MULTICS

GCOS ENVIRONMENT SIMULATOR ADDENDUM A

SUBJECT

Changes to the Manual

SPECIAL INSTRUCTIONS

This is the first addendum to AN05-02 dated December 1983. Refer to the Preface for "Significant Changes."

Insert the attached pages into the manual according to the collating instructions on the back of this cover. Throughout the manual, change bars in the margins indicate technical additions and asterisks denote deletions.

Note:

Insert this cover after the manual cover to indicate the updating of the document with Addendum A.

SOFTWARE SUPPORTED

Multics Software Release 11.0

ORDER NUMBER

AN05-02A

December 1985

44744
5C186
Printed in U.S.A.

Honeywell

COLLATING INSTRUCTIONS

To update the manual, remove old pages and insert new pages as follows:

Remove	Insert
TP, Preface	TP, Preface
iii through viii	iii through viii
4-1 through 4-4	4-1 through4-4
7-3, blank	7-3, blank
A-5, blank	A-5, A-6 A-7, blank
B-1 through B-16	B-1 through B-12 B-13, blank
i-1 through i-4	i-1 through i-4
TP Remarks Form (12-83)	TP Remarks Form (12-85)

File No.: 1L13

AN05-02A

MULTICS

GCOS ENVIRONMENT SIMULATOR

SUBJECT

Organization and Operation of the GCOS Environment Simulator

SPECIAL INSTRUCTIONS

This manual supersedes AN05, Revision 1, dated October 1978, and its addenda, AN05-01A, dated November 1980, and AN05-01B, dated July 1981. Refer to the Preface for Significant Changes.

Throughout the manual, change bars in the margin indicate technical additions and asterisks denote deletions.

SOFTWARE SUPPORTED

Multics Software Release 10.2
Series 60 Level 66 Software Release 4S3 and DPS 1.3
Series 6000 Software Release JS3

ORDER NUMBER

AN05-02

December 1983

Honeywell

PREFACE

The purpose of this manual is to provide the user with an introduction to, and instructions for using, the GCOS Environment Simulator. The simulator, together with other Multics facilities, provides the ability for some GCOS jobs to run under the control of Multics.

This manual contains descriptions of the following Multics Priced Separate Products (PSPs); some of them may not be installed in your system.

SGS6801
GCOS (III) TimeSharing Environment Facility

SGS6804
GCOS (III) Batch Environment Facility

The following are the primary reference manuals for user and system programmers of the Multics system. These manuals contain general information and may be referenced throughout this document. For convenience, these references are as follows:

Document	Referenced In Text As
Multics Programmer's Reference Manual (Order No. AG91)	Programmer's Reference
Multics Command and Active Functions (Order No. AG92)	Commands
Multics Subroutines and I/O Modules (Order No. AG93)	Subroutines

There are also frequent references within the text to the titles of the GCOS manuals listed below by title and order number.

General Comprehensive Operating Supervisor (GCOS), Order No. DD19

Control Cards Reference Manual, Order No. DD31

File and Record Control, Order No. DD07

Time Sharing System Reference Manual, Order No. DJ31

 File No.: 1L13 AN05-02A

Often, user typed lines and lines displayed by Multics are shown together in the same examples. To differentiate between these lines, an exclamation mark (!) precedes user-typed text. This is done only to distinguish user text from system-generated text, it is not to be included as part of the input line. Also, a "carriage return" is implied at the end of every user-typed line.

Note: Because of page constraints in this document, certain character strings of data used in examples may not match exactly the information as seen on a user's terminal. That is, the character strings in examples may be folded (contained on several lines), whereas the actual interactive (live) session may display the same information on a single line or multiple lines with different line breaks than shown.

Significant Changes in AN05-02A

- Deleted Section 3 (material is duplicated elsewhere in this manual, or is contained in other GCOS manuals).
- Moved complete portions of Appendix B to Appendix A and renamed Appendix B. The moved information (i.e., Appendix B to Appendix A) was under title of:

  Maintenance of the GCOS User Table (and all sub-headings)
  Installing the GCOS Daemon (and all sub-headings)
  Creating the GCOS SysDaemon User Directory Structure

  None of the moved information was changed, therefore no change bars appear on the moved data, or the Contents.
- Deleted "Utility Commands Example" in Appendix B.
- Added "Steps to Build GCOS Software and Library Files" in Appendix B.

CONTENTS

ILLUSTRATIONS

SECTION 1

GCOS SIMULATION WITHIN MULTICS

The GCOS batch and time sharing simulators provide for executing GCOS slave jobs and time sharing subsystems on Multics. Other facilities in this group include the GCOS daemon and several commands that can be used for manipulating GCOS files.

In this document, the term "GTSS" refers to the GCOS time sharing simulator on Multics, while the term "GCOS time sharing" refers to the native GCOS facility.

These facilities are briefly described in this introductory section and more fully described elsewhere in this manual. For additional information on Multics facilities and terminology, refer to the Multics reference manuals listed in the preface of this document.

MULTICS PROCESSES

In learning to use either of the simulators, it should be kept in mind that Multics is a time sharing system and is primarily oriented toward interactive users. Each logged-in interactive user has what is known as a process, which comprises the set of procedures (programs) and data that are being used and the path of execution through these programs. A process is similar to a GCOS job, or time sharing session. Commands entered by the user are analogous to the activities in the GCOS job. For example, one command line is used to compile a program and another is used to execute the program. However, a process differs from a job in that the user can intervene after, or even during, each command to determine what is done next.

Multics has a batch processing facility that allows a user to queue an absentee job. An absentee job is a file containing a set of Multics commands that are to execute at some later time and which is run without terminal interactions. If no time is specified, the job is run as soon as those preceding it in the queue have finished. When the job is run, an absentee process is logged in for the user and the commands in the absentee input (absin) file are executed as if they had been entered on the console by an interactive user.

For absentee processing, output intended for the console is automatically directed to an absentee output (absout) file, which can later be printed and examined by the user. It is assumed that there are no user interactions, or that the user has anticipated them and has placed appropriate responses in the absin file.

In addition to interactive and absentee processes, there is another type of process known as the daemon process. Daemon processes are logged in by the Multics operator and usually remain logged in during the entire time Multics is operating. These processes perform functions that cannot be performed in a user's process for reasons of security or efficiency.

For example, daemon processes are used to run the high-speed printers, the card reader, and the card punch. Normal user processes usually are not permitted to attach these devices. User processes queue requests for the input or output of files. Daemon processes then carry out these requests in the order in which they appear in the queues. This method of processing is analogous to the processing of SYSOUT by GCOS.

Each process in the system has associated with it a personid and a projectid. These normally are written in the form person.project.

- The personid is the registered name of the user (i.e., the name typed when logging in).
- The projectid is the name of the project group in which the user works. It is used both for accounting and for access control purposes. It is possible for a user to be registered on several projects. Some examples include:

IO.SysDaemon	runs printers, etc.
GCOS.SysDaemon	runs GCOS batch jobs
Smith.Applications	ordinary user
Jones.Multics	{Multics system programmer
Jones.Applications	{Jones works on two projects}

While GCOS allows both accounting and file access information to be changed within a job (via multiple $ IDENT and $ USERID control cards), Multics does not permit such changes within a process.

GCOS BATCH ENVIRONMENT SIMULATOR

The GCOS batch environment simulator is invoked via the Multics gcos command. The simulator immediately runs one GCOS job in the user's process. While the job is running, the user's process is completely dedicated to it and is not able to execute other Multics commands.

The user's console is treated as the GCOS operator's console. Input and output intended for the GCOS operator's console are directed to the user's console; they do not appear on the Multics operator's console.

When the job is finished, the user's process is free to execute other Multics commands or to run another GCOS job. It should be noted that the simulator runs only one job per call. No facilities are built into the simulator for batch processing, multiprogramming, communication between jobs, job scheduling, or resolving conflicting use of files or peripherals.

The simulated interfaces are those seen by the job stream and by the executing slave area program. The simulator reads card images from an input file, interprets the control cards, allocates the necessary files, and performs other initialization functions necessary to run each activity.

During an activity, the simulator intercepts all MMEs (master mode entries) and performs the processing function requested by them. No provision is made for jobs that require special privileges, such as use of the MME ".EMM".

The simulator runs as an unprivileged user program under Multics and makes use of Multics facilities for all privileged operations such as input/output. Other GCOS features not simulated include:

- Transaction processing
- Remote I/O (except to the user's terminal)
- Checkpoint, restart
- File transfer between the simulator and native GCOS on any media other than punched cards or magnetic tape
- Functions of the GCOS file system that have no counterparts in the Multics file system

GCOS TIME SHARING ENVIRONMENT SIMULATOR

The GTSS is invoked via the Multics gcos_tss command. The user may invoke the Multics gcos_set_environment command before executing gcos_tss to modify parameters regulating successive calls to gcos_tss.

GCOS time sharing simulation involves an interactive user interface that looks like GCOS time sharing and a simulated environment that allows a user to execute GCOS time sharing subsystems and user programs. The following functions are provided:

1. Recognition of GCOS command language for each subsystem using tables extracted from the GCOS time sharing executive (module TSSA). Each command recognized has a corresponding list of primitives to be interpreted.

2. Interpretation of GCOS time sharing primitives. There are a total of 12 primitives to be interpreted. They provide for stacking the current subsystem and calling a new one, initiating the loading and execution of the current subsystem program, initiating the building of input, returning to the subsystem at the previous level, manipulating bits in the subsystem switch word, and conditionally executing blocks of primitives under control of specified bits in the subsystem switch word.

3. A command loader function to allow a user program stored on an H* file to be loaded and executed. The command loader is invoked whenever an unrecognized command is given.

4. A basic line editor which takes input from the terminal and merges it in line-numbered sequence with the current file.

5. A static derail (DRL) handler similar to the MME handler in the GCOS batch simulator. This handler uses a transfer vector to cause the appropriate routine to be executed for each DRL.

GCOS DAEMON

The GCOS daemon provides batch processing and some job scheduling facilities. Jobs can be submitted to Multics operations in either card deck or IMCV (input media conversion) tape form. The daemon examines the job and interprets some of the control cards for security, scheduling, and accounting purposes. It then submits an absentee job, which consists of commands, to call the simulator. The simulator runs the job and queues requests for the I/O daemon to print and punch the output.

A separate absentee job is submitted for each GCOS job processed by the daemon. (See Appendix A for information on the use of the GCOS daemon facility.)

FILE SYSTEM

The following discussion applies to both the batch and the time sharing simulators; although the interfaces are slightly different, the functions are the same. References that are simulator-specific are given as such.

The GCOS file system is not simulated. Instead, references to permanent files from $ PRMFL or $ SELECT control cards, or from a MME GEFSYE are mapped by the simulator into references to files in the Multics file system. Many of the GCOS file system functions can be mapped into Multics. However, there are many comprehensive facilities within GCOS, for operating system-managed file integrity and concurrent access control, that are not accommodated.

The Multics file system has several similarities to the GCOS file system. Multics files are identified by pathnames, which are analogous to the GCOS file string. They comprise a series of directory names, which are analogous to GCOS catalog names, followed by an entry name, which is analogous to a GCOS file name. Passwords are not included in a Multics pathname.

Multics literature uses the term "segment" when referring to items contained in the Multics file system. The term "file" is used in the special case of a segment that is being accessed by explicitly programmed I/O rather than via the normal Multics method of direct-segment addressing.

References to permanent files can be made from the simulator via a Multics pathname or via a GCOS file string. The Multics pathname can be used in place of the GCOS file string on the $ PRMFL card.

File System Naming Conventions

- The GCOS simulator character set for names may be composed of lowercase alphanumerics, period, and minus signs. Names generated from within GCOS object programs are always mapped to lowercase.
- A name consisting of zeros is specifically prohibited. Blanks are not permitted. If multiple-word names are desired then the words must be separated by periods or minus signs, not blanks.
- A maximum of eight characters is normally used for file names. Catalog names may be up to 12 characters and composed of the same characters as file names.

- To access a file with a name longer than eight characters in GTSS, an alternate name must be given to the file using a name from one to eight characters in length.

- The greater-than (>) character is specifically prohibited in entrynames, since it is used to form pathnames. Other characters not recommended for entrynames are:

  Less-than (<), asterisk (*), question mark (?), percent (%), equal sign (=), dollar sign ($), quotation mark ("), left slant (\), all ASCII control characters (tab, carriage return, etc.), and parentheses.

- Non-ASCII characters are not permitted in entrynames.

- The period must not be the first or last character of a name. Although this is allowed in native GCOS, it conflicts with the naming conventions of native Multics.

CATALOG AND FILE PERMISSIONS

Each file has associated with it a Multics access control list (ACL), which is set by the owner of the file. The ACL can specify combinations of read, write, and execute permissions to individual users or to all users in a specific group.

Access to permanent files from the simulator is determined only by Multics access control and is based on:

1. The person.project of the process in which the simulator is running.
2. The access granted to that person.project by the ACLs of the files being referenced.

When the simulator is running in an interactive user process, access privileges to any permanent files are the accesses granted to that person.project via the ACLs of the referenced files. The presence or absence of a $ USERID control card has no effect on this access. (The $ USERID card is ignored by the simulator.)

