procedure uses the sort utility specified by the SORTPGM system option. Sorting can be done by SAS or the **syncsort** utility. You can use all of the options available to the SAS sort utility, such as the SORTSEQ and NODUPKEY options. In some situations, you can improve your performance by using the NOEQUALS option. If you specify an option that is not supported by the host sort, then the SAS sort will be used instead. For more information about all of the options that are available, see the SORT procedure in *Base SAS Procedures Guide*.

SORTSIZE= Option

Limiting the Amount of Memory Available to PROC SORT

You can use the SORTSIZE= option in the PROC SORT statement to limit the amount of memory available to the SORT procedure. This option can reduce the amount of swapping SAS must do to sort the data set.

Note: If you do not specify the SORTSIZE= option, PROC SORT uses the value of the SORTSIZE system option. The SORTSIZE system option can be defined on the command line or in the SAS configuration file. △

Syntax of the SORTSIZE= Option

The syntax of the SORTSIZE= option is as follows:

SORTSIZE=*memory-specification*

where *memory-specification* can be one of the following:

n	specifies the amount of memory in bytes.
*n*K	specifies the amount of memory in 1-kilobyte multiples.
*n*M	specifies the amount of memory in 1-megabyte multiples.
*n*G	specifies the amount of memory in 1-gigabyte multiples.

Default Value of the SORTSIZE= Option

The default SAS configuration file sets this option based on the value of the SORTSIZE system option. The default for the SORTSIZE system option is MAX; however, the value of MAX depends on your operating system. To view the value of MAX for your operating environment, run the following code:

```
proc options option=sortsize;
run;
```

You can override the default value of the SORTSIZE system option by

- specifying a different SORTSIZE= value in the PROC SORT statement
- submitting an OPTIONS statement that sets the SORTSIZE system option to a new value
- setting the SORTSIZE system option on the command line during the invocation of SAS.

Improving Performance with the SORTSIZE= Option

In general, you should set the SORTSIZE= option no larger than the amount of physical memory available to the SAS process. If the SORTSIZE= value is larger than the

amount of available memory, then the operating system will be forced to page excessively. If the SORTSIZE= value is too small, then not all of the sorting can be done in memory, which also results in more disk I/O.

When the SORTSIZE= value is large enough to sort the entire data set in memory, you can achieve optimal sort performance. If the entire data set to be sorted will not fit in memory, SAS creates a temporary utility file to store the data. In this case, SAS uses a sort algorithm that is tuned to sort using disk space instead of memory.

Note: You can also use the SORTSIZE system option, which has the same effect as the SORTSIZE= option in the PROC SORT statement. △

TAGSORT Option

The TAGSORT option in the PROC SORT statement is useful when there might not be enough disk space to sort a large SAS data set. When you specify the TAGSORT option, only the sort keys (that is, the variables specified in the BY statement) and the observation number for each observation are stored in the temporary utility files. The sort keys, together with the observation number, are referred to as *tags*. At the completion of the sorting process, the tags are used to retrieve the records from the input data set in sorted order. Thus, in cases where the total number of bytes of the sort keys is small compared with the length of the record, temporary disk use is reduced considerably.

You must have enough disk space to hold an additional copy of the data set (the output data set) and the utility file that contains the tags. By default, this utility file is stored in the Work library. If this directory is too small, you can change this directory using the WORK system option. For more information, see “WORK System Option” on page 381.

Note: If you are using a host sort utility, then you can use the SORTDEV system option to change the location of your temporary files. For more information, see “SORTDEV System Option” on page 366. △

Note that while using the TAGSORT option may reduce temporary disk use, the processing time could be higher. However, on computers with limited available disk space, the TAGSORT option might enable sorts to be performed in situations where they would otherwise not be possible.

Disk Space Considerations for PROC SORT

You need to consider the following items when determining the amount of the disk space needed to run PROC SORT:

input SAS data set
: PROC sort uses the input data set specified by the DATA= option.

output SAS data set
: PROC SORT stores the output data set in the location specified by the OUT= option. If the OUT= option is not specified, PROC SORT stores the output SAS data set in the Work library.

utility file stored in the Work library
: This utility file is approximately the size of the input SAS data set.

temporary output SAS data set
: During the sort, PROC SORT creates its output in the directory specified in the OUT= option (or directory of the input data set if the OUT= option is not specified). The temporary data set has the same filename as the original data set, except it has an extension of .lck. After the sort completes successfully, the

original data set is deleted, and the temporary data set is renamed to match the original data set. Therefore, you need to have enough available space in the target directory to hold two copies of the data set.

You can reduce the amount of disk space needed by specifying the OVERWRITE option on the PROC SORT statement. When you specify this option, SAS replaces the input data set with the sorted data set. This option should only be used with a data set that is backed up or with a data set that you can reconstruct. For more information, see the SORT procedure in *Base SAS Procedures Guide*.

Performance Tuning for PROC SORT

How SAS Determines the Amount of Memory to Use Generally, SAS uses the memory value specified in the REALMEMSIZE system option. However, this value is limited by the SORTSIZE value (which is limited by the value of the MEMSIZE system option). If SORTSIZE is set to the default value of MAX, then PROC SORT uses the REALMEMSIZE value to determine the amount of memory to use. For information about setting the REALMEMSIZE system option, see "Guidelines for Setting the REALMEMSIZE System Option" on page 285.

Note: If you receive an out of memory error, then increase the value of MEMSIZE. For more information, see "MEMSIZE System Option" on page 344. △

Guidelines for Setting the REALMEMSIZE System Option Since PROC SORT uses the REALMEMSIZE system option to determine how much memory to use, it is important that the REALMEMSIZE value reflects the amount of memory that is available on your system. If REALMEMSIZE is set too high, then PROC SORT might use more memory than is actually available. Using too much memory will cause excessive paging and adversely impact system performance.

In general, REALMEMSIZE should be set to the amount of physical memory (not including swap space) that you expect to be available to SAS at run time. A good starting value is the amount of physical memory installed on the computer less the amount that is being used by running applications and the operating system. You can experiment with the REALMEMSIZE value until you reach optimum performance for your environment. In some cases, optimum performance can be achieved with a very low REALMEMSIZE value. A low value could cause SAS to use less memory and leave more memory for the operating system to perform I/O caching.

For more information, see "REALMEMSIZE System Option" on page 353.

Creating Your Own Collating Sequences

If you want to provide your own collating sequences or change a collating sequence provided for you, use the TRANTAB procedure to create or modify translation tables. For more information about the TRANTAB procedure, see *SAS National Language Support (NLS): User's Guide*. When you create your own translation tables, they are stored in your Sasuser.Profile catalog, and they override any translation tables by the same name that are stored in the Host catalog.

Note: System managers can modify the Host catalog by copying newly created tables from the Profile catalog to the Host catalog. Then all users can access the new or modified translation table. △

If you are using the SAS windowing environment and want to see the names of the collating sequences that are stored in the Host catalog, issue the following command from any window:

```
catalog sashelp.host
```

If you are not using the SAS windowing environment, then issue the following statements to generate a list of the contents of the Host catalog:

```
proc catalog catalog=sashelp.host;
contents;
run;
```

Entries of type TRANTAB are the collating sequences.

To see the contents of a particular translation table, use the following statements:

```
proc trantab table=table-name;
list;
run;
```

The contents of collating sequences are displayed in the SAS log.

Specifying the Host Sort Utility

Introduction to Using the Host Sort UNIX has one host sort utility, **syncsort**. You can use this sorting application as an alternative sorting algorithm to the SAS sort. SAS determines which sort to use by the values that are set for the SORTNAME, SORTPGM, SORTCUT, and SORTCUTP system options.

Setting the Host Sort Utility as the Sort Algorithm To specify a host sort utility as the sort algorithm, complete the following steps:

1. Specify the name of the host utility (**syncsort**) in the SORTNAME system option.
2. Set the SORTPGM system option to tell SAS when to use the host sort utility.
   - If SORTPGM=HOST, then SAS will always use the host sort utility.
   - If SORTPGM=BEST, then SAS chooses the best sorting method (either the SAS sort or the host sort) for the situation. For more information, see "Sorting Based on Size or Observations" on page 286.

Sorting Based on Size or Observations The sort routine that SAS uses can be based on either the number of observations in a data set or on the size of the data set. When the SORTPGM system option is set to BEST, SAS uses the first available and pertinent sorting algorithm based on this order of precedence:

- host sort utility
- SAS sort utility

SAS looks at the values for the SORTCUT and SORTCUTP system options to determine which sort to use.

The SORTCUT option specifies the number of observations above which the host sort utility is used instead of the SAS sort. The SORTCUTP option specifies the number of bytes in the data set above which the host sort utility is used.

If SORTCUT and SORTCUTP are set to zero, SAS uses the SAS sort routine. If you specify both options and either condition is met, SAS uses the host sort utility.

When the following OPTIONS statement is in effect, the host sort utility is used when the number of observations is 501 or greater:

```
options sortpgm=best sortcut=500;
```

In this example, the host sort utility is used when the size of the data set is greater than 40M:

```
options sortpgm=best sortcutp=40M;
```

For more information about these sort options, see "SORTCUT System Option" on page 364, "SORTCUTP System Option" on page 365, and "SORTPGM System Option" on page 368.

Changing the Location of Temporary Files Used by the Host Sort Utility By default, the host sort utilities use the location that is specified in the -WORK option for temporary files. To change the location of these temporary files, specify a location by using the SORTDEV system option. Here is an example:

```
options sortdev=''/tmp/host'';
```

For more information, see "SORTDEV System Option" on page 366.

Passing Options to the Host Sort Utility To specify options for the sort utility, use the SORTANOM system option. For a list of valid options, see "SORTANOM System Option" on page 363.

Passing Parameters to the Host Sort Utility To pass parameters to the sort utility, use the SORTPARM system option. The parameters that you can specify depend on the host sort utility. For more information, see "SORTPARM System Option" on page 367.

Specifying the SORTSEQ= Option with a Host Sort Utility

CAUTION:

If you are using a host sort utility to sort your data, then specifying the SORTSEQ= option might corrupt the character BY variables if the sort sequence translation table and its inverse are not one-to-one mappings. In other words for the sort to work, the translation table must map each character to a unique weight, and the inverse table must map each weight to a unique character variable. △

If your translation tables do not map one-to-one, then you can use one of the following methods to perform your sort:

- create a translation table that maps one-to-one. Once you create a translation table that maps one-to-one, you can easily create a corresponding inverse table using the TRANTAB procedure. If your table is not mapped one-to-one, then you will receive the following note in the SAS log when you try to create an inverse table:

  ```
  NOTE:  This table cannot be mapped one to one.
  ```

  For more information, see TRANTAB Procedure in *SAS National Language Support (NLS): User's Guide*.
- use the SAS sort. You can specify the SAS sort using the SORTPGM system option. For more information, see "SORTPGM System Option" on page 368.
- specify the collation order options of your host sort utility. See the documentation for your host sort utility for more information.
- create a view with a dummy BY variable. For an example, see "Example: Creating a View with a Dummy BY Variable" on page 287.

Note: After using one of these methods, you might need to perform subsequent BY processing using either the NOTSORTED option or the NOBYSORTED system option. For more information about the NOTSORTED option, see BY Statement in *SAS Language Reference: Dictionary*. For more information about the NOBYSORTED system option, see BYSORTED System Option in *SAS Language Reference: Dictionary*. △

Example: Creating a View with a Dummy BY Variable The following code is an example of creating a view using a dummy BY variable:

```
options no date nostimer ls=78 ps=60;
options sortpgm=host msglevel=i;
```

```
data one;
   input name $ age;
   datalines;
   anne 35
   ALBERT 10
   JUAN 90
   janet 5
   bridget 23
   BRIAN 45
   ;

data oneview / view=oneview;
   set one;
   name1=upcase(name);
run;

proc sort data=oneview out=final(drop=name1);
   by name1;
run;

proc print data=final;
run;
```

The output is the following:

Output 15.2 Creating a View with a Dummy BY Variable

```
  The SAS System
Obs        name       age
 1         ALBERT      10
 2         anne        35
 3         BRIAN       45
 4         bridget     23
 5         janet        5
 6         JUAN        90
```

See Also

- "TRANTAB Procedure" in *SAS National Language Support (NLS): User's Guide*
- "MEMSIZE System Option" on page 344
- "REALMEMSIZE System Option" on page 353
- "SORTANOM System Option" on page 363
- "SORTCUT System Option" on page 364
- "SORTCUTP System Option" on page 365
- "SORTDEV System Option" on page 366
- "SORTNAME System Option" on page 367
- "SORTPARM System Option" on page 367
- "SORTPGM System Option" on page 368
- "SORTSIZE System Option" on page 368
- "UTILLOC System Option" in *SAS Language Reference: Dictionary*

CHAPTER
16
Statements under UNIX

SAS Statements under UNIX

This section describes SAS statements that exhibit behavior or syntax that is specific to UNIX environments. Each statement description includes a brief “UNIX specifics” section that explains which aspect of the statement is specific to UNIX. If the information under the "UNIX specifics" says "all," then the statement is described only in this documentation. Otherwise, the statement is described in both this documentation and in *SAS Language Reference: Dictionary*.

Dictionary

ABORT Statement

Stops executing the current DATA step, SAS job, or SAS session

Valid: in a DATA step

UNIX specifics: values of *n*

See: ABORT Statement in *SAS Language Reference: Dictionary*

Syntax

ABORT <ABEND|RETURN><*n*>;

Details

The *n* option enables you to specify the value of the exit status code that SAS returns to the shell when it stops executing. The value of *n* can range from 0 to 255.

See Also

- "Determining the Completion Status of a SAS Job in UNIX Environments" on page 22

ATTRIB Statement

Associates a format, informat, label, and/or length with one or more variables

Valid: in a DATA step

UNIX specifics: length specification

See: ATTRIB Statement in *SAS Language Reference: Dictionary*

Syntax

ATTRIB *variable-list*-1 *attribute-list*-1 <...*variable-list-n attribute-list-n*>;

Note: Following is a simplified explanation of the ATTRIB statement syntax. For complete syntax and its explanation, see the ATTRIB statement in *SAS Language Reference: Dictionary*. △

attribute-list

LENGTH=<$>*length*
specifies the length of the variables in *variable-list*. The minimum length that you can specify for a numeric variable depends on the floating-point format used by your system. Because most systems use the IEEE floating-point format, the minimum is 3 bytes.

See Also

- Chapter 8, "Data Representation," on page 197

FILE Statement

Specifies the current output file for PUT statements

Valid: in a DATA step

UNIX specifics: valid values for *file-specification*, *host-options*, and *encoding-value*

See: FILE Statement in *SAS Language Reference: Dictionary*

Syntax

FILE *file-specification* <ENCODING='*encoding-value*' > <*options*> <*host-options*>;

file-specification
can be any of the file specification forms that are discussed in the "Accessing an External File or Device in UNIX Environments" on page 133.

ENCODING='*encoding-value*'
specifies the encoding to use when writing to the output file. The value for ENCODING= indicates that the output file has a different encoding from the current session encoding.

When you write data to the output file, SAS transcodes the data from the session encoding to the specified encoding.

For valid encoding values, see "Encoding Values in SAS Language Elements" in *SAS National Language Support (NLS): User's Guide*.

options
can be any of the options for the FILE statement that are valid in all operating environments. See the *SAS Language Reference: Dictionary* for a description of these options.

host-options
are specific to UNIX environments. These options can be any of the following:

BLKSIZE=
BLK=
specifies the number of bytes that are physically written in one I/O operation. The default is 8K. The maximum is 1G-1.

TERMSTR=
controls the end-of-line/record delimiters in PC and UNIX formatted files. This option enables the sharing of UNIX and PC formatted files between the two hosts. The following are values for the TERMSTR= option:

CRLF	Carriage Return Line Feed. This parameter is used to create PC format files.
NL	Newline. This parameter is used to create UNIX format files. NL is the default format.

Use TERMSTR=CRLF when you are writing to a file that you want to read on a PC. If you use this option when creating the file, then you do not need to use TERMSTR=NL when reading the file on the PC.

LRECL=
specifies the logical record length. Its meaning depends on the record format in effect (RECFM). The default is 256. The maximum length is 1G.

- If RECFM=F, then the value for the LRECL= option determines the length of each output record. The output record is truncated or padded with blanks to fit the specified size.
- If RECFM=N, then the value for the LRECL= option must be at least 256.
- If RECFM=V, then the value for the LRECL= option determines the maximum record length. Records that are longer than the specified length are divided into multiple records.

MOD
indicates that data written to the file should be appended to the file.

NEW | OLD
indicates that a new file is to be opened for output. If the file already exists, then it is deleted and re-created. If you specify OLD, then the previous contents of the file are replaced. NEW is the default.

RECFM=
specifies the record format. Values for the RECFM= option are

D	default format (same as variable).
F	fixed format. That is, each record has the same length. Do not use RECFM=F for external files that contain carriage-control characters.
N	binary format. The file consists of a stream of bytes with no record boundaries.
P	print format. SAS writes carriage-control characters.
V	variable format. Each record ends with a newline character.
S370V	variable S370 record format (V).
S370VB	variable block S370 record format (VB).
S370VBS	variable block with spanned records S370 record format (VBS).

UNBUF
tells SAS not to perform buffered writes to the file on any subsequent FILE statement. This option applies especially when you are writing to a data collection device.

Details

The ENCODING= option is valid only when the FILE statement includes a file specification that is not a reserved fileref. If the FILE statement includes the ENCODING= argument and the reserved filerefs Log or Print as the *file-specification*, then SAS issues an error message. The ENCODING= value in the FILE statement overrides the value of the ENCODING= system option.

You can set the permissions of the output file by issuing the **umask** command from within the SAS session. For more information, see “Executing Operating System Commands from Your SAS Session” on page 13.

See Also

- Chapter 5, “Using External Files and Devices,” on page 131

FILENAME Statement

Associates a SAS fileref with an external file or output device

Valid: anywhere

UNIX specifics: *device-type*, *external-file*, *host-options*, and *encoding-value*

See: FILENAME Statement in *SAS Language Reference: Dictionary*

Syntax

FILENAME *fileref* <*device-type*> *'external-file'* <ENCODING=*'encoding-value'*> <*host-options*>;

FILENAME *fileref device-type* <*'external-file'*> <ENCODING=*'encoding-value'*> <*'host-options'*>;

FILENAME *fileref* (*'pathname-1' ... 'pathname-n'*) <ENCODING=*'encoding-value'*> <*'host-options'*>;

FILENAME *fileref directory-name* <ENCODING=*'encoding-value'*>;

FILENAME *fileref* <*access-method*> *'external-file' access-information*;

FILENAME *fileref* CLEAR | _ALL_ CLEAR;

FILENAME *fileref* LIST | _ALL_ LIST;

fileref
is the name by which you reference the file. Under UNIX, the value of *fileref* can be an environment variable. See "Using Environment Variables to Assign Filerefs in UNIX Environments" on page 139 for more information.

device-type
specifies a device for the output, such as a disk, terminal, printer, pipe, and so on. The device-type keyword must follow *fileref* and precede *pathname*. Table 16.1 on page 296 describes the valid device types. DISK is the default device type. If you are associating the fileref with a DISK file, then you do not need to specify the device type.

'external-file'
differs according to device type. Table 16.1 on page 296 shows the information appropriate to each device. Remember that UNIX filenames are case sensitive. See "Specifying Pathnames in UNIX Environments" on page 133 for more information.

ENCODING=*'encoding-value'*
specifies the encoding to use when reading from or writing to the external file. The value for ENCODING= indicates that the external file has a different encoding from the current session encoding.

When you read data from an external file, SAS transcodes the data from the specified encoding to the session encoding. When you write data to an external file, SAS transcodes the data from the session encoding to the specified encoding.

For valid encoding values, see "Encoding Values in SAS Language Elements" in *SAS National Language Support (NLS): User's Guide*.

'host-options'
are specific to UNIX environments. These options can be any of the following:

BLKSIZE=
BLK=
specifies the number of bytes that are physically written or read in one I/O operation. The default is 8K. The maximum is 1G-1. If you specify RECFM=S370VBS, then you should specify BLKSIZE=32760 in order to avoid errors with records longer than 255 characters.

TERMSTR=
controls the end of line/record delimiters in PC and UNIX formatted files. This option enables the sharing of UNIX and PC formatted files between the two hosts. The following values for the TERMSTR= option:

CRLF	Carriage Return Line Feed. This parameter is used to read and write PC format files.
NL	Newline. This parameter is used to read and write UNIX format files. NL is the default format.

If you are working on UNIX and reading a file that was created on a PC, specify TERMSTR=CRLF unless the file was created with the TERMSTR=NL option. If you are writing a file that will be read on a PC, specify TERMSTR=CRLF.

If you are working on a PC and reading a file that was created on UNIX, specify TERMSTR=NL unless the file was created with the TERMSTR=CRLF option. If you are writing a file that will be read on UNIX, specify TERMSTR=NL.

LRECL=
specifies the logical record length. Its meaning depends on the record format in effect (RECFM). The default is 256. The maximum length is 1G.

- If RECFM=F, then the value for the LRECL= option determines either the number of bytes to be read as one record or the length of each output record. The output record is truncated or padded with blanks to fit the specified size.
- If RECFM=N, then the value for the LRECL= option must be at least 256.
- If RECFM=V, then the value for the LRECL= option determines the maximum record length. Records that are longer than the specified length are divided into multiple records on output and truncated on input.
- If RECFM=S370VBS, then you should specify LRECL=32760 in order to avoid errors with records longer than 255 characters.

MOD
indicates that data written to the file should be appended to the file.

NEW | OLD
indicates that a new file is to be opened for output. If the file already exists, then it is deleted and re-created. If you specify OLD, then the previous contents of the file are replaced. NEW is the default.

RECFM=
specifies the record format. Values for the RECFM= option are

D	default format (same as variable).
F	fixed format. That is, each record has the same length. Do not use RECFM=F for external files that contain carriage-control characters.
N	binary format. The file consists of a stream of bytes with no record boundaries.
P	print format. On output, SAS writes carriage-control characters.

V	variable format. Each record ends with a newline character.
S370V	variable S370 record format (V).
S370VB	variable block S370 record format (VB).
S370VBS	variable block with spanned records S370 record format (VBS). If you specify RECFM=S3270VBS, then you should specify BLDSIZE=32760 and LRECL=32760 in order to avoid errors with records longer than 255 characters.

The RECFM= option is used for both input and output.

UNBUF
tells SAS not to perform buffered writes to the file on any subsequent FILE statement. This option applies especially when you are reading from or writing to a data collection device. As explained in *SAS Language Reference: Dictionary*, it also prevents buffered reads on INFILE statements.

'pathname-1'...'pathname-n'
are pathnames for the files that you want to access with the same fileref. Use this form of the FILENAME statement when you want to concatenate filenames. Concatenation of filenames is available only for DISK files, so you do not have to specify the *device-type*. Separate the pathnames with either commas or blank spaces. Enclose each pathname in quotation marks. Table 4.6 on page 115 shows character substitutions you can use when specifying a pathname. If the fileref that you are defining is to be used for input, then you can also use wildcards as described in "Using Wildcards in Pathnames (Input Only)" on page 134. Remember that UNIX filenames are case-sensitive.

directory-name
specifies the directory that contains the files that you want to access. For more information, see "Assigning a Fileref to a Directory (Using Aggregate Syntax)" on page 138.

access-method
can be CATALOG, SOCKET, FTP, or URL. Table 16.1 on page 296 describes the information expected by these access methods.

access-information
differs according to the access method. Table 16.1 on page 296 shows the information appropriate to each access method.

CLEAR
clears the specified fileref or, if you specify _ALL_, clears all filerefs that are currently defined.

Note: You cannot clear a fileref that is defined by an environment variable. Filerefs that are defined by environment variables are assigned for the entire SAS session. △

ALL
refers to all filerefs currently defined. You can use this keyword when you are listing or clearing filerefs.

LIST
prints to the SAS log the pathname of the specified fileref or, if you specify _ALL_, lists the definition for all filerefs that are currently defined. Filerefs defined as environment variables appear only if you have already used those filerefs in a SAS statement. If you are using the Bourne shell or the Korn shell, SAS cannot determine the name of a preopened file, so it displays the following string instead of a filename:

```
<File Descriptor number>
```

See “Using Environment Variables to Assign Filerefs in UNIX Environments” on page 139 for more information.

Table 16.1 Device Information in the FILENAME Statement

Device or Access Method	Function	External-file
CATALOG	references a SAS catalog as an external file	is a valid two-, three-, or four-part SAS catalog name followed by catalog options (if needed). See *SAS Language Reference: Dictionary* for details.
DISK	associates the fileref with a DISK file	is either the pathname for a single file or, if you are concatenating filenames, a list of pathnames separated by blanks or commas and enclosed in parentheses. The level of specification depends on your location in the file system. Table 4.6 on page 115 shows character substitutions that you can use when specifying a UNIX pathname.
DUMMY	associates a fileref with a null device	None. DUMMY enables you to debug your application without reading from or writing to a device. Output to this device is discarded.
EMAIL	sends electronic mail to an address	is an address and e-mail options. See “Sending Electronic Mail Using the FILENAME Statement (EMAIL)” on page 143 for details.
FTP	reads or writes to a file from any machine on a network that is running an FTP server	is the pathname of the external file on the remote machine followed by FTP options. See the *SAS Language Reference: Dictionary* and “Assigning Filerefs to Files on Other Systems (FTP and SOCKET Access Types)” on page 137 for details. If you are transferring a file to UNIX from the z/OS operating environment and you want to use either the S370V or S370VB format to access that file, then the file must be of type RECFM=U and BLKSIZE=32760 before you transfer it. If you FTP to an z/OS machine, only one member of an z/OS PDS can be written to at a time. If you need to write to multiple members at the same time, an z/OS PDSE or a UNIX System Services directory should be used.
PIPE	reads input from or writes output to a UNIX command	is a UNIX command. See Chapter 6, “Printing and Routing Output,” on page 153 for details.
PLOTTER	sends output to a plotter	is a device name and plotter options. See “Using PRTFILE and PRINT with a Fileref” on page 160 and “Using the PRINTTO Procedure in UNIX Environments” on page 161 for details.
PRINTER	sends output to a printer	is a device name and printer option. See “Using PRTFILE and PRINT with a Fileref” on page 160 and “Using the PRINTTO Procedure in UNIX Environments” on page 161 for details.
SOCKET	reads and writes information over a TCP/IP socket	depends on whether the SAS application is a server application or a client application. In a client application, *external-file* is the name or IP address of the host and the TCP/IP port number to connect to followed by any TCP/IP options. In a server application, *external-file* is the port number to create for listening, followed by the SERVER keyword, and then any TCP/IP options. See the *SAS Language Reference: Dictionary* for details.