When the simulator is running jobs submitted by the GCOS daemon, there is a security problem. Normally, access to a Multics process (and its file access privileges) is validated by a password typed by the user at login time. However, a GCOS password is contained on a $ USERID card and is, therefore, much more susceptible to theft. Thus, in addition to the normal Multics file access controls, some additional restrictions are placed on jobs submitted by the daemon to protect the security of the Multics file system (see "Restrictions on Daemon Jobs" described later in this section).

FILE SYSTEM ACCESS MODES

Multics Access Modes

The access modes for segments are:

execute, e
: an executing procedure can transfer to this segment, and words of this segment can then be interpreted as instructions and executed by a processor.

null, n
: no access.

read, r
: this process can execute instructions that cause data to be fetched (loaded) from the segment.

write, w
: this process can execute instructions that cause data in the segment to be modified.

The access modes for directories are:

append, a
: new segments, directories, and links can be created in the directory.

modify, m
: the attributes of existing segments, directories, and links contained in the directory, and certain attributes of the directory itself can be modified. In addition, existing segments, directories, and links contained in the directory can be deleted.

null, n
: no access.

status, s
: the attributes of segments, directories, and links contained in the directory, and certain attributes of the directory itself can be obtained.

GCOS Permissions

The permissions for both files and catalogs are:

append, a
: allow transfer of information from file to program, but not from program to file (equivalent to read access).

create, c
: allow catalogs and files to be entered as subordinate to this catalog.

exclude, x
: the specified user has no access to the catalog or file.

execute, e
: allow object code in the file to be executed. Execute permission is applicable only to time sharing.

lock, l
: allow MME or directive to security lock the file or catalog, or to

remove an existing security lock (security lock does not apply to a user with lock permission).

modify, m
: allow catalog or file descriptor to be modified. Allow entries to be made in catalog for subordinate files or catalogs. Users permitted to modify are allowed to perform any actions -- hence, modify includes create, lock, and purge, which in turn includes recovery and thus write and read.

purge, p
: allow file to be deleted, or catalog and all subordinate files to be deleted. Users with purge permission can also perform any of the actions permitted by recovery, including write and read.

read, r
: allow transfer of information from file to program, but not from program to file.

recovery, rec
: allow write when the file is abort locked, or when the file has defective space. Also, accept MME or directive to abort lock the file, or to reset an existing abort lock. Users with recovery permission are also given write and read permission.

write, w
: allow transfer of information from file to program and program to file (users with write permission thus have read permission).

The following chart defines the actual Multics permissions used to simulate the desired GCOS permissions. To determine which Multics permissions are set, find the desired GCOS permissions (at the top of the chart) and read the corresponding (vertical) Multics permissions for that column.

(create catalog)

Multics	GCOS r	w	a	e	p	m	l	c	x
3 r	X	X	X	X	X	X	X	X	
3 w		X			X	X		X	
3 e	X	X	X	X	X	X		X	
n									X
4 s						X			
4 m					1	2		1	
4 a						X		X	

(create files)

Multics	GCOS r	w	a	e	p	m	l	x
5 r	X	X	X	X	X	X	X	
5 w		X			X	X		
5 e	X	X		X	X	X		
n								X
6 s								
6 m					1	2		
6 a								

Notes:

1 Needed to allow deletion by other users. (This access also allows these users to delete other segments.)

2 Modify, which is necessary to allow changing of file permissions, also allows deletion of all segments under the directory.

3 Permissions set on initial ACL s on segments.

4 ACLs set on the created directory.

5 ACLs set on the segment created.

6 ACLs which must be set on the superior directory.

In order to access a gtss file, the user must have:

a. sma on the containing directory
b. s on the directory superior to the containing directory

MAPPING OF FILE STRINGS TO PATHNAMES

Because the structure of the Multics file system is different from that of the GCOS file system, the appropriate method of mapping a GCOS file string into a Multics pathname is not an obvious one. The default method used allows many GCOS jobs to run immediately and requires that some initialization be performed in the Multics file system before other jobs can run.

Both the GCOS and Multics file systems are organized in tree-structured hierarchies. However, while the GCOS file system holds only user files, the Multics file system holds the entire Multics system; user files are held in a subdirectory of the total hierarchy.

The user file subdirectory contains project directories and each of these contains individual user directories. The user file subdirectory is analogous to the system master catalog (SMC) of the GCOS file system. However, the project directories that it contains are not analogous to the user master catalogs (UMC) in the GCOS file system, since Multics users are not normally permitted to create or modify files in the project directories.

User directories are more nearly analogous to the GCOS UMCs, but they differ in that they have two-component names, while UMCs have only one-component names. This makes it impossible to map UMC names directly into user directory names without obtaining additional information from some source or making some assumptions.

The following paragraphs describe in detail how each mapping method works.

A user directory, which is contained in a project directory, is known as a home directory in Multics terminology. The form of a home directory pathname is:

```
>udd>project>person
```

The greater-than sign (>) is used to separate components of a Multics pathname (instead of the slash (/) that is used in GCOS file strings). The leading > indicates the pathname is relative to the root of the hierarchy rather than relative to the working directory. The directory udd (user_directory_directory) contains all project directories. Every user's home directory is contained in some project directory. For example, the home directory pathname of the user Smith.Applications is:

```
>udd>Applications>Smith
```

User files can be placed in the home directory. Users also can create subdirectories in the home directory and can place files in them to organize and/or restrict access to groups of files. Multics does not associate passwords with individual directories or files; access is controlled only by the ACL of the directory or file in question.

Each process has a working directory. Initially, this directory is the home directory of the user. However, it can be changed by the user.

With Multics conventions, files can be referenced by an absolute pathname or by a relative pathname. An absolute pathname begins with a greater-than sign (>) and contains the names of all the directories superior to the file in the hierarchy. For example:

```
>udd>Applications>Smith>data_file
```

A relative pathname does not begin with a greater-than (>) sign and the complete pathname is assumed to begin with the pathname of the working directory. The simplest example of a relative pathname is an entry name (analogous to a GCOS file name):

```
data_file
```

This identifies the same file as the previous example, provided the working directory is:

```
>udd>Applications>Smith
```

Similarly, a GCOS catalog/file string that uses a leading UMC name can be considered to be an absolute pathname, and a file or catalog string that does not have a leading UMC name is considered to be a relative pathname.

The method of mapping file strings to pathnames differs between the GCOS batch simulator and the GTSS. These differences are described in the following paragraphs.

Mapping Under GCOS Batch Simulator

Multics absolute or relative pathnames can be used on $ PRMFL and $ SELECT cards. They are interpreted as previously described. If a GCOS file string is used on one of these cards or in a MME GEFSYE, it is mapped into a Multics pathname:

1. All passwords, along with the dollar signs that precede them, are removed from the string and ignored.
2. All slashes (/) are changed to greater-than signs (>).
3. The first catalog name in the string (that of the UMC) is replaced by the default working directory pathname (not the current working directory pathname). The default working directory is normally the same as the home directory, but it can be altered by the user with the change_default_wdir command. If no catalog string precedes the final string, the final string is appended to the pathname for the user's current working directory.

Therefore, the file string:

```
SMITH/JONES$CAT/Y$DOE
```

is transformed to the pathname:

```
>udd>Applications>Smith>jones>y
```

for the user Smith.Applications. Note that the retained portion of the file string is indicated in lowercase letters, while the original file string is indicated in all capital letters. This illustrates a common situation in which a file string is encountered on a BCD (binary coded decimal) card that was used as input via the GCOS daemon. (Alphabetic BCD characters are translated into lowercase ASCII characters for internal processing by the simulator.) However, if the input is a Multics ASCII file, the original characters (uppercase or lowercase) in pathnames and file strings are preserved. A complete description of the use of the ASCII and BCD character sets is included in Section 2, under "Data Formats."

Problems that arise while mapping file strings into pathnames (while accessing the files of another user) can be solved in two ways:

1. Repunch the cards using Multics pathnames.

2. Place links in the home directory, which points to the files of interest in the user's directory. (See the Programmer's Reference manual for link information.)

Mapping Under GTSS

GTSS does not allow the user to directly specify Multics pathnames in GCOS time sharing commands. This would be impractical because such commands are processed directly by GCOS time sharing subsystems.

The GTSS user is provided with a choice of three rules for mapping GCOS file strings to Multics pathnames. The rules are UMC (User Master Catalog), SMC (System Master Catalog), and WD (working directory). The default is WD, which may be changed through the gcos_set_environment command.

The first two steps of the mapping process are the same for GTSS as for the GCOS batch simulator:

1. All passwords, together with the dollar signs that precede them, are removed from the string and ignored.
2. All slashes (/) are changed to greater-than signs (>).

Further mapping action depends on the selection of one of the mapping rules described in the following paragraphs.

MAPPING RULE: WD

The first catalog name in the string (that of the UMC) is replaced by the working directory pathname.

If the user's working directory is:

```
>udd>Applications>Smith>temp_dir
```

then the file string:

```
SMITH/JONES$CAT/Y$DOE
```

is transformed to the pathname:

```
>udd>Applications>Smith>temp_dir>jones>y
```

MAPPING RULE: UMC

This rule for pathname mapping converts the leading UMC name in the GCOS catalog/file string into the string >udd>umc_name>umc_name. The purpose of this mode is to allow direct mapping of pathnames in either direction with no explicit action on the part of the individual user. This mode is used by the gcos_fms command for the loading of GCOS user SAVE tapes onto Multics.

The use of this rule requires that the Multics system administrator add the lowercase version of the UMC name to the project directory under >udd. The project administrator must create a directory by the same name directly below the project directory. This second directory is the equivalent to the catalog

for the given UMC on native GCOS. For example, a project on GCOS has the UMC name of DEBUG. This project is also registered on Multics, but with the name GDEBUG. The following steps must be taken to use the UMC mapping rule:

1. add_name >udd>GDEBUG debug
2. create_dir >udd>debug>debug
3. set_acl >udd>debug>debug sma *.GDEBUG
4. set_iacl_seg >udd>debug>debug rw *.GDEBUG

The GCOS catalog/file string of SMITH/JONESCAT/YDOE would be mapped to:

>udd>smith>smith>jones>y.

MAPPING RULE: SMC

This rule for pathname mapping sets a directory pathname specified by the user to be the root directory for all subsequent mappings. All UMCs are looked for or created directly under the SMC directory. Thus, the user has complete control over the mapping of GCOS catalog/file strings to their targets on Multics. Typically, the user creates links with the names of UMCs that point to the corresponding directories containing the desired subcatalogs and files. For example, the user has specified:

-directory_mapping smc -smc_pathname >udd>GDEBUG

in a command invocation of gcos_set_environment. With this mode, the GCOS catalog/file string of SMITH/JONESCAT/YDOE is mapped to >udd>GDEBUG>smith>jones>y.

GCOS File Attributes

An interim mechanism is used for the processing of GCOS file attributes. File attributes include the sequential/random creation mode, maximum file size, and current file size.

The data attribute required by GTSS is converted to ASCII representation and saved as added names on the branch. The general form of the attribute name is:

<entryname>.<attributename>.<attributevalue>

Only GTSS requires these attributes. For files and catalogs created under GTSS, no user action is necessary to set or maintain this information. If files are created outside of GTSS, but are to be accessed under GTSS, the user must manually set this information. An exec_com is provided to allow GCOS-oriented users to manipulate these added names. To use the exec_com, type "ec >unb>gcos_file_attributes.ec path", where path is the pathname of the file to be accessed under GTSS.

RESTRICTIONS ON DAEMON JOBS

To prevent the security of the Multics file system from being compromised by a GCOS job submitted via the daemon, three mechanisms are used.

1. The daemon submits all jobs under the process identification "Anonymous.GCOS.g". The Anonymous person identification is the standard Multics name used to identify an unknown user (whose right to use the system has not been verified) as well as to identify a known user. User files whose ACLs contain the entry "null Anonymous.*.*" are completely protected from daemon jobs.

2. The Multics ring protection mechanism is used to force daemon jobs to run in a less privileged ring than interactive user processes normally run in. The ring protection mechanism is described in the Programmer's Reference manual.

   This mechanism is an extension of the two-level (supervisor/user) protection feature found in most computer systems. Ring protection provides eight levels of privilege (ring 0-7) with ring 0 having the highest privilege and ring 7 the lowest. A process has a current execution level equal to one of the ring numbers and is said to be executing in that ring. The hardcore supervisor executes in ring 0. Normally, user processes execute commands in ring 4 (the "user ring").

   Each file has a set of ring brackets, which are specified by three ring numbers. The ring brackets operate in addition to the ACL to restrict access to a file. A file whose ring brackets are [1 5 5] can be executed in rings 1-5, written in rings 0 and 1, and read in rings 0-5. (This is determined from the first two numbers (1 and 5). The third number specifies the gate bracket, which controls changes in the ring of execution.) A file whose ring brackets are [4 4 4] can be written and read in rings 0-4 and executed only in ring 4.

   System commands and subroutines (including the GCOS simulators) have ring brackets of [1 5 5]. User-created files have brackets of [4 4 4] by default, since they are created by a process executing in ring 4. User file ring brackets can be changed using the set_ring_brackets command.