Device or Access Method	Function	External-file
TAPE	associates a fileref with a tape	is the pathname for a tape device. The name specified should be the name of the special file associated with the tape device. See "Processing Files on TAPE in UNIX Environments" on page 149 for more information.
TEMP	associates a fileref with an external file stored in the Work data library	None
TERMINAL	associates a fileref with a terminal	is the pathname of a terminal.
UPRINTER	sends output to the default printer that was set up through the Printer Setup dialog box	None
URL	enables you to use the URL of a file to access it remotely	is the name of the file that you want to read from or write to on a URL server. The URL must be in one of these forms: **http://*hostname/file*** **http://*hostname*:*portno/file***

See Also

- Chapter 5, "Using External Files and Devices," on page 131
- Chapter 6, "Printing and Routing Output," on page 153

FOOTNOTE Statement

Prints up to ten lines of text at the bottom of the procedure output

Valid: anywhere

UNIX specifics: maximum length of footnote

See: FOOTNOTE Statement in *SAS Language Reference: Dictionary*

Syntax

FOOTNOTE <*n*> <*'text'* | "*text*">;

Details

The maximum footnote length is 255 characters. If the length of the specified footnote is greater than the value of the LINESIZE option, SAS truncates the footnote to the line size.

%INCLUDE Statement

Includes and executes SAS statements and data lines

Valid: anywhere

UNIX specifics: *source*, if a file specification is used; valid values for *encoding-value*

See: %INCLUDE Statement in *SAS Language Reference: Dictionary*

Syntax

%INCLUDE *source-1* <*...source-n*> </<SOURCE2>
<S2=*length*><ENCODING='*encoding-value*'><*host-options*>>;

source
describes the location you want to access with the %INCLUDE statement. The three possible sources are a file specification, internal lines, or keyboard entry. The file specification can be any of the file specification forms that are discussed in "Accessing an External File or Device in UNIX Environments" on page 133.

Note: When using aggregate syntax, if the member name contains a leading digit, enclose the member name in quotation marks. If the member name contains a macro variable reference, use double quotation marks. △

ENCODING='*encoding-value*'
specifies the encoding to use when reading from the specified source. The value for ENCODING= indicates that the specified source has a different encoding from the current session encoding.

When you read data from the specified source, SAS transcodes the data from the specified encoding to the session encoding.

For valid encoding values, see "Encoding Values in SAS Language Elements" in *SAS National Language Support (NLS): User's Guide*.

host-options
consists of statement options that are valid under UNIX. The following options are available:

BLKSIZE=*block-size*
BLK=*block-size*
specifies the number of bytes that are physically read or written in an I/O operation. The default is 8K. The maximum is 1M.

LRECL=*record-length*
specifies the record length (in bytes). Under UNIX, the default is 256. The value of *record-length* can range from 1 to 1,048,576 (1 megabyte).

RECFM=*record-format*
specifies the record format. The following values are valid under UNIX:

D	default format (same as variable).
F	fixed format. That is, each record has the same length.
N	binary format. The file consists of a stream of bytes with no record boundaries.
P	print format.

V variable format. Each record ends with a newline character.

S370V variable S370 record format (V).

S370VB variable block S370 record format (VB).

S370VBS variable block with spanned records S370 record format (VBS).

The S370 values are valid with files laid out as z/OS files only. That is, files that are binary, have variable-length records, and are in EBCDIC format. If you want to use a fixed-format z/OS file, first copy it to a variable-length, binary z/OS file.

Details

If you specify any options on the %INCLUDE statement, remember to precede the options list with a forward slash (/).

See Also

- Chapter 5, "Using External Files and Devices," on page 131

INFILE Statement

Specifies an external file to be read with an INPUT statement

Valid: in a DATA step

UNIX specifics: valid values for *encoding-value*, *file-specification*, and *host-options*

See: INFILE Statement in *SAS Language Reference: Dictionary*

Syntax

INFILE *file-specification* <ENCODING='*encoding-value*'> <*options*> <*host-options*>;

file-specification
can be any of the file specification forms that are discussed in the "Accessing an External File or Device in UNIX Environments" on page 133.

ENCODING='*encoding-value*'
specifies the encoding to use when reading from the external file. The value for ENCODING= indicates that the external file has a different encoding from the current session encoding.

When you read data from an external file, SAS transcodes the data from the specified encoding to the session encoding.

For valid encoding values, see "Encoding Values in SAS Language Elements" in *SAS National Language Support (NLS): User's Guide*.

host-options
are specific to UNIX environments. These options can be any of the following:

BLKSIZE=
BLK=
specifies the number of bytes that are physically read in one I/O operation. The default is 8K. The maximum is 1G-1.

TERMSTR=
controls the end of line/record delimiters in PC and UNIX formatted files. This option enables the sharing of UNIX and PC formatted files between the two hosts. The following are values for the TERMSTR= option:

CRLF	Carriage Return Line Feed. This parameter is used to read PC format files.
NL	Newline. This parameter is used to read UNIX format files. NL is the default.

Use TERMSTR=CRLF to read a file that was created on the PC. If this PC format file was created using TERMSTR=NL, then the TERMSTR option is unnecessary.

LRECL=
specifies the logical record length. Its meaning depends on the record format in effect (RECFM). The default is 256. The maximum length is 1G.

- If RECFM=F, then the value for the LRECL= option determines the number of bytes to be read as one record.
- If RECFM=N, then the value for the LRECL= option must be at least 256.
- If RECFM=V, then the value for the LRECL= option determines the maximum record length. Records that are longer than the specified length are truncated.

RECFM=
specifies the record format. The following values are valid under UNIX:

D	default format (same as variable).
F	fixed format. That is, each record has the same length.
N	binary format. The file consists of a stream of bytes with no record boundaries.
P	print format.
V	variable format. Each record ends with a newline character.
S370V	variable S370 record format (V).
S370VB	variable block S370 record format (VB).
S370VBS	variable block with spanned records S370 record format (VBS).

Details

The ENCODING= option is valid only when the INFILE statement includes a file specification that is not a reserved fileref. If the INFILE statement includes the ENCODING= argument and the reserved filerefs DATALINES or DATALINES4 as a *file-specification*, then SAS issues an error message. The ENCODING= value in the INFILE statement overrides the value of the ENCODING= system option.

See Also

- Chapter 5, “Using External Files and Devices,” on page 131

LENGTH Statement

Specifies the number of bytes that SAS uses to store a variable’s value

Valid: in a DATA step

UNIX specifics: valid numeric variable lengths

See: LENGTH Statement in *SAS Language Reference: Dictionary*

Syntax

LENGTH <*variable-1*><*...variable-n*> <$> *length* <DEFAULT=*n*>

length
can range from 3 to 8 for numeric variables under UNIX. The minimum length you can specify for a numeric variable depends on the floating-point format used by your system. Because most systems use the IEEE floating-point format, the minimum is 3 bytes.

DEFAULT=*n*
changes the default number of bytes that are used for storing the values of newly created numeric variables from 8 to the value of *n*. Under UNIX, *n* can range from 3 to 8.

See Also

- Chapter 8, "Data Representation," on page 197

LIBNAME Statement

Associates or disassociates one or more SAS data libraries with a libref; lists the characteristics of a SAS data library

Valid: anywhere

UNIX specifics: *engine*, *library*, and *engine/host-options*

See: LIBNAME Statement in *SAS Language Reference: Dictionary*

Syntax

LIBNAME *libref* <*engine*> '*SAS-data-library*' <*options*> <*engine/host-options*>;

LIBNAME *libref* <*engine*> ('*library-1*'<,...'*library-n*'>) <*options*>;

LIBNAME *libref* ('*library-1*'|*libref-1*,...,'*library-n*'|*librefn*);

LIBNAME *libref* CLEAR|_ALL _ CLEAR;

LIBNAME *libref* LIST|_ALL _ LIST;

libref
is any valid libref as documented in *SAS Language Reference: Dictionary*. SAS reserves some librefs for special system libraries. See "Librefs Assigned by SAS in UNIX Environments" on page 118 for more information.

engine
is one of the library engines supported under UNIX. See “Details” on page 303 for a description of the engines. If no engine name is specified, SAS determines which engine to use as described in “Omitting Engine Names from the LIBNAME Statement” on page 304.

'SAS-data-library'
differs according to the engine that you specify and according to your current working directory. Table 16.2 on page 303 describes what each engine expects for this argument. Specify directory pathnames as described in “Specifying Pathnames in UNIX Environments” on page 114. You cannot create directories with the LIBNAME statement. The directory that you specify here must already exist, and you must have permissions to it. Enclose the data library name in quotation marks. Remember that UNIX pathnames are case-sensitive.

'library-n'|libref-n
are pathnames or librefs (that have already been assigned) for the data libraries that you want to access with one libref. Use these forms of the LIBNAME statement when you want to concatenate data libraries. Separate the pathnames with either commas or blank spaces. Enclose library pathnames in quotation marks. Do not enclose librefs in quotation marks. See “Assigning a Libref to Several Directories (Concatenating Directories)” on page 115 for more information.

options
are LIBNAME statement options that are available in all operating environments. See *SAS Language Reference: Dictionary* for information about these options.

engine/host-options
can be any of the options described in “Engine/Host Options” on page 305.

ALL
refers to all librefs currently defined. You can use this keyword when you are listing or clearing librefs.

CLEAR
clears the specified libref or, if you specify _ALL_, clears all librefs that are currently defined. Sasuser, Sashelp, and Work remain assigned.

Note: When you clear a libref defined by an environment variable, the variable remains defined, but it is no longer considered a libref. You can still reuse it, either as a libref or a fileref. See “Using Environment Variables as Librefs in UNIX Environments” on page 117 for more information. △

SAS automatically clears the association between librefs and their respective data libraries at the end of your job or session. If you want to associate an existing libref with a different SAS data library during the current session, you do not have to end the session or clear the libref. SAS automatically reassigns the libref when you issue a LIBNAME statement for the new SAS data library.

LIST
prints to the SAS log the engine, pathname, file format, access permissions, and so on, that are associated with the specified libref or, if you specify _ALL_, prints this information for all librefs that are currently defined. Librefs defined as environment variables appear only if you have already used those librefs in a SAS statement.

Details

There are two main types of engines:

View engines
: enable SAS to read SAS data views that are described by SAS/ACCESS software, the SQL procedure, and DATA step views. The use of SAS view engines is automatic because the name of the view engine is stored as part of the descriptor portion of the SAS data set.

Library engines
: control access at the SAS data library level. Every SAS data library has an associated library engine, and the files in that library can be accessed only through that engine. There are two types of library engines:

  native engines
  : access SAS files created and maintained by SAS. See the following table for a description of these engines.

  interface engines
  : treat other vendors' files as if they were SAS files. See the following table and "Accessing BMDP, OSIRIS, or SPSS Files in UNIX Environments" on page 125 for more information.

Table 16.2 Engine Names and Descriptions

Engine Type	Name (Alias)	Description	SAS-data-library
default	V9 (BASE) V8	enables you to create new SAS data files and access existing SAS data files that were created with Version 8 or SAS 9. The V8 and V9 engines are identical. This engine enables read access to data files that were created with some earlier versions of SAS, but this engine is the only one that supports SAS 9 catalogs. This engine allows for data set indexing and compression and is also documented in *SAS Language Reference: Dictionary*.	is the pathname of the directory containing the library.
sequential	V9TAPE (TAPE) V8TAPE	accesses SAS data files that were created in a sequential format, whether on tape or on disk. This engine requires less overhead than the default engine because sequential access is simpler than random access. This engine is also documented in *SAS Language Reference: Dictionary*.	is the name of the special file (see "Introduction to External Files and Devices in UNIX Environments" on page 132) associated with the sequential device, such as **`/dev/rmt/0mn`**.
	V6TAPE	accesses V6 SAS data files that were created in a sequential format. This engine is read-only.	is the name of the special file (see "Introduction to External Files and Devices in UNIX Environments" on page 132) associated with the sequential device, such as **`/dev/rmt/0mn`**.
compatibility	V6	accesses any data file that was created by Releases 6.09 through 6.12. This engine is read-only.	is the pathname of the directory containing the library.

Engine Type	Name (Alias)	Description	SAS-data-library
servers	SPDS	enables communication between a client session and a data server. You must have the Scalable Performance Data Server licensed on your client machine to use this engine. Refer to *Scalable Performance Data Server User's Guide, Version 2* for more information.	is the logical LIBNAME domain name for an SPDS data library on the server machine. The name server resolves the domain name into the physical path for the library.
	MDDB	enables communication between a client session and an MDDB server. You must have SAS/MDDB Server licensed either or your client machine or on your server machine to use this engine. Refer to *SAS MDDB Server Software: Administration Guide* for complete information.	
transport	XPORT	accesses transport data sets. This engine creates machine-independent SAS transport files that can be used under all hosts running Release 6.06 or later of SAS. This engine is documented in *Moving and Accessing SAS Files*.	is the pathname of either a sequential device or a disk file.
XML	XML	generates (writes) and processes (reads) any XML document, which is an application- and machine-independent file.	is the pathname of the XML document.
interface	BMDP	provides read-only access to BMDP files. This engine is available only on AIX, HP-UX, and Solaris.	is the pathname of the data file.
	OSIRIS	provides read-only access to OSIRIS files.	is the pathname of the data file.
	SPSS	provides read-only access to SPSS files	is the pathname of the data file.

Omitting Engine Names from the LIBNAME Statement It is always more efficient to specify the engine name than to have SAS determine the correct engine. However, if you omit an engine name in the LIBNAME statement or if you define an environment variable to serve as a libref, SAS determines the appropriate engine.

If you have specified the ENGINE= system option, SAS uses the engine name that you specified. See "ENGINE System Option" on page 327 for a discussion of the ENGINE= system option.

Note: The ENGINE= system option specifies the default engine for data libraries on disk only. △

If you did not specify the ENGINE= system option, SAS looks at the extensions of the files in the given directory and uses these rules to determine an engine:

- If all the SAS data sets in the library were created by the same engine, the libref is assigned using that engine.

  Note: If the engine used to create the data sets is not the same as the default engine, then you will not be able to create a view or stored program. See "Using Multiple Engines for a Library in UNIX Environments" on page 116 for more information. △

- If there are no SAS data sets in the given directory, the libref is assigned using the default engine.
- If there are SAS data sets from more than one engine, the system issues a message about finding mixed engine types and assigns the libref using the default engine.

Engine/Host Options The LIBNAME statement accepts the following options:

FILELOCKS=NONE | FAIL | CONTINUE
specifies whether file locking is on or off for the library that you are defining. This LIBNAME statement option works like the FILELOCKS system option, except that it applies only to the library that you are defining. See "FILELOCKS System Option" on page 329 for more information.

You can also specify any of the options supported by the SPDS engines. SPDS is the Scalable Performance Data Server. Refer to *Scalable Performance Data Server User's Guide, Version 2* for a description of these options.

See Also

- Chapter 4, "Using SAS Files," on page 101

SYSTASK Statement

Executes asynchronous tasks

Valid: anywhere

UNIX specifics: all

Syntax

SYSTASK COMMAND "*host-command*"
<WAIT|NOWAIT>
<TASKNAME=*taskname*>
<MNAME=*name-variable*>
<STATUS=*status-variable*>
<SHELL<="*shell-command*">>
<CLEANUP>;

SYSTASK LIST <_ALL_ | *taskname*> <STATE> <STATVAR>;

SYSTASK KILL *taskname* <*taskname*...>;

COMMAND
executes the *host-command*.

LIST
lists either a specific active task or all of the active tasks in the system. A task is *active* if it is running or if it has completed and has not been waited for using the WAITFOR statement.

KILL
forces the termination of the specified task(s).

host-command
specifies the name of a UNIX command (including any command-specific options) or the name of an X Windows or Motif application. Enclose the command in either single or double quotation marks. If the command options require quotation marks, repeat them for each option. For example:

```
SYSTASK COMMAND "xdialog -m ""There was an error."" -t ""Error"" -o";
```

Note: The *host-command* that you specify cannot require input from the keyboard. △

WAIT | NOWAIT
determines whether SYSTASK COMMAND suspends execution of the current SAS session until the task has completed. NOWAIT is the default. For tasks that start with the NOWAIT option, you can use the WAITFOR statement when necessary to suspend execution of the SAS session until the task has finished. See "WAITFOR Statement" on page 308.

TASKNAME=*taskname*
specifies a name that identifies the task. Task names must be unique among all active tasks. A task is *active* if it is running or if it has completed and has not been waited for using the WAITFOR statement. Duplicate task names generate an error in the SAS log. If you do not specify a task name, SYSTASK will automatically generate a name. If the task name contains a blank character, enclose it in quotation marks.

Task names cannot be reused, even if the task has completed, unless you either issue the WAITFOR statement for the task or you specify the CLEANUP option.

MNAME=*name-variable*
specifies a macro variable in which you want SYSTASK to store the task name that it automatically generated for the task. If you specify both the TASKNAME option and the MNAME option, SYSTASK copies the name that you specified with TASKNAME into the variable that you specified with MNAME.

STATUS=*status-variable*
specifies a macro variable in which you want SYSTASK to store the status of the task. Status variable names must be unique among all active tasks.

SHELL<="*shell-command*">
specifies that the *host-command* should be executed with the host shell command. If you specify a *shell-command*, SYSTASK uses the shell command that you specify to invoke the shell; otherwise, SYSTASK uses the default shell. Enclose the shell command in quotes.

Note: The SHELL option assumes that the shell command that you specify uses the **-i** option to pass statements. Usually, your shell command will be **sh**, **csh**, **ksh**, or **bash**. △

CLEANUP
specifies that the task should be removed from the LISTTASK output when the task completes. You can then reuse the task name without issuing the WAITFOR statement.

Details

SYSTASK enables you to execute host-specific commands from within your SAS session or application. Unlike the X statement, SYSTASK runs these commands as

asynchronous tasks, which means that these tasks execute independently of all other tasks that are currently running. Asynchronous tasks run in the background, so you can perform additional tasks while the asynchronous task is still running.

For example, to start a new shell and execute the UNIX **cp** command in that shell, you might use this statement:

```
systask command "cp /tmp/sas* ~/archive/" taskname="copyjob1"
                status=copysts1 shell;
```

The return code from the **cp** command is saved in the macro variable COPYSTS1.

The output from the command is displayed in the SAS log.

Note: Program steps that follow the SYSTASK statements in SAS applications usually depend on the successful execution of the SYSTASK statements. Therefore, syntax errors in some SYSTASK statements will cause your SAS application to abort. △

There are two types of asynchronous processes that can be started from SAS:

Task
: All tasks started with SYSTASK COMMAND are of type Task. For these tasks, if you do not specify STATVAR or STATE, then SYSTASK LIST displays the task name, type, and state, and the name of the status macro variable. You can use SYSTASK KILL to kill only tasks of type Task.

SAS/CONNECT Process
: Tasks started from SAS/CONNECT with the RSUBMIT statement are of type SAS/CONNECT Process. For SAS/CONNECT processes, SYSTASK LIST displays the task name, type, and state. You can use SYSTASK KILL to kill a SAS/CONNECT process. For information about starting SAS/CONNECT processes, refer to *SAS/CONNECT User's Guide*.

  Note: The preferred method for displaying any task (not just SAS/CONNECT processes) is to use the LISTTASK statement instead of SYSTASK LIST. The preferred method for ending a task is using the KILLTASK statement in place of SYSTASK KILL. △

The SYSRC macro variable contains the return code for the SYSTASK statement. The status variable that you specify with the STATUS option contains the return code of the process started with SYSTASK COMMAND. To ensure that a task executes successfully, you should monitor both the status of the SYSTASK statement and the status of the process that is started by the SYSTASK statement.

If a SYSTASK statement cannot execute successfully, the SYSRC macro variable will contain a non-zero value. For example, there might be insufficient resources to complete a task or the SYSTASK statement might contain syntax errors. With the SYSTASK KILL statement, if one or more of the processes cannot be killed, SYSRC is set to a non-zero value.

When a task is started, its status variable is set to NULL. You can use the status variables for each task to determine which tasks failed to complete. Any task whose status variable is NULL did not complete execution. If a task terminates abnormally, then its status variable will be set to **-1**. See "WAITFOR Statement" on page 308 for more information about the status variables.

Unlike the X statement, you cannot use the SYSTASK statement to start a new interactive session.

See Also

- "WAITFOR Statement" on page 308
- "X Statement" on page 310

□ “Executing Operating System Commands from Your SAS Session” on page 13

TITLE Statement

Specifies title lines for SAS output

Valid: anywhere
UNIX specifics: maximum length of title
See: TITLE Statement in *SAS Language Reference: Dictionary*

Syntax

TITLE <*n*> <*'text'* | "*text*">;

Details

In interactive modes, the maximum title length is 254 characters; otherwise, the maximum length is 200 characters. If the length of the specified title is greater than the value of the LINESIZE option, the title is truncated to the line size.

WAITFOR Statement

Suspends execution of the current SAS session until the specified tasks finish executing

Valid: anywhere
UNIX specifics: all

Syntax

WAITFOR <_ANY | _ALL_> *taskname* <*taskname*...> <TIMEOUT=*seconds*>;

taskname
specifies the name of the task(s) that you want to wait for. See “SYSTASK Statement” on page 305 for information about task names. The task name(s) that you specify must match exactly the task names assigned through the SYSTASK COMMAND statement. You cannot use wildcards to specify task names.

ANY | _ALL_
suspends execution of the current SAS session until either one or all of the specified tasks finishes executing. The default setting is _ANY_, which means that as soon as one of the specified task(s) completes executing, the WAITFOR statement will finish executing.

TIMEOUT=*seconds*
specifies the maximum number of seconds that WAITFOR should suspend the current SAS session. If you do not specify the TIMEOUT option, WAITFOR will suspend execution of the SAS session indefinitely.

Details

The WAITFOR statements suspends execution of the current SAS session until the specified task(s) finish executing or until the TIMEOUT= interval (if specified) has elapsed. If the specified task was started with the WAIT option, then the WAITFOR statement ignores that task. See "SYSTASK Statement" on page 305 for a description of the WAIT option.

For example, the following statement starts three different X Windows programs and waits for them to complete:

```
systask command "xv" taskname=pgm1;
systask command "xterm" taskname=pgm2;
systask command "xcalc" taskname=pgm3;
waitfor _all_ pgm1 pgm2 pgm3;
```

The WAITFOR statement can be used to execute multiple concurrent SAS sessions. The following statements start three different SAS jobs and suspend the execution of the current SAS session until those three jobs have finished executing:

```
systask command "sas myprog1.sas" taskname=sas1;
systask command "sas myprog2.sas" taskname=sas2;
systask command "sas myprog3.sas" taskname=sas3;
waitfor _all_ sas1 sas2 sas3;
```

Note: In this method, SAS terminates after each command which can result in reduced performance. SAS/CONNECT can also be used for executing parallel SAS sessions. See *SAS/CONNECT User's Guide* for more information. △

The SYSRC macro variable contains the return code for the WAITFOR statement. If a WAITFOR statement cannot execute successfully, the SYSRC macro variable will contain a non-zero value. For example, the WAITFOR statement may contain syntax errors. If the number of seconds specified with the TIMEOUT option elapses, then the WAITFOR statement finishes executing, and SYSRC is set to a non-zero value if

- you specify a single task that does not finish executing
- you specify more than one task and the _ANY_ option (which is the default setting), but none of the tasks finishes executing
- you specify more than one task and the _ALL_ option, and any one of the tasks does not finish executing.

Any task whose status variable is still NULL after the WAITFOR statement has executed did not complete execution. See "SYSTASK Statement" on page 305 for a description of status variables for individual tasks.

See Also

- "SYSTASK Statement" on page 305
- "X Statement" on page 310
- *SAS/CONNECT User's Guide*
- "Executing Operating System Commands from Your SAS Session" on page 13

X Statement

Issues an operating system command from within a SAS session

Valid: anywhere

UNIX specifics: valid operating system command

See: X Statement in *SAS Language Reference: Dictionary*

Syntax

X <'*host-command*'>;

host-command
specifies the UNIX command. If you specify only one UNIX command, you do not need to enclose it in quotation marks. Also, if you are running SAS from the Korn shell, you cannot use aliases.

Details

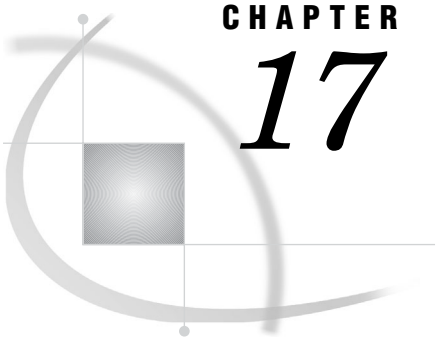
The X statement issues a UNIX command from within a SAS session. SAS executes the X statement immediately.

Neither the X statement nor the %SYSEXEC macro program statement is intended for use during the execution of a DATA step. The CALL SYSTEM routine, however, can be executed within a DATA step. See "CALL SYSTEM Routine" on page 239 for an example.

Note: The X statement is not supported without arguments under the X Window System. △

See Also

- "Executing Operating System Commands from Your SAS Session" on page 13



CHAPTER

17

System Options under UNIX

SAS System Options under UNIX

This section describes SAS system options that have behavior or syntax that is specific to UNIX environments. Each system option description includes a brief "UNIX specifics" section that explains which aspect of the system option is specific to UNIX. If the information under "UNIX specifics" is "all," then the system option is described only in this documentation. Otherwise, the system option is described in both this documentation and in *SAS Language Reference: Dictionary*.

See "Summary of All SAS System Options in UNIX Environments" on page 384 for a table of all of the system options available under UNIX.

Determining How a System Option Was Set

Because of the relationship between some SAS system options, SAS might modify an option's value. This modification might change your results.

To determine how an option was set, enter the following code in the SAS Program Editor:

```
proc options option=option value;
run;
```

After you submit this code, the SAS log will explain how the option was set. For example, the following output is displayed when you enter

```
proc options option=CATCACHE value;
run;
```

Output 17.1 Log Output for the CATCACHE System Option

```
Option Value Information for SAS Option CATCACHE
      Option Value:  0
      Option Scope:  Default
      How option value was set:  Shipped Default
```

Options that are set by SAS will often say "Internal" in the **How option value was set** field. Some SAS options are only internal. You cannot specify an internal option as the **option=** value in the preceding code. If you do, SAS will return an error stating an unrecognized option value.

Dictionary

ALTLOG System Option

Specifies the destination for an alternate SAS log

Default: no alternate SAS log is made

Valid in: configuration file, SAS invocation, SASV9_OPTIONS environment variable

Category: Environment control: Files
PROC OPTIONS GROUP= ENVFILES
UNIX specifics: all

Syntax

–ALTLOG *destination* | –NOALTLOG

-ALTLOG *destination*
specifies the location where an alternate SAS log is to be sent. The *destination* argument can be any valid UNIX path to a directory, a filename, or an environment variable that is associated with a path. If you specify only the path to a directory, the SAS log is placed in a file in the specified directory. The name of the file will be *filename*.log, where *filename* is the name of your SAS job. If you are running SAS interactively and specify only the path to a directory, the log is written to a file named Sas.log within that path.