   The daemon causes the absentee processes that it creates to run jobs for execution in ring 5. These processes thus will be denied access to all files whose ring brackets are [4 4 4], even if the ACLs do not deny access to the process. Files created by these processes have brackets of [5 5 5] and are accessible to interactive users running in ring 4.

   Interactive users who wish to grant daemon jobs access to their files can do so by setting brackets of [5 5 5] on the files and placing Anonymous.GCOS.g on the ACLs.

3. To create and delete files, a daemon job needs both modify and append permissions on the directory containing the files. To avoid the necessity of interactive users giving Anonymous.GCOS.g those permissions on the directories they use interactively, a separate hierarchy of GCOS files is maintained. This hierarchy is analogous to >udd (its name is >gdd), but it contains project and user directories only for GCOS users.

   The absentee jobs submitted by the daemon begin with a command that changes the home directory to >gdd>project>person, where person and project are determined by a table lookup of the account number on the $ IDENT control card. This causes GCOS file strings on $ PRMFL and $ SELECT control cards to be transformed into pathnames in the >gdd hierarchy. (However, Multics pathnames on those cards are not changed.)

For example, in the GCOS card deck, which includes:

1. A $ IDENT card with a user accounting code that transforms to Smith.Test

2. A $ PRMFL card with the string:

JONES$PASSWORD/CATALOG1/CATALOG2/FILENAME1

the file is interpreted as >gdd>Test>Smith>catalog1>catalog2>filename1.

The interactive Multics user who also submits batch GCOS jobs via the daemon should be allowed status/modify/append access to the directory >gdd>project>person to permit the sharing of data bases between these two environments. Since the user's execution level is 4, no difficulties are encountered in attempting read or write accesses to files (bracketed [5 5 5]) under this subtree.

Similarly, when it is necessary for a batch GCOS job to access a file that is created interactively, the ACL must be set appropriately for Anonymous.GCOS.g and its brackets set to [5 5 5] (or higher).

JOB SCHEDULING

The $ MSG3 control card is used by the daemon to defer the start of an absentee job until the time requested on the card. The daemon uses the $ SNUMB card urgency field to select the priority queue in which the absentee job should be placed. The daemon rejects a job if another job with the same snumb previously was submitted and has not yet cleared the system.

With the exception of these two cards, there are no facilities in either the daemon or the simulator for scheduling, holding, or resolving conflicting resource usage among jobs. Jobs attempting to modify the same permanent files must be manually scheduled to avoid conflicts. Jobs that use many tape drives must be scheduled so as to avoid lockups between them (i.e., each job waiting for tape drives that were allocated to the other).

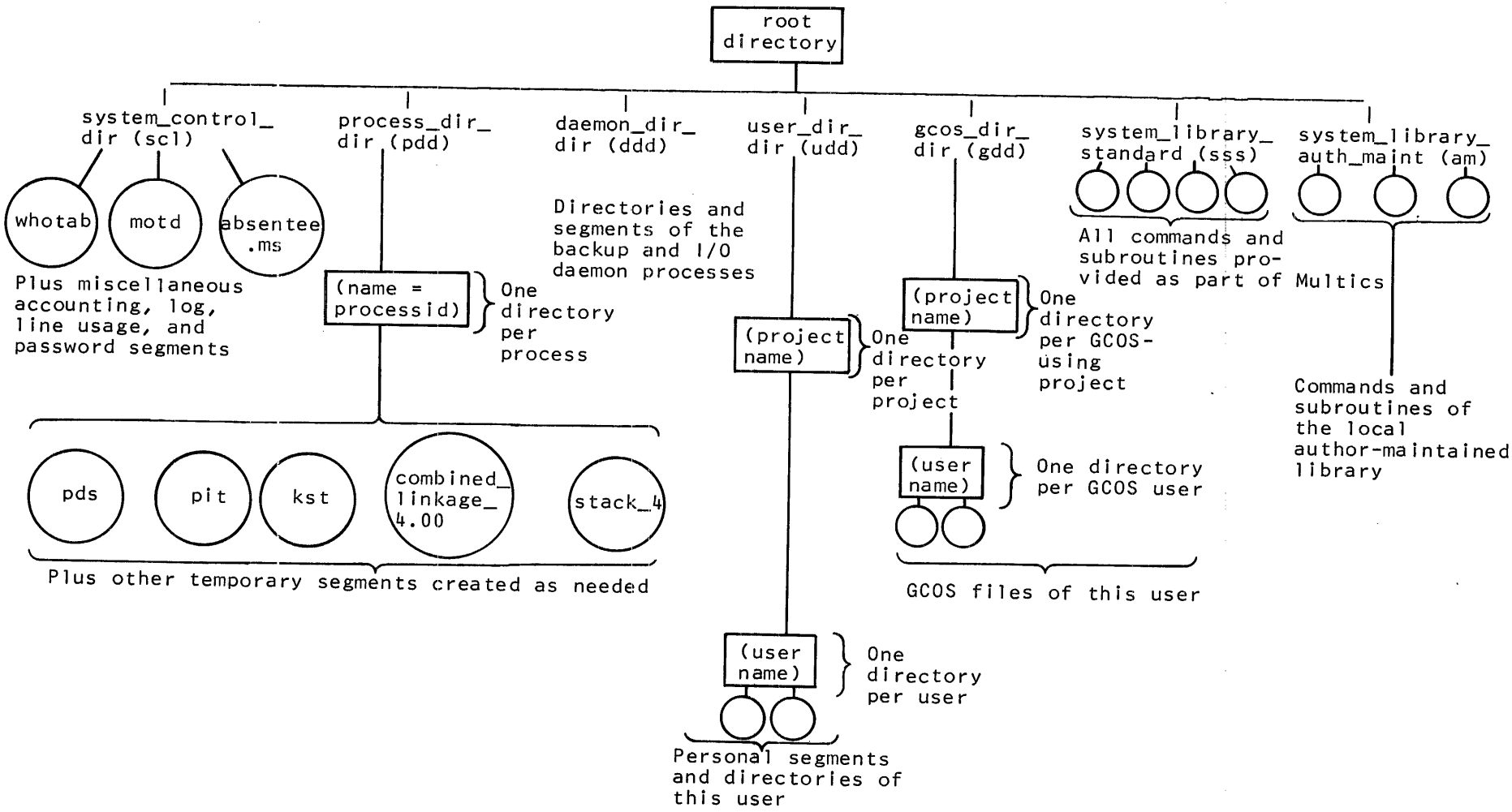


Figure 1-1. Multics Storage System Hierarchy

SECTION 2

GCOS BATCH ENVIRONMENT SIMULATOR DESIGN AND OPERATION

ORGANIZATION

The GCOS batch environment simulator (invoked by the gcos command) comprises three major parts: the input reader, the allocator, and the MME simulator.

The input reader reads the input file (which can be either ASCII or BCD), merges in any $ SELECT files (each of which can be either ASCII or BCD), translates the card images to the format expected by the allocator, and writes the images into the job stream file. The input reader also checks for execution activities (for later setting of program switch word bit 5) and performs a limited amount of checking for control card errors. Most of the control card error checking, however, is performed by the allocator.

The allocator reads the control cards and data cards for each activity from the job stream file that is created by the input reader; allocates the files to be used by the activity; interprets some control cards; and writes other control cards and data cards onto files for later interpretation by a slave program. The allocator then calls the MME simulator.

The MME simulator replaces the signal_ pointer in the header of the user's stack, so that the simulator's MME handler is the first user ring program to receive control following a fault (rather than signal_, which is the normal user ring fault handler). The MME simulator also establishes a handler for faults, such as overflow and underflow, that is called by signal_ if control ever reaches it. A MME GECALL, which contains the name of the slave program to be executed, is constructed and placed in the slave area segment. Control then is transferred to the MME GECALL.

The software library is searched, the program is loaded, and execution begins.

Whenever a MME or any other fault occurs, the Multics hardcore supervisor performs initial processing. If the MME or fault is not one that must be handled by the supervisor, it is passed on to the fault handler in the user ring. Since the MME simulator substitutes its own handler for signal_, that handler receives control.

If the fault is a GCOS MME, the simulator calls a subroutine to perform the service requested by the MME. Control then is returned to the slave program.

If the fault is not a MME, the simulator calls signal_. Eventually, if the fault is one that can be handled by the slave program, the simulator's fault handler examines the fault vector in the slave program and either transfers to a handler in the slave program or takes appropriate action itself.

When a MME GEFINI occurs, the MME simulator returns control to the allocator.

All activities are run in this manner. When all activities have been run, the simulator converts SYSOUT print and punch files to ASCII and raw files respectively and submits them to the I/O daemon for printing or punching. (The conversion and the daemon requests can be prevented by using gcos command options.)

Note that successive compilations are run as separate activities and can cause the limit of 63 to be exceeded.

A single segment is used to simulate the slave area (the slave area is limited to 256K). The slave service area does not exist in the GCOS environment simulator. While the slave area prefix exists, its contents are not always guaranteed to be identical to that in native GCOS. The slave area is placed in the process directory and is named gcos_slave_area_seg.

GCOS SOFTWARE DATA BASES

The subroutine library (L*) is a multisegment file named gcos_library_subroutines_. (This library is known to gcos_tss by its add_name gtss_Lstar_.) It is a reformatted version of the file SOFTW-SYSLIB from a GCOS total system tape. Other subroutine libraries may be installed with the commands described in Appendix B and accessed via the $ LIBRARY and $ PRMFL cards in the job deck. (Any of these alternate subroutine libraries installed in the same directory as gcos_library_subroutines_ and given the add_name gtss_starL_ becomes known to gcos_tss as the *L library.)

The software library, which contains compilers, etc., is a multisegment file named gcos_system_software_. Its catalog is not in the same format as the one created by the system editor and, therefore, it cannot be updated by the system editor. The programs described in Appendix B are used to update this file. The GECALL, GESAVE, and GERSTR MMEs all accept this nonstandard catalog format, as well as the system editor random library format.

The simulator normally uses the copies of these two libraries that are found in the Multics system library. However, if the -userlib control argument is given on the gcos command line, the simulator searches for and uses (if found) copies of these libraries supplied by the user. It also uses a secondary library, which is searched after the software library. The effects of the -userlib argument are:

1. The search rules used in the search for the libraries are modified by placing the referencing_dir rule last. This allows a file contained in a directory in the user's search rules (such as the working directory) that has the same name as one of the libraries to be found before the file of that name in the Multics system libraries.

2. The file named gcos_second_software_ is sought by using the modified search rules previously described. If it is found, the file is searched via MME GECALL if the requested module is not found in the primary software library (gcos_system_software_). This differs from a ** file, which is searched before the primary software library.

Both of these user-supplied libraries may be constructed using either the programs described in Appendix B, or the system editor.

INPUT/OUTPUT OPERATIONS

Slave program I/O operations under the simulator, as under native GCOS, are initiated via a MME GEINOS instruction, which is followed by a select sequence specifying additional information pertaining to the I/O operation. This gives control to the simulator. The simulator then calls the standard Multics procedures to initiate activity on the peripheral subsystems.

The simulator performs actual device specific I/O only to magnetic tape devices. All other I/O devices are simulated by files in the Multics storage system. Print and punch output files are (optionally) converted to Multics file formats and printed or punched by Multics, after the GCOS job has ended. A separate Multics file is used to simulate each GCOS file used by the job. (The term "file" refers either to a segment or a multisegment file.)

Physical I/O, either to tapes or files, is performed by the Multics I/O system, which has complete control over the allocation of devices to user processes and the initiation of actual I/O operations. Since all I/O is done by Multics, the function performed in the native GCOS environment by MME GEROAD is not required under the simulator.

In addition, the simulator's I/O procedures provide the following capabilities:

- Symbolic file code to physical unit translation.

  Files are referred to symbolically with two-character file codes. This provides maximum flexibility of file assignment at execution time because files that a program can use are device independent.

- Simulated magnetic tape processing on disk.

  The simulator provides a method of disk processing that simulates the serial mode of processing normally associated with magnetic tape. This mode of processing (GCOS-linked) not only provides a degree of device independence to computer configuration, but also provides the opportunity to reduce program setup time by eliminating the use of magnetic tape for some files.

- File system protection.

  The simulator ensures that the user has been granted permission for permanent file access. This is accomplished in two ways. First, the standard Multics access controls ensure that the GCOS user has at least the minimum permissions necessary to access another user's permanent files. In addition, simulator I/O procedures validate all I/O requests against the I/O modes requested on the $ PRMFL control card. This prevents a job from requesting only read permission on a permanent file and then erroneously writing on that file.

Other native GCOS IOS (Input Output Supervisor) functions such as I/O interrupt processing, queuing of program I/O requests, and channel crossbar management are performed by Multics and are not part of the GCOS environment simulator procedures.

The simulator, operating under Multics, performs the logical functions necessary to interface the GCOS slave program I/O request to the Multics I/O system:

- Linking files to devices, which are allocated to the job through a Multics I/O stream
- File code to I/O stream translation
- File protection of user files
- Pseudo magnetic tape processing on disk/drum
- Simulation of the GCOS courtesy call dispatching mechanism

GCOS ENVIRONMENT SIMULATOR FILES

The Multics storage system is used by the simulator for:

1. The GCOS subroutine and software libraries (including assemblers and compilers)
2. GCOS environment permanent executable modules, permanent control tables, and temporary segments containing information specific to each job
3. GCOS save user temporary files
4. The slave core image seen by the executing slave program
5. SYSOUT collection space
6. GCOS user permanent files
7. System temporary files as required by the various language processors' utility programs

For a description of native GCOS system files, refer to the General Comprehensive Operating Supervisor (GCOS) manual.