-NOALTLOG
causes any previous ALTLOG specifications to be ignored.

Details

All messages that are written to the SAS log are written to the location specified in *destination*. You can use this option to capture procedure output for printing.

Note: You can use the LOG option in the PRINTTO procedure to redirect any portion of the log to an external file. The code for PROC PRINTTO will not appear in the SAS log for the current session, but it will appear in the SAS log that you created with the ALTLOG system option. △

See Also

- “ALTPRINT System Option” on page 314
- “PRINTTO Procedure” on page 281
- “Using SAS System Options to Route Output” on page 163

ALTPRINT System Option

Specifies the destination for a copy of the SAS procedure output file

Default: no copy of SAS output is made
Valid in: configuration file, SAS invocation, SASV9_OPTIONS environment variable
Category: Environment control: Files
PROC OPTIONS GROUP= ENVFILES
UNIX specifics: all

Syntax

-ALTPRINT *destination* | –NOALTPRINT

-ALTPRINT *destination*
specifies the location where a copy of the procedure output is to be sent. The *destination* can be any valid UNIX path to a directory, a filename, or an environment variable that is associated with a path. If you specify only the path to a directory, the copy is place in a file in the specified directory. The file's name will be *filename*.lst, where *filename* is the name of your SAS job. If you are running SAS interactively and specify only the path to a directory, the filename is Sas.lst.

-NOALTPRINT
causes any previous ALTPRINT specifications to be ignored.

Details

All messages that are written to the SAS procedure output file are also written to the location specified in *destination*. You can use this option to capture the procedure output for printing.

See Also

- "ALTLOG System Option" on page 313
- "Using SAS System Options to Route Output" on page 163

APPEND System Option

Appends the specified pathname to the existing value of the specified system option

Default: none
Valid in: configuration file, SAS invocation, SASV9_OPTIONS environment variable
PROC OPTIONS GROUP= ENVFILES
UNIX specifics: all

Syntax

-APPEND *system-option new-pathname*

system-option
can be HELPLOC, MAPS, MSG, SASAUTOS, SASHELP.

new-pathname
is the new pathname that you want to append to the current value of *system-option*.

Details

By default, if you specify the HELPLOC, MAPS, MSG, SASAUTOS, or SASHELP system option more than one time, the last value that is specified is the value that SAS uses. If you want to add additional pathnames to the pathnames already specified by one of these options, you must use the APPEND system option to add the new pathname. For example, if you entered the following SAS command, the only location

that SAS will look for help files is **/apps/help** and the output of PROC OPTIONS will show only **/apps/help**:

```
sas -helploc /sas/help -helploc /apps/help
```

If you want SAS to look in both locations for help files, you must use the APPEND option:

```
sas -helploc /sas/help -append helploc /apps/help
```

For the value of the HELPLOC option, PROC OPTIONS will now show

```
('/sas/help' '/apps/help')
```

See Also

- "INSERT System Option" on page 337

AUTOEXEC System Option

Specifies the autoexec file to be used

Default: **autoexec.sas** (see Details below)
Valid in: configuration file, SAS invocation, SASV9_OPTIONS environment variable
Category: Environment control: Files
PROC OPTIONS GROUP= ENVFILES
UNIX specifics: all

Syntax

-AUTOEXEC *filename* | -NOAUTOEXEC

-AUTOEXEC *filename*
specifies the autoexec file to be used. The *filename* must resolve to a valid UNIX pathname.

-NOAUTOEXEC
specifies that SAS is not to process any autoexec files.

Details

The autoexec file contains SAS statements that are executed automatically when you invoke SAS or when you start another SAS process. The autoexec file can contain any SAS statements. For example, your autoexec file can contain LIBNAME statements for SAS data libraries that you access routinely in SAS sessions.

SAS looks for this option in the following order:

1. on the command line
2. in the SASV9_OPTIONS environment variable
3. in the configuration file.

It uses the first AUTOEXEC option it encounters and ignores all others.

If neither AUTOEXEC nor NOAUTOEXEC is specified, SAS searches three directories for an autoexec.sas file in the following order:

1. your current directory
2. your home directory
3. the **!SASROOT** directory (see Appendix 1, "The !SASROOT Directory," on page 397).

SAS uses the first file it finds to initialize the session.

If you want to see the contents of the autoexec file for your session, use the ECHOAUTO system option when you invoke SAS.

See Also

- "Customizing Your SAS Session Using System Options" on page 18

AUTOSAVELOC System Option

Specifies the location of the Program Editor autosave file

Default: none

Valid in: configuration file, SAS invocation, OPTIONS statement, SAS System Options window

Category: Environment control: Display

PROC OPTIONS GROUP= ENVDISPLAY

UNIX specifics: valid values of *pathname*

Syntax

-AUTOSAVELOC *fileref* | *pathname*

AUTOSAVELOC= *fileref* | *pathname*

fileref
specifies a fileref to the location where the autosave file is saved.

pathname
specifies the pathname of the autosave file. The *pathname* must be a valid UNIX pathname.

Details

By default, SAS saves the Program Editor autosave file, pgm.asv, in the current folder. You can use the AUTOSAVELOC system option to specify a different location for the autosave file.

See Also

- "SETAUTOSAVE Command" on page 215

BATCH System Option

Specifies whether batch settings for LINESIZE, PAGESIZE, and SOURCE are in effect when SAS executes

Default: NOBATCH

Valid in: configuration file, SAS invocation, SASV9_OPTIONS environment variable

Category: Environment control: Initialization and operation

PROC OPTIONS GROUP= EXECMODES

UNIX specifics: default setting

See: BATCH System Option in *SAS Language Reference: Dictionary*

Syntax

-BATCH | -NOBATCH

-BATCH
specifies that SAS use the batch settings of LINESIZE=132, PAGESIZE=60, and SOURCE. At the start of an interactive SAS session, you can use the BATCH setting to simulate the behavior of the system in batch mode.

-NOBATCH
specifies that SAS not use the batch settings for LINESIZE, PAGESIZE, and SOURCE. While in batch mode, you can specify NOBATCH to use the default (nonbatch) settings for the options LINESIZE, PAGESIZE, and NOSOURCE.

Details

The setting of the BATCH option does not specify the method of operation. BATCH only sets the appropriate batch settings for a collection of options that are in effect when SAS executes. The LINESIZE and PAGESIZE option values for batch mode can be dynamically set based on terminal characteristics when executing interactively.

In addition, the BATCH system option does not specify whether a terminal is present.

See Also

- "LINESIZE System Option" on page 338
- "PAGESIZE System Option" on page 350

BUFNO System Option

Specifies the number of buffers to be allocated for processing SAS data sets

Default: 1

Valid in: configuration file, SAS invocation, OPTIONS statement, SAS System Options window, SASV9_OPTIONS environment variable

Category: Files: SAS Files

PROC OPTIONS GROUP= SASFILES, PERFORMANCE

UNIX specifics: default value

See: BUFNO System Option in *SAS Language Reference: Dictionary*

Syntax

-BUFNO *n* | *n*K | *n*M| *n*G | *hex*X | MIN | MAX

BUFNO=*n* | *n*K | *n*M| *n*G| *hex*X | MIN | MAX

***n* | *n*K | *n*M | *n*G**
specifies the number of buffers in multiples of 1 (bytes); 1,024 (kilobytes); 1,048,576 (megabytes); or 1,073,741,824 (gigabytes). You can specify decimal values for the number of kilobytes, megabytes, or gigabytes. For example, a value of **`8`** specifies 8 buffers, a value of **`.782k`** specifies 801 buffers, and a value of **`3m`** specifies 3,145,728 buffers.

***hex*X**
specifies the number of buffers as a hexadecimal value. You must specify the value beginning with a number (0–9), followed by hex digits (0–9, A-F), and then followed by an X. For example, **`2dx`** specifies 45 buffers.

MIN
sets the number of buffers to 0, and requires SAS to use the default value of 1.

MAX
sets the number of buffers to 2,147,483,647.

Details

The number of buffers is not a permanent attribute of the data set; it is valid only for the current SAS session or job.

BUFNO= applies to SAS data sets that are opened for input, output, or update.

Using BUFNO= can improve execution time by limiting the number of input/output operations that are required for a particular SAS data set. The improvement in execution time, however, comes at the expense of increased memory consumption.

Under UNIX, the maximum number of buffers that you can allocate is determined by the amount of memory available.

BUFSIZE System Option

Specifies the permanent buffer page size for output SAS data sets

Default: 0

Valid in: configuration file, SAS invocation, OPTIONS statement, SAS System Options window, SASV9_OPTIONS environment variable

Category: Files: SAS Files

PROC OPTIONS GROUP= SASFILES, PERFORMANCE

UNIX specifics: valid range
See: BUFSIZE System Option in *SAS Language Reference: Dictionary*

Syntax

-BUFSIZE *n* | *n*K | *n*M | *n*G | *hex*X | MIN | MAX
BUFSIZE=*n* | *n*K | *n*M | *n*G | *hex*X | MIN | MAX

***n* | *n*K | *n*M | *n*G**
specifies the buffer page size in multiples of 1 (bytes); 1,024 (kilobytes); 1,048,576 (megabytes); or 1,073,741,824 (gigabytes). You can specify decimal values for the number of kilobytes, megabytes, or gigabytes. For example, a value of **8** specifies 8 bytes, a value of **.782k** specifies 801 bytes, and a value of **3m** specifies 3,145,728 bytes.

***hex*X**
specifies the buffer page size as a hexadecimal value. You must specify the value beginning with a number (0–9), followed by hex digits (0–9, A-F), and then followed by an X. For example, **2dx** sets the buffer page size to 45 bytes.

MIN
sets the buffer page size to 0. When the buffer size is 0, the BASE engine calculates a buffer size to optimize CPU and I/O use. This size is the smallest multiple of 8K that can hold 80 observations but is not larger than 64K.

MAX
sets the buffer page size to 2,147,483,647.

Details

The buffer page size can range from 1K to 2G-1.

If you specify a nonzero value when you create a SAS data set, the BASE engine uses that value. If that value cannot hold at least one observation or is not a multiple of 1K, the engine rounds the value up to a multiple of 1K.

CATCACHE System Option

Specifies the number of SAS catalogs to keep open

Default: 0
Valid in: configuration file, SAS invocation, SASV9_OPTIONS environment variable
Category: Files: SAS Files
PROC OPTIONS GROUP= SASFILES
UNIX specifics: Valid values for *n*
See: CATCACHE System Option in *SAS Language Reference: Dictionary*
SAS Language Reference: Dictionary

Syntax

-CATCACHE *n* | *n*K | MIN | MAX

***n* | *n*K**
specifies the number of open-file descriptors to keep in cache memory in multiples of 1 (*n*) or 1,024 (*n*K). You can specify decimal values for the number of kilobytes. For example, a value of **8** specifies 8 open-file descriptors, a value of **.782k** specifies 801 open-file descriptors, and a value of **3k** specifies 3,072 open-file descriptors.

If $n > 0$, SAS places up to that number of open-file descriptors in cache memory instead of closing the catalogs.

MIN
sets the number of open-file descriptors that are kept in cache memory to 0.

MAX
sets the number of open-file descriptors that are kept in cache memory to 32,767.

Details

By using the CATCACHE system option to specify the number of SAS catalogs to keep open, you can avoid the repeated opening and closing of the same catalogs.

If SAS is running on an z/OS server and the MINSTG system option is in effect, SAS sets the value of CATCACHE to 0.

See Also

- The section on optimizing system performance in *SAS Language Reference: Concepts*

CLEANUP System Option

Specifies how to handle out-of-resource conditions

Default: CLEANUP for interactive modes; NOCLEANUP otherwise

Valid in: configuration file, SAS invocation, OPTIONS statement, SASV9_OPTIONS environment variable

Category: Environment control: Error handling

PROC OPTIONS GROUP= ERRORHANDLING

UNIX specifics: behavior when running in interactive line mode and batch mode

See: CLEANUP System Option in *SAS Language Reference: Dictionary*

Syntax

-CLEANUP| -NOCLEANUP

CLEANUP | NOCLEANUP

CLEANUP
specifies that during the entire session, SAS attempts to perform automatic, continuous clean up of resources that are not essential for execution. Nonessential resources include those that are not visible to the user (for example, cache memory) and those that are visible to the user (for example, the KEYS windows).

CLEANUP does not prompt you before SAS attempts to clean up your disk. However, when an out-of-disk-space condition occurs and your display is attached to the process, you are prompted with a menu selection even if the CLEANUP option is on. If you do not want to be prompted for out-of-disk-space conditions, use the CLEANUP option in conjunction with the NOTERMINAL option.

When the CLEANUP option is on, SAS performs automatic continuous cleanup. If not enough resources are recovered, the request for the resource fails, and an appropriate error message is written to the SAS log.

CLEANUP is the default in batch mode because there is no display attached to the process to accommodate prompting.

NOCLEANUP
specifies that SAS allows the user to choose how to handle an out-of-resource condition. When NOCLEANUP is in effect and SAS cannot execute because of a lack of resources, SAS automatically attempts to clean up resources that are not visible to the user (for example, cache memory). However, resources that are visible to the user (for example, the KEYS windows) are not automatically cleaned up. Instead, SAS prompts you before attempting to clean up your disk.

Details

The CLEANUP system option indicates whether you should be prompted with a menu of items to be cleaned up when SAS encounters an out-of-resource condition. In batch mode, SAS ignores this option, and if an out-of-resource condition occurs, the SAS session terminates.

CONFIG System Option

Specifies the name of an alternative configuration file

Default: `sasv9.cfg` (see "Order of Precedence for SAS Configuration Files" on page 17)

Valid in: configuration file, SAS invocation, SASV9_OPTIONS environment variable, SASV9_CONFIG environment variable

Category: Environment control: Files

PROC OPTIONS GROUP= ENVFILES

UNIX specifics: all

Syntax

-CONFIG *filename* | –NOCONFIG

-CONFIG *filename*
specifies a configuration file to be read. The *filename* must resolve to a valid UNIX filename.

-NOCONFIG
specifies that any previous CONFIG specification should be ignored and that the default system options should be used.

Details

Configuration files contain system option specifications that execute automatically whenever SAS is invoked.

If you specify the CONFIG= system option in a configuration file, the option is ignored.

See Also

□ "Customizing Your SAS Session Using System Options" on page 18

DBCS System Option

Recognizes double-byte character sets (DBCS)

Default: NODBCS

Valid in: configuration file, SAS invocation, SASV9_OPTIONS environment variable

Category: Environment control: Language control

PROC OPTIONS GROUP= LANGUAGECONTROL

UNIX specifics: all

See: DBCS System Option in *SAS National Language Support (NLS): User's Guide*

DBCSLANG System Option

Specifies a double-byte character set (DBCS) language

Default: NONE

Valid in: configuration file, SAS invocation, SASV9_OPTIONS environment variable

Category: Environment control: Language control

PROC OPTIONS GROUP= LANGUAGECONTROL

UNIX specifics: all

See: DBCSLANG System Option in *SAS National Language Support (NLS): User's Guide*

DBCSTYPE System Option

Specifies a double-byte character set (DBCS) encoding method

Default: Depends on UNIX environment

Valid in: configuration file, SAS invocation, SASV9_OPTIONS environment variable

Category: Environment control: Language control

PROC OPTIONS GROUP= LANGUAGECONTROL

UNIX specifics: all

See: DBCSTYPE System Option in *SAS National Language Support (NLS): User's Guide*

DEVICE System Option

Specifies a device driver for graphics output for SAS/GRAPH software

Default: none

Valid in: configuration file, SAS invocation, OPTIONS statement, SAS System Options window, SASV9_OPTIONS environment variable, GOPTIONS statement

Category: Graphics: Driver settings

PROC OPTIONS GROUP= GRAPHICS

UNIX specifics: valid device drivers

See: DEVICE System Option in *SAS Language Reference: Dictionary*

Syntax

-DEVICE *device-driver-name*

DEVICE=*device-driver-name*

device-driver-name
specifies the name of a device driver for graphics output.

Details

To see the list of device drivers that are available under UNIX, you can use the GDEVICE procedure. If you are using the SAS windowing environment, submit the following statements:

```
proc gdevice catalog=sashelp.devices;
run;
```

If you are running SAS in interactive line mode or batch mode, submit the following statements:

```
proc gdevice catalog=sashelp.devices nofs;
   list _all_;
run;
```

See Also

- *SAS/GRAPH Reference, Volumes 1 and 2*

ECHO System Option

Specifies a message to be echoed to the computer

Default: NOECHO
Valid in: configuration file, SAS invocation, SASV9_OPTIONS environment variable
Category: Log and procedure output control: SAS log
PROC OPTIONS GROUP= LOGCONTROL
UNIX specifics: all

Syntax

-ECHO "*message*" | -NOECHO

ECHO "*message*"
specifies the text of the message to be echoed to the computer. The text must be enclosed in single or double quotation marks if the message is more than one word. Otherwise, the quotation marks are not needed.

NOECHO
specifies that no messages are to be echoed to the computer.

Details

Messages that result from errors in the autoexec file are printed in the SAS log regardless of how the ECHO system option is set.

You can specify multiple ECHO options. The strings are displayed in the order in which SAS encounters them. See "How SAS Processes System Options Set in Multiple Places" on page 20 for information on how that order is determined.

Example

For example, you can specify the following:

```
-echo ''SAS 9.1 under UNIX is initializing.''
```

The message appears in the LOG window as SAS initializes.

See Also

- "ECHOAUTO System Option" in *SAS Language Reference: Dictionary*

EDITCMD System Option

Specifies the host editor to be used with the HOSTEDIT command

Default: none
Valid in: configuration file, SAS invocation, OPTIONS statement, SASV9_OPTIONS environment variable
Category: Environment control: Display
PROC OPTIONS GROUP= ENVDISPLAY
UNIX specifics: all

Syntax

-EDITCMD "*host-editor-pathname editor-options*"

EDITCMD="*host-editor-pathname editor-options*"

Details

The EDITCMD system option specifies the command that is issued to the operating environment. If you are using a terminal-based editor, such as vi, you must specify a command that runs the editor inside a terminal emulator window.

You can define the EDITCMD option using the SASV9_OPTIONS environment variable as part of a configuration file or on the command line to make the definition available automatically to SAS. The option must be specified as a quoted string. You can use either single or double quotation marks. You can change the value for the EDITCMD option during a SAS session by issuing an OPTIONS statement.

The host editor that you specify is used when you issue the HOSTEDIT command. The HOSTEDIT command is valid only when you are running SAS in a windowing environment.

If you do not specify the full pathname, SAS searches the pathnames specified in the $PATH environment variable. Quotation marks are needed only when you specify editor options. If you are using a terminal-based editor, you must specify a command that runs the editor inside a terminal emulator window. For example, to use vi, you would specify

```
sas -editcmd "/usr/bin/X11/xterm -e /usr/bin/vi"
```

See Also

- "Configuring SAS for Host Editor Support in UNIX Environments" on page 49

EMAILSYS System Option

Specifies which email system to use for sending electronic mail from within SAS

Default: SMTP (Simple Mail Transfer Protocol)

Valid in: configuration file, SAS invocation

Category: Communications: Email

PROC OPTIONS GROUP= EMAIL

UNIX specifics: all

Syntax

-EMAILSYS SMTP | *name-of-script*

SMTP
specifies the Simple Mail Transfer Protocol electronic mail interface.

name-of-script
specifies which script to use for sending electronic mail from within SAS. Some external scripts do not support sending e-mail attachments. These scripts are not supported by SAS.

Details

The EMAILSYS system option specifies which e-mail system to use for sending electronic mail from within SAS. Specifying SMTP supports sending e-mail attachments on UNIX, but might require changing the values of the EMAILHOST= and EMAILPORT=, depending on your site configuration.

See Also

- "Sending Mail from within Your SAS Session in UNIX Environments" on page 47
- "Sending Electronic Mail Using the FILENAME Statement (EMAIL)" on page 143
- "EMAILHOST System Option" in *SAS Language Reference: Dictionary*
- "EMAILPORT System Option" in *SAS Language Reference: Dictionary*
- "The SMTP E-mail Interface" in *SAS Language Reference: Concepts*

ENCODING System Option

Specifies the default character-set encoding for processing external data

Default: latin1
Valid in: configuration file, SAS invocation
Category: Environment control: Language control
PROC OPTIONS GROUP= LANGUAGECONTROL
UNIX specifics: valid values
See: ENCODING= System Option in *SAS National Language Support (NLS): User's Guide*

ENGINE System Option

Specifies the default access method to use for SAS libraries

Default: V9
Valid in: configuration file, SAS invocation, SASV9_OPTIONS environment variable
Category: Files: SAS Files
PROC OPTIONS GROUP= SASFILES
UNIX specifics: valid values of *engine-name*

Syntax

-ENGINE *engine-name*

engine-name
can be one of the following under UNIX:

BASE | V9
specifies the default SAS engine for SAS 9 and SAS 9.1 files.

V8
specifies the SAS engine for all SAS Version 8 files.

V7
specifies the SAS engine for all Version 7 files.

V6
specifies the SAS engine for Release 6.09 through Release 6.12. This engine is read-only.

TAPE | V9TAPE
specifies the default sequential engine for SAS 9 and SAS 9.1 files.

V8TAPE | V7TAPE
specifies the SAS sequential engine for all Version 8 and Version 7 files. These engines are identical to the V9TAPE engine.

V6TAPE
specifies the SAS sequential engine for Version 6 files. This engine is read-only.

See Also

- “Accessing SAS Files across Compatible Machine Types in UNIX Environments” on page 108
- “ENGINE System Option” in *SAS Language Reference: Dictionary*
- *SAS Language Reference: Concepts*

FILELOCKS System Option

Specifies whether file locking is on or off and what action should be taken if a file cannot be locked

Default: FAIL

Valid in: configuration file, SAS invocation, SASV9_OPTIONS environment variable

Category: Environment control: Files

PROC OPTIONS GROUP= ENVFILES

UNIX specifics: all

Syntax

-FILELOCKS NONE | FAIL | CONTINUE

NONE
turns file locking off. SAS files are not protected from shared update access.

FAIL
turns file locking on. If a file cannot be locked, an attempt to open it fails.

CONTINUE
turns file locking on. If a file is already locked by someone else, an attempt to open it fails. If the file cannot be locked for some other reason, the file is opened and a warning message is sent to the log.

Details

When FILELOCKS is set to FAIL, SAS prevents these situations:

- two SAS sessions simultaneously opening the same SAS file for update or output.
- one SAS session reading a SAS file that another SAS session has open for update or output.
- one SAS session writing to a file that another SAS session has open in read mode.

With file locking on, multiple SAS sessions will be able to simultaneously read the same SAS file.

For file locking to work on some hosts, you must have the host's file locking service running. This usually involves having a lock daemon (such as **`lockd`**) and a stat daemon (such as **`statd`**) running. You might also need to execute other commands. For specifics, refer to the man pages or system administration instructions for your host.

In addition, if you are working with NFS-mounted files, the "file locking service" must be running both on your local host and on the remote host.

To prevent data corruption, setting FILELOCKS to NONE or CONTINUE is not recommended.

See Also

- "WORKINIT System Option" on page 382

FONTSLOC System Option

Specifies the directory that contains the SAS fonts that are loaded by some Universal Printer drivers

Default: `!SASROOT/misc/font`
Valid in: configuration file, SAS invocation
Category: Environment control: Display
PROC OPTIONS GROUP= ENVDISPLAY
UNIX specifics: valid pathname

Syntax

FONTSLOC "*directory-specification*"

"*directory-specification*"
specifies the directory that contains the SAS fonts that are loaded during the SAS session. The *directory-specification* must be enclosed in double quotation marks.

Details

The directory must be a valid operating system path name.

FSDBTYPE System Option

Specifies full-screen double-byte character set (DBCS) encoding method

Default: DEFAULT
Valid in: configuration file, SAS invocation, SASV9_OPTIONS environment variable
Category: Environment control: Language control
PROC OPTIONS GROUP= LANGUAGECONTROL
UNIX specifics: all
See: FSDBTYPE System Option in *SAS National Language Support (NLS): User's Guide*

FSIMM System Option

Specifies full-screen double-byte character set (DBCS) input method modules (IMMs)

Default: none
Valid in: configuration file, SAS invocation, SASV9_OPTIONS environment variable
Category: Environment control: Language control

PROC OPTIONS GROUP= LANGUAGECONTROL
UNIX specifics: all
See: FSIMM System Option in *SAS National Language Support (NLS): User's Guide*

FSIMMOPT System Option

Specifies options for full-screen double-byte character set (DBCS) input method modules

Default: none
Valid in: configuration file, SAS invocation, SASV9_OPTIONS environment variable
Category: Environment control: Language control
PROC OPTIONS GROUP= LANGUAGECONTROL
UNIX specifics: all
See: FSIMMOPT System Option in *SAS National Language Support (NLS): User's Guide*

FULLSTIMER System Option

Writes all available system performance statistics to the SAS log

Default: NOFULLSTIMER
Valid in: configuration file, SAS invocation, OPTIONS statement, SASV9_OPTIONS environment variable
Category: Log and procedure output control: SAS log
PROC OPTIONS GROUP= LOGCONTROL
UNIX specifics: all

Syntax

-FULLSTIMER | -NOFULLSTIMER

FULLSTIMER | NOFULLSTIMER

FULLSTIMER
writes to the SAS log a list of the host dependent resources that were used for each step and for the entire SAS session.

NOFULLSTIMER
does not write to the SAS log a complete list of resources.

Details

SAS calls the **`getrusage()`** and **`times()`** UNIX system calls for your operating environment to obtain the statistics presented with FULLSTIMER. The following is an example of FULLSTIMER output.

Output 17.2 FULLSTIMER Output

```
real time             1.34 seconds
user cpu time         0.04 seconds
system cpu time       0.29 seconds
Memory                                208k
Page Faults                           116
Page Reclaims                         656
Page Swaps                            0
Voluntary Context Switches            601
Involuntary Context Switches          24
Block Input Operations                10
Block Output Operations               7
```

Note: If both FULLSTIMER and STIMER are set, the FULLSTIMER statistics are printed. △

FULLSTIMER displays the following statistics:

Table 17.1 Description of FULLSTIMER Statistics

Statistic	Description
Real Time	the amount of time spent to process the SAS job. Real time is also referred to as elapsed time.
User CPU	the CPU time spent to execute your SAS code.
System CPU	the CPU time spent to perform system overhead tasks on behalf of the SAS process.
Memory	the amount of memory required to run a step.
Page Faults	the number of pages that SAS tried to access but were not in main memory and required I/O activity.
Page Reclaims	the number of pages that can be accessed without I/O activity.
Page Swaps	the number of times a process was swapped out of main memory.
Voluntary Context Switches	the number of times that the SAS process had to give up on the CPU because of a resource constraint such as a disk drive.
Involuntary Context Switches	the number of times that the operating system forced a process into an inactive state.
Block Input Operations	the number of I/O operations performed to read the data into memory.
Block Output Operations	the number of I/O operations performed to write the data to file.

For more information about these statistics, see the man pages for the **`getrusage()`** and **`times()`** UNIX system calls for your operating environment.

Note: Starting in SAS 9, some procedures use multiple threads. On computers with multiple CPUs, the operating system can run more than one thread simultaneously. Consequently, CPU time might exceed real time in your FULLSTIMER output.

For example, a SAS procedure could use two threads that run on two separate CPUs simultaneously. The value of CPU time would be calculated as the following:

```
CPU1 time + CPU2 time = total CPU time
1 second + 1 second = 2 seconds
```

Since CPU1 can run a thread at the same time that CPU2 runs a separate thread for the same SAS process, you can theoretically consume 2 CPU seconds in 1 second of real time. △

See Also

- "STIMER System Option" on page 376

GISMAPS System Option

Specifies the location of the SAS data library that contains U.S. Census Tract maps supplied by SAS/GIS

Default: GISMAPS, if defined

Valid in: configuration file, SAS invocation, SASV9_OPTIONS environment variable, LIBNAME statement

Category: Graphics: Driver settings

PROC OPTIONS GROUP= GRAPHICS

UNIX specifics: valid values for *library-specification* and *path-to-library*

See: GISMAPS= System Option in *SAS Language Reference: Dictionary*

Syntax

GISMAPS=*library-specification* | *path-to-library*

library-specification* | *path-to-library
specifies either a library or a physical path to a library that contains U.S. Census Tract maps supplied by SAS/GIS.

HELPINDEX System Option

Specifies one or more index files to be used by SAS Help and Documentation

Default: **`/help/common.hlp/index.txt`**, **`/help/common.hlp/keywords.htm`**, **`common.hhk`**

Valid in: configuration file, SAS invocation

Category: Environment control: Help

PROC OPTIONS GROUP= HELP

UNIX specifics: applet and HTML files must reside in the path specified by the HELPLOC option

Syntax

HELPINDEX = *index-pathname-1* < *index-pathname-2* < *index-pathname-3*>>

index-pathname
specifies the partial pathname for the index that is to be used by SAS Help and Documentation. The *index-pathname* can be any or all of the following:

/help/*applet-index-filename*
specifies the partial pathname of the index file that is to be used by the SAS Documentation Java applet in a UNIX environment. *applet-index-filename* must have a file extension of .txt and it must reside in a path that is specified by the HELPLOC system option. The default is **`/help/common.hlp/index.txt`**.

See the default index file for the format that is required for an index file.

/help/*accessible-index-filename*
specifies the partial pathname of an accessible index file that is to be used by SAS Help and Documentation in UNIX, OpenVMS, or z/OS environments. An accessible index file is an HTML file that can be used by Web browsers. *accessible-index-filename* must have a file extension of .htm and it must reside in a path that is specified by the HELPLOC system option. The default pathname is **`/help/common.hlp/keywords.htm`**.

See the default index file for the format that is required for an index file.

HTML-Help-index-pathname
specifies the pathname of the Microsoft HTML Help index that is to be used by SAS Help and Documentation in Windows environments. The default pathname is **`common.hhk`**. For information about creating an index for Microsoft HTML Help, see your Microsoft HTML Help documentation.

Details

Use the HELPINDEX option if you have a customized index that you want to use in place of the SAS-supplied index. If you use one configuration file to start SAS in more than one operating environment, you can specify all of the partial pathnames in the HELPINDEX option. The order of the pathnames is not important, although only one pathname of each type can be specified.

When the HELPINDEX option specifies a pathname for UNIX, OpenVMS, or z/OS operating environments, SAS determines the complete path by replacing **`/help/`** in the partial pathname with the pathname specified in the HELPLOC option. If the HELPLOC option contains more than one pathname, SAS searches each path for the specified index.

For example, when the value of HELPINDEX is **`/help/common.hlp/myindex.htm`** and the value of HELPLOC is **`/u/myhome/myhelp`**, the complete path to the index is **`/u/myhome/myhelp/common.hlp/myindex.htm`**.

See Also

□ “HELPLOC System Option” on page 334

HELPLOC System Option

Specifies the location of the text and index files for the facility that is used to view SAS Help and Documentation

Default: **`!SASROOT/X11/native_help`**

Valid in: configuration file, SAS invocation

Category: Environment control: Help

PROC OPTIONS GROUP= HELP

UNIX specifics: default *pathname*

Syntax

-HELPLOC (*pathname*<,*pathname-2*...,*pathname-n*>)

pathname
specifies one or more directory pathnames in which SAS Help and Documentation files are located.

Details

Specifying a value for the HELPLOC system option causes SAS to insert that value at the start of a list of concatenated values, the last of which is the default value. This enables you to access the help for your site without losing access to SAS Help and Documentation.

To insert or append pathnames by specifying an additional HELPLOC option, you must use the INSERT or APPEND system option.

Example

The following command contains two specifications of HELPLOC:

```
sas -helploc /app1/help -helploc /app2/help
```

The value of the system option is the following:

```
/app1/help, /app2/help, !SASROOT/X11/native_help
```

See Also

- "INSERT System Option" on page 337
- "APPEND System Option" on page 315

HELPTOC System Option

Specifies the table of contents files to be used by SAS Help and Documentation

Default: **`/help/helpnav.hlp/config.txt`**, **`/help/common.hlp/toc.htm`**, **`common.hhc`**

Valid in: configuration file, SAS invocation

Category: Environment control: Help

PROC OPTIONS GROUP= HELP

UNIX specifics: applet and HTML files must reside in the path specified by the HELPLOC option

Syntax

HELPTOC = *TOC-pathname-1 < TOC-pathname-2 < TOC-pathname-3>>*

TOC-pathname
specifies a partial pathname for the table of contents that is to be used by SAS Help and Documentation. *TOC-pathname* can be any or all of the following:

/help/*applet-TOC-filename*
specifies the partial pathname of the table of contents file that is to be used by the SAS Documentation Java applet in a UNIX environment. The *applet-TOC-filename* must have a file extension of .txt and it must reside in a path that is specified by the HELPLOC system option. The default is **/help/helpnav.hlp/config.txt**.

See the default table of contents file for the format that is required for an index file.

/help/*accessible-TOC-filename*
specifies the partial pathname of an accessible table of contents file that is to be used by SAS Help and Documentation in UNIX, OpenVMS, or z/OS environments. An accessible table of contents file is an HTML file that can be used by Web browsers. The *accessible-TOC-filename* have a file extension of .htm and it must reside in a path that is specified by the HELPLOC system option. The default pathname is **/help/common.hlp/toc.htm**.

See the default table of contents file for the format that is required for a table of contents.

HTML-Help-TOC-pathname
specifies the complete pathname to the Microsoft HTML Help table of contents that is to be used by SAS Help and Documentation in Windows environments. The default pathname is **common.hhc**. For information about creating an index for Microsoft HTML Help, see your Microsoft HTML Help documentation.

Details

Use the HELPTOC option if you have a customized table of contents that you want to use in place of the SAS supplied table of contents. If you use one configuration file to start SAS in more than one operating environment, you can specify all of the partial pathnames in the HELPTOC option. The order of the pathnames is not important, although only one pathname of each type can be specified.

When the HELPTOC option specifies the pathname for UNIX, OpenVMS, or z/OS operating environments, SAS determines the complete path by replacing **/help/** in the partial pathname with the pathname specified in the HELPLOC option. If the HELPLOC option contains more than one pathname, SAS searches each path for the table of contents.

For example, when HELPTOC is **/help/common.hlp/mytoc.htm** and the value of HELPLOC is **/u/myhome/myhelp**, the complete path to the table of contents is **/u/myhome/myhelp/common.hlp/mytoc.htm**.

See Also

□ “HELPLOC System Option” on page 334

INSERT System Option

Inserts the specified pathname at the beginning of the value of the specified system option

Default: none

Valid in: configuration file, SAS invocation, SASV9_OPTIONS environment variable

PROC OPTIONS GROUP= ENVFILES

UNIX specifics: all

Syntax

-INSERT *system-option new-pathname*

system-option
can be HELPLOC, MAPS, MSG, SASAUTOS, or SASHELP.

new-pathname
is the new pathname that you want to insert at the front of the current value of *system-option*.

Details

By default, if you specify the HELPLOC, MAPS, MSG, SASAUTOS, or SASHELP system option more than one time, the last specification is the one that SAS uses. If you want to insert additional pathnames to front of the search paths already specified by one of these options, you must use the INSERT system option to add the new pathname. For example, if you entered the following SAS command, the only location that SAS will look for help files is **/apps/help** and the output of PROC OPTIONS will show only **/apps/help**:

```
sas -helploc /apps/help
```

If you want SAS to look in both the current path for help files and in **/apps/help** and if you want SAS to search **/apps/help** first, then you must use the INSERT option:

```
sas -insert helploc /apps/help
```

If your current path for help files is **!SASROOT/X11/native_help**, then for the value of the HELPLOC option, PROC OPTIONS will now show

```
('/apps/help' '!SASROOT/X11/native_help')
```

See Also

- "APPEND System Option" on page 315

JREOPTIONS System Option

Identifies Java Runtime Environment (JRE) options for SAS

Default: none
Valid in: configuration file, SAS invocation
Category: Environment control: Initialization and operation
PROC OPTIONS GROUP= EXECMODES
UNIX specifics: all

Syntax

-JREOPTIONS (*-JRE-option-1*<*-JRE-option-n*>)

JREOPTIONS (*-JRE-option-1*<*-JRE-option-n*>)

-JRE-option
specifies one or more Java Runtime Environment options. JRE options must begin with a hyphen (-). Use a space to separate multiple JRE options. Valid values for *JRE-option* depend on your installation's Java Runtime Environment. For information about JRE options, see your installation's Java documentation.

Details

The set of JRE options must be enclosed in parentheses. If you specify multiple JREOPTIONS system options, SAS appends JRE options to JRE options that are currently defined. Incorrect JRE options are ignored.

Examples

```
-jreoptions (-verbose)
```

```
-jreoptions (-Djava.class.path=myjava/classes/myclasses.jar:myjava2/
             classes/myclasses.jar -oss600k)
```

LINESIZE System Option

Specifies the line size of the SAS Log and Output windows

Default: the display width setting for the interactive modes; 132 for batch mode
Valid in: configuration file, SAS invocation, OPTIONS statement, SAS System Options window, SASV9_OPTIONS environment variable
Category: Log and procedure output control: SAS log and procedure output
PROC OPTIONS GROUP= LOG_LISTCONTROL
UNIX specifics: default values
See: LINESIZE System Option in *SAS Language Reference: Dictionary*

Syntax

-LINESIZE *n* | *hex*X | MIN | MAX

LINESIZE=*n* | *hex*X | MIN | MAX

n
specifies the line size in characters. Valid values range between 64 and 256.

***hex*X**
specifies the line size as a hexadecimal value. You must specify the value beginning with a number (0–9), followed by hex digits (0–9, A-F), and then followed by an X. For example, **`2dx`** specifies 45 characters.

MIN
sets the line size to 64 characters.

MAX
sets the line size to 256 characters.

See Also

- "Controlling the Content and Appearance of Output in UNIX Environments" on page 165

LOADMEMSIZE System Option

Specifies a suggested amount of memory needed for executable programs loaded by SAS

Default: 0
Valid in: configuration file, SAS invocation, SASV9_OPTIONS environment variable
Category: System administration: Memory
PROC OPTIONS GROUP= MEMORY
UNIX specifics: all

Syntax

-LOADMEMSIZE *n* | *n*K | *n*M | *n*G | *hex*X | MIN | MAX

***n* | *n*K | *n*M | *n*G**
specifies the memory size in multiples of 1 (bytes); 1,024 (kilobytes); 1,048,576 (megabytes); or 1,073,741,842 (gigabytes). You can specify decimal values for the number of kilobytes, megabytes, or gigabytes. For example, a value of **`8`** specifies 8 bytes, a value of **`.782k`** specifies 801 bytes, and a value of **`3m`** specifies 3,145,728 bytes.

***hex*X**
specifies the amount of memory as a hexadecimal value. You must specify the value beginning with a number (0–9), followed by hex digits (0–9, A-F), and then followed by an X. For example, **`2dx`** sets the amount of memory to 45 bytes.

MIN
specifies 0 bytes, which indicates that there is no limit on the total amount of memory that can be used.

MAX
specifies that the maximum amount of memory for executable programs is limited only by the amount of memory available.

Details

An executable program is a complied executable. If SAS needs to use a function that is stored in an executable program, SAS loads the executable program. Then SAS can access the function that is compiled within the executable program.

When LOADMEMSIZE is set to 0, the memory that is used for executable programs that are loaded by SAS is limited only by the amount of system memory available. If LOADMEMSIZE is set to 1, executable programs are purged from memory when they are no longer in use.

For values of two or greater, SAS first checks the amount of memory available for SAS executable programs. If the total amount of memory available is greater than the value of LOADMEMSIZE, SAS purges the SAS loaded executable programs that are not in use until the memory that is used is less than the value of the LOADMEMSIZE option, or until there are no other SAS loaded executable programs that can be purged. If all executable programs have been purged and more memory is needed, additional system memory is used as long as it is available.

LOCALE System Option

Specifies attributes that reflect the language, local conventions, and culture for a geographic region and that are used to establish the default working environment for a SAS session

Default: English

Valid in: configuration file, SAS invocation, OPTIONS statement, SAS System Options window, SASV9_OPTIONS environment variable

Category: Environment control: Language control

PROC OPTIONS GROUP= LANGUAGECONTROL

UNIX specifics: all

See: LOCALE= System Option in *SAS National Language Support (NLS): User's Guide*

LOG System Option

Specifies a destination to which the SAS log is written in batch mode

Default: a file in the current directory with the same filename as the SAS source file and an extension of .log.

Valid in: configuration file, SAS invocation, SASV9_OPTIONS environment variable

Category: Environment control: Files

PROC OPTIONS GROUP= ENVFILES

UNIX specifics: all

Syntax

-LOG *destination* | -NOLOG

-LOG *destination*
specifies the destination for the SAS log. The *destination* can be a any valid UNIX path to a directory, a filename, or an environment variable that is associated with a path. If you specify only the path to a directory, the log file is created in the specified directory. The default name for this file is *filename*.log, where *filename* is the name of your SAS job.

-NOLOG
suppresses the creation of the SAS log. Do not use this value unless your SAS program is thoroughly debugged.

Details

LOG is valid in batch mode; it is ignored in the interactive modes.

See Also

□ "Using SAS System Options to Route Output" on page 163

LPTYPE System Option

Specifies which UNIX command and options settings will be used to route files to the printer

Default: SYSV

Valid in: configuration file, SAS invocation, OPTIONS statement, SASV9_OPTIONS environment variable

Category: Log and procedure output: Procedure output

PROC OPTIONS GROUP= LISTCONTROL

UNIX specifics: all

Syntax

-LPTYPE BSD | SYSV

LPTYPE=BSD | SYSV

Details

The LPTYPE option determines whether SAS is to use the **`lpr`** or the **`lp`** UNIX command to print files.

The LPTYPE option has two forms:

-LPTYPE BSD
causes SAS to use the **lpr** command to send files to the printer. The **lpr** command is usually supported on UNIX operating systems developed at Berkeley, such as HP-UX.

-LPTYPE SYSV
causes SAS to use the **lp** command to send files to the printer. The **lp** command is usually supported on operating systems derived from UNIX System V, such as Solaris.

If you do not know whether to specify BSD or SYSV, check with your System Administrator.

By default, SAS uses the **lpr** command if your operating system is derived from Berkeley's version; otherwise, it uses the **lp** command.

See Also

□ "PRINTCMD System Option" on page 352

MAPS System Option

Specifies the name of the SAS data library containing the SAS/GRAPH map data sets

Default: **!SASROOT/maps** (set in the installed **!SASROOT/sasv9.cfg** file)

Valid in: configuration file, SAS invocation, OPTIONS statement, SAS System Options window, SASV9_OPTIONS environment variable

Category: Graphics: Driver settings

PROC OPTIONS GROUP= GRAPHICS

UNIX specifics: default value and *location-of-maps*

See: MAPS System Option in *SAS Language Reference: Dictionary*

Syntax

-MAPS *location-of-maps*

MAPS=*location-of-maps*

location-of-maps
specifies a libref, a valid UNIX pathname, or an environment variable associated with a pathname. Do not use a specific filename.

Details

You can reassign the MAPS libref, but you cannot clear it.

Map files might have to be uncompressed before they are used. Use the CONTENTS statement in the DATASETS procedure to determine whether they are compressed.

See Also

- "INSERT System Option" on page 337
- "APPEND System Option" on page 315

MAXMEMQUERY System Option

Specifies the maximum amount of memory that is allocated per request for certain procedures

Default: 6M

Valid in: configuration file, SAS invocation, OPTIONS statement, SASV9_OPTIONS environment variable

Category: System administration: Memory

PROC OPTIONS GROUP= MEMORY

UNIX specifics: all

Syntax

-MAXMEMQUERY *n* | *n*K | *n*M | *n*G | *hex*X | MIN | MAX

MAXMEMQUERY= *n* | *n*K | *n*M | *n*G | *hex*X | MIN | MAX

***n* | *n*K | *n*M | *n*G**
specifies the limit in multiples of 1 (bytes); 1,024 (kilobytes); 1,048,576 (megabytes); or 1,073,741,824 (gigabytes). You can specify decimal values for the number of kilobytes, megabytes, or gigabytes. For example, a value of **8** specifies 8 bytes, a value of **.782k** specifies 801 bytes, and a value of **3m** specifies 3,145,728 bytes.

***hex*X**
specifies the amount of memory as a hexadecimal value. You must specify the value beginning with a number (0–9), followed by hex digits (0–9, A-F), and then followed by an X. For example, **2dx** sets the amount of memory to 45 bytes.

MIN
specifies 0 bytes, which indicates that there is no limit on the total amount of memory that can be allocated per request by each SAS procedure. These memory allocations are limited by the value of MEMSIZE.

MAX
specifies a limit of 2,147,483,647 bytes.

Details

Some SAS procedures attempt to allocate all of the memory that they can, up to the amount specified by the MEMSIZE option. If this amount of memory is not available, SAS attempts to use paging. If the amount of page space available is less than the value of MEMSIZE, SAS generates an error message. The MAXMEMQUERY option specifies the maximum amount of memory that SAS can request at one time. If your system has small system paging devices, you might want to lower the value of MAXMEMQUERY.

MEMSIZE System Option

Specifies the limit on the total amount of memory that can be used by each SAS session

Default: value set in SAS configuration file **!SASROOT/sasv9.cfg**
Valid in: configuration file, SAS invocation, SASV9_OPTIONS environment variable
Category: System administration: Memory
PROC OPTIONS GROUP= MEMORY
UNIX specifics: all

Syntax

-MEMSIZE *n* | *n*K | *n*M | *n*G | *hex*X | MIN | MAX

***n* | *n*K | *n*M | *n*G**
specifies the limit in multiples of 1 (bytes); 1,024 (kilobytes); 1,048,576 (megabytes); or 1,073,741,824 (gigabytes). You can specify decimal values for the number of kilobytes, megabytes, or gigabytes. For example, a value of **8** specifies 8 bytes, a value of **.782k** specifies 801 bytes, and a value of **3m** specifies 3,145,728 bytes.

***hex*X**
specifies the amount of memory as a hexadecimal value. You must specify the value beginning with a number (0–9), followed by hex digits (0–9, A-F), and then followed by an X. For example, **2dx** sets the amount of memory to 45 bytes.

MIN
specifies 0 bytes, which indicates that there is no limit on the total amount of memory that can be used by each SAS session.

MAX
specifies to set the memory value to the largest possible setting. This value depends on the system limit.

Details

The MEMSIZE option specifies the total amount of memory available to each SAS session. Too low a value will result in out-of-memory conditions.

SAS does not automatically reserve or allocate the amount of memory that you specify in the MEMSIZE option. SAS will only use as much memory as it needs to complete a process. For example, a DATA step might only require 20M of memory, so even though MEMSIZE is set to 500M, SAS will use only 20M of memory.

While your SAS jobs are running, you can monitor the effect of larger memory settings by using system monitoring tools, such as **top** or **vmstat**.

Note: Setting MEMSIZE to 0 is not recommended except for debugging and testing purposes. The optimal setting for this option depends on the other applications running and system resources available at your site. The amount of memory available to SAS processes can also be limited by your system administrator.

To determine this optimal value, run the SAS procedure or DATA step with MEMSIZE set to 0 and the FULLSTIMER option. Note the amount of memory used by the process and then set MEMSIZE to a larger amount. △

See Also

- "REALMEMSIZE System Option" on page 353
- "SORT Procedure" on page 282

MSG System Option

Specifies the library that contains SAS error messages

Alias: SASMSG

Default: **`!SASROOT/sasmsg`** (set in the installed **`!SASROOT/sasv9.cfg`** file)

Valid in: configuration file, SAS invocation, SASV9_OPTIONS environment variable

Category: Environment control: Files

PROC OPTIONS GROUP= ENVFILES

UNIX specifics: all

Syntax

-MSG *pathname*

-MSG ('*pathname*' '*pathname*' ...)

pathname
must resolve to a valid UNIX pathname. You can use an environment variable that resolves to a valid pathname.

Details

This option is set in the installation process and is not normally changed after installation.

See Also

- "INSERT System Option" on page 337
- "APPEND System Option" on page 315

MSGCASE System Option

Determines whether SAS displays notes, warnings, and error messages in uppercase

Default: NOMSGCASE

Valid in: configuration file, SAS invocation, SASV9_OPTIONS environment variable

Category: Log and procedure output: SAS log

PROC OPTIONS GROUP= LOGCONTROL

UNIX specifics: all

Syntax

MSGCASE | NOMSGCASE

MSGCASE
displays messages in uppercase characters.

NOMSGCASE
displays messages in uppercase and lowercase characters.

Details

User-generated messages and source lines are not affected by the MSGCASE system option.

MSYMTABMAX System Option

Specifies the maximum amount of memory available to the macro variable symbol table(s)

Default: 4M (set in the installed **`!SASROOT/sasv9.cfg`** file)

Valid in: configuration file, SAS invocation, OPTIONS statement, SAS System Options window, SASV9_OPTIONS environment variable

Category: Macro: SAS macro

PROC OPTIONS GROUP= MACRO

UNIX specifics: default value

See: MSYMTABMAX System Option in *SAS Macro Language: Reference*

Syntax

-MSYMTABMAX *n* | *n*K | *n*M | *n*G | *hex*X | MIN | MAX

MSYMTABMAX=*n* | *n*K | *n*M | *n*G | *hex*X | MIN | MAX

***n* | *n*K | *n*M | *n*G**
specifies the maximum amount of memory that is available in multiples of 1 (bytes); 1,024 (kilobytes); 1,048,576 (megabytes); or 1,073,741,824 (gigabytes). You can specify decimal values for the number of kilobytes, megabytes, or gigabytes. For example, a value of **`8`** specifies 8 bytes, a value of **`.782k`** specifies 801 bytes, and a value of **`3m`** specifies 3,145,728 bytes.

***hex*X**
specifies the maximum amount of memory that is available as a hexadecimal value. You must specify the value beginning with a number (0–9), followed by hex digits (0–9, A-F), and then followed by an X. For example, **`2dx`** sets the maximum amount of memory to 45 bytes.

MIN
sets the amount of memory that is available to the minimum setting, which is 0 bytes. This causes all macro symbol tables to be written to disk.

MAX
sets the amount of memory that is available to the maximum setting, which is 2,147,483,647 bytes.

MVARSIZE System Option

Specifies the maximum size for in-memory macro variables

Default: 32K (set in the installed **`!SASROOT/sasv9.cfg`** file)
Valid in: configuration file, SAS invocation, OPTIONS statement, SAS System Options window, SASV9_OPTIONS environment variable
Category: Macro: SAS macro
PROC OPTIONS GROUP= MACRO
UNIX specifics: default value
See: MVARSIZE System Option in *SAS Macro Language: Reference*

Syntax

-MVARSIZE *n* | *n*K | *n*M | *n*G | *hex*X | MIN | MAX

MVARSIZE=*n* | *n*K | *n*M | *n*G | *hex*X | MIN | MAX

***n* | *n*K | *n*M | *n*G**
specifies the maximum macro variable size in multiples of 1 (bytes); 1,024 (kilobytes); 1,048,576 (megabytes); or 1,073,741,824 (gigabytes). You can specify decimal values for the number of kilobytes, megabytes, or gigabytes. For example, a value of **`8`** specifies 8 bytes, a value of **`.782k`** specifies 801 bytes, and a value of **`3m`** specifies 3,145,728 bytes.

***hex*X**
specifies the maximum macro variable size as a hexadecimal value. You must specify the value beginning with a number (0–9), followed by hex digits (0–9, A-F), and then followed by an X. For example, **`2dx`** sets the maximum macro variable size to 45 bytes.

MIN
sets the macro variable size to the minimum setting, which is 0 bytes. This causes all macro variable values to be written to disk.

MAX
sets the macro variable size to the maximum setting, which is 2,147,483,647 bytes.

NEWS System Option

Specifies a file that contains messages to be written to the SAS log

Default: **!SASROOT/misc/base/news** (set in the installed **!SASROOT/sasv9.cfg** file)
Valid in: configuration file, SAS invocation, SASV9_OPTIONS environment variable
Category: Environment control: Files
PROC OPTIONS GROUP= ENVFILES
UNIX specifics: -NONEWS option
See: NEWS System Option in *SAS Language Reference: Dictionary*

Syntax

-NEWS *file-specification* | –NONEWS

-NEWS *file-specification*
specifies an external file. This file contains the messages for the SAS log.

-NONEWS
specifies that the contents of the NEWS file is not displayed in the SAS log, even if the file exists. This option causes any previous NEWS specifications to be ignored.

Details

The contents of the NEWS file are displayed in the SAS log immediately after the SAS header.