GCOS Pool

The GCOS pool is a project directory of the GCOS daemon's directory (>ddd>GCOS) that functions in a manner similar to the SYSOUT pool in native GCOS. It can be used to contain the temporary files of a GCOS job. These files include the system-created input and working files, as well as files to contain SYSOUT output. In addition, the pool can be used to hold output "written" to the pseudoprinter and the pseudopunch under the simulator.

For the purposes of accounting, project and user directories containing permfiles exist under >gdd.

User Files

GCOS environment simulator users operate on mass storage files through the facilities of the Multics storage system. Both permanent files (nonremovable) and temporary files are processed by GCOS programs in the normal manner. The Multics features of paging and segmentation are transparent to simulator users. All GCOS files are actually Multics segments with no special attributes. Since they are one and the same, access to the files can be either by GCOS slave programs operating under the simulator or by any Multics procedure.

File Formats

Files to be used by a job executing under the simulator can exist either in the Multics storage system (in the form of paged segments) or on magnetic tape in the same logical structure as that under native GCOS. Files on magnetic tape can be processed by the same slave program operating under either the simulator or native GCOS.

Data formats are identical to those for native GCOS. Since the simulator interfaces with the slave software at the MME level, all Multics-supported GCOS slave program content managers (such as file and record control) define the content and format of the data to be read or written.

Standard GCOS access modes (sequential and random) are supported by the simulator. Thus, the access mode for a particular mass storage file, as determined at the time of file allocation (or as defined by the MME GEMORE mode change), causes the routine that simulates the MME GEINOS either to supply the desired I/O address for the user or to use the user-supplied I/O address.

File Orientation

The GCOS environment simulator assigns pseudoperipheral devices to an activity via file code designators and manages these peripherals during I/O operations. The programmer references all peripherals by using the file code designators (two alphanumeric characters) that are referenced in two ways:

- On file control cards that specify those files needed to execute an activity
- In communicating to the file and record control program, or to the simulator's input/output supervisor

The file code designators and their assigned file information blocks (FIB) are used by the simulator for peripheral identification and management. The FIBs are similar to the peripheral assignment tables (PAT) used by native GCOS in that each contains a description of a file and each can be pointed to by one or more file codes.

File Names

Temporary and output files created by the simulator are placed in the Multics file system hierarchy. The complete pathname of each file comprises a directory name (the complete pathname of the directory containing the file) and an entry name (the equivalent of a GCOS file name).

Each directory in the hierarchy either has a quota (a limit on the total space occupied by the files it contains), or borrows a quota from a superior directory. Quota is measured in records. One record equals one Multics page, which is 1024 words.

Jobs that produce a large amount of output or use large temporary files could fail to complete normally if there is not enough quota in the directories being used to hold the files created by the job. The simulator allows the files to be placed in up to four different directories (three of which can be specified by the user) to help alleviate the quota problem.

The four directories are: pdir (process directory), wdir (working directory), syot_dir (SYSOUT directory), and temp_dir (temporary directory).

The pdir and wdir directories are defined by Multics. Each process is assigned a process directory when it is created (having a quota determined by the system). The current value of this quota is 512 records. Each process has a working directory, which is initially the home directory of the user but can be changed (using the change_wdir command) to any directory to which the user has access.

The syot_dir and temp_dir directories are defined by the simulator. They are, by default, equal to wdir and pdir respectively, but they can be set to any directories to which the user has access (using the -syot_dir and -temp_dir control arguments on the gcos command line).

Files are placed in directories on the following basis:

Directory	Files
pdir	gcos_slave_area_seg
temp_dir	all temporary GCOS files
syot_dir	all GCOS output (print, punch, and SYSOUT files)
wdir	converted (ASCII or raw) copies of output files

The entry names of temporary and output files begin with the job identification (job_id). By default, the job_id is the entry name of the input file with the gcos suffix removed (if it is present). However, the job_id can be set to any name by using the -id control argument on the gcos command line.

Standard output files have the names:

job_id.sysprint
job_id.syspunch
job_id.exec_report

The exec_report file is copied into the sysprint file and deleted at the end of the job. However, it might be found in the syot_dir if simulation terminated abnormally. The sysprint and syspunch files contain all print and punch records, respectively, as written via the MME GESYOT instruction.

Any additional output files and all temporary files have names of the form:

job_id.a<activity number>.<+file code>

where: <activity number> is the number of the activity in which the file was first created.

<+filecode> is the file code, with asterisks replaced by plus signs (to avoid interfering with the Multics star convention, which assigns a special meaning to asterisks in file names).

For example, a job whose input file is named testjob.gcos has output files named:

testjob.sysprint
testjob.syspunch
testjob.exec_report

If the job consists of a compilation followed by an execution, the B* file containing the object code from the compilation is named:

```
testjob.a1.b+
```

and is placed in temp_dir. If the second activity contains a $ PRINT...P* control card, the file used to simulate the printer is named:

```
testjob.a2.p+
```

and is placed in syot_dir.

If the simulator default output processing (converting GCOS output files to Multics format) is not overridden by control arguments, the converted output files are named by appending "list" or "raw" to the GCOS file names for print and punch files respectively. Continuing the preceding example, the converted output files would be named:

```
testjob.sysprint.list
testjob.a2.p+.list
```

and (only if punched card output is actually produced by the job):

```
testjob.syspunch.raw
```

Note that the exec_report file does not have a corresponding list file because it is copied onto the sysprint file and then placed at the beginning of the sysprint.list file at the time the rest of the SYSOUT is sorted by report code.

JOB INPUT

The simulator is invoked via the gcos command to run one GCOS job at a time in one Multics process. Many users can be calling the gcos command simultaneously. Batch processing is provided by the GCOS daemon.

Functions performed inline by programs comprising the simulator include control card interpretation and MME processing. Other functions are provided by GCOS slave software that is executed under simulator control in the same manner as a user-written program (file and record control, general loader, and language translators).

GCOS Command

The gcos command invokes the GCOS environment simulator and initiates job processing. The command is used after the job stream has been entered into a file by the user or by the GCOS daemon. The format of the gcos command is:

```
gcos job_deck_path {-control_arguments}
```

This command is described in detail in Section 3.

Job Decks

Jobs punched on cards can be submitted to the simulator in two ways:

- The job deck can be submitted to the GCOS daemon. The deck is read by an online card reader under control of the daemon and an absentee job is submitted to run the GCOS job. (See Appendix A for more details.)

- The job deck can be submitted for input to a segment by using the Multics bulk card input facility (refer to the Programmer's Reference manual for information on this facility). The raw input mode must be used. Then, the gcos_card_utility command must be used to convert the resulting segment to a format acceptable to the simulator. The converted segment can be used as input to the simulator either in an interactive process or in an absentee job submitted by an interactive process.

IMCV Tapes

IMCV options available under native GCOS also are available under the GCOS daemon. IMCV tapes need only be in GCOS system standard format, as created by the bulk media conversion (BMC) program. (See Appendix A for more details.)

Data Formats

The job deck file supplied to the simulator (and any $ SELECT files) can be in either of two formats:

- Multics ASCII, as created by the Multics editors
- GCOS system standard format containing mixed BCD and binary card images

ASCII format is assumed. GCOS format must be specified, either by the gcos suffix on the name of the input file, by the control argument -gcos on the gcos command line, or in the comments field of a $ SELECT control card.

When a mixed BCD and binary card deck is submitted to the daemon, the deck is read in raw mode and translated to GCOS system standard format. If a card deck is read into a user's segment via bulk card input in raw mode, the deck must be converted to ASCII or to GCOS system standard format before being used as input to the simulator. (If the deck contains binary cards, it cannot be converted to ASCII.)

Like all other Multics software, the simulator uses the ASCII character set internally. However, the GCOS slave software that runs under control of the simulator uses the BCD character set internally. Therefore, the images of cards processed by the simulator must be in ASCII, while those processed by the slave software must be in BCD.

All nondollar cards (refer to Section 5 for definition) are processed only by the slave software and, therefore, should be in BCD (unless they are binary cards). All dollar (control) cards are processed by the simulator. However, many also are written on files for processing by the slave software. Therefore, both an ASCII and a BCD copy of each dollar card are needed.

The simulator's input reader accepts whatever format input file is provided and creates a job stream file containing the necessary ASCII and BCD card images. This involves some translations from ASCII to BCD and vice versa. Since the ASCII character set contains both uppercase and lowercase alphabetic characters and BCD contains only uppercase, some problems arise in translation.

Following the Multics convention of using lowercase alphabetic characters for most purposes, the simulator uses lowercase characters for control card names, control card options, file codes, and other character strings that GCOS users are accustomed to seeing printed in uppercase (BCD).

All BCD-to-ASCII translation performed by the simulator produces lowercase ASCII alphabetic characters (job output is an exception).

Control cards must be in lowercase when entered into an ASCII segment via one of the Multics editors. (Nondollar cards, which are translated to BCD, can be in either case, since both uppercase and lowercase ASCII letters are translated to the same BCD codes.)

The only exception to the requirement for lowercase ASCII on control cards is in Multics pathnames on $ PRMFL and $ SELECT cards. Multics pathnames can, and generally do, contain uppercase letters. These uppercase letters most frequently are the first letter of person and project names.

Since BCD is translated to lowercase ASCII, it is not possible to punch a Multics pathname containing any uppercase letters on a BCD card. However, GCOS file strings have their first catalog name replaced by the home directory pathname of the process running the simulation. Therefore, if the pathname of the file (relative to the home directory) contains no uppercase characters, the pathname can be punched as a file string. This is likely to occur, since uppercase letters appear most frequently in person and project names that are part of the home directory pathname.

Tab Replacement

An ASCII job deck segment (or an ASCII $ SELECT file) can be typed by using tabs to position fields to the card columns required by GCOS. The tabs are translated into the proper number of spaces by the simulator's input reader. This translation is known as canonicalization.

Tabstop positions vary with context. For nondollar cards, these positions are determined by the preceding activity card. For example, data cards following a $ FORTRAN card are canonicalized using a single logical tabstop in column 7.

The following sets of tabstops are used:

Tabstops	Used For
8, 16, 32, 73	dollar cards
8, 16, 32, 73	GMAP, 355MAP
7, 73	FORTRAN, FORT77
8, 12, 73	COBOL
10, 20, 30, 40, 50, 60, 70	PL1, ALGOL, JOVIAL, IDS, IDS2, CIDS2, RPG2

The columns following the last tabstop in each set are treated as if each contained a logical tabstop. Also, cards for which no tabstops are defined are treated as if each column contained a logical tabstop. Therefore, a tab occurring in a position where no tabstops are defined is translated to a single space rather than moving the logical character position to the end of the card.

The canonicalization process accepts the sequence tab-backspace to set the logical character position to the column before the next logical tabstop. However, the Multics terminal input software does not accept this sequence directly. Therefore, the backspace following the tab must be entered as some other character

and later changed to a backspace (using an editor). The user can choose some character that will not be used elsewhere in the file, type the character wherever a backspace is intended to follow a tab, and then change all occurrences of that character to a backspace. This is done by using a global substitution.

If the file contains no tabs, processing time can be saved by using the -no_canonicalize control argument on the gcos command line. This causes the canonicalization process to be omitted.

ASCII input lines are checked for length exceeding 80 characters. If canonicalization is being done, the length after canonicalization is checked. If the -truncate control argument is given on the gcos command line, lines that exceed 80 characters are truncated to 80 with no warning. Otherwise, a line that exceeds 80 characters is treated as a nonfatal error.

The action taken for a nonfatal error is controlled by the -brief and -continue control arguments. (See the introductory paragraphs of Section 5, concerning -brief and -continue.)

Identification Process

The simulator scans the first two records of the job input stream to ensure they are the $ SNUMB and $ IDENT control cards. If the job is submitted via the GCOS daemon, the $ SNUMB card must contain a sequence code (number) that is not already in the system (because the $ SNUMB code is registered in a table and is used to uniquely identify certain working files for the job.)

Job Limits

A GCOS job executing under the simulator is subject to resource usage limits in the following higher level functions:

- Slave memory must be less than 256K.
- Maximum of 63 activities per job.
- All printer and card punch output is stored online until the end of the job. This is true even though the job thinks it has the device directly attached. Therefore, sufficient space is required.
- A maximum of 40 files can be open at any one time. However, five of these files are dedicated to the subroutine library (L*), the execution report, the SYSOUT print and punch collector files, and the loader file (R*). The number of files that can be created per activity is 35 minus the number of files currently saved from preceding activities. For example, if activity 1 saved four files and activity 2 saved two files, only 29 new files can be created by subsequent activities (35 minus 4 minus 2) until some of the previously saved files are released. Note that the system files allocated for each activity are deducted from the 35 files, thereby leaving a smaller number of files than can be allocated by a user. Each activity can refer to any or all of the 35 files.