NLSCOMPATMODE System Option

Provides national language compatibility with previous releases of SAS

Default: NONLSCOMPATMODE
Valid in: configuration file, SAS invocation
Category: Environment control: Language control
PROC OPTIONS GROUP= LANGUAGECONTROL
UNIX specifics: all
See: NLSCOMPATMODE System Option in *SAS National Language Support (NLS): User's Guide*

OBS System Option

Specifies which observation SAS will process last

Default: MAX
Valid in: configuration file, SAS invocation, OPTIONS statement, SAS System Options window, SASV9_OPTIONS environment variable
Category: Files: SAS Files

PROC OPTIONS GROUP= SASFILES
UNIX specifics: default value
See: OBS System Option in *SAS Language Reference: Dictionary*

Syntax

-OBS *n* | *n*K | *n*M | *n*G | *n*T | *hex*X | MIN | MAX

OBS=*n* | *n*K | *n*M | *n*G | *n*T | *hex*X | MIN | MAX

***n* | *n*K | *n*M | *n*G | *n*T**
specifies a number to indicate when to stop processing. Using one of the letter notations results in multiplying the integer by a specific value. That is, specifying K (kilo) multiplies the integer by 1,024, M (mega) multiplies by 1,048,576, G (giga) multiplies by 1,073,741,824, or T (tera) multiplies by 1,099,511,627,776. You can specify a decimal value for *n* when it is used to specify a K, M, G, or T value. For example, a value of **`20`** specifies 20 observations or records, a value of **`.782k`** specifies 801 observations or records, and a value of **`3m`** specifies 3,145,728 observations or records.

***hex*X**
specifies a number to indicate when to stop processing as a hexadecimal value. You must specify the value beginning with a number (0–9), followed by hex digits (0–9, A-F), and then followed by an X. For example, the hexadecimal value F8 must be specified as **`0F8x`** in order to specify the decimal equivalent of 248. For example, the value **`2dx`** specifies the decimal equivalent of 45.

MIN
sets the number to indicate when to stop processing to 0.

If OBS=0 and the NOREPLACE option is in effect, SAS might still be able to take certain actions. See *SAS Language Reference: Dictionary* for more information.

MAX
sets the number to indicate when to stop processing to 9,007,199,254,740,992 . On 32-bit systems, MAX is 2,147,483,647.

OPLIST System Option

Writes the settings of SAS system options to the SAS log

Default: NOOPLIST
Valid in: configuration file, SAS invocation, SASV9_OPTIONS environment variable
Category: Log and procedure output control: SAS log
PROC OPTIONS GROUP= LOGCONTROL
UNIX specifics: all

Syntax

-OPLIST | -NOOPLIST

Details

The OPLIST system option echoes only the system options specified on the command line; it does not echo any system options specified in the configuration file or in the SASV9_OPTIONS environment variable. (If you want to echo the contents of the configuration file, use the VERBOSE option.) For example, suppose you invoke SAS with the following command:

```
sas -nodms -fullstimer -nonews -oplist
```

SAS writes this line to the SAS log:

```
NOTE: SAS command line: -nodms -fullstimer -nonews -oplist
```

See Also

- "VERBOSE System Option" on page 380

PAGESIZE System Option

Specifies the number of lines that compose a page of SAS output

Default: number of lines on your display for interactive modes; 60 for batch mode

Valid in: configuration file, SAS invocation, OPTIONS statement, SAS System Options window, SASV9_OPTIONS environment variable

Category: Log and procedure output control: SAS log and procedure output

PROC OPTIONS GROUP= LOG_LISTCONTROL

UNIX specifics: default values and range

See: PAGESIZE System Option in *SAS Language Reference: Dictionary*

Syntax

-PAGESIZE *n* | *n*K | *hex*X | MIN | MAX

PAGESIZE=*n* | *n*K | *hex*X | MIN | MAX

***n* | *n*K**
specifies the number of lines that compose a page in multiples of 1 (*n*) or 1,024 (*n*K). You can specify decimal values for the number of kilobytes. For example, a value of **`800`** specifies 800 lines, a value of **`.782k`** specifies 801 lines, and a value of **`3k`** specifies 3,072 lines.

***hex*X**
specifies the number of lines that compose a page as a hexadecimal value. You must specify the value beginning with a number (0–9), followed by hex digits (0–9, A-F), and then followed by an X. For example, the value **`2dx`** specifies 45 lines.

MIN
sets the number of lines that compose a page to the minimum setting, which is 15.

MAX
sets the number of lines that compose a page to the maximum setting, which is 32,767.

Details

The default for interactive modes is the number of lines on your display. For batch mode, the default is 60.

See Also

- "Controlling the Content and Appearance of Output in UNIX Environments" on page 165

PATH System Option

Specifies the search path for SAS executable modules

Default: **`!SASROOT/sasexe`** (set in the installed **`!SASROOT/sasv9.cfg`** file)
Valid in: configuration file, SAS invocation, SASV9_OPTIONS environment variable
Category: Environment control: Files
PROC OPTIONS GROUP= ENVFILES
UNIX specifics: all

Syntax

-PATH *pathname*

Details

The PATH option identifies the search paths for SAS executable files. You can specify multiple PATH options to define the search order. The paths are searched in the order in which SAS encounters them; therefore, specify at the front of the list the paths for the products that you run most frequently. See "How SAS Processes System Options Set in Multiple Places" on page 20 for information about how that order is determined when you specify the PATH system option more than once.

PRINT System Option

Specifies the destination for SAS output in batch mode

Default: the SAS output from a batch SAS program is written to a file in the current directory with the same filename as the SAS source file and an extension of .lst.
Valid in: configuration file, SAS invocation, SASV9_OPTIONS environment variable
Category: Environment control: Files

PROC OPTIONS GROUP= ENVFILES

UNIX specifics: all

Syntax

-PRINT *destination* | -NOPRINT

-PRINT *destination*
specifies the location for the SAS procedure output file. The *destination* can be any valid UNIX path to a directory, a filename, or an environment variable that is associated with a path. If you specify only the path to a directory, the procedure output file is created in the specified directory. The default name for this file is *filename*.lst, where *filename* is the name of your SAS job.

-NOPRINT
suppresses the creation of the SAS procedure output file.

Details

PRINT is valid in batch mode; it is ignored in interactive modes.

See Also

- "Using SAS System Options to Route Output" on page 163

PRINTCMD System Option

Specifies the print command SAS is to use

Default: none

Valid in: configuration file, SAS invocation, OPTIONS statement, SASV9_OPTIONS environment variable

Category: Log and procedure output control: Procedure output

PROC OPTIONS GROUP= LISTCONTROL

UNIX specifics: all

Syntax

-PRINTCMD "*print-command*"

PRINTCMD="*print-command*"

Details

The syntax of the options passed to the print command is controlled by the LPTYPE system option. If LPTYPE is set to BSD, the command uses **`lpr`** command options; if LPTYPE is set to SYSV, the command uses **`lp`** command options.

If your site uses a print command (spooler) other than **lp** or **lpr**, *print-command* specifies its name. The PRINTCMD option overrides the LPTYPE setting.

When specified in an options statement, the PRINTCMD option will not change the print commands assigned to previously defined filenames. For example, consider the following code:

```
filename pc1 printer;
proc printto print=pc1;
run;
proc print data=sales.week;
run;

options printcmd="netlp";

filename pc2 printer;
proc printto print=pc2;
run;
proc print data=sales.month;
run;
```

Output associated with PC2 will use the **netlp** command; output associated with PC1 will use the default print command.

See Also

- Chapter 6, "Printing and Routing Output," on page 153
- "LPTYPE System Option" on page 341

REALMEMSIZE System Option

Indicates the amount of real memory SAS can expect to allocate

Default: 0

Valid in: configuration file, SAS invocation, SASV9_OPTIONS environment variable

Category: System administration: Memory

PROC OPTIONS GROUP= MEMORY

UNIX specifics: valid values

Syntax

-REALMEMSIZE *n* | *n*K | *n*M | 1G | *hex*X | MIN | MAX

***n* | *n*K | *n*M**
specifies the amount of memory to reserve in multiples of 1 (bytes); 1,024 (kilobytes); or 1,048,576 (megabytes). You can specify decimal values for the number of kilobytes or megabytes. For example, a value of **8** specifies 8 bytes, a value of **.782k** specifies 801 bytes, and a value of **3m** specifies 3,145,728 bytes.

1G
specifies the amount of memory to reserve is 1,073,741,824 (1 gigabyte).

***hex*X**
specifies the amount of memory as a hexadecimal value. You must specify the value beginning with a number (0–9), followed by hex digits (0–9, A-F), and then followed by an X. For example, the value **`2dx`** sets the amount of memory to 45 bytes.

MIN
specifies a value of 0, which indicates that the memory usage is determined by SAS when SAS starts.

MAX
specifies to set the memory size to the largest permissible value.

Details

Use the REALMEMSIZE system option to optimize the performance of SAS procedures that alter their algorithms and memory usage. Setting the value of REALMEMSIZE too low might result in less than optimal performance. For better performance, set REALMEMSIZE to the amount of memory (excluding swap space) that is available to the SAS session at invocation.

See Also

- “MEMSIZE System Option” on page 344
- “SORT Procedure” on page 282

RSASUSER System Option

Controls whether members of the Sasuser data library can be opened for update or for read-only access

Default: NORSASUSER

Valid in: configuration file, SAS invocation, SASV9_OPTIONS environment variable

Category: Environment control: Files

PROC OPTIONS GROUP= ENVFILES

UNIX specifics: network considerations

See: RSASUSER System Option in *SAS Language Reference: Dictionary*

Syntax

-RSASUSER | -NORSASUSER

Details

RSASUSER
limits access to the Sasuser data library to read-only access. (If the Sasuser data library is being shared by multiple users or the same user is running SAS multiple times simultaneously, the Sasuser data library is often shared.) By default, if one user has a member of the Sasuser data library open for update, all other users are denied access to that SAS data library member. For example, if one user is writing

to the Sasuser.Profile catalog, no other user can even read data from the Profile catalog.

Specifying RSASUSER enables a group of users to share Sasuser data library members by allowing all users read-only access to members. In the Profile catalog example, if RSASUSER is in effect, all users can open the Profile catalog for read-only access, allowing other users to concurrently read from the Profile catalog. However, no user can write information out to the Profile catalog; you receive an error message if you try to do so.

Specifying RSASUSER from the command line affects only that session's access to files. To enable a group of users to share members in the Sasuser data library, the system manager should set RSASUSER in a common SAS configuration file, which is shared by all users who will be sharing the Sasuser data library.

If you specify RSASUSER but no Profile catalog exists in the Sasuser data library, the Profile catalog is created in the Work data library.

Note: While the RSASUSER option is extremely useful for sharing information (such as the Profile catalog) stored in the Sasuser data library, it is less practical when used in conjunction with SAS/ASSIST software or other SAS modules that require update access to the Sasuser data library. △

NORSASUSER
prevents users from sharing members of the Sasuser data library because it allows a user to open a file in the Sasuser library for update access. Update access requires exclusive rights to the data library member.

See Also

- "Sharing Files in UNIX Environments" on page 124

RTRACE System Option

Produces a list of resources that are read during SAS execution

Default: NONE

Valid in: configuration file, SAS invocation, SASV9_OPTIONS environment variable

Category: Log and procedure output control: SAS log

PROC OPTIONS GROUP= LOGCONTROL

UNIX specifics: all

Syntax

-RTRACE ALL | NONE

ALL
traces both files that are read and executable files that are loaded.

NONE
tells SAS not to produce any trace information.

Details

The RTRACE system option produces a list of resources that are read or loaded during the execution of SAS. If you specify ALL but do not specify the RTRACELOC option, the output is written to the SAS log.

See Also

□ "RTRACELOC System Option" on page 356

RTRACELOC System Option

Specify the pathname of the file to which RTRACE information is written

Default: none

Valid in: configuration file, SAS invocation, OPTIONS statement, SAS System Options window, SASV9_OPTIONS environment variable

Category: Environment control: Files

PROC OPTIONS GROUP= ENVFILES

UNIX specifics: all

Syntax

-RTRACELOC *pathname*

pathname
specifies the file to which RTRACE information is written. The *pathname* must include the path and the filename for the RTRACE output.

Details

The RTRACELOC system option specifies the pathname of the file to which RTRACE information is written. If the *pathname* does not include the filename, the output will be directed to standard output. If you specify RTRACE ALL but do not specify RTRACELOC, the output is written to the SAS log.

See Also

□ "RTRACE System Option" on page 355

S System Option

Specifies the length of statements on each line of source statements and the length of data on the line following a DATALINES statement

Default: 0

Valid in: configuration file, SAS invocation, OPTIONS statement, SAS System Options window, SASV9_OPTIONS environment variable

Category: Input control: Data processing

PROC OPTIONS GROUP= INPUTCONTROL

UNIX specifics: valid values for *n*

See: S System Option in *SAS Language Reference: Dictionary*

Syntax

-S *n* | *n*K | *n*M | *n*G | *hex*X | MIN | MAX

S=*n* | *n*K | *n*M | *n*G | *hex*X | MIN | MAX

***n* | *n*K | *n*M | *n*G**
specifies the length of statements and data in multiples of 1 (bytes); 1,024 (kilobytes); 1,048,576 (megabytes); or 1,073,741,824 (gigabytes). You can specify decimal values for the number of kilobytes, megabytes, or gigabytes. For example, a value of **8** specifies 8 bytes, a value of **.782k** specifies 801 bytes, and a value of **3m** specifies 3,145,728 bytes.

***hex*X**
specifies the length of statements and data as a hexadecimal value. You must specify the value beginning with a number (0–9), followed by hex digits (0–9, A-F), and then followed by an X. For example, the value **2dx** sets the length of statements and data to 45 bytes.

MIN
specifies the length of statements to be 0, which causes SAS to use the default value.

MAX
specifies to use the maximum line length allowed under UNIX, which is 2,147,483,647 bytes.

Details

Input can be from either fixed- or variable-length records. Both fixed-length and variable-length records can be either unsequenced or sequenced. Unsequenced records do not contain sequence fields.

SAS determines whether the input contains sequence numbers that are based on the value of S. The S system option specifies the length of statements, exclusive of sequence numbers, on each line of SAS source statements and the length of data, exclusive of sequence numbers, on lines following a DATALINES statement.

The default value of 0 enables SAS to read a file with any line length up to MAX.

S2 System Option

Specifies the length of secondary source statements

Default: 0

Valid in: configuration file, SAS invocation, OPTIONS statement, SAS System Options window, SASV9_OPTIONS environment variable

Category: Input control: Data processing

PROC OPTIONS GROUP= INPUTCONTROL

UNIX specifics: valid values for *n*

See: S2 System Option in *SAS Language Reference: Dictionary*

Syntax

S2=S | *n* | *n*K | *n*M | *n*G | *hex*X | MIN | MAX

S
uses the current value of the S system option to compute the record length of text that comes from the %INCLUDE statement, an autoexec file, or an autocall macro file.

***n* | *n*K | *n*M | *n*G**
specifies the value by which to compute the record length of text that comes from an %INCLUDE statement, an autoexec file, or an autocall macro file. *n* can be between 0 and 2,147,483,647, or you can specify the value in multiples of 1 (bytes); 1,024 (kilobytes); 1,048,576 (megabytes); or 1,073,741,824 (gigabytes). You can specify decimal values for the number of kilobytes, megabytes, or gigabytes. For example, a value of **8** specifies 8 bytes, a value of **.782k** specifies 801 bytes, and a value of **3m** specifies 3,145,728 bytes.

***hex*X**
specifies the length of statements and data as a hexadecimal value. You must specify the value beginning with a number (0–9), followed by hex digits (0–9, A-F), and then followed by an X. For example, the value **2dx** sets the length of statements and data to 45 bytes.

MIN
uses the value of 0, indicating no length restriction.

MAX
uses the value of 2,147,483,647.

Details

The S2 system option operates exactly like the S system option, except that the S2 system option controls input from only an %INCLUDE statement, an autoexec file, or an autocall macro file.

SASAUTOS System Option

Specifies the autocall library

Default: SASAUTOS fileref

Valid in: configuration file, SAS invocation, OPTIONS statement, SAS System Options window, SASV9_OPTIONS environment variable

Category: Environment control: Files
Macro: SAS macro
PROC OPTIONS GROUP= ENVFILES
MACRO
UNIX specifics: syntax for specifying multiple *dir-name*s
See: SASAUTOS System Option in *SAS Macro Language: Reference*

Syntax

-SASAUTOS '*dir-name*'|*fileref*

-SASAUTOS ('*dir-name1*'|*fileref1*,...,'*dir-name-n*'|*filerefn*)

-NOSASAUTOS

SASAUTOS='*dir-name*'|*fileref*

SASAUTOS =('*dir-name1*'|*fileref1*,...,'*dir-name-n*'|*filerefn*)

NOSASAUTOS

Note: The SASAUTOS option uses filerefs, not librefs. △

Details

Each autocall macro library consists of files in a UNIX directory. The *dir-name* can be the pathname of a UNIX directory, a fileref, or an environment variable.

If you specify the pathname of a directory, you must enclose the name in quotation marks. You can omit the quotation marks only if you are specifying the option in the configuration file, in the SAS command, or in the SASV9_OPTIONS environment variable, and if the name cannot be taken to be a fileref.

If you specify a fileref, you must define it before attempting to use any of the autocall macros. You can define the fileref in a FILENAME statement, in an environment variable, or with the FILENAME function (see "Assigning Filerefs to External Files or Devices with the FILENAME Statement" on page 135).

How you specify multiple directory names, filerefs, or environment variables depends on where you specify the SASAUTOS option:

- If you specify the SASAUTOS option in the configuration file or in the SASV9_OPTIONS environment variable, use either multiple SASAUTOS options, or enclose the directory names in parentheses. Separate the names with a comma or a blank space.
- If you specify the SASAUTOS option in the SAS command, use the APPEND or INSERT system options to append to the end or insert on the beginning of the current SASAUTOS value. For example, the following code adds **/users/userid/also** to the end of the current SASAUTOS value, **/users/userid/here**:

```
sas -sasautos /users/userid/here -append sasautos /users/userid/also
```

- If you specify the SASAUTOS option in the OPTIONS statement or in the System Options window, you must enclose the directory names in parentheses. Separate the names with a comma or a blank space.

At configuration time, SAS concatenates all directories specified for SASAUTOS. However, after the session starts, any new directories you specify override any current autocall libraries.

The NOSASAUTOS option causes SAS to ignore all previous SASAUTOS specifications (whether specified in the SAS command, in the configuration file, or in the SASV9_OPTIONS environment variable).

The default value of the SASAUTOS option is the SASAUTOS fileref. There is no UNIX directory assigned to the fileref, so you must define the SASAUTOS fileref if you want to use it as your autocall library.

Example: Specifying Multiple Environment Variables in the OPTIONS Statement

The following example shows the syntax to use if you are specifying multiple environment variables in the OPTIONS statement:

```
options sasautos=(AUTODIR, SASAUTOS);
```

The environment variables that you specify must be defined. For example, you could define the AUTODIR environment variable at SAS invocation by using the following code:

```
-set AUTODIR /tmp/sasautos
```

For more information about how to define an environment variable, see "SET System Option" on page 363.

See Also

- "APPEND System Option" on page 315
- "INSERT System Option" on page 337
- *SAS Macro Language: Reference*

SASHELP System Option

Specifies the locations of Sashelp libraries

Default: **`!SASROOT/sashelp`** (set in the installed **`!SASROOT/sasv9.cfg`** file)
Valid in: configuration file, SAS invocation, SASV9_OPTIONS environment variable
Category: Environment control: Files
PROC OPTIONS GROUP= ENVFILES
UNIX specifics: *pathname* can also be an environment variable
See: SASHELP System Option in *SAS Language Reference: Dictionary*

Syntax

–SASHELP *pathname*

–SASHELP (*'pathname'*, *'pathname'*...)

Details

This option is set in the installation process and is not normally changed after installation. An environment variable can be specified as the value of SASHELP.

See Also

- "INSERT System Option" on page 337
- "APPEND System Option" on page 315

SASSCRIPT System Option

Specifies one or more storage locations of SAS/CONNECT script files

Default: `!SASROOT/misc/connect`

Valid in: configuration file, SAS invocation, OPTIONS statement, SAS System Options window, SASV9_OPTIONS environment variable

Category: Communications: Networking and encryption

PROC OPTIONS GROUP= COMMUNICATIONS

UNIX specifics: syntax for specifying multiple directory names

Syntax

-SASSCRIPT '*dir-name*' | ('*dir-name-1*',...,'*dir-name-n*')

SASSCRIPT='*dir-name*' | ('*dir-name-1*',...,'*dir-name-n*')

Details

How you specify multiple directory names in the same SASSCRIPT option depends on where you specify the SASSCRIPT option:

- If you specify the option in the configuration file or in the SASV9_OPTIONS environment variable, use either multiple SASSCRIPT options, or enclose the directory names in parentheses. Separate the names with a comma or a blank space.
- If you specify the option in the SAS command, use multiple SASSCRIPT options, since parentheses cause syntax errors.
- If you specify the option in the OPTIONS statement or in the System Options window, you must enclose the directory names in parentheses. Separate the names with a comma or a blank space.

See Also

- "SASSCRIPT System Option" in *SAS/CONNECT User's Guide*

SASUSER System Option

Specifies the name of the Sasuser library

Default: `~/sasuser.v91` (set in the installed `!SASROOT/sasv9.cfg` file)

Valid in: configuration file, SAS invocation, SASV9_OPTIONS environment variable

Category: Environment control: Files

PROC OPTIONS GROUP= ENVFILES

UNIX specifics: *pathname* can be an environment variable

See: SASUSER System Option in *SAS Language Reference: Dictionary*

Syntax

–SASUSER *pathname*

Details

The *pathname* identifies the directory for the Sasuser library that contains a user's Profile catalog. You can use an environment variable to specify the pathname, for example:

```
sas -sasuser $HOME
```

SEQENGINE System Option

Specifies the default access method for SAS sequential data libraries

Default: TAPE

Valid in: configuration file, SAS invocation, OPTIONS statement, SAS System Options window, SASV9_OPTIONS environment variable

Category: Files: SAS Files

PROC OPTIONS GROUP= SASFILES

UNIX specifics: all

Syntax

SEQENGINE=*engine-name*

engine-name can be one of the following under UNIX:

V9TAPE|TAPE
specifies the default sequential engine for SAS 9 and SAS 9.1. TAPE is the default value.

V8TAPE
specifies the sequential engine for SAS Version 8. This engine and the V9TAPE engine are identical.

V6TAPE
specifies the sequential engine for Version 6. This engine is read-only.

Details

The SEQENGINE= option specifies the default access method, or engine, that is used when you are creating new sequential-format SAS data libraries. The engine that is used with an existing sequential library is determined by the first data set in that library and is not affected by this option.

See Also

- "Accessing Sequential-Format Data Libraries in UNIX Environments" on page 122

SET System Option

Defines an environment variable

Default: none

Valid in: configuration file, SAS invocation, OPTIONS Statement, SASV9_OPTIONS environment variable

Category: Environment control: Files

PROC OPTIONS GROUP= ENVFILES

UNIX specifics: all

Syntax

–SET *variable-name value*

Details

The SET option lets you define an environment variable that is valid within the SAS session and any shell started from within the SAS session. Using the SET option is similar to using the SAS **setenv** command. See "Executing Operating System Commands from Your SAS Session" on page 13 for details.

A special use for the SET option is to specify the name of the **!SASROOT** directory:

```
-set SASROOT pathname
```

The pathname specified can then be used to expand **!SASROOT** (as shown in Table 4.6 on page 115).

After exiting your SAS session, environment variables that are set with the SET option no longer exist.

See Also

- Appendix 1, "The !SASROOT Directory," on page 397
- "Defining Environment Variables in UNIX Environments" on page 21

SORTANOM System Option

Specifies certain options for the host sort utility

Default: none

Valid in: configuration file, SAS invocation, OPTIONS statement, SASV9_OPTIONS environment variable

Category: Sort: Procedure options

PROC OPTIONS GROUP= SORT

UNIX specifics: all

Syntax

SORTANOM=*option(s)*

–SORTANOM *option(s)*

The *option(s)* can be any one or more of the following:

b tells **syncsort** to run in multi-call mode instead of single-call mode. (Refer to the documentation for **syncsort** for more information.)

Note: This option is available for **syncsort** only. △

t prints statistics about the external sorting process in the SAS log.

v prints (in the SAS log) all of the commands that are passed to the host sort utility.

SORTCUT System Option

Specifies the number of observations above which the host sort program is used instead of the SAS sort program

Default: 0

Valid in: configuration file, SAS invocation, OPTIONS statement, SASV9_OPTIONS environment variable

Category: Sort: Procedure options

PROC OPTIONS GROUP= SORT

UNIX specifics: all

Syntax

SORTCUT=*n* | *n*K | *n*M | *n*G | *hex*X | MIN | MAX

-SORTCUT *n* | *n*K | *n*M | *n*G | *hex*X | MIN | MAX

***n* | *n*K | *n*M | *n*G**
specifies the number of observations in multiples of 1 (*n*); 1,024 (*n*K); 1,048,576 (*n*M); or 1,073,741,824 (*n*G). You can specify decimal values for the number of kilobytes, megabytes, or gigabytes. For example, a value of **800** specifies 800 observations, a value of **.782k** specifies 801 observations, and a value of **3m** specifies 3,145,728 observations.

***hex*X**
specifies the number of observations as a hexadecimal value. You must specify the value beginning with a number (0–9), followed by hex digits (0–9, A-F), and then followed by an X. For example, the value **2ffx** specifies 767 observations.

MIN
specifies 0 observations.

MAX
specifies 2,147,483,647 observations.

Details

When you specify SORTPGM=BEST, SAS uses the value of the SORTCUT and SORTCUTP options to determine whether to use the host sort or the SAS sort. If the number of observations in the data set is greater than the number that you specify with SORTCUT, the host sort will be used. If both SORTCUT and SORTCUTP are either not defined or are set to 0, the SAS sort is used. If you specify both options and both conditions are true, SAS chooses the host sort.

See Also

- "SORTCUTP System Option" on page 365
- "SORTPGM System Option" on page 368

SORTCUTP System Option

Specifies the number of bytes above which the host sort program is used instead of the SAS sort program

Default: 0

Valid in: configuration file, SAS invocation, OPTIONS statement, SASV9_OPTIONS environment variable

Category: Sort: Procedure options

PROC OPTIONS GROUP= SORT

UNIX specifics: all

Syntax

SORTCUTP=*n* | *n*K | *n*M | *n*G | *hex*X | MIN | MAX

-SORTCUTP *n* | *n*K | *n*M | *n*G | *hex*X | MIN | MAX

***n* | *n*K | *n*M | *n*G**
specifies the number of bytes in multiples of 1 (bytes); 1,024 (kilobytes); 1,048,576 (megabytes); or 1,073,741,824 (gigabytes). You can specify decimal values for the number of kilobytes, megabytes, or gigabytes. For example, a value of **`8`** specifies 8 bytes, a value of **`.782k`** specifies 801 bytes, and a value of **`3m`** specifies 3,145,728 bytes.

***hex*X**
specifies the number of bytes as a hexadecimal value. You must specify the value beginning with a number (0–9), followed by hex digits (0–9, A-F), and then followed by an X. For example, the value **`2dx`** specifies 45 bytes.

MIN
specifies 0 bytes.

MAX
specifies 2,147,483,647 bytes.

Details

When you specify SORTPGM=BEST, SAS uses the value of the SORTCUT and SORTCUTP options to determine whether to use the host sort or the SAS sort. If the data set to be sorted is larger than the number of bytes (or kilobytes or megabytes) that you specify with SORTCUTP, the host sort (external) program will be used instead of the SAS sort (internal) program. The value you specify must be less than or equal to 2,147,483,647 bytes. If both SORTCUT and SORTCUTP are either not defined or are set to 0, the SAS sort is used. If you specify both options and both conditions are true, SAS chooses the host sort.

The following equation computes the number of bytes to be sorted:

```
number-of-bytes= ((length-of-obs)+(length-of-all-keys))
*number-of-obs
```

See Also

- "SORTANOM System Option" on page 363
- "SORTCUT System Option" on page 364
- "SORTPGM System Option" on page 368

SORTDEV System Option

Specifies the pathname used for temporary files created by the host sort utility

Default: same location as -WORK, which is set in the installed **!SASROOT/sasv9.cfg** file

Valid in: configuration file, SAS invocation, OPTIONS statement, SASV9_OPTIONS environment variable

Category: Sort: Procedure options

PROC OPTIONS GROUP= SORT

UNIX specifics: all

Syntax

SORTDEV='*pathname*'

-SORTDEV *pathname*

Details

The SORTDEV option specifies an alternative pathname for temporary files created by the host sort.

SORTNAME System Option

Specifies the name of the host sort utility

Default: none

Valid in: configuration file, SAS invocation, OPTIONS statement, SASV9_OPTIONS environment variable

Category: Sort: Procedure options

PROC OPTIONS GROUP= SORT

UNIX specifics: all

Syntax

SORTNAME='*host-sort-utility-name*'

-SORTNAME *host-sort-utility-name*

Details

The SORTNAME option specifies the name of the default host sort utility, **`syncsort`**.

See Also

- "SORTPGM System Option" on page 368

SORTPARM System Option

Specifies parameters for the host sort utility

Default: none

Valid in: configuration file, SAS invocation, OPTIONS statement, SASV9_OPTIONS environment variable

Category: Sort: Procedure options

PROC OPTIONS GROUP= SORT

UNIX specifics: all

Syntax

SORTPARM='*parameters*'

–SORTPARM '*parameters*'

The *parameters* are any parameters that you want to pass to the sort utility. For a description of these parameters, refer to the documentation for the sort that you are using.

SORTPGM System Option

Specifies whether the SAS sort or the host sort is used

Default: BEST

Valid in: configuration file, SAS invocation, OPTIONS statement, SASV9_OPTIONS environment variable

Category: Sort: Procedure options

PROC OPTIONS GROUP= SORT

UNIX specifics: all

Syntax

–SORTPGM SAS | HOST | BEST

SORTPGM=SAS | HOST | BEST

Details

The SORTPGM system option tells SAS whether to use the SAS sort, to use the host sort, or to determine which sort is best for the data set.

SAS
tells SAS to use the SAS sort.

HOST
tells SAS to use the sort specified by the SORTNAME system option.

BEST
tells SAS to determine the best routine to sort the data set: the SAS Software sort or the host sort specified by the SORTNAME system option. The settings of the SORTCUT and SORTCUTP system options determine whether SAS chooses the SAS Software sort or the host sort.

See Also

- “SORTCUTP System Option” on page 365
- “SORTNAME System Option” on page 367
- “SORTSIZE System Option” on page 368

SORTSIZE System Option

Specifies the amount of memory available to the SORT procedure

Default: MAX

Valid in: configuration file, SAS invocation, OPTIONS statement, SAS System Options window, SASV9_OPTIONS environment variable

Category: Sort: Procedure options
System administration: Memory

PROC OPTIONS GROUP= SORT
MEMORY
UNIX specifics: value of MAX
See: SORTSIZE System Option in *SAS Language Reference: Dictionary*

Syntax

–SORTSIZE *n* | *n*K | *n*M | *n*G | *hex*X | MIN | MAX

SORTSIZE=*n* | *n*K | *n*M | *n*G | *hex*X | MIN | MAX

***n* | *n*K | *n*M | *n*G**
specifies the number of bytes in multiples of 1 (bytes); 1,024 (kilobytes); 1,048,576 (megabytes); or 1,073,741,824 (gigabytes). You can specify decimal values for the number of kilobytes, megabytes, or gigabytes. For example, a value of **8** specifies 8 bytes, a value of **.782k** specifies 801 bytes, and a value of **3m** specifies 3,145,728 bytes.

***hex*X**
specifies the amount of memory as a hexadecimal value. You must specify the value beginning with a number (0–9), followed by hex digits (0–9, A-F), and then followed by an X. For example, the value **2dx** sets the amount of memory to 45 bytes.

MIN
specifies 0 bytes, which indicates that there is no limit except the limitation specified by the MEMSIZE system option.

MAX
specifies 2,147,483,647 bytes.

Details

The SORTSIZE option might reduce the amount of swapping that SAS must do to sort a data set. If the SORT procedure needs more memory than you specify, it creates a temporary utility file in your SAS Work directory. The SORT procedure's algorithm can swap unneeded data more efficiently than the operating system can.

The amount of memory that SAS uses for the SORT procedure also depends on the values of the MEMSIZE and REALMEMSIZE system options. For more information, see the "How SAS Determines the Amount of Memory to Use" on page 285 section for the SORT procedure.

In most cases, you can set SORTSIZE=MAX since this value will limit the amount of memory used by the SORT procedure.

See Also

- "REALMEMSIZE System Option" on page 353
- "SORTDEV System Option" on page 366
- "SORT Procedure" on page 282

SSLCALISTLOC System Option

Specifies the location of digital certificates for trusted certificate authorities

Default: none

Valid in: configuration file, SAS invocation, OPTIONS statement, SAS System Options window

Category: Communications: Networking and encryption

PROC OPTIONS GROUP= COMMUNICATIONS

UNIX specifics: all

Syntax

SSLCALISTLOC="*file-path*"

file-path
specifies the location of a file that contains the digital certificates for the trusted certificate authorities (CA).

Details

This option is used to identify, for SSL, the certificate authorities that it should trust. This option is always required for clients because they must trust at least one CA in order to validate a server's certificate. This option is required for a server only if client authentication is enabled.

See Also

- "SSLCERTLOC System Option" on page 370
- "SSLCLIENTAUTH System Option" on page 371
- "SSLCRLCHECK System Option" on page 372
- "SSLCRLLOC System Option" on page 373
- "SSLPVTKEYLOC System Option" on page 373
- "SSLPVTKEYPASS System Option" on page 374
- Appendix 3, "Using SSL in UNIX Environments," on page 403
- *SAS/CONNECT User's Guide*

SSLCERTLOC System Option

Specifies the name of the file that contains the digital certificate that is used for authentication

Default: none

Valid in: configuration file, SAS invocation, OPTIONS statement, SAS System Options window

Category: Communications: Networking and encryption

PROC OPTIONS GROUP= COMMUNICATIONS

UNIX specifics: all

Syntax

SSLCERTLOC="*file-path*"

file-path
The name of a file that contains a digital certificate.

Details

SSLCERTLOC is required for a server. It is needed on a client only if client authentication is being performed.

See Also

- "SSLCALISTLOC System Option" on page 369
- "SSLCLIENTAUTH System Option" on page 371
- "SSLCRLCHECK System Option" on page 372
- "SSLCRLLOC System Option" on page 373
- "SSLPVTKEYLOC System Option" on page 373
- "SSLPVTKEYPASS System Option" on page 374
- Appendix 3, "Using SSL in UNIX Environments," on page 403
- *SAS/CONNECT User's Guide*

SSLCLIENTAUTH System Option

Specifies whether a server should perform client authentication

Default: NOSSLCLIENTAUTH

Valid in: configuration file, SAS invocation, OPTIONS statement, SAS System Options window

Category: Communications: Networking and encryption

PROC OPTIONS GROUP= COMMUNICATIONS

UNIX specifics: all

Syntax

SSLCLIENTAUTH | NOSSLCLIENTAUTH

SSLCLIENTAUTH | NOSSLCLIENTAUTH
specifies whether the server should require SSL to provide client authentication.

Details

Server authentication is always performed, but SSLCLIENTAUTH enables a user to control client authentication. This option is meaningful only when used on a server.

See Also

- “SSLCALISTLOC System Option” on page 369
- “SSLCERTLOC System Option” on page 370
- “SSLCRLCHECK System Option” on page 372
- “SSLCRLLOC System Option” on page 373
- “SSLPVTKEYLOC System Option” on page 373
- “SSLPVTKEYPASS System Option” on page 374
- Appendix 3, “Using SSL in UNIX Environments,” on page 403
- *SAS/CONNECT User's Guide*

SSLCRLCHECK System Option

Specifies whether Certificate Revocation Lists (CRLs) are checked when a digital certificate is validated

Default: NOSSLCRLCHECK

Valid in: configuration file, SAS invocation, OPTIONS statement, SAS System Options window

Category: Communications: Networking and encryption

PROC OPTIONS GROUP= COMMUNICATIONS

UNIX specifics: all

Syntax

SSLCRLCHECK | NOSSLCRLCHECK

SSLCRLCHECK | NOSSLCRLCHECK
controls whether CRLs are checked when digital certificates are validated.

Details

Certificate Revocation Lists (CRLs) are published by Certificate Authorities (CAs) and contain a list of revoked digital certificates. The list contains only the revoked certificates that were issued by that particular certificate authority. This option is relevant for servers only if client authentication is used. Because clients always check server certificates, this option is always relevant for clients.

See Also

- “SSLCALISTLOC System Option” on page 369
- “SSLCERTLOC System Option” on page 370
- “SSLCLIENTAUTH System Option” on page 371
- “SSLCRLLOC System Option” on page 373
- “SSLPVTKEYLOC System Option” on page 373
- “SSLPVTKEYPASS System Option” on page 374

- Appendix 3, "Using SSL in UNIX Environments," on page 403
- *SAS/CONNECT User's Guide*

SSLCRLLOC System Option

Specifies the location for a Certificate Revocation List (CRL)

Default: none

Valid in: configuration file, SAS invocation, OPTIONS statement, SAS System Options window

Category: Communications: Networking and encryption

PROC OPTIONS GROUP= COMMUNICATIONS

UNIX specifics: all

Syntax

SSLCRLLOC="*file-path*"

file-path
specifies the location of a Certificate Revocation List (CRL).

Details

This option is relevant only when the SSLCRLCHECK option is enabled.

See Also

- "SSLCALISTLOC System Option" on page 369
- "SSLCERTLOC System Option" on page 370
- "SSLCLIENTAUTH System Option" on page 371
- "SSLCRLCHECK System Option" on page 372
- "SSLPVTKEYLOC System Option" on page 373
- "SSLPVTKEYPASS System Option" on page 374
- Appendix 3, "Using SSL in UNIX Environments," on page 403
- *SAS/CONNECT User's Guide*

SSLPVTKEYLOC System Option

Specifies where to find the private key that corresponds to the digital certificate that was specified with the SSLCERTLOC option

Default: none

Valid in: configuration file, SAS invocation, OPTIONS statement, SAS System Options window

Category: Communications: Networking and encryption

PROC OPTIONS GROUP= COMMUNICATIONS

UNIX specifics: all

Syntax

SSLPVTKEYLOC="*file-path*"

file-path
specifies the name of the file that contains the private key that corresponds to the digital certificate that was specified with the SSLCERTLOC= option.

Details

The value of this option must be the name of the file that contains the private key. This option is required only when SSLCERTLOC is specified.

See Also

- "SSLCALISTLOC System Option" on page 369
- "SSLCERTLOC System Option" on page 370
- "SSLCLIENTAUTH System Option" on page 371
- "SSLCRLCHECK System Option" on page 372
- "SSLCRLLOC System Option" on page 373
- "SSLPVTKEYPASS System Option" on page 374
- Appendix 3, "Using SSL in UNIX Environments," on page 403
- *SAS/CONNECT User's Guide*

SSLPVTKEYPASS System Option

Specifies the password that SSL should use to decrypt the private key that is stored in the file specified by SSLPVTKEYLOC

Default: none

Valid in: configuration file, SAS invocation, OPTIONS statement, SAS System Options window

Category: Communications: Networking and encryption

PROC OPTIONS GROUP= COMMUNICATIONS

UNIX specifics: all

Syntax

SSLPVTKEYPASS="*password*"

password
specifies the password that SSL should use to decrypt the private key that is stored in the file that is specified by SSLPVTKEYLOC.

Details

The option is required only when the private key is encrypted.

See Also

- "SSLCALISTLOC System Option" on page 369
- "SSLCERTLOC System Option" on page 370
- "SSLCLIENTAUTH System Option" on page 371
- "SSLCRLCHECK System Option" on page 372
- "SSLCRLLOC System Option" on page 373
- "SSLPVTKEYLOC System Option" on page 373
- Appendix 3, "Using SSL in UNIX Environments," on page 403
- *SAS/CONNECT User's Guide*

STDIO System Option

Specifies whether SAS should use stdin, stdout, and stderr

Default: NOSTDIO
Valid in: configuration file, SAS invocation, SASV9_OPTIONS environment variable
Category: Input control: Data processing
PROC OPTIONS GROUP= INPUTCONTROL
UNIX specifics: all

Syntax

–STDIO | –NOSTDIO

Details

This option tells SAS to take its input from standard input (stdin), to write its log to standard error (stderr), and to write its output to standard output (stdout).

This option is designed for running SAS in batch mode or from a shell script. If you specify this option interactively, SAS starts a line mode session. The STDIO option overrides the DMS, DMSEXP, and EXPLORER system options.

The STDIO option does not affect the assignment of the Stdio, Stdin, and Stderr filerefs. See "Filerefs Assigned by SAS in UNIX Environments" on page 140 for more information.

For example, in the following SAS command, the file MyInput is used as the source program, and files MyOutput and MyLog are used for the procedure output and log respectively.

```
sas -stdio < myinput > myoutput 2> mylog
```

If you are using the C shell, you should use parentheses:

```
(sas -stdio < myinput > myoutput ) >& output_log
```

See Also

- Chapter 6, “Printing and Routing Output,” on page 153

STIMEFMT System Option

Control the format of output produced by FULLSTIMER and STIMER

Default: M

Valid in: configuration file, SAS invocation, OPTIONS statement, SASV9_OPTIONS environment variable

Category: Log and procedure output control: SAS log

PROC OPTIONS GROUP= LOGCONTROL

UNIX specifics: all

Syntax

-STIMEFMT *value*

STIMEFMT=*value*

Details

The STIMEFMT system option controls the format of output produced by the FULLSTIMER and STIMER system options.

STIMEFMT takes the following values:

HOURS, H, or Z
: prints time statistics in **hh:mm:ss.ss** format.

MINUTES or M
: prints time statistics in **mm:ss.ss** format.

SECONDS or S
: prints time statistics in **ss.ss** format.

NORMAL or N
: prints time statistics in **ss.ss** format if the time is less than one minute, in **mm:ss.ss** format if the time is less than one hour, or in **hh:mm:ss.ss** format otherwise.

STIMER System Option

Writes a subset of system performance statistics to the SAS log

Default: STIMER

Valid in: configuration file, SAS invocation, OPTIONS statement, SASV9_OPTIONS environment variable

Category: Log and procedure output control: SAS log

PROC OPTIONS GROUP= LOGCONTROL

UNIX specifics: all

Syntax

–STIMER | –NOSTIMER

STIMER | NOSTIMER

STIMER
writes only real time and CPU time to the SAS log.

NOSTIMER
does not write any statistics to the SAS log.

Details

The STIMER system option specifies whether a subset of all the performance statistics of your system that are available to SAS are written to the SAS log. The following is an example of STIMER output.

Output 17.3 STIMER Output

```
real time   1.34 seconds
cpu time    0.04 seconds
```

STIMER displays the following statistics:

Table 17.2 Description of STIMER Statistics

Statistic	Description
Real Time	the amount of time spent to process the SAS job. Real time is also referred to as elapsed time.
CPU time	the total time spent to execute your SAS code and spent to perform system overhead tasks on behalf of the SAS process. This value is the combination of the user cpu and system cpu statistics from FULLSTIMER.

If both STIMER and FULLSTIMER are set, the FULLSTIMER statistics are printed.

Note: Starting in SAS 9, some procedures use multiple threads. On computers with multiple CPUs, the operating system can run more than one thread simultaneously. Consequently, CPU time might exceed real time in your STIMER output.

For example, a SAS procedure could use two threads that run on two separate CPUs simultaneously. The value of CPU time would be calculated as the following:

```
CPU1 time + CPU2 time = total CPU time
1 second + 1 second = 2 seconds
```

Since CPU1 can run a thread at the same time that CPU2 runs a separate thread, you can theoretically consume 2 CPU seconds in 1 second of real time. △

See Also

- "FULLSTIMER System Option" on page 331

SYSIN System Option

Specifies the default location of SAS source code when running in batch mode

Default: none
Valid in: configuration file, SAS invocation, SASV9_OPTIONS environment variable
Category: Environment control: Files
PROC OPTIONS GROUP= ENVFILES
UNIX specifics: all

Syntax

–SYSIN *filename*

The *filename* must be a valid UNIX filename.

Details

This option applies only when you are using batch mode. It is not necessary to precede the filename with the SYSIN option if the filename immediately follows the keyword SAS. For example, the following two SAS commands are equivalent:

```
sas saspgms/report1.sas    sas –sysin saspgms/report1.sas
```

See Also

- "Starting SAS Sessions in UNIX Environments" on page 4

SYSPRINT System Option

Specifies the destination for printed output

Default: default system printer
Valid in: configuration file, SAS invocation, OPTIONS statement, SASV9_OPTIONS environment variable
Category: Log and procedure output control: Procedure output
PROC OPTIONS GROUP= LISTCONTROL
UNIX specifics: all

Syntax

–SYSPRINT *destination* | *'destination option-list'*

SYSPRINT=*destination* | *'destination option-list'*

destination
is the name of a hardcopy device at your site. Consult your system administrator for a list of available destinations.

option-list
is the list of options to pass to the **lp** (or **lpr**) command.

Details

The SYSPRINT option specifies a destination for printed output other than default system printer. You can also use the option list to pass options to the **lp** (or **lpr**) command.

Note: When a fileref is assigned, the SYSPRINT option is queried. If the value of the SYSPRINT option is later changed, the fileref does not pick up this change. △

For details, see "Changing the Default Print Command in UNIX Environments" on page 165.

See Also

- "PRINTCMD System Option" on page 352
- Chapter 6, "Printing and Routing Output," on page 153

TAPECLOSE System Option

Specifies the default CLOSE disposition when reading and writing a SAS data library on tape

Default: REREAD

Valid in: configuration file, SAS invocation, OPTIONS statement, SAS System Options window, SASV9_OPTIONS environment variable

Category: Files: SAS Files

PROC OPTIONS GROUP= SASFILES

UNIX specifics: all

Syntax

–TAPECLOSE *disposition*

TAPECLOSE=*disposition*

The *disposition* can be one of the following values:

REREAD
rewind to the beginning of this file when it is closed. This is the default.

REWIND
rewind to the beginning of the tape after closing each member.

LEAVE
perform no tape positioning when you close a member.

FREE
rewind and dismount the tape when the next member is closed.

USER System Option

Specifies the name of the default permanent SAS data library

Default: none

Valid in: configuration file, SAS invocation, OPTIONS statement, SAS System Options window, SASV9_OPTIONS environment variable

Category: Environment control: Files

PROC OPTIONS GROUP= ENVFILES

UNIX specifics: *pathname* must be a valid UNIX pathname

See: USER System Option in *SAS Language Reference: Dictionary*

Syntax

-USER *pathname*

USER='*pathname*' | *libref*

pathname
identifies the directory containing your default permanent SAS data library. It must be a directory name.

libref
is the libref associated with the directory containing your default permanent SAS data library. It must already be assigned.

See Also

- "Using One-Level Names To Access Permanent Files (User Data Library)" on page 120

VERBOSE System Option

Controls whether SAS writes the settings of SAS system options to the terminal

Default: NOVERBOSE

Valid in: configuration file, SAS invocation, SASV9_OPTIONS environment variable

Category: Log and procedure output control: SAS log

PROC OPTIONS GROUP= LOGCONTROL

UNIX specifics: all

Syntax

–VERBOSE | –NOVERBOSE

-VERBOSE
writes the settings of SAS system options from the configuration file, the SAS command, and the SASV9_OPTIONS environment variable to the terminal. For the CONFIG option, VERBOSE lists the name of the configuration file or files.

-NOVERBOSE
does not write the settings of the system options to the terminal.

Details

SAS sends the system option information to standard output. If the standard output is a terminal, the list is displayed with the **more** command. You can also use the **more** command to scroll the file. The RETURN key scrolls one line; the space bar scrolls the entire display.

See Also

- "Customizing Your SAS Session Using System Options" on page 18
- "OPLIST System Option" on page 349

WORK System Option

Specifies the name of the Work library

Default: set in the installed **!SASROOT/sasv9.cfg** file

Valid in: configuration file, SAS invocation, SASV9_OPTIONS environment variable

Category: Environment control: Files

PROC OPTIONS GROUP= ENVFILES

UNIX specifics: all

See: WORK System Option in *SAS Language Reference: Dictionary*

Syntax

–WORK *pathname*

pathname
specifies the directory (not a filename) where your Work SAS data library can be created or found. SAS will create the directory if it does not exist. You must have write permission to *pathname*.

See Also

- "WORKINIT System Option" on page 382

WORKINIT System Option

Initializes the Work data library

Default: WORKINIT
Valid in: configuration file, SAS invocation, SASV9_OPTIONS environment variable
Category: Environment control: Files
PROC OPTIONS GROUP= ENVFILES
UNIX specifics: WORKINIT does not erase files from previous sessions
See: WORKINIT System Option in *SAS Language Reference: Dictionary*

Syntax

–WORKINIT | –NOWORKINIT

WORKINIT
specifies that a new subdirectory is to be created in the directory specified in the WORK option.

NOWORKINIT
specifies that the system is to use the directory specified by the WORK option.

- If the system does not find any old subdirectories, it creates a new one.
- If the system finds more than one old subdirectory, it uses the latest one.
- If file locking is in effect (see FILELOCKS option), the system looks for the latest unlocked directory. If it finds none, it creates a new one.

Details

The WORKINIT option controls whether the Work data library is initialized at SAS invocation.

See Also

- "FILELOCKS System Option" on page 329
- "WORK System Option" on page 381

WORKPERMS System Option

Sets the permissions of the SAS Work library when it is initially created

Default: `umask`
Valid in: configuration file, SAS invocation
Category: Environment control: Files

PROC OPTIONS GROUP= ENVFILES

UNIX specifics: all

Syntax

–WORKPERMS *permission value*

permission value
specifies the octal value representing the permissions desired for the SAS Work directory. Values can be any octal value setting the permission of a UNIX directory. Examples of values include: **`umask`**, 700, 755, 770, 775, and 777.

XCMD System Option

Specifies whether the X command is valid in the current SAS session

Default: XCMD

Valid in: configuration file, SAS invocation, SASV9_OPTIONS environment variable

Category: Environment control: Display

PROC OPTIONS GROUP= ENVDISPLAY

UNIX specifics: all

Syntax

XCMD | NOXCMD

XCMD
specifies that the X command is valid in the current SAS session.

NOXCMD
specifies that the X command is not valid in the current SAS session.

Details

The XCMD system option specifies whether the X command is valid in the current SAS session.

You cannot use several SAS statements, objects, and facilities if you use the NOXCMD system option. Examples of these statements, objects, and facilities include:

- the PIPE device type on the FILENAME statement
- the CALL SYSTEM routine
- the %SYSEXEC macro
- any facility that SAS uses to execute a shell-level command.

See Also

- "Executing Operating System Commands from Your SAS Session" on page 13
- "X Command" on page 221
- "FILENAME Statement" on page 293
- "CALL SYSTEM Routine" on page 239
- %SYSEXEC macro in "Macro Statements in UNIX Environments" on page 265

Summary of All SAS System Options in UNIX Environments

The following table lists every SAS system option available under UNIX. Many of these options have no host-specific behavior and are described in *SAS Language Reference: Dictionary*. If an option is available only under UNIX, it is described in this documentation. If an option is available under all environments but has some environment-specific behavior, it is described in both *SAS Language Reference: Dictionary* and this documentation. Use the following legend to see where to find more information on an option.

Access	indicates that the option is described in the SAS/ACCESS section of the SAS Help and Documentation.
Comp	indicates that the option is described in this section.
Connect	indicates that the option is described in *SAS/CONNECT User's Guide*.
DQ	indicates that the option is described in *SAS Data Quality Server: Reference*.
IT	indicates that the option is described in the documentation for SAS Integration Technologies, either with the SAS Integration Technologies software or on the SAS Web site.
LR	indicates that the option is described in *SAS Language Reference: Dictionary*.
Macro	indicates that the option is described in *SAS Macro Language: Reference*.
Methods	indicates that the option is described in *Communications Access Methods for SAS/CONNECT and SAS/SHARE*.
NLS	indicates that the option is described in *SAS National Language Support (NLS): User's Guide*.
Share	indicates that the option is described in *SAS/SHARE User's Guide*.
SPDE	indicates that the option is described in *SAS Scalable Performance Data Engine: Reference*.

The table also shows the default value for each option and where you can specify the option:

- at initialization: in the SAS command, in the SASV9_OPTIONS environment variable, or in the configuration file
- in the OPTIONS statement
- in the System Options window.