Variable parameters, which can be varied via Multics startup control cards or by operator commands, include the number of tape drives that a process is permitted to attach and the number of concurrent absentee processes being processed in parallel.

ALLOCATION

There are several differences between Multics and native GCOS in the manner in which the allocation, execution, and termination functions are accomplished. However, it is not the intent of the simulator to duplicate the implementation of native GCOS, but only to simulate its effects on jobs and slave programs. Thus, allocation of all system resources (peripherals, core memory, and processor time) comes under final control of Multics. The GCOS environment simulator serves only as the intermediary.

When a new job is received, an initial pass is made through the input stream. At this time, canonicalization of any ASCII input is performed, the contents of any $ SELECT files are copied into the input stream, the presence of execution activities is noted, and some limited control card checking is performed.

During the second pass, detailed processing of the control cards for each activity occurs just before the activity is run. Control card errors cause the job to be aborted at this point, but preceding activities have already run.

When an executing slave program initiates remote teleprinter I/O, the terminal I/O is to the user_input and user_output Multics I/O stream. These normally are attached to the interactive user's console. The program then can interact with a Multics user at the console in the normal manner.

Peripheral allocation has three categories:

1. Real devices (magnetic tapes, interactive user consoles, and all temporary mass storage files requested on the various $ control cards).
2. Pseudo devices (printers and card punches).
3. Devices not permitted (card reader, paper tape reader or punch, and removable disk packs).

The simulator job stream input procedure reads all cards from the canonicalized source up to the next activity definition card or activity termination card. All peripheral allocation control cards for magnetic tape result in allocation of tape handlers. During this phase, tape mounting instructions are issued via the Multics attach mechanism.

Pseudoperipherals are allocated as the result of printer and card punch allocation control cards. (The job cannot allocate a card reader.) A pseudoperipheral appears to be the actual peripheral device for the executing slave program, but it actually is a segment in which the simulator retains the output. This segment then is processed in a manner similar to that used to convert SYSOUT output to a form acceptable to the Multics I/O daemon for printing or punching.

After the peripherals and scratch files are allocated, memory is allocated for the activity. The simulator determines the amount of memory required (from the appropriate $ LIMITS control card definition) and creates a segment of this size to hold the executing slave program.

All mass storage allocation cards requesting temporary file space are processed in the same manner as those requesting peripheral allocation. A Multics segment is created with the appropriate size limitations. Both random and sequential access modes are processed through the simulator's I/O routines.

The simulator makes access to permanent files for the slave program. These files are permanent segments in the Multics storage system and are subject to the Multics permission modes defined for that segment. Passwords on $ PRMFL control cards are ignored.

Where appropriate, peripheral allocation for GCOS activities run under the simulator takes advantage of existing and newly implemented Multics facilities. Resource allocation conflicts, such as making access to a data base for updating, must be resolved by manually scheduling conflicting jobs for processing at different times.

When the assigned peripherals are ready (sufficient memory is assumed), the activity passes into the execution phase. No job activity is initiated until all prior activities of that job have been completed.

EXECUTION

During execution, the activity is initiated in the same manner as native GCOS. Each activity is executed under the supervision of the simulator. All user communication with the simulator is provided at execution time by the MME routines, which provide requests for service, exchange of information, and control functions. Many of the MME routines are automatically called by the compilers and by file and record control programs.

Once loaded and executing, the slave program is similar to other procedures executing under Multics in that the segment is paged as necessary by the Multics paging mechanism. However, a significant difference is that the segment containing the executing GCOS slave program must have Write permission for the user because procedures and data are intermixed.

After termination of the first activity, control is returned to the simulator and processing for the second activity begins.

Fault Processing

Responses to any exception conditions or processor faults are executed during this phase. When an exception condition or fault is encountered, either standard actions are performed or the user is given the option of intervening, depending upon the particular error and the I/O request.

The method by which faults are processed depends upon whether the fault was caused by the simulator or by the slave program.

If a fault (or other error) occurs within the simulator (rather than within the slave program), the user fault vector is not examined. Instead, the reason for the fault or error is printed on the user's terminal. Then, by default, the activity is aborted via a simulated MME GEBORT. However, if the -debug control argument is given on the gcos command line, the user is given the opportunity to intervene, using Multics debugging tools, to determine the cause of the error and possibly to correct it.

Slave program faults are interpreted by the simulator procedures. If the fault is because of a MME, the MME is analyzed and the appropriate simulator procedures are called to simulate the effects of native GCOS.

For program faults, the fault causes interrogation of the user fault vectors. If the user specifies a fault processing routine, control is transferred via this vector back to the slave activity. If no fault processing routine is specified in the user fault vector, a MME GEBORT is set in the user program and control is transferred to MME GEBORT. GCOS allows processing of the following program faults by a user error subroutine:

memory
divide check
overflow
command
illegal operation code
fault tag
derail

A user can process an unlimited number of divide check, overflow, and derail faults. However, a second attempt to process a fault tag, command, illegal operation code, or memory fault causes the program to abort, except under a special condition. The special condition is that the Instruction Counter and Indicator Register (IC and IR) word (even) of the fault vector pair is zero and the second word is nonzero. Under this condition, the simulator allows the user to process one of these faults after the first fault is processed.

The user cannot invoke a fault processing routine for the following hardware faults:

lockup
parity
startup
execute
connect
shutdown
timer runout
operation not complete

TERMINATION

Three types of termination (normal, abnormal, or simulator fault) can occur when running a user activity under the GCOS environment simulator.

In native GCOS, successive GMAP or FORTRAN activities (without intervening $ IDENT or file control cards) are run as a single GMAP or FORTRAN activity. Thus, a job containing ten FORTRAN compilations, followed by a $ EXECUTE control card, is run as two activities (a FORTRAN compilation of ten routines and an execution activity). Under the simulator, each compilation or assembly is a separate activity that undergoes termination.

Normal Termination

Normal activity termination occurs when a MME GEFINI is executed by a user program. If an activity terminates normally, the user does not get a dump, wrapup, or an abort subactivity, regardless of whether the slave program prefix contains a wrapup address.

The execution report is generated, peripheral files are released, and memory is released.

If the activity is the last activity of the job (delimited by a $ ENDJOB control card), a *EOJ message is issued on the console and the job is released for printer or punched output. If the activity is not the last activity, allocation of the next activity begins immediately.

Abnormal Termination

Abnormal termination can be caused by:

simulator-detected errors
slave program requests (MME GEBORT)
hardware-generated/system faults

Any of these results in an immediate interrupt of the activity. Since all I/O is synchronous in the simulator, no I/O is lost in an abnormal termination.

SIMULATOR-DETECTED ERRORS

The locations of the fault or error and the reason code are placed in bits 0-17 of word 13 (octal) of the slave program prefix. A *ABT message is issued on the user's output stream. If bit 0 of the program switch word is on, a postmortem dump is generated. If word 27 (octal) of the slave program prefix contains a wrapup address, the word is cleared and the user is given control at that address. (This word normally is initialized by the ".SETU." routine if this is an execute activity.) After the (optional) wrapup terminates, bit 12 of the program switch word is examined. If the bit is set, a Utility subactivity is executed. Upon return from this subactivity, the activity is terminated.

MME GEBORT

In the case of a MME GEBORT, the program supplies the reason code; the simulator provides the remaining data. The IC and reason code are inserted in word 13 (octal) of the slave program prefix:

bits	0-17	IC (location of termination)
	18-20	2 (MME GEBORT reason code)
	21-23	Zero
	24-35	Reason code

Memory and peripherals allocated to the activity are deallocated, and dismounting instructions are issued to the operator on the console. A postmortem memory dump is written on the execution report if bit 0 of the program switch word = 1. If the address (bits 0-17) of word 27 (octal) of the slave program prefix is within slave program limits, return is made to the user for wrapup. Then, the check for abort subactivity is made.

HARDWARE-GENERATED AND SYSTEM FAULTS

If a fault or other error occurs during the running of an activity (between the SRT and FIN messages on the user's terminal) and the simulator is unable to handle it, control passes to the Multics command processor. Execution of the job is suspended at the point of the fault. The user may wish to use Multics debugging tools to determine the cause of the error. After this is done, there are two possible courses of action: 1) to terminate the simulation immediately

with all output being lost, type "release", and 2) to reenter the simulator, possibly continue execution of the job, and to obtain output, type:

```
gcos_mme_bort_$simulatorØØxx
```

where ØØ is one or more spaces and xx is any two characters (representing an abort code). Control is returned to the simulator, which simulates a MME GEBORT utilizing the two-character abort code required by the program.

Refer to the General Comprehensive Operating Supervisor (GCOS) manual for a description of the dump, wrapup, and abort subactivities and terminate messages.

If a fault occurs in the simulator before or between activities and causes control to pass to the Multics command processor, simulation of a MME GEBORT causes unpredictable results.

There is an alternative to the preceding command that causes the job to terminate immediately. Output buffers are forced out, files are closed, and SYSOUT is generated (if possible). The command is:

```
gcos_cc_endjob_
```

This command causes the simulator to be entered at the $ ENDJOB control card processor, which performs the end-of-job processing previously described.

JOB OUTPUT

Job output comprises the SYSOUT (sysprint and syspunch) collection files, any print or punch files created by activities, and any permfiles created by the job. Permfiles remain in the Multics storage system when the job is finished. By default, all SYSOUT print and punch files are converted from GCOS to Multics format, placed in the working directory, and queued for output by the Multics I/O daemon. Control arguments on the gcos command line can be used to request output processing other than the defaults.

The sysprint generated by the slave programs is in BCD. The execution report produced by the simulator contains both BCD and ASCII records. ASCII is used for lines that might contain Multics pathnames, since uppercase and lowercase letters are significant and should be preserved. By default, BCD sysout is translated to uppercase ASCII to simulate the appearance of SYSOUT printed by native GCOS. The -lower_case control argument can be used on the gcos command line to cause BCD SYSOUT to be translated to lowercase ASCII.

*

The GCOS SYSOUT line limits are enforced. The lines counted include only those supplied by the slave program, including MME GEBORT core dumps and MME GESNAP print lines. Refer to the Control Card Reference Manual, Order No. DD31 for limits regarding various activities.

SECTION 3

USER COMMANDS

This section contains descriptions of the GCOS environment simulator user commands, presented in alphabetical order. Each description contains the name of the command (including the abbreviated form), discusses the purpose of the command, and shows the correct usage. Notes and examples are included when necessary for clarity.

The commands are:

gcos, gc
: invokes GCOS batch environment simulator.

gcos_card_utility, gcu
: copies card image files, translating from GCOS format to ASCII or vice versa.

gcos_create_file, gcf
: creates GCOS random files.

gcos_fms, gfms
: performs certain functions of the GCOS file management supervisor from Multics command level.

gcos_label_tape, gclt
: writes a GCOS label on tape.

gcos_set_environment, gse
: sets and modifies the user-controlled runtime parameters.

gcos_sysprint, gsp
: converts GCOS BCD sysout print files to ASCII files suitable for use with the dprint command.

gcos_syspunch, gspn
: converts GCOS BCD sysout punch files to files suitable for use with the dpunch command.

gcos_tss, gtss
: invokes GCOS time sharing environment simulator.

Name: gcos, gc

The gcos command invokes the GCOS batch environment simulator to run a single GCOS job in the user's process.

Related facilities include the GCOS daemon, which provides batch processing for GCOS jobs under Multics, and the gcos_sysprint, gcos_syspunch, and gcos_card_utility commands, which may be used to manipulate GCOS format files that reside in the Multics storage system (these commands are described later in this section). More information on the simulator and the GCOS daemon may be found in earlier sections of this manual.

Usage

gcos job_deck_path {-control_args}

where:

1. job_deck_path
   is the pathname of a file (segment or multisegment file) containing a GCOS job deck. The file may contain ASCII lines (as produced by one of the Multics editors) representing card images; or it may be a GCOS standard system format file, containing BCD and binary card images. It is assumed to contain ASCII lines unless the GCOS format is specified. One way of specifying GCOS format is to have the name of the file end with the gcos suffix. The other way is the -gcos control argument, described below.

2. control_args
   may be chosen from the following control arguments. They may appear in any order and may precede or follow the job_deck_path.

Control arguments specifying the input supplied to the simulator are:

-ascii, -aci
the input file contains ASCII lines, as produced by one of the Multics editors. This is the default, but this argument may be used if the name of an ASCII file ends with the suffix gcos (to avoid the necessity of renaming the file).

-gcos, -gc
the input file is in GCOS standard system format, containing BCD and binary card images. Such a file could have been produced as output of a previous GCOS job, or by the gcos_card_utility command.

-no_canonicalize, -nocan
may be used to save processing time when an ASCII input file contains no tab characters, and the fields on all the card images are aligned in the columns required by GCOS. Normally, an ASCII input file may contain tabs separating the fields on each line. The process of replacing these tabs by the appropriate number of blanks to align the fields in the columns required by GCOS is known as canonicalization. Logical tab stops are known for GCOS $ control cards and for all the languages supported by the simulator. See Section 2 of this manual for more information on canonicalization.

-truncate, -tc
if any ASCII input file (the job deck file, or any $ SELECTed file) contains lines longer than 80 characters (after canonicalization), the extra characters are assumed to be part of comments, and are discarded without warning. If this argument is not given, the first line longer than 80 characters causes the job to be rejected.