Table 17.3 Summary of All SAS System Options

		Can Be Specified In			
Name	Default	SAS Invocation, SASV9_OPTIONS, Configuration File	OPTIONS Statement	System Options Window	See
ALTLOG	no copy	X			Comp
ALTPRINT	no copy	X			Comp
APPLETLOC	!SASROOT/misc/applets	X			LR
ARMAGENT	sasarmmg	X	X	X	LR
ARMLOC	none	X	X	X	LR
ARMSUBSYS	ARM_NONE	X	X	X	LR
ASYNCHIO	see LR	X			LR
AUTHPROVIDERDOMAIN	NULL	X			LR
AUTOEXEC	see description	X			Comp
AUTOSAVELOC	none	X	X	X	Comp
AUTOSIGNON	NOAUTOSIGNON	X	X	X	Connect
BATCH	NOBATCH	X			Comp/LR
BINDING	DEFAULT	X	X	X	LR
BLKSIZE	256	X	X	X	Comp
BOTTOMMARGIN	0	X	X	X	LR
BUFNO	1	X	X	X	Comp/LR
BUFSIZE	0	X	X	X	Comp/LR
BYERR	BYERR	X	X	X	LR
BYLINE	BYLINE	X	X	X	LR
BYSORTED	BYSORTED	X			LR
CAPS	NOCAPS	X	X	X	LR
CARDIMAGE	NOCARDIMAGE	X	X	X	LR
CATCACHE	0	X			Comp/LR
CBUFNO	0				LR
CENTER	CENTER	X	X	X	LR
CHARCODE	NOCHARCODE	X	X	X	LR
CLEANUP	see description	X	X	X	Comp/LR
CMDMAC	NOCMDMAC	X	X	X	Macro
CMPLIB	NULL	X	X	X	LR

		Can Be Specified In			
Name	Default	SAS Invocation, SASV9_OPTIONS, Configuration File	OPTIONS Statement	System Options Window	See
CMPOPT	none	X	X	LR	
COLLATE	NOCOLLATE	X	X	X	LR
COLORPRINTINT	COLORPRINTINT	X	X	X	LR
COMAMID	TCP	X			Connect/ Share/ Methods
COMAUX1	X				Methods
COMAUX2	X				Methods
COMPRESS	NO	X	X	X	LR
CONFIG	see description	X (also SASV9_CONFIG environment variable)			Comp
CONNECTPERSIST	YES	X			Connect
CONNECTREMOTE	none	X	X	X	Connect
CONNECTSTATUS	X	X	X	X	Connect
CONNECTWAIT	X	X	X	X	Connect
COPIES	1	X	X	X	LR
CPUCOUNT	8	X	X	X	LR
CPUID	CPUID	X			LR
DATASTMTCHK	COREKEYWORDS	X	X	X	LR
DATE	DATE	X	X	X	LR
DATESTYLE	MDY	X	X	X	LR
DBCS	NODBCS	X			NLS
DBCSLANG	see description	X			NLS
DBCSTYPE	see description	X			NLS
DBSLICEPARM	(THREADED_APPS, 2)	X	X	X	LR
DBSRVTP	NONE	X			Access
DETAILS	NODETAILS	X	X	X	LR
DEVICE	none	X	X	X	Comp/ LR
DFLANG	ENGLISH	X	X	X	NLS
DKRICOND	ERROR	X	X	X	LR
DKROCOND	WARN	X	X	X	LR
DLDMGACTION	REPAIR	X	X	X	LR
DMR	NODMR	X			Connect
DMS	DMS	X			LR
DMSEXP	NODMSEXP	X			LR

		Can Be Specified In			
Name	Default	SAS Invocation, SASV9_OPTIONS, Configuration File	OPTIONS Statement	System Options Window	See
DMSLOGSIZE	99999	X			LR
DMSOUTSIZE	99999	X			LR
DMSYNCHK	NODMSSYNCHK	X	X	X	LR
DQLOCALE	NULL	X	X	X	DQ
DQSETUPLOC	NULL	X	X	X	DQ
DSNFERR	DSNFERR	X	X	X	LR
DTRESET	NODTRESET	X	X	X	LR
DUPLEX	NODUPLEX	X	X	X	LR
ECHO	none	X			Comp
ECHOAUTO	NOECHOAUTO	X			LR
EDITCMD	none	X	X		Comp
EMAILAUTHPROTOCOL	LOGIN	X			LR
EMAILHOST	localhost	X			LR
EMAILID	none	X			LR
EMAILPORT	25	X			LR
EMAILPW	NULL	X			LR
EMAILSYS	SMTP	X (except SASV9_OPTIONS environment variable)			Comp
ENCODING	latin1	X			NLS
ENGINE	V9	X			Comp/ LR
ERRORABEND	NOERRORABEND	X	X	X	LR
ERRORBYABEND	NOERRORBYABEND	X	X	X	LR
ERRORCHECK	NORMAL	X			LR
ERRORS	20	X	X	X	LR
EXPLORER	NOEXPLORER	X			LR
FILELOCKS	FAIL	X			Comp
FIRSTOBS	1	X	X	X	LR
FMTERR	FMTERR	X	X	X	LR
FMTSEARCH	WORK library	X	X	X	LR
FSDBTYPE	none	X			NLS
FORMCHAR	`|----|+|---+=| -/\<>*`	X	X	X	LR
FORMDLIM	none	X	X	X	LR
FORMS	DEFAULT	X	X	X	LR

		Can Be Specified In			
Name	Default	SAS Invocation, SASV9_OPTIONS, Configuration File	OPTIONS Statement	System Options Window	See
FONTSLOC	!SASROOT/misc/font	X			Comp/LR
FSIMM	none	X			NLS
FSIMMOPT	none	X			NLS
FULLSTIMER	NOFULLSTIMER	X	X		Comp
GISMAPS	see description				LR
GWINDOW	GWINDOW	X	X	X	LR
HELPADDR	NULL	X	X	X	Web
HELPBROWSER	SAS	X	X	X	Web
HELPENCMD	none	X			LR
HELPHOST	NULL	X	X	X	Web
HELPINDEX	/help/common.hlp/index.txt, /help/common.hlp/keywords.htm, common.hhk	X			Comp/LR
HELPLOC	!SASROOT/X11/native_help	X			Comp/LR
HELPPORT	0	X	X	X	Web
HELPTOC	/help/common.hlp/contents.txt, /help/common.hlp/toc.htm, common.hhc	X			Comp/LR
IBUFSIZE	0	X	X	X	LR
IMPLMAC	NOIMPLMAC	X	X	X	Macro
INITCMD	none	X			LR
INITSTMT	none	X			LR
INVALIDDATA	.	X	X	X	LR
JREOPTIONS	none	X			Comp
LABEL	LABEL	X	X	X	LR
LAST	_NULL_	X	X	X	LR
LEFTMARGIN	0	X	X	X	LR
LINESIZE	see description	X	X	X	Comp/LR
LOADMEMSIZE	0	X			Comp
LOCALE	English	X	X	X	NLS
LOG	see description	X			Comp

		Can Be Specified In			
Name	Default	SAS Invocation, SASV9_OPTIONS, Configuration File	OPTIONS Statement	System Options Window	See
LOGPARM	none	X			LR
LPTYPE	see description	X	X		Comp
MACRO	MACRO	X			Macro
MAPS	!SASROOT/maps	X	X	X	LR/ Comp
MAUTOLOCDISPLAY	NOMAUTO-LOCDISPLAY	X	X	X	Macro
MAUTOSOURCE	MAUTOSOURCE	X	X	X	Macro
MAXMEMQUERY	6M	X			Comp
MAXSEGRATIO	75	X	X	X	SPDE
MCOMPILENOTE	NONE	X	X	X	Macro
MERGENOBY	NOWARN	X	X	X	LR
MERROR	MERROR	X	X	X	Macro
MEMSIZE	value set in **!SASROOT/ sasv9.cfg**	X			Comp
METAAUTORESOURCES	NULL	X			LR
METACONNECT	NULL	X	X	X	LR
METAENCRYPTALG	NONE	X			LR
METAENCRYPTLEVEL	EVERYTHING	X			LR
METAID	none	X			LR
METAPASS	none	X	X	X	LR
METAPORT	0	X	X	X	LR
METAPROFILE	NULL	X			LR
METAPROTOCOL	BRIDGE	X	X	X	LR
METAREPOSITORY	DEFAULT	X			LR
METASERVER	none	X	X	X	LR
METAUSER	none	X	X	X	LR
MINDELIMITER	NULL	X	X	X	Macro
MFILE	NOMFILE	X	X	X	Macro
MINPARTSIZE	0	X			SPDE
MISSING	.	X	X	X	LR
MLOGIC	NOMLOGIC	X	X	X	Macro
MLOGICNEST	NOMLOGICNEST	X	X	X	Macro
MPRINT	NOMPRINT	X	X	X	Macro
MPRINTNEST	NOMPRINTNEST	X	X	X	Macro

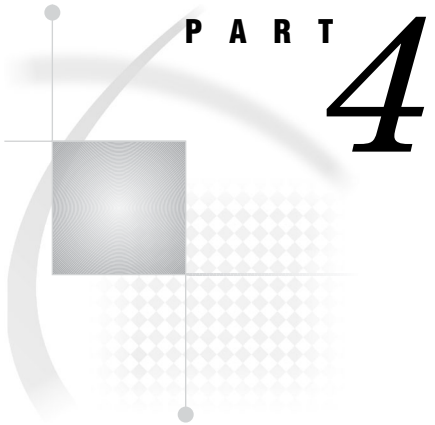
		Can Be Specified In			
Name	Default	SAS Invocation, SASV9_OPTIONS, Configuration File	OPTIONS Statement	System Options Window	See
MRECALL	NOMRECALL	X	X	X	Macro
MSG	!SASROOT/sasmsg	X			Comp
MSGCASE	NOMSGCASE	X			LR
MSGLEVEL	N	X	X	X	LR
MSTORED	NOMSTORED	X	X	X	Macro
MSYMTABMAX	4M	X	X	X	Comp/ Macro
MULTENVAPPLE	NOMULTENVAPPLE	X	X		LR
MVARSIZE	32K	X	X	X	Comp/ Macro
NETENCRYPT	NONETENCRYPT	X	X	X	Connect
NETENCRYPTALGORITHM	none	X	X	X	Connect
NETENCRYPTKEYLEN	0	X	X	X	Connect
NETMAC	NETMAC	X	X	X	Connect
NEWS	!SASROOT/misc/base/news	X			Comp/ LR
NLSCOMPATMODE	NONLSCOMPATMODE	X			NLS
NOTES	NOTES	X	X	X	LR
NUMBER	NUMBER	X	X	X	LR
OBJECTSERVER	NOOBJECTSERVER	X			LR
OBS	MAX	X	X	X	Comp/ LR
OPLIST	NOOPLIST	X			Comp
ORIENTATION	PORTRAIT	X	X	X	LR
OVP	NOOVP	X	X	X	LR
PAGEBREAK INITIAL	NOPAGEBREAK INITIAL	X			LR
PAGENO	1	X	X	X	LR
PAGESIZE	see description	X	X	X	Comp/ LR
PAPERDEST	none	X	X	X	LR
PAPERSIZE	LETTER	X	X	X	LR
PAPERSOURCE	none	X	X	X	LR
PAPERTYPE	PLAIN	X	X	X	LR

		Can Be Specified In			
Name	Default	SAS Invocation, SASV9_OPTIONS, Configuration File	OPTIONS Statement	System Options Window	See
PARM	none	X	X	X	LR
PARMCARDS	FT15F001	X	X	X	LR
PATH	!SASROOT/sasexe	X			Comp
PRINT	see description	X			Comp
PRINTCMD	none	X	X		Comp
PRINTERPATH	none	X (Available on MIPS ABI and Intel ABI only)			LR
PRINTINIT	NOPRINTINIT	X			LR
PRINTMSGLIST	PRINTMSGLIST	X	X	X	LR
PRODTOC	NULL	X			Web
QUOTELENMAX	QUOTELENMAX	X	X	X	LR
REALMEMSIZE	0	X			COMP
REPLACE	REPLACE	X	X	X	LR
REUSE	NO	X	X	X	LR
RIGHTMARGIN	0	X	X	X	LR
RSASUSER	NORSASUSER	X			Comp/ LR
RTRACE	none	X			Comp
RTRACELOC	none	X			Comp
S	0	X	X	X	Comp/ LR
S2	0	X	X	X	Comp/ LR
SASAUTOS	SASAUTOS fileref	X	X	X	Comp/ Macro
SASCMD	none	X			Connect
SASFRSCR	none	Valid in SAS Component Language			Connect
SASHELP	!SASROOT/ sashelp	X			Comp/ LR
SASMSTORE	none	X	X	X	Macro
SASSCRIPT	!SASROOT/misc/ connect	X	X	X	Comp/ Connect
SASUSER	~SASUSER.800	X			Comp/ LR
SEQ	8	X	X	X	LR
SEQENGINE	V9TAPE	X	X	X	Comp
SERROR	SERROR	X	X	X	Macro
SET	none	X	X		Comp

		Can Be Specified In			
Name	Default	SAS Invocation, SASV9_OPTIONS, Configuration File	OPTIONS Statement	System Options Window	See
SETINIT	NOSETINIT	X			LR
SKIP	0	X	X	X	LR
SOLUTIONS	SOLUTIONS	X			LR
SORTANOM	none	X	X		Comp
SORTCUT	0	X	X		Comp
SORTCUTP	0	X	X		Comp
SORTDEV	see description	X	X		Comp
SORTDUP	physical	X	X	X	LR
SORTEQUALS	SORTEQUALS	X	X	X	LR
SORTNAME	none	X	X		Comp
SORTPARM	none	X	X		Comp
SORTPGM	BEST	X	X		Comp
SORTSEQ	none	X	X	X	NLS
SORTSIZE	MAX	X	X	X	Comp/ LR
SOURCE	SOURCE	X	X	X	LR
SOURCE2	NOSOURCE2	X	X	X	LR
SPDEINDEXSORTSIZE	32M	X	X	X	SPDE
SPDEMAXTHREADS	0	X			SPDE
SPDESORTSIZE	32M	X	X	X	SPDE
SPDEUTILLOC	NULL	X			SPDE
SPDEWHEVAL	COST	X			SPDE
SPOOL	NOSPOOL	X	X	X	LR
SSLCALISTLOC	none	X	X	X	Comp/ Connect
SSLCERTLOC	none	X	X	X	Comp/ Connect
SSLCLIENTAUTH	NOSSLCLIENTAUTH	X	X	X	Comp/ Connect
SSLCRLCHECK	NOSSLCRLCHECK	X	X	X	Comp/ Connect
SSLCRLLOC	none	X	X	X	Comp/ Connect
SSLPVTKEYLOC	none	X	X	X	Comp/ Connect
SSLPVTKEYPASS	none	X	X	X	Comp/ Connect

		Can Be Specified In			
Name	Default	SAS Invocation, SASV9_OPTIONS, Configuration File	OPTIONS Statement	System Options Window	See
STARTLIB	STARTLIB	X			LR
STDIO	NOSTDIO	X			Comp
STIMEFMT	M	X			Comp
STIMER	STIMER	X	X		Comp
SUMSIZE	0	X			LR
SYMBOLGEN	NOSYMBOLGEN	X	X	X	Macro
SYNCHIO	see LR	X			LR
SYNTAXCHECK	SYNTAXCHECK	X	X	X	LR
SYSIN	none	X			Comp
SYSPARM	none	X	X	X	Macro
SYSPRINT	default system printer	X			Comp
SYSPRINTFONT	none	X	X	X	LR
SYSRPUTSYNC	NO	X			Connect
TAPECLOSE	REREAD	X	X	X	Comp
TBUFSIZE	0	X	X	X	Connect
TCPPORTFIRST	0	SAS invocation on the remote host			Connect
TCPPORTLAST	0	SAS invocation on the remote host			Connect
TERMINAL	TERMINAL	X			LR
TERMSTMT	none	X			LR
TEXTURELOC	none	X	X	X	LR
THREADS	NOTHREADS	X	X	X	LR
TOOLSMENU	TOOLSMENU	X			LR
TOPMARGIN	0	X	X	X	LR
TRAINLOC	none	X			LR
TRANTAB	none	X	X	X	NLS
UNIVERSALPRINT	UNIVERSALPRINT	X			LR
USER	none	X	X	X	Comp/ LR
UTILLOC	none	X			LR
UUIDCOUNT	100	X	X	X	IT
UUIDGENDHOST	none	X			IT
V6CREATEUPDATE	ERROR	X			LR
VALIDFMTNAME	LONG	X	X	X	LR
VALIDVARNAME	V7	X	X	X	LR

		Can Be Specified In			
Name	Default	SAS Invocation, SASV9_OPTIONS, Configuration File	OPTIONS Statement	System Options Window	See
VERBOSE	NOVERBOSE	X			Comp
VIEWMENU	VIEWMENU	X			LR
VNFERR	VNFERR	X	X	X	LR
WORK	see description	X			Comp/ LR
WORKINIT	WORKINIT	X			Comp/ LR
WORKPERMS	WORKPERMS	X			Comp
WORKTERM	WORKTERM	X	X	X	LR
XCMD	XCMD	X			Comp
YEARCUTOFF	1920	X	X	X	LR



PART 4

Appendices

APPENDIX

1

The !SASROOT Directory

Introduction to the !SASROOT Directory

When SAS is installed, its entire directory structure is placed on a node in your file system. This node, which forms the root of SAS, is called the **!SASROOT** directory. Although **!SASROOT** can be located anywhere in your file system, the system administrator typically installs it in **/usr/local/sas91**. The SET system option is used to specify the name of the **!SASROOT** directory. (See "SET System Option" on page 363.)

Contents of the !SASROOT Directory

The **!SASROOT** directory contains the files required to use SAS. This directory includes invocation points, configuration files, sample programs, catalogs, data sets, and executable files. You do not need to know the organization of these directories to use SAS.

Many of the files in these directories are not text files, so do not try to read a file unless the UNIX **file** command verifies that it contains ASCII text. Typically, files with a .sas, .cfg, .scr, or .txt extension are readable. If all available SAS products are installed on your system, the **!SASROOT** directory contains the files and directories listed in the following tables:

Table A1.1 SAS Files in the !SASROOT Directory

SAS File	Description of Contents
sas	is the default invocation point for SAS.
sassetup	is the invocation point for SAS Setup, the installation program for SAS.
setinit.sas	is the SAS program used for updating licensing information.
sasv9.cfg	is the system configuration file for SAS

Table A1.2 SAS Directories in the !SASROOT Directory

SAS Directory	Description of Contents
bin	contains the invocation scripts for each language listed in the **nls** directory. This directory also contains the sasenv script that sets the environment variables that are required by SAS.
dbcs	contains the subdirectories for a DBCS installation.
gismaps	contains Census Tract maps for SAS/GIS software.
install	contains the SAS Setup program files.
maps	is a SAS data library that contains SAS data sets used by SAS/GRAPH software to produce maps. You receive some maps with SAS/GRAPH software. Additional maps are available in the SAS Map Data Library Series.
misc	contains miscellaneous components, such as Java applets, fonts, and textures. This directory also contains components for various SAS products, such as script files for SAS/CONNECT and thin client interfaces for SAS/SHARE.
nls	contains subdirectories for national language support. These directories include: **en** (English), **ja** (Japanese), **ko** (Korean), and **zh** (Simplified Chinese). Each language directory contains a **sascfg** subdirectory that contains the SAS data files created during installation.
samples	contains sample programs for different SAS products. These programs are organized by product subdirectory. Sample programs are optionally installed. Consequently, this directory might not include samples for every SAS product, or the directory might be empty.
sasautos	contains predefined SAS macros. See “Using Autocall Libraries in UNIX Environments” on page 266.
sasexe	contains executable files for different SAS products.
sashelp	is a SAS data library that contains online help files, menus, descriptions of graphics devices, and other catalogs used by SAS procedures that support windows.
sasmsg	contains files that contain all of the messages and notes that are used by SAS.
saspgm	contains various components of SAS products.
sastest	contains files that are used by the Feature Testing Tool.
utilities	contains man pages and utility programs. See “The Utilities Directory in UNIX Environments” on page 399 for more information.
X11	contains the files needed to run SAS with the X Window System. These files include bitmap files, online help files, and resource files.

APPENDIX

2

Tools for the System Administrator

The Utilities Directory in UNIX Environments

The **!SASROOT/utilities** directory contains two important subdirectories:

man contains the online manual pages for SAS. "Installing Manual Pages" on page 399 describes how to make these pages accessible to users through the UNIX **man** command. This directory contains two subdirectories: **man1**, which contains unformatted man pages, and **cat1**, which contains formatted man pages.

bin contains the executable files for administrative tools. "Utilities in the /bin Directory" on page 400 describes some of the tools in this directory.

Installing Manual Pages

To be able to read the manual pages in the **utilities/man** directory, move the files to the **man1** subdirectory of the location of the other man files for your system. This location is usually **/usr/man** or **/usr/local/man**. Execute the **man man** command to determine the appropriate pathname for your system. When you have found the correct pathname, use the following command to move the SAS man files:

```
cp -r sasroot/utilities/man/* pathname
```

where *pathname* is the directory location of your system man files.

For example, the following command enables you to access online help by moving the SAS man files from the **!SASROOT** directory to the **man1** file in your system's man directory.

```
cp /usr/local/sas91/utilities/man/* /usr/local/man/man1
```

After you have issued this command, you can access online help with the **man sas** command.

You can also add the directory to your system's MANPATH environment variable if it has been previously defined.

Utilities in the /bin Directory

The following table briefly describes some of the tools in the **utilities/bin** directory. You can also use the **man** command for information on these utilities. You will need ROOT permissions to execute these commands.

Table A2.1 Tools for the System Administrator

Tool Name	Function
cleanwork	deletes any leftover Work directories whose associated SAS process has terminated.
patchname	resets the name of the **sasroot** directory in the specified executable file.

cleanwork Command

Deletes any leftover Work and Utility directories whose associated SAS process has ended

Syntax

cleanwork *directory*

directory
names the directory containing the Work and Utility directories. The name must match the value specified in the WORK system option (which is typically **/usr/tmp** and installed in the !SASROOT/sasv9.cfg file) or the value specified in the UTILLOC system option.

Note: Unless **cleanwork** is run by root, user permissions might prevent you from deleting a directory. △

Details

The **cleanwork** command deletes any directories that were assigned to the Work data library or directories assigned by the UTILLOC system option. **cleanwork** deletes only the SAS jobs that are on the UNIX box that your SAS session is running on. Each SAS process is in the format

```
SAS_workcode_nodename
```

or

```
SAS_utilcode_nodename
```

code
is a 12-character code. The first four characters are randomly generated numbers. The next eight characters are based on the hexadecimal process ID of the SAS session. Processes that are active are not deleted.

nodename
specifies the name of the UNIX box where the SAS process is running.

If you are working on nodename *jupiter*, then the **cleanwork** command deletes all directories with inactive processes on *jupiter*. **cleanwork** does not delete a directory that is associated with an orphaned process if that process is still showing up as active. In this case, you need to manually kill the process and then rerun **cleanwork**.

Note: The **cleanwork** utility for Version 8 will work on Version 8 of SAS as well as prior versions of SAS. △

See Also

- "Work Data Library" on page 120

patchname Command

Resets the name of the !SASROOT directory in the specified executable file

Syntax

patchname *filepath sasroot-directory-pathname*

filepath
specifies the absolute pathname of the file in which to set the **!SASROOT** directory.

sasroot-directory-pathname
specifies the absolute pathname of the new **!SASROOT** directory.

Details

The **patchname** command resets the name of the **!SASROOT** directory in the specified executable file to the specified directory. When you install SAS, the installation program uses **patchname** to write the name of the **!SASROOT** directory to the file that needs this information: the executable file containing the **sas** command. If you change the **!SASROOT** directory, you must use **patchname** to alter this file.

APPENDIX

3

Using SSL in UNIX Environments

What Is SSL?

SSL (Secure Sockets Layer)

SSL is a protocol that provides secure network communications. Developed by Netscape Communications, SSL uses the encryption algorithms that were developed by RSA Security, Inc. and other cryptography experts.

In addition to providing encryption services, SSL performs client and server authentication and uses message authentication codes. SSL is supported by both Netscape Navigator and Internet Explorer. Many Web sites use this protocol to protect confidential user information, such as credit card numbers. URLs that require an SSL connection begin with https: instead of http:. The SSL protocol is application independent, which allows protocols such as HTTP, FTP, and Telnet to be transparently layered above it. SSL is optimized for HTTP.

Certification Authorities (CAs)

As e-business proliferates, there is a great need to ensure the confidentiality of business transactions over a network between an enterprise and its consumers, between enterprises, and within an enterprise. Cryptography products provide security services by exploiting digital certificates, public-key cryptography, private-key cryptography, and digital signatures. Certification authorities (CAs) create and maintain digital certificates, which also help preserve confidentiality.

Various commercial CAs, such as VeriSign and Thawte, provide competitive services for the e-commerce market. You can also develop your own CA by using products from companies such as RSA Security and Microsoft or from the Open Source Toolkit OpenSSL. From a trusted CA, members of an enterprise can obtain digital certificates to facilitate their e-business needs. The CA provides a variety of ongoing services to the business client that include handling digital certificate requests, issuing digital certificates, and revoking digital certificates.

Public and Private Keys

Public-key cryptography uses a public and a private key pair. The public key can be known by anyone, therefore, anyone can send a confidential message. The private key is confidential and known only to the owner of the key pair, therefore, only the owner can read the encrypted message. The public key is used primarily for encryption, but it can also be used to verify digital signatures. The private key is used primarily for decryption, but it can also be used to generate a digital signature.

Digital Signatures

A digital signature affixed to an electronic document or to a network data packet is like a personal signature that concludes a hand-written letter or that validates a credit card transaction. Digital signatures are a safeguard against fraud. A unique digital signature results from using a private key to encrypt a message digest. Receipt of a document that contains a digital signature enables the receiver to verify the source of the document. Electronic documents can be verified if you know where the document came from, who sent it, and when it was sent. Another form of verification comes from MACs, which ensure that a document has not been changed since it was signed.

Digital Certificates

Digital certificates are electronic documents that ensure the binding of a public key to an individual or an organization. Digital certificates provide protection from fraud.

Usually, a digital certificate contains a public key, a user's name, and an expiration date. It also contains the name of the certification authority (CA) that issued the digital certificate and a digital signature that is generated by the CA. The CA's validation of an individual or an organization allows that individual or organization to be accepted at sites that trust the CA.

Using SSL

Overview of SSL Set-Up Process

The details for installing and setting up SSL at your site are based on the operating environment and the digital certificate services software that you use. However, the following tasks are basic:

1. Access the appropriate software for installing and setting up digital certificate services under your operating environment.
2. Define a Certificate Authority (CA).
3. Request that digital certificates be generated by the CA for users, machines, and other CAs.
4. Store the digital certificates in a trusted repository.
5. View the properties of the generated digital certificates.
6. Start a server.
7. Connect to the server.

SSL for SAS

You can set SAS system options to use SSL in a SAS session. See the SAS options that are appropriate to your operating environment.

SSL for UNIX

System and Software Requirements for SSL under UNIX

The system and software requirements for using SSL under UNIX operating environments are:

- A computer that runs UNIX.
- Internet access and a Web browser such as Netscape Navigator or Internet Explorer.
- The TCP/IP communications access method.
- Access to the OpenSSL utility at www.openssl.org/source if you plan to use the OpenSSL CA.
- Knowledge of your site's security policy, practices, and technology. The properties of the digital certificates that you request are based on the security policies that have been adopted at your site.

Setting Up SSL under UNIX

Perform the following tasks to set up and use SSL:

1. Download and build SSL.
2. Create digital certificate requests.
3. Generate digital certificates from requests.
4. View the digital certificates.
5. Terminate the OpenSSL utility.
6. Create a trusted list of CAs.