Control arguments specifying the disposition of output from the simulator are:

-dprint, -dp
queue the converted print files for printing by the Multics I/O daemon, followed by deletion. (The -list argument is implied and need not be given.)

-dprint_options "options", -dpo "options"
queue the converted print files for printing by the I/O daemon, but use the dprint control arguments supplied in the options string instead of the default of -delete. The options must be enclosed in quotation marks if they contain blanks or other delimiter characters recognized by the command processor. The dprint command is called via cu_$cp so that a user-defined abbreviation for dprint (that supplies default heading and destination arguments, for example) would be used in this call. (The -list and -dprint arguments are implied and need not be given.)

-dpunch, -dpn
queue the converted punch files for punching by the I/O daemon in raw mode, followed by deletion. (The -raw argument is implied and need not be given.)

-dpunch_options "options", -dpno "options"
queue the converted punch files for punching by the I/O daemon, but use the dpunch control arguments supplied in the options string. The -raw argument is always used for dpunch, since the converted punch files are not suitable for punching in any other mode. The explanations under -dprint_options above, regarding quotation marks and abbreviations, apply to this argument as well. (The -raw and -dpunch arguments are implied and need not be given.)

-hold, -hd
do not perform the default conversion and daemon output of print and punch files. The default is:

-dpo -dl -dpno "-dl -raw"

Since the default for each file type (print or punch) is overridden when any of the above arguments is specified for the given file type, the -hold argument is only required when one of the file types is to be left in GCOS standard system format, with no conversion or daemon output being performed.

-list, -ls
convert print files (both SYSOUT and simulated printer) from BCD to ASCII and delete the BCD copy. (This conversion is performed by a call to the gcos_sysprint command for each file.)

-lower_case, -lc
translate alphabetic BCD characters in print files to lowercase ASCII. (Default is uppercase)

-raw
convert punch files (both SYSOUT and simulated card punch) from BCD

(or binary) to an internal format suitable for punching by the Multics I/O daemon in raw mode (960 bits per card image) and delete the BCD copy. (This conversion is performed by a call to the gcos_syspunch command for each file.)

Control arguments governing the creation and use of files by the simulator are:

-job_id id, -id id
use the job identification specified by id in the names of files created by the simulator for this job. See Section 2 of this manual for more information on the naming of files. The id may be any character string up to 18 characters to be used in file names, or it may be one of the following control arguments:

-unique
use a Multics unique name as the job id. (A unique name is a 15-character string, generated by the unique_chars_ subroutine, beginning with an exclamation point and guaranteed to be unique within the system.)

-jd_seg, -jd
use the entryname of the job deck segment as the job id. If the entryname ends with gcos, that suffix is removed from the id. (Default)

-syot_dir path, -sd path
use the pathname of a directory specified by path for the GCOS format copies of print, punch, and sysout files. By default, the working directory is used. (The converted copies of these files are always placed in the working directory.)

-temp_dir path, -td path
use the pathname of a directory specified by path for temporary GCOS files. (Default is the process directory)

Other control arguments are:

-parameter STRs, -pm STRs
where STR strings override GCOS JCL parameter values specified on the $ PARAM card (i.e., specify the #n values in the JCL input). For example:

job_deck: (in segment tape_copy)

```
$       snumb       joba
$       ident       apert,name
$       param       ,99999
$       utility
$       tape9       01,x1d,,#1,,data-orig,,den8 -noring
$       tape9       02,x2d,,#2,,data-copy,,den16
$       futil       01,02,copy/1f/
$       endjob
```

Multics commands:

```
gcos tape_copy -ls -pm X2753
```

copies tape X2753 to a scratch tape.

```
gcos tape_copy -ls -pm X2753 X2754
```

copies tape X2753 to tape X2754.

Null parameters may be specified in the command with "". Parameters beginning with "-" must be preceded with -string. For example, to specify that parameter 1 is equal to "X2753" and parameter 3 is equal to "-copy", enter:

```
gcos tape_copy -pm X2753 "" -string -copy
```

-brief, -bf
: suppresses the printing of all terminal output produced by the simulator except for fatal error messages. Output from the slave program is not suppressed.

-continue, -ctu
: continues processing of the job when a nonfatal error occurs. Unless the -brief control argument is given, a warning message is printed on the user's terminal. If this argument is not given, the first nonfatal error causes the job to be rejected. Nonfatal errors occur mainly in control card processing, and are described in detail in Section 5.

-debug, -db
: invokes interactive debugging aid to be used in the Multics environment (refer to the debug command in the Commands manual). Normally, unrecoverable errors cause an immediate abort of the current function and a reset to command level. Use of this control argument causes batch to call the debugging tool for further analysis of the problem and possible correction.

-long, -lg
: requests certain lines from the execution report, including the begin and end activity lines (containing program switch word, etc.) to be printed in addition to the normal terminal output.

-userlib
: enables the use of GCOS slave software libraries supplied by the user, instead of, or in addition to, the copies of the libraries installed in the system. The use of this argument is described in Section 2 under "GCOS Software Data Bases."

Notes

If no control arguments are given, the defaults are such that the command:

```
gcos path
```

is equivalent to the command:

```
gcos path -aci -dpo -dl -dpno "-dl -raw" -id -jd
```

Any $TAPE control card in the job deck issues a mount message (to the operator) for the designated tape. The mount message requests insertion of a write ring in the tape reel unless the -noring or -nr control is included as a comment on the $TAPE card (see the job deck example included in the -parameter control argument above).

gcos

For tape processing, the simulator assigns drive numbers according to the order in which tape devices are requested. These numbers are used in reporting information to the caller and have no correspondence to the tape unit identifiers used by Multics.

When tape density of 1600 or less is specified, and a tape drive capable of both 1600 and 6250 bpi is used, difficulties with tape processing may occur. The problem can be circumvented by using the Multics assign_resource command to request a tape drive (i.e., one that does not support both 1600 and 6250 bpi).

gcos_card_utility

Name: gcos_card_utility, gcu

The gcos_card_utility command copies GCOS card image files, altering their format, content, and medium, as specified by the user. The command can make a partial copy of an input file, separate the input file into several output files, or combine several input files into one output file.

The acceptable formats are: Multics ASCII (as produced by Multics editors); GCOS standard system format (SSF) (defined in the File and Record Control manual); or raw (one of the formats used by the Multics I/O daemon for card input and output). GCOS SSF or raw format files can contain compressed source decks (COMDK) (described under "RDREC" in the above-referenced manual).

The contents of the files may include: one or more complete GCOS jobs (IMCV); one or more source or object decks (in the format produced by the GCOS source/object library editor); binary card images; or any other card images (partial GCOS jobs) subject to certain restrictions.

The medium may be magnetic tape or a segment or multisegment file in the Multics storage system.

Usage

gcu input_specification output_specification

The two specifications may appear in either order. The first must be completed before the second is begun. They consist of pathnames (or tape numbers) and control arguments specifying format, content, medium, partial copying, and the amount of information that should be printed on the user's terminal.

The arguments that can be used in the specifications are described below. Some are meaningful only for input or for output; others have slightly different meanings for input and output. These restrictions are noted in the argument descriptions.

A group of arguments, preceded by certain control arguments, is defined to be a list. A list must follow several of the control arguments described below. The format of a list is described below, under "Input and Output Lists."

The argument descriptions should be read sequentially for a better understanding of the operation of the command. The meanings of, and restrictions on, combinations of arguments are mentioned in the descriptions themselves, if they fit the context. Otherwise, they are mentioned later in the "Notes" portion of this command. In addition, some examples are given at the end of this command description.

Except where otherwise noted, the arguments within the input and output specifications may appear in any order.

gcos_card_utility

Input and Output Specifications

The following control arguments are used to signify the beginning of the input or output specifications:

-input, -in
signifies the beginning of the input specification. It may be omitted if the input specification appears first, and the input consists of a single file, and the pathname of that file is the first item of the input specification (i.e., if the pathname of the single input file is the first argument of the command).

-output, -out
signifies the beginning of the output specification. It may be omitted if the output specification is second, and the output consists of a single file, and the pathname of that file is the first item of the output specification, and the input specification does not end with a list (i.e., if the pathname of the single output file can be recognized as such, thus ending the input specification and beginning the output specification).

Pathname

The following argument is used to specify an input or output file in the Multics storage system:

path
the relative or absolute pathname of an input or output file. If the entry portion of the pathname ends with any of the suffixes ascii, gcos, raw, or comdk, it is interpreted as having the same meaning as the corresponding control argument (-ascii, -gcos, -raw, or -comdk).

A conflict between a suffix and a control argument is treated as an error, unless the control argument precedes the pathname, in which case the control argument overrides the suffix. This allows a file whose format does not match its suffix to be processed without having to be renamed first.

For multiple file input (or output), the format of the input (or output) files must be the same. If the format is not specified by a control argument (which must precede the entire list if the pathnames have suffixes), the suffixes determine the format, and they must be consistent.

The star and equal conventions may not be used in the pathname argument.

gcos_card_utility

File Formats

The following control arguments are used to specify the format of the input and output files:

-ascii, -aci
Input and output:

indicates that the file is in Multics ASCII format, as produced by one of the Multics editors. The lines are of variable length and end with the ASCII newline character.

Input:

no canonicalization of the input lines is performed unless specifically requested by the -canonicalize control argument. If canonicalization is requested, the logical tabstops must be determinable from context or be given with the -tabs control argument (described below). If the file is to be translated to BCD, it can contain only those ASCII characters that have BCD equivalents.

Output:

binary input card images are discarded. A warning message is printed for each contiguous group of card images discarded, giving the size of the group. If the input is BCD, an ASCII newline character is appended to each translated BCD card image, resulting in ASCII output lines of 81 characters each. (See the -truncate control argument below.)

-comdk, -cdk
Input:

indicates that if any COMDKs are present in the input file, they are to be decompressed. (This does not require that any COMDKs be present.) If this argument is not given, COMDK cards are treated as binary cards, which are copied unchanged if the output format is GCOS or raw, but discarded if the output format is ASCII.

Output:

indicates that all cards except binary cards and dollar cards (GCOS control cards that have a "$" in column 1) are to be compressed before being written in the output file.

This control argument is designed to compress individual source decks, and may produce unusable results if run on complete jobs, since there are some nondollar cards that GCOS slave programs do not expect to be compressed.

Input and output:

only GCOS and raw format files may contain COMDKs. If raw format is not specified, GCOS format is assumed (and need not be specified explicitly) for a file containing COMDKs.

-gcos, -gc
Input and output:

indicates that the file is in GCOS standard system format. It contains 320-word blocks (320 or fewer, for tape). Blocks begin with a block

control word (BCW) and contain variable length records, each beginning with a record control word (RCW). The records may contain BCD or binary card images, or ASCII lines (media codes 1, 2, or 6).

-gcos_ascii, -gca
Output only:

indicates that the output file is written in GCOS standard system format, but the records contain ASCII characters (media code 6) and are of varying length. Trailing new lines are removed, and any unused characters in the last word are filled with ASCII PAD (octal 177) characters.

-gcos_bcd, -gcb
Output only:

indicates that the output file is written in GCOS BCD format.

-no_canonicalize, -ncan
ASCII input only:

indicates that canonicalization should not be performed. (Default on all ASCII input files)

ASCII card images may be typed in free format, with tabs separating fields. Canonicalization is the process of replacing the tabs with the correct number of spaces to align the fields in the columns required by GCOS. Logical tabstops are known for dollar cards (columns 8, 16, and 32) and for all of the languages supported by the GCOS environment simulator.

If the card images contain no tabs, processing time can be saved by omitting the canonicalization.

-raw
Input and output:

specifies that the file contains card images, in the format produced (and accepted) by the Multics I/O daemon, for card input (and output) in raw mode. Each card image is a 960-bit string, 12 rows by 80 columns, with each 12-bit string representing the punches in one column, a "1"b signifying a punched hole.

The actual reading and punching of cards is not done by this command. See the description of the dpunch command in this document and the "Bulk Input and Output" description in the Programmer's Reference manual for information on card input and output.

-tabs tabstops
ASCII input only:

used to supply logical tabstops to be used for canonicalization. A list of from 1 to 10 column numbers, in increasing numerical order, separated by spaces, must follow this argument.

Normally, the set of logical tabstops to be used for each group of nondollar cards is determined from the system call card (e.g., $ FORTRAN, $ GMAP) that precedes them. If the input file contains nondollar cards not preceded by a system call card and canonicalization is in effect (-canonicalize is specified), the tabstops must be supplied with the -tabs control argument. When they are supplied this way, they are used for all nondollar cards in the file instead of the tabstops implied by any system call cards in the file.

gcos_card_utility

-truncate, -tc
ASCII input:

indicates that ASCII input lines might be longer than 80 characters (after canonicalization, if it is being performed), and that any such lines are to be truncated to 80 characters without warning.

In the absence of this argument, the first occurrence of an ASCII line longer than 80 characters causes the command to be aborted.

ASCII Output:

trailing blanks are removed from ASCII output lines, resulting in lines of varying length, 81 characters or shorter, including NL.

The format of the input or output files can be specified explicitly by one of the above control arguments (with the exception of -comdk), or by a suffix on the pathname. It may also be implied by some other argument or combinations of arguments, such as specifying -tape or specifying -comdk and not -raw, both of which imply -gcos. In the absence of any indication to the contrary, ASCII format is assumed.

Only one of the above control arguments (-ascii, -gcos, -gcos_ascii, -gcos_bcd, or -raw) may be used in one I/O specification. The control arguments (-no_canonicalize, -tabs, and -truncate) may be used only for an ASCII file.

File Contents

The following control arguments are used to specify the content of the input and output files:

-imcv list
Input only:

specifies that the input file contains several complete GCOS jobs that are to be separated, either to copy each into a separate output file, or to select a subset of all the jobs for copying.

This argument must be followed by one or more arguments specifying the snumbs of the jobs to be copied. The format of this list of snumbs is described below under "Input and Output Lists."

-library list, -lib list
Input only:

specifies that the input file contains several source or object decks, each preceded by a $ GMAP, $ 355MAP, or $ OBJECT card with the four-character edit name in columns 73-76 (the format produced by the GCOS source/object library editor), and that the decks are to be separated, either to copy each into a separate output file, or to select a subset of all the decks for copying.

This argument must be followed by one or more arguments specifying the edit names of the source decks to be copied. The format of this list of edit names is described below under "Input and Output Lists."

gcos_card_utility

Only one of these control arguments (-imcv or -library) may be used in an I/O specification.

Tape Files

The following control arguments are used when the medium of the input or output file is tape:

-attached, -att
may be used instead of a tape number, to indicate that the tape remains attached from a previous use of this command. Multifile reels may be read or written by using this command repeatedly and keeping the tape attached between uses with the -retain argument.

This argument may also follow (not necessarily immediately) the tape number, so that the number of an attached tape can be supplied for writing in the label.

-detach, -det
detaches a tape that was retained by a previous use of this command. There are two uses:

- If it is the only argument on the command line, any previously retained tapes, either input or output, are detached, and no other processing takes place.
- If it is used within an input (or output) specification, any previously retained input (or output) tape is detached, allowing a different input (or output) tape to be attached.

-label label, -lbl label
used to select a file from a multifile input tape and to specify position and file name for an output tape. Use of this argument is optional. label has three possible formats:

n
file_name
n,file_name

where n is the position of the file on the tape to be read or written and file_name is the 12-character name to be found (or written) in words 9-10 of the tape label.

On input, if -label is not given, or on output, if n is not given, current tape position is assumed. The tape is positioned at its beginning, except for tapes that are retained from previous activities. (The -label control argument should not be given for a retained input tape that is positioned correctly, since it causes the tape to be rewound and searched from the beginning for the specified position or file name. An output tape remains positioned if only the file_name is given, but is rewound if the position is given.)

If both n and file_name are given on input, and file_name does not match words 9-10 of the nth label, the user is given the option to quit or continue.

If file_name is not given on output, words 9-10 are set to blanks. If file_name is the string -no_labels or -nl, then label processing

gcos_card_utility

is omitted and every record read or written is treated as a data record.

-retain, -ret
specifies that the tape is to remain attached, and positioned at the end of the file read or written, when the command finishes. It is then available to subsequent uses of this command without requiring the operator to mount and dismount it repeatedly. This argument must be given in each use of the command except the last, to keep the tape mounted.

-tape tape_number, -tape7 tape_number, -tape9 tape_number
Input and output:

specifies a GCOS magnetic tape as input or output. For -tape the installation default number of tracks is used. For all three, the installation default density (for the given number of tracks) is used, unless density is specified in tape_number (e.g., "-tape9 12345,den=1600").

One of these -tape control arguments must precede the -retain and -label control arguments, if they are given.

The tape is assumed to be labeled with standard GCOS labels, as described in the File and Record Control manual. (See -no_labels under the -label control argument below.) Multifile reels are supported, but only as described below. Multireel files are not supported.

tape_number is the tape reel serial number (word four of the label). It may be up to five characters long. It is used in the mount message to the Multics operator and written in the header label, for output.

Only GCOS format files can be read from or written onto tape; the -gcos control argument is implied and need not be given.

If a tape file is specified for input (or output), it must be the only input (or output) file specified.

Partial Copying

The following control arguments may be used to specify that a subset of the card images in each input file should be copied:

-count N, -ct N
where N is the number of card images to be copied.

-first N, -ft N
where N is the number of the first card image to be copied.

-last N, -lt N
where N is the number of the last card image to be copied.

Counting is performed on card images sent to the output file, but before they are compressed, if COMDK output was specified. Input card images have already been decompressed, and binary card images discarded for ASCII output, before counting takes place.

gcos_card_utility

If -first is not given, copying begins with the first card image. If neither -last nor -count is given, copying proceeds to the end of the file. The -last and -count control arguments may not both be used in the same specification.

For IMCV or library content, only the specified subset is searched for the jobs or decks to be copied, and copying stops in the middle of a job or deck if the end of the subset is reached. Also, COMDKs that are not being copied are not decompressed, so COMDK card images, rather than decompressed card images, are counted during the search for the next job or deck to be copied.

Output File Name Duplication

The following control argument may be used to specify the action to be taken when the output file already exists:

-append, -app
Output (nontape) only:

specifies that, if an output file already exists, the output is to be appended to it, immediately following its current contents. The format of the existing file must be the same as that of the output to be written. If it is not, the results are unpredictable.

If this argument is not given, and the output file already exists, the user is given the choice of aborting the command, or deleting the existing file and continuing.

Input and Output Lists

The following arguments are used to specify a list of Multics files, or of jobs or decks within a single file, to be copied:

-all
Input only:

may follow -imcv or -library to indicate that the entire contents of the input file are to be copied, each job or deck into a separate output file.

-file input path, -file path, -fi path
Input and output:

path is the pathname of a segment containing the file names, or snumbs, or edit names, one name per line. This form may be used (after -list, -imcv, or -library) when the list of names is long and inconvenient to type on the command line.

gcos_card_utility

-list list, -ls list
Input and output:

specifies a list of Multics files to be copied from or into. The list that follows it has the same format as the lists after -imcv or -gmap, except that -all cannot be included in it.

-name, -nm
Output only:

may follow -list to indicate that the names of output files should be taken from the snumbs or edit names of jobs or decks from an IMCV or library input file. The output files are placed in the working directory and have entrynames of the form snumb.suffix or edit_name.suffix, where suffix is ascii, gcos, or raw, depending on the format of the output file.

name1 ... namen
Input and output:

names of input output files, input snumbs, or input edit names may be given on the command line, after -list, -imcv, or -library.

Terminal Output

The following control arguments are used to specify the amount of information to be printed on the user's terminal:

-brief, -bf
Input and output:

specifies that messages warning of nonfatal errors, or giving detailed information about errors, should not be printed on the terminal. The argument may be used in both the input and output specifications, to suppress messages connected with the reading of input or the writing of output.

-debug, -db
Input or output:

may be given once on the command line, in either the input or the output specification. Its meaning is independent of its location. It indicates that whenever an error occurs, the Multics debug command (see the Commands manual) is to be called after the error message is printed, giving the user the opportunity to determine the cause of the error and possibly correct it and cause processing to continue.

-long, -lg
Input:

specifies that the names of input items (snumbs, edit names, tape file names, Multics pathnames) be printed on the terminal as they are read from input.

Output:

specifies that the names of items selected for copying be printed on the terminal as they are written to output.

gcos_card_utility

Input and output:

if this argument is given for both input and output, all input item names are printed, and those selected for copying are so indicated.

This argument is not inconsistent with -brief. The two arguments control the printing of different types of information, and both may be given on the same command line.

Notes

Only one of the control arguments specifying a list of items (-list, -imcv, or -library) may be given in an input or output specification. The items can be file names, snumbs, or edit names.

If a list of output file names is given, and the number of input items does not match the number of output file names, copying proceeds until one of the lists is exhausted, and then a warning message is printed on the terminal.

When a list of snumbs or edit names is given, the input file is read sequentially and the list is searched for a match with each snumb or edit name in the input file to determine whether the job or deck should be copied. Thus the order of items in the output file is the same as their order in the input file, and the order of the list is not significant. However, a list of input (or output) file names is read (or written) in the order in which they appear in the list.

Alphabetic BCD input is translated to lowercase ASCII. Arguments containing edit names, tape labels, etc., must be typed in lowercase in order to compare with corresponding BCD names from the input file.

Examples

To convert the Multics ASCII segment named readable.jcl into a GCOS BCD card image file named GJOB utilizing canonicalization and tab stops at 8, 16, and 32, enter:

```
gcu readable.jcl -ascii -can -tabs 8 16 32 -out GJOB -gcos_bcd
```

To convert the GCOS TSS formatted file named SPCWRS into a Multics ASCII segment, enter:

```
gcu SPCWRS -gcos space_wars.fortran
```

To extract the source program named TSSA from the GCOS 800-bpi tape U8421 which is a library of COMDKs (a K* tape), enter:

```
gcu -in -tape u8421 -comdk -lib tssa -out tssa.ascii
```

To convert the GCOS BCD file named pswdlst.gcos to a Multics ASCII text file, enter:

```
gcu pswdlst.gcos password.list.ascii
```

gcos_card_utility

To copy a BCD file contained in the GCOS 1600-bpi tape identified as 12DO to a Multics ASCII text file called gcos_manuals, enter:

```
gcu -input -tape 12do,den=1600 -output gcos_manuals -ascii
```

gcos_create_file

Name: gcos_create_file, gcf

The gcos_create_file command is used to create files that are compatible with GCOS random files.

Usage

gcf filename {-control_arg}

where:

1. filename
   is the Multics pathname of the GCOS file.

2. control_arg
   may be either -links N or -llinks N (where N is the file size in either links or llinks). (Default is -llinks 1)

Note

This command is used when a GCOS job (or activity) requires space in the form of a random file. The gcos_create_file command must be used to create the random file before the GCOS fob is executed.

Name: gcos_fms, gfms

The gcos_fms command unloads files from a GCOS MASTER or USER SAVE tape. This provides a convenient means of transporting files to be used with the Multics GCOS simulator (gcos and gtss comands) from a GCOS environment. Multics directory hierarchy is created, along with any segments or multi-segment files, to contain the GCOS catalog structure.

Usage

gfmc reel_number {-control_args}

where:

1. reel_number
   is the tape reel identifier for the input save tape and is in the form of a string of alphanumeric characters (e.g., AAAAA).

2. control_args
   can be chosen from the following:

   -control_file path, -cf path
   path specifies the pathname of a Multics segment containing a list of (only) those GCOS user names whose files are to be unloaded. The contents of this segment are in free-form format as:

   restorelist: user_name1 {, user_name2}... ;

   -density N, -den N
   N specifies tape density as 200, 556, 800, 1600, or 6250.

   -directory_mapping rule_name, -drm rule_name
   establishes the directory mapping rule for the restore process. The rule_name should be entered as "umc", "smc", or "wd". If smc is selected, then the control_arg "-smc_pathname path" must also be supplied. (See "Notes" below.) (Default is wd)

   -no_acl
   prevents the ACL for the segment containing the GCOS files from being determined from the permission information on the SAVE TAPE. This ACL information is usually unrelated to Multics ACLs.

   -restore umc_name, -rest umc_name
   restores GCOS catalog structure identified by umc_name (user master catalog name). The default is obtained from the catalog on the tape. (See "Notes" below.)

   -smc_pathname path, -spn path
   establishes the pathname for smc directory mapping if the -directory_mapping smc argument is supplied.

   -track N, -tk N
   N is either "7" or "9" to specify 7- or 9-track tape. (If the track argument is not included, the system Defaults to 9-track.)

Notes

If the SAVE tape format is unknown, an invocation of gcos_fms reel_number attempts to determine tape density and the SAVE tape (MASTER or USER) USER_ID, and uses the current working directory as the default area in which to unload the contents of the specified SAVE tape. Error messages are output if the tape cannot be processed.

The restored structure is placed in the Multics storage system depending on the option given with the -directory_mapping argument. If the mapping rule is umc, the initial catalog (UMC) is placed beneath the directory ">udd>umc_name", where umc_name represents the name given after the -restore argument. Thus all subordinate structures are placed under the directory ">udd>umc_name>umc_name". If the mapping rule is smc, the UMC catalog is placed immediately under the directory named with the -smc_pathname argument. If the mapping rule is wd, the current working directory is taken to be synonymous with the UMC catalog. All subordinate structure is placed below the working directory.

The user process must have sma access to the directory below which the restore is to be done. For a restore with mapping rule "umc", this usually requires the cooperation of the System Administrator to either create the directory >udd>user_name, or to add the name user_name to an existing project.

The gcos_fms program does not currently restore permissions or other individual attributes of catalogs or files.

If multiple occurrences of any of the control arguments -track, -restore, -directory_mapping, or -smc_pathname appear in the same command line, the last occurrence of the respective control argument is used.

Attempting to unload a very large number of files may cause the known segment table (KST) to overflow. Three choices of action are available should this condition occur: request the project administrator to increase the size of the KST; break the request down into several partial saves of GCOS files (DOONLYs or catalog saves); have the GCOS site administrator restore the GCOS userid on the GCOS system from a MASTER SAVE tape and re-create the save tape.

gcos_label_tape

Name: gcos_label_tape, gclt

The gcos_label_tape command writes a GCOS label on a tape.