Downloading and Building SSL under UNIX

If you want to use OpenSSL as your trusted Certificate Authority (CA), follow the instructions for downloading and building OpenSSL that are given at www.openssl.org/source. For complete documentation about the OpenSSL utility, visit www.openssl.org/docs/apps/openssl.html.

Information about alternative CAs and their Web sites follows:

- For VeriSign, see www.verisign.com
- For Thawte, see www.thawte.com

Creating Digital Certificate Requests under UNIX

To enable an SSL connection at your site, you must

- obtain a digital certificate from a certification authority (CA).
- create a digital certificate request from which a digital certificate is generated.
- request one or more digital certificates for the CA (if you will be running your own CA), the server, and the client (optional).

The tasks that you perform to request a digital certificate for the CA, the server, and the client are similar, however, the values that you specify will be different.

In this example, Proton, Inc. is the organization that is applying for certification authority status by using OpenSSL. After Proton, Inc. becomes a CA, it can serve as a Certificate Authority for issuing digital certificates to clients (users) and servers on its network.

Perform the following tasks:

1. Select the **apps** subdirectory of the directory where OpenSSL was built.
2. Initialize OpenSSL.

```
$ openssl
```

3. Issue the appropriate command to request a digital certificate. (See Table A3.1 on page 407.) The functions of the arguments used in the commands are shown in Table A3.2 on page 407

Table A3.1 Open SSL Commands for Requesting a Digital Certificate

Request Certificate for	OpenSSL Command
CA	req -config ./openssl.cnf -new -out sas.req -keyout saskey.pem -nodes
Server	req -config ./openssl.cnf -new -out server.req -keyout serverkey.pem
Client	req -config ./openssl.cnf -new -out client.req -keyout clientkey.pem

Table A3.2 Arguments and Values Used in OpenSSL Commands

OpenSSL Arguments and Values	Functions
req	requests a certificate
-config ./openssl.cnf	specifies where the configuration details for the OpenSSL program are stored
-new	identifies the request as new
-out sas.req	specifies where the certificate request will be stored
-keyout saskey.pem	specifies where the private key will be stored
-nodes	prevents the private key from being encrypted

4 Informational messages are displayed and prompts for additional information appear according to the specific request.

To accept a default value, press the Return key. To change a default value, type the appropriate information and press the Return key.

Note: Unless the -NODES option is used in the OpenSSL command when creating a digital certificate request, OpenSSL will prompt you for a password before allowing access to the private key. △

The following is an example of a request for a digital certificate:

```
OpenSSL> req -config ./openssl.cnf -new -out sas.req -keyout saskey.pem -nodes
Using configuration from ./openssl.cnf
Generating a 1024 bit RSA private key
.........................++++++
.........................................++++++
writing new private key to 'saskey.pem'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [US]:
State or Province Name (full name) [North Carolina]:
Locality Name (city) [Cary]:
Organization Name (company) [Proton INC.]:
Organizational Unit Name (department) [IDB]:
Common Name (YOUR name) []: Joe Bass
```

```
Email Address []:Joe.Bass@proton.com


Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
OpenSSL>
```

The request for a digital certificate is complete.

Note: For the server, the Common Name must be the name of the machine on which the server runs; for example, apex.serv.com. △

Generating Digital Certificates on UNIX

Perform the following tasks to generate digital certificates for a CA, a server, and a client.

1 Issue the appropriate command to generate a digital certificate from the digital certificate request. (See Table A3.3 on page 408.)

Table A3.3 OpenSSL Commands for Generating Digital Certificates under UNIX

Generate Certificate for	OpenSSL Command
CA	x509 req -in sas.req -signkey saskey.pem -out sas.pem
Server	ca -config ./openssl.cnf -in server.req -out server.pem -nodes
Client	ca -config ./openssl.cnf -in client.req -out client.pem

The functions performed by the OpenSSL arguments and values are shown in Table A3.4 on page 408.

Table A3.4 Arguments and Values Used in OpenSSL Commands on UNIX

OpenSSL Arguments and Values	Functions
x509	identifies the certificate display and signing utility
req	specifies that a certificate be generated from the request
ca	identifies the certificate authority utility
-config ./openssl.cnf	specifies where the configuration details for the OpenSSL utility are stored
-in filename.req	specifies where the input for the certificate request is stored
-out filename.pem	specifies where the certificate will be stored
-signkey saskey.pem	specifies the private key that will be used to sign the certificate that is generated by the certificate request

2 Informational messages are displayed and prompts for additional information appear according to the specific request.

To accept a default value, press the Return key. To change a default value, type the appropriate information, and press the Return key.

Sample dialog for creating a server digital certificate follows:

Note: The password is for the CA's private key. △

```
Using configuration from ./openssl.cnf
Enter PEM pass phrase: password
Check that the request matches the signature
Signature ok
The Subjects Distinguished Name is as follows
countryName           :PRINTABLE:'US'
stateOrProvinceName   :PRINTABLE:'NC'
localityName          :PRINTABLE:'Cary'
organizationName      :PRINTABLE:'Proton, Inc.'
organizationalUnitName:PRINTABLE:'Development'
commonName            :PRINTABLE:'Server'
Certificate is to be certified until Oct 16 17:48:27 2003 GMT (365 days)
Sign the certificate? [y/n]:y
1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries Data Base Updated
```

The subject's Distinguished Name is obtained from the digital certificate request.

A root CA digital certificate is self-signed. *Self-signed* means that the digital certificate is signed with the private key that corresponds to the public key that is in the digital certificate. Except for root CAs, digital certificates are usually signed with a private key that corresponds to a public key that belongs to someone else, usually the CA.

Viewing Digital Certificates

To view a digital certificate, issue the following command:

```
openssl> x509  -text -in filename.pem
```

A digital certificate contains data that was collected to generate the digital certificate timestamps, a digital signature, and other information. However, because the generated digital certificate is encoded (usually in PEM format), it is unreadable.

Terminating OpenSSL

To terminate OpenSSL, type **`quit`** at the prompt.

Creating a CA Trust List

After generating digital certificates for the CA, the server, and the client (optional), you must identify for the OpenSSL client application one or more CAs that are to be trusted. This list is called a *trust list*.

If there is only one CA to trust, specify the name of the file that contains the OpenSSL CA digital certificate, in the client application.

If multiple CAs are to be trusted, create a new file and copy-and-paste into it the contents of all the digital certificates for CAs to be trusted by the client application.

Use the following template to create a CA trust list:

```
Certificate for OpenSSL CA

-----BEGIN CERTIFICATE-----

<PEM encoded certificate>

-----END CERTIFICATE-----


Certificate for Keon CA

-----BEGIN CERTIFICATE-----

<PEM encoded certificate>

-----END CERTIFICATE-----


Certificate for Microsoft CA

-----BEGIN CERTIFICATE-----


-----END CERTIFICATE-----
```

Because the digital certificate is encoded, it is unreadable. Therefore, the content of the digital certificate in this example is represented as **<PEM encoded certificate>** . The content of each digital certificate is delimited with a **-----BEGIN CERTIFICATE-----** and **-----END CERTIFICATE-----** pair. All text outside the delimiters is ignored. Therefore, you might want to use undelimited lines for descriptive comments. In the preceding template, the file that is used contains the content of digital certificates for the CAs: OpenSSL, Keon, and Microsoft.

Note: If you are including a digital certificate that is stored in DER format, you must first convert it to PEM format. For more information, see "Converting between PEM and DER File Formats" on page 410. △

Converting between PEM and DER File Formats

By default, OpenSSL files are created in PEM (Privacy Enhanced Mail) format. SSL files that are created in Windows operating environments are created in DER (Distinguished Encoding Rules) format.

Under Windows, you can import a file that is created in either PEM or DER format. However, a digital certificate that is created in DER format must be converted to PEM format before it can be included in a trust list under UNIX.

An example of converting a server digital certificate from PEM input format to DER output format follows:

```
OpenSSL> x509 -inform PEM -outform DER -in server.pem -out server.der
```

An example of converting a server digital certificate from DER input format to PEM output format follows:

```
OpenSSL> x509 -inform DER -outform PEM -in server.der -out server.pem
```

SSL Language Elements

APPENDIX

4

SAS Releases in UNIX Environments

SAS Releases in UNIX Environments

The following table lists the releases of SAS that were shipped for each UNIX operating environment.

Table A4.1 SAS Releases in UNIX Environments

UNIX Operating Environment	SAS Versions and Releases							
	9.1	**9**[1]	**8**	**7**	**6.12**	**6.11**	**6.10**	**6.09**
AIX	•	•	•	•	•	•		•
HP-UX[2]	•	•	•	•	•	•		•
Tru64 UNIX[3]	•	•	•	•	•	•	•	
Solaris[4]	•	•	•	•	•	•		•
Linux[5]	•	•	•					
Intel ABI			•[6]			•	•	
IRIX[7]			•					
SunOS[8]					•	•		•
MIPS ABI						•	•	

[1]Starting in SAS 9, AIX, HP-UX, and Solaris are 64-bit enabled.

[2]For SAS 9 and 9.1, both HP-UX PA Risc in 64-bit environments and HP-UX for Itanium are supported.

[3]formerly DIGITAL UNIX.

[4]refers to SUN systems starting with release 2.5 (also called 5.5).

[5]refers to Linux for Intel in 32-bit environments.

[6]refers to NCR MP/RAS, UnixWare, Sequent Dynix/Ptx, and Solaris for Intel.

[7]IRIX (the operating system created by Silicon Graphics) was formerly included in the MIPS ABI architectural group, which SAS supported only in SAS Releases 6.10 and 6.11.

[8]refers to SUN system releases up to and including release 2.4.

APPENDIX
5

Recommended Reading

Recommended Reading

Here is the recommended reading list for this title:

- *SAS Language Reference: Concepts*
- *SAS Language Reference: Dictionary*
- *Base SAS Procedures Guide*
- *SAS Macro Language: Reference*
- *SAS National Language Support (NLS): User's Guide*

For a complete list of SAS publications, see the current *SAS Publishing Catalog*. To order the most current publications or to receive a free copy of the catalog, contact a SAS representative at

SAS Publishing Sales
SAS Campus Drive
Cary, NC 27513
Telephone: (800) 727-3228*
Fax: (919) 677-8166
E-mail: **sasbook@sas.com**
Web address: **support.sas.com/publishing**
* For other SAS Institute business, call (919) 677-8000.

Customers outside the United States should contact their local SAS office.

Glossary

access descriptor
a SAS/ACCESS file that describes data that is managed by a data management system. After creating an access descriptor, you can use it as the basis for creating one or more view descriptors.

active window
a window that is open, is displayed, and to which keyboard input is directed. Only one window can be active at a time.

aggregate storage location
a location on an operating system that can contain a group of distinct files. Different host operating systems call an aggregate grouping of files different names, such as a directory, a maclib, or a partitioned data set. The standard form for referencing an aggregate storage location from within SAS is fileref(name), where fileref is the entire aggregate and (name) is a specific file or member of that aggregate.

application work space (AWS)
a window that contains other windows (child windows) or from which other windows can be invoked, but which is not contained within any parent window that is part of the same software application.

ASCII
an acronym for the American Standard Code for Information Interchange. ASCII is a 7-bit character coding scheme (8 bits when a parity check bit is included) including graphic (printable) and control (nonprintable) codes.

ASCII collating sequence
an ordering of characters that follows the order of the characters in the American Standard for Information Interchange (ASCII) character coding scheme. SAS uses the same collating sequence as its host operating environment.

autoexec file
a file that contains SAS statements that are executed automatically when SAS is invoked. The autoexec file can be used to specify some SAS system options, as well as librefs and filerefs to data sources that are used frequently. See also fileref and libref.

background process
in UNIX environments, a process that executes independently of the shell. When a command is executing in a background process, you can enter other commands or

start other background processes without waiting for your initial command to finish executing.

batch file
a file that contains operating-system commands, which are processed sequentially when the file is executed.

batch mode
a method of executing SAS programs in which a file that contains SAS statements plus any necessary operating environment commands is submitted to the computer's batch queue. After you submit the program, control returns to the terminal or workstation, where you can perform other tasks. Batch mode is sometimes referred to as running in the background. The program output can be written to files or printed on an output device.

Under UNIX, place statements that you want to execute in a file. Then specify that file when you run SAS in the background.

buffer
an area of computer memory that is reserved for use in performing input/output (I/O) operations.

button
a component of a graphical user interface. A button is usually in the form of a rectangle or square that contains a label. The button is programmed to execute a command, to open a window, or to perform some other function when a user selects it. For example, many graphical user interfaces include buttons that have labels such as OK, Cancel, and Help.

catalog
See SAS catalog.

catalog entry
See SAS catalog entry.

class name
a name that provides a way to group individual X resources together. For example, DMSboldFont and DMSFont are two separate X resources, but they are both part of the Font class.

client
(1) a computer or application that requests services, data, or other resources from a server. (2) in the X Window System, an application program that interacts with the X server and can perform tasks such as terminal emulation or window management. For example, SAS is a client because it requests windows to be created, results to be displayed, and so on.

command line
the location in any SAS windowing environment window designated with Command ===>.

command prompt
the symbol after which you enter operating system commands. In UNIX environments, different shells use different command prompts. The default command prompt for the Bourne shell and the Korn shell is $, and the default prompt for the C shell is %.

configuration file
in SAS software, an external file that contains SAS system options. These system options take effect each time you invoke SAS.

container window
any SAS window that contains interior windows.

converting SAS files
the process of changing the format of a SAS file from the format that is appropriate for one version of SAS to the format that is appropriate for another version in the same operating environment.

current directory
the directory you are working in at any given time. When you log on, your current directory is the starting point for relative pathnames. See also working directory.

data set option
a SAS language element that specifies actions that apply only to a particular SAS data set. For example, data set options enable you to rename variables, to select only the first or last in observations for processing, to drop variables from processing or from the output data set, and to specify a password for a SAS data set.

descriptor information
information about the contents and attributes of a SAS data set. SAS creates and maintains descriptor information within every SAS data set.

dialog box
a type of window that opens to prompt you for additional information or to ask you to confirm a request.

directory
a named subdivision on a disk or diskette used in organizing files. A directory also contains information about the file such as size and date of last change.

download
to copy a file from the remote host to the local host.

drag
to press and hold a mouse button while moving the mouse.

engine
a component of SAS software that reads from or writes to a file. Each engine allows SAS to access files with a particular format. There are several types of engines. See also interface engine, library engine, native engine, and view engine.

environment variable
in UNIX environments, a shell variable whose value or values can be accessed by any program that is executed from that shell. The shell assigns default values to some environment variables. For example, the type of terminal and the type of command prompt are specified by the default values of two environment variables.

error message
a message in the SAS log or Message window that indicates that SAS was not able to continue processing the program.

external file
a file that is maintained by the host operating environment or by some software product. SAS can read data from and route output to external files. External files can contain raw data, SAS programming statements, procedure output, or output that was created by the PUT statement. An external file is not a SAS data set.

file descriptor
under UNIX operating systems, a nonnegative integer identifier used to refer to a file opened for reading or writing or both.

file extension
the classification of a file in a directory that identifies what type of information is stored in the file. For example, .SCAT is the file extension for SAS catalogs. See also member type.

fileref
a name that is temporarily assigned to an external file or to an aggregate storage location such as a directory or folder. The fileref identifies the file or the storage location to SAS.

Under the UNIX operating system and its derivatives, you can assign a fileref with a FILENAME statement, or you can define it as an environment variable.

font
a complete set of all the characters of the same design and style. The characters in a font can be figures or symbols as well as alphanumeric characters.

foreground process
in UNIX environments, a process that executes while you wait for the command prompt to reappear. You cannot execute additional commands while the initial command is being executed in a foreground process.

function key
a keyboard key that can be defined to have a specific action in a specific software environment. For example, Keypad 3 is defined as PASTE in the PROGRAM EDITOR window, but in the context of the FSEDIT procedure, Keypad 3 is defined as DUP.

home directory
under UNIX operating systems, the directory in which a user is placed after logging in. The home directory is also called the login directory.

icon
in windowing environments, a pictorial representation of an object. An icon usually represents a window or an object associated with an action such as printing or filing.

index
in SAS software, a component of a SAS data set that enables SAS to access observations in the SAS data set quickly and efficiently. The purpose of SAS indexes is to optimize WHERE-clause processing and facilitate BY-group processing.

interactive line mode
a method of running SAS programs in which you enter one line of a SAS program at a time at the SAS session prompt. SAS processes each line immediately after you press the ENTER or RETURN key. Procedure output and informative messages are returned directly to the display device.

interface engine
a SAS engine that reads and writes file formats supported by other vendors' software. See also engine and native engine.

interior window
a window within an application workspace that is controlled by SAS. SAS/ASSIST software is an example of an application with interior windows. See also application work space (AWS).

kernel
the memory-resident part of a UNIX operating system that manages the computer's resources. The kernel allocates memory, schedules programs for execution, monitors devices, and so on.

library engine
an engine that accesses groups of files and puts them into the correct form for processing by SAS utility windows and procedures. A library engine also determines the fundamental processing characteristics of the library, presents lists of files for the library directory, and supports view engines. See also engine and view engine.

libref
a name that is temporarily associated with a SAS data library. For example, in the name Sasuser.Accounts, the name Sasuser is the libref. You assign a libref with a LIBNAME statement or with an operating system command.

local SAS session
a SAS session running on the local host. The local session accepts SAS statements and passes those that are remote submitted to the remote host for processing. The local session manages the output and messages from both the local session and the remote session.

login directory
See home directory.

login shell
under UNIX operating systems, the program (or command interpreter) started when a user logs in.

member
a SAS file in a SAS data library.

member type
a SAS name assigned that identifies the type of information stored in a SAS file. Member types include ACCESS, DATA, CATALOG, PROGRAM, and VIEW.

menu bar
the primary list of items in a window which represent the actions or classes of actions that can be executed. Selecting an item executes an action, opens a pull-down menu, or opens a dialog box that requests additional information. See also pop-up menu and pull-down menu.

methods of running SAS
standard methods of operation used to run SAS programs. These methods are SAS/ASSIST software, SAS windowing environment, interactive line mode, noninteractive mode, and batch mode.

Multiple Engine Architecture (MEA)
a feature of SAS that enables it to access a variety of file formats through sets of instructions called engines. See also engine.

native engine
an engine that accesses forms of SAS files created and processed only by SAS. See also engine.

network
an interconnected group of computers.

noninteractive mode
a method of running SAS programs in which you prepare a file of SAS statements and submit the program to the operating system. The program runs immediately and occupies your current session.

note
in the SAS log, an informative message or explanation.

path
the route through a hierarchical file system leading to a particular file or directory

permanent SAS data library
a SAS library that is not deleted when a SAS session terminates, and which is therefore it is available to subsequent SAS sessions. Unless the User libref is defined, you use a two-level name to access a file in a permanent library. The first-level name is the libref, and the second-level name is the member name.

physical filename
the name the operating system uses to identify a file.

pipe
under UNIX operating systems and derivatives, the facility that links one command to another so that the standard output of one becomes the standard input of the other.

PMENU facility
a menuing system in SAS that is used instead of the command line as a way to execute commands. The PMENU facility consists of a menu bar, pull-down menus, and dialog boxes.

pop-up menu
a menu that appears when requested. Pop-up menus are context-specific, depending on which window is active and on the cursor location.

primary windows
in the SAS windowing environment, the PROGRAM EDITOR, LOG, and OUTPUT (LISTING), and OUTPUT MANAGER windows.

procedure output file
an external file that contains the result of the analysis or the report produced. Most procedures write output to the procedure output file by default. Reports that DATA steps produce using PUT statements and a FILE statement with the PRINT destination also go to this file.

process ID (PID)
a unique number assigned by the operating system to each process

Profile catalog
See Sasuser.Profile catalog.

protocol
a set of rules governing data communications between computers and peripheral devices.

pull-down menu
the list of menu item or choices that appears when you choose an item from a menu bar or from another menu. See also PMENU facility.

random access
the ability to retrieve records in a file without reading all records sequentially.

remote host
a computer physically removed from yours that you can log in to.

return code
a code passed to the operating system that indicates whether a command or job step has executed successfully.

SAS catalog
a SAS file that stores many different kinds of information in smaller units called entries. A single SAS catalog can contain several different types of catalog entries. See also SAS catalog entry.

SAS catalog entry
a separate storage unit within a SAS catalog. Each entry has an entry type that identifies its purpose to SAS. Some catalog entries contain system information such as key definitions. Other catalog entries contain application information such as window definitions, Help windows, formats, informats, macros, or graphics output.

SAS command
a command that invokes SAS. This command may vary depending on the operating environment and site. See also SAS invocation.

SAS data file
a SAS data set that contains data values as well as descriptor information that is associated with the data.

SAS data library
a collection of one or more SAS files that are recognized by SAS and which are referenced and stored as a unit. Each file is a member of the library.

SAS data set
a file whose contents are in one of the native SAS file formats. There are two types of SAS data sets: SAS data files and SAS data views. SAS data files contain data values in addition to descriptor information that is associated with the data. SAS data views contain only the descriptor information plus other information that is required for retrieving data values from other SAS data values from other SAS data sets or from files whose contents are in other software vendors' file formats. See also descriptor information.

SAS data set option
an option that appears in parentheses after a SAS data set name. Data set options specify actions that apply only to the processing of that SAS data set.

SAS data view
a type of SAS data set that retrieves data values from other files. A SAS data view contains only descriptor information such as the data types and lengths of the variables (columns), plus other information that is required for retrieving data values from other SAS data sets or from files that are stored in other software vendors' file formats.

SAS file
a specially structured file that is created, organized, and, optionally, maintained by SAS. A SAS file can be a SAS data set, a catalog, a stored program, or an access descriptor.

SAS initialization
the setting of global characteristics that must be in place at start-up for a SAS programming environment. SAS performs initialization by setting certain SAS system options called initialization options. Invoking SAS software initiates SAS initialization.

SAS invocation
the process of calling or starting up SAS software by an individual user through execution of the SAS command. Invoking SAS initiates SAS initialization. See also SAS initialization.

SAS log
a file that contains a record of the SAS statements that you enter as well as messages about the execution of your program.

SAS session
See session.

SAS system option
an option that affects the processing of an entire SAS program or interactive SAS session from the time the option is specified until it is changed. Examples of items controlled by SAS system options include appearance of SAS output, handling of some files that are used by SAS, use of system variables, the processing observations in SAS data sets, features of SAS initialization, and the way SAS interacts with your host operating environment.

SAS windowing environment
an interactive windowing interface to SAS software. In this environment you can issue commands by typing them on the command line, by pressing function keys, or by selecting items from menus or menu bars. Within one session, you can perform many different tasks, including preparing and submitting programs, viewing and printing results, and debugging and resubmitting programs.

Sashelp library
a SAS data library supplied by SAS software that stores text for HELP windows, default function key and window definitions, and menus.

Sasuser library
a default, permanent SAS data library that is created at the beginning of your first SAS session. The Sasuser library contains a Profile catalog that stores the tailoring features you specify for SAS. You can also store other SAS files in this library.

Sasuser.Profile catalog
a SAS catalog in which SAS stores information about attributes of your SAS windowing environment. For example, this catalog contains function-key definitions, fonts for graphics applications, window attributes, and other information that is used by interactive SAS procedures. See also SAS catalog.

sequential access
a method of file access in which the records are read or written one after the other from the beginning of the file to the end.

server
(1) in a network, a computer that is reserved for servicing other computers in the network. Servers can provide several different types of services, such as file services and communication services. Servers can also enable users to access shared resources such as disks, data, and modems. (2) in the X Window System, software that transfers information between various clients on a network. Through their clients, users can submit input to and request output from a server. See also client.

session
a single period during which a software application is in use, from the time the application is invoked until its execution is terminated.

session gravity
in the X Window interface to SAS, the resource that controls the region of the workstation display in which SAS attempts to place its windows.

shell
a UNIX command interpreter. Sample shells are sh, csh, and ksh.

shell script
a file containing commands that can be read and executed by the shell. A shell script is also called a shell procedure or a shell program.

special file
under UNIX operating systems, an interface to an input or output device. Writing to or reading from the file activates the device.

standard error
under UNIX operating systems, the destination of the program's error messages.

standard input
the primary source of data going into a command. Standard input comes from the keyboard unless it is being redirected from a file or piped from another command.

standard output
the primary destination of data coming from a command. Standard output goes to the display unless it is being redirected to a file or piped to another command.

swapping
the action of moving segments from memory to disk and vice versa.

system option
See SAS system option.

temporary SAS data library
a library that exists only for the current SAS session or job. The most common temporary library is the Work library. See also Work data library.

toggle
an option, parameter, or other mechanism that enables you to turn on or turn off a processing feature.

Toolbox
a feature of SAS that enables you to associate an icon with any SAS command or macro. Selecting the icon executes its associated command or string of commands.

toolset
a set of predefined tools that is associated with an application. Toolsets make it easier for individual users to customize their application toolboxes.

Universal Printing
a feature of SAS software that enables you to send SAS output to PDF, Postscript, and PCL files, as well as directly to printers. The Universal Printing system also provides many options that enable you to customize your output, and it is available in all of the operating environments that SAS supports.

upload
to copy a file from the local host to the remote host.

User data library
a SAS data library defined with the libref User. When the libref User is defined, SAS uses it as the default libref for one-level names.

view engine
an engine that enables SAS to process SAS data views. A view engine performs in a transparent manner. See also SAS data view.

warning
in the SAS log, a message that informs you of a potential problem.

Work data library
the SAS data library that is automatically defined by SAS at the beginning of each SAS session or SAS job. The Work library contains SAS files that are temporary by default. When the libref User is not defined, SAS uses Work as the default library for SAS files created with one-level names.

working directory
the directory in which a software application is stored. When the application is started, the working directory becomes the current directory unless you specify a

different current directory. Therefore, the working directory for a SAS session is usually the directory from which you invoked SAS.

X resource
a characteristic of a window interface, such as font type, font size, color, gravity, and window size.

X resource file
in the X Window System, a file that stores attribute specifications for the windowing environment, such as color, gravity, font types and sizes, and window sizes.

X window system
a graphical windowing system that was developed at the Massachusetts Institute of Technology.

Index

Numbers

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z

Your Turn

If you have comments or suggestions about *SAS 9.1 Companion for UNIX Environments*, please send them to us on a photocopy of this page or send us electronic mail.

Send comments about this book to

SAS Institute Inc.
Publications Division
SAS Campus Drive
Cary, NC 27513
E-mail: **yourturn@unx.sas.com**

Send suggestions about the software to

SAS Institute Inc.
Technical Support Division
SAS Campus Drive
Cary, NC 27513
E-mail: **suggest@unx.sas.com**