Usage

gclt reel_num {new_num} {-control_args}

where:

1. reel_num
   is the number on the label of the tape reel to be used.

2. new_num
   is the new serial number to be written on the tape. If it is the same as the reel_num, it may be omitted.

3. control_args
   can be chosen from the following:

   -density N, -den N
   where N specifies the tape density. (Default is 1600 bpi)

   -erase
   erases the tape before labeling it. Default is to overwrite the old label (if it exists), and leave the remaining data intact.

   -no_erase
   does not erase the tape before labeling it. (Default)

   -track N, -tk N
   where N specifies 7- or 9-track tape. (Default is 9-track)

Notes

If no control arguments are given, the command:

gclt xyz12

is equivalent to the command:

gclt xyz12 xyz12 -tk 9 -den 1600 -no_erase

If conflicting control arguments are given, the rightmost control argument is used (e.g., "gclt m1266 -tk 7 -tk 9" results in a 9-track tape label.)

Name: gcos_set_environment, gse

This command may be given in the user's process before the gcos_tss command is invoked, to modify the user-controlled run-time parameters for GTSS.

Usage

gse {-control_args}

where control_args may be chosen from the following control arguments:

-directory_mapping mapping_rule, -drm mapping_rule
sets the directory mapping to use the specified mapping_rule. The following mapping rules are allowed: umc, smc, and wd. The default is wd. See Section 1 for a discussion on "Mapping File Strings to Pathnames."

-gcos_debug_pathname path, -gdbpn path
sets the path of the control file to be used when the gcos_debug command is invoked. (If path is not specified a Default of >udd>Project_id>Person_id>Person_id.gdb is used.)

-modes modes_string
sets the modes, according to modes_string. Mode names are separated by commas, each one optionally preceded by a circumflex (^) to turn the specified mode off. Modes not specified in modes_string are left unchanged.

ast
prints the asterisk for each input request in GTSS build mode. When off, no prompt character is given. (Default is on)

drl
prints trace information on each derail. The derail name and its location are printed. (Default is off)

gdb
gcos_debug breaks are inserted in slave object code as it is loaded. The gcos_debug command is invoked when such a break is encountered during execution. (Default is off)

mcmd
allows the use of the e request at GTSS command level. When allowed, the remainder of the input line is passed on to the Multics command processor. (Default is off)

mquit
GTSS goes to a new Multics command level in response to the quit (or break) key. When off, depressing the quit key on the terminal resets the current GCOS time sharing subsystem function and returns to GTSS command level. (Default is off)

ss
prints trace information about each subsystem call. The subsystem name is printed. (Default is off)

-print, -pr
displays on the user_output switch the current control information set by this command. If there are multiple control arguments, -print is the last one processed.

-reset, -rs
sets all modes to the default.

-smc_pathname path, -spn path
sets path as the root directory to be used when catalog/file strings are mapped into Multics pathnames using the system master catalog (SMC) directory mapping rule. This control argument is required when -directory_mapping smc is specified.

-umc_name name, -unm name
sets the user master catalog (UMC) name to be used for catalog/file string operations where UMC name is only implied. Name may be given in upper or lowercase, but otherwise is equivalent to the user's GCOS login USERID. This control argument is required when -directory_mapping umc or -directory_mapping smc is specified. The UMC name is mapped to lowercase for use in constructing Multics pathnames.

Notes

The user must have at least status access to the directory implied by the directory mapping_rule and to its containing directory.

If any error is detected by the gcos_set_environment command, no changes are made to any of the user-controlled runtime parameters.

gcos_sysprint

Name: gcos_sysprint, gsp

The gcos_sysprint command converts a print file (either SYSOUT or simulated printer) produced by the GCOS environment simulator from GCOS ASCII or BCD to Multics ASCII for printing or viewing on a terminal. This command is called automatically by the simulator to convert any print files produced by the job, unless the -hold option is given on the gcos command line. It can also be called by the user to convert a file that was not converted automatically.

Usage

gsp input_path {output_path} {-control_arg}

where:

1. input_path
   is the pathname of a print file produced by the simulator.

2. output_path
   is the pathname of a file into which the ASCII output lines are to be written. If the file already exists, it is overwritten with no warning. If output_path is omitted, the lines are displayed on the user's terminal as they are converted.

3. control_arg

   -lower_case, -lc
   translates alphabetic characters to lowercase ASCII. (Default is uppercase)

*

Notes

The star and equal conventions are not implemented in this command.

If the first record in the file is the SYSOUT header record written by the simulator, then the records are ordered by report code (within each activity) before being printed, and all records from the execution report are printed first. Otherwise the records are printed in the order in which they appear in the input file, with no sorting or reordering of any kind.

The records are assumed to be GCOS print line images, ending with GCOS printer control characters. These latter are converted to ASCII newline and newpage characters. To convert a BCD card image file to ASCII, with newline characters being added to the end of each line, use the gcos_card_utility command.

gcos_syspunch

Name: gcos_syspunch, gspn

The gcos_syspunch command converts a GCOS standard system format file containing BCD and binary card images to a format suitable for punching using the Multics dpunch command with the -raw argument. This command is called automatically by the GCOS environment simulator to convert any punch files produced by the job, unless the -hold argument is given on the gcos command line. It can also be called by the user to convert any GCOS card image file for dpunching.

Usage

gspn path

where path is the pathname of the card image file to be converted.

Notes

The output is written into a segment in the working directory whose entryname consists of the entryname portion of path plus the suffix raw. If this segment already exists, it will be overwritten with no warning.

The star convention is not implemented in this command.

The conversion produces a 960-bit string for each input card image. Each 12-bit string represents one of the 80 card columns, with a 1-bit indicating that a hole should be punched in the corresponding row and column. This conversion is such that, when the output file is punched using the dpunch -raw command, the result is valid GCOS BCD or binary cards containing the same information that was in the records of the input file. *

gcos_tss

Name: gcos_tss, gtss

The gcos_tss command invokes the GTSS to run a single GCOS time sharing user in the user's process.

Usage

gtss

Notes

All parameters controlling the operation of gcos_tss are set with the gcos_set_environment command. If gtss is invoked with no prior call to gcos_set_environment, the following values are assumed:

-drm wd -modes ast,^drl,^gbd,^mcmd,^mquit,^ss

The gtss SAVE command will not set the bit counts on segments created via this command, since they are not needed by the simulator. If, however, the user wishes to manipulate the files via programs external to the simulator (such as gcos_card_utility), the bit counts must first be set modulo 320 words.

The gtss command may not be invoked while the caller is using window_call (video mode). The GTSS subsystems cannot handle the special I/O required to support this terminal interface.

SECTION 4

MASTER MODE ENTRIES

This section describes the differences in MME processing between native GCOS and the GCOS environment simulator. Complete documentation of the GCOS MMEs can be found in the General Comprehensive Operating Supervisor (GCOS) manual.

OVERVIEW OF MME PROCESSING

Every attempt is made by the simulator to process MMEs in the same manner as native GCOS. Since the level 68 hardware is a superset of the level 66 hardware, the hardware mechanisms for MME processing are identical. The only differences that may exist, therefore, are due to features of some MMEs that are not yet supported by the simulator.

Whenever a MME (op code = 001 octal) is executed by the slave job, a fault is signalled in the user ring. The simulator, before starting an activity execution, sets up a static handler for processing this fault. When the fault is signalled, the simulator's MME processor is called and simulation of native GCOS actions begin.

Once the MME processor is in control, it extracts the address from the faulting instruction and uses it to index a table of MME processing subroutines. The appropriate subroutine is called and the MME simulation is carried out. When this simulation is completed, control returns to the MME processor which restarts the slave job.

MME DIFFERENCES

.EMM - is not supported

GEBORT - is supported
including the storing of appropriate data in the slave prefix. Also, due to the fact that all I/O is performed time-sequential, no user I/O is lost.

GECALL - is supported
as if the simulated GCOS system had no production library.

GECHEK - is supported as qualified.
processed as if the requested checkpoint dump was successful. However, no checkpoint dump is written to the specified file.

GEENDC - is supported

GEFADD - is supported as qualified
1. All physical device addresses are manufactured by the GCOS environment.
2. ASCII and 51-column capabilities are not provided for card punches.
3. ASCII capability is not available for printers.
4. All fields are returned for mass storage files and tapes.

GEFCON - is supported
notice that the physical device address is a simulated value as with the GEFADD MME.

GEFILS - is not supported
execution of this MME causes the slave activity to abort.

GEFINI - is supported

GEFRCE - is supported as qualified
the undocumented interfaces for UFAS are not supported.

GEFSYE - is supported as qualified
functions: 2 catalog create; 3 file create; 9 file release; and 22 file purge.

GEIDSE - is supported as qualified
execution of this MME is allowed for the I-D-S/I open function. The open information always indicates that disk accounting is used and that no special FMS functions are provided.

GEINFO - is supported as qualified
1. SSA copying of the first two words only is supported.
2. Requesting a program number for a SNUMB other than the current job's SNUMB causes an abort of the slave program.

GEINOS - is supported as qualified
1. The intercom facility is not supported;

2. The following list of I/O commands are not supported:

write printer (WPR, WPRC)
write printer ASCII (WPRASC, WPRCAS, WPREAS, WPRECA)
reread tape decimal (RRTD)
reread tape binary (RRTB)
set file protect (SFP)
read EBCDIC (RTE)
read ASCII/EBCDIC (RTAE)
read ASCII (RTA)
write EBCDIC (WTE)
write ASCII/EBCDIC (WTAE)
write ASCII (WTA)
set file permit (SFPR)
tape load (TPLD)
write drum and verify (WDRV)
drum compare and verify (DRCV)
write disk - continuous and verify (SDICV)
compare and verify (DICV)
write disk - continuous and compare (WDICV)
write disk ASCII (WDASC)
read disk ASCII (RDASC)
read disk nonstandard size (RDNSZ)

GELAPS - is supported

GELBAR - is supported

GELOOP - is not supported
execution of this MME causes the slave activity to abort.

GEMORE - is supported
see "Mapping of file Strings to Pathnames" in Section 1 for a description of catalog/filename to pathname mapping.

GEMREL - is supported

GENEWS - is not supported
execution of this MME causes the slave activity to abort.

GEPRIO - is ignored
no special action is necessary due to the time-sequential operation of all I/O simulation.

GERELC - is ignored
no special action is necessary due to the time-sequential operation of all I/O simulation.

GERELS - is supported

GERETS - is supported

GEROAD - is ignored
no special action is necessary due to the time-sequential operation of all I/O simulation.

GEROLL - is not supported
execution of this MME causes the slave activity to abort.

GEROUT - is supported as qualified
The following operations are implemented:

3 (octal)	-	direct access output
4 (octal)	-	direct access output, then input
5 (octal)	-	user program inquiry to terminal
6 (octal)	-	program requests terminal type
17 (octal)	-	program requests line disconnect
20 (octal)	-	direct access current line status

All operations not implemented fall into the category of paper tape manipulation, line switching, extended output length, or BTT load/start operations.

Output is written to the user_output stream and input is read from the user_input stream. Data is translated to and from the Multics standard data format. This allows the stream names to be associated with any type of device. The edited I/O mode is used for output to suppress escape printing.

A terminal break is simulated by a QUIT (user depresses the appropriate key on the terminal: QUIT, BREAK, ATTN, etc.) followed by the program_interrupt command.

Direct access is established (if not already) on any correct call with operation code 3, 4, or 5.

GERSTR - is supported as qualified
either native GCOS or GCOS environment simulator system format disk files may be used. Tape files are not supported.

GESAVE - is supported as qualified
the format of the system file produced is not the same as that produced by the GESAVE MME in native GCOS. The gcos_tss command cannot read these files, it requires files created by native GCOS, the System Editor ($SYSEDIT), or some TSS subsystem in gtss.

GESECR - is not supported

GESETS - is supported

GESNAP - is supported

GESNUM - is not supported
execution of this MME causes the slave activity to abort.

GESPEC - is not supported
execution of this MME causes the slave activity to abort.

GESYOT - is supported as qualified
requests for backdoor file processing are denied as if no backdoor file were configured.

GETIME - is supported

GEUSER - is supported as qualified
execution of this MME causes a search using the Multics search rules for a subroutine mme_geuser. If found, this subroutine is called in the same manner as any of the MME processing subroutines of the GCOS environment simulator. If the subroutine mme_geuser is not found, the slave activity is aborted.

GEWAKE - is not supported

GEXLIT - is not supported

GMODER - is not supported

GMODES - is not supported

SECTION 5

CONTROL CARDS

This section describes the differences in control card processing between native GCOS and the GCOS environment simulator. Complete documentation of the GCOS control cards can be found in the Control Cards Reference Manual.

OVERVIEW OF CONTROL CARD PROCESSING

The simulator attempts to process valid job decks in the same manner as native GCOS. However, the internal functioning of the simulator differs from that of GCOS and some features of GCOS, and responses to incorrect jobs may not be the same. To assist users in coping with such differences, the card processing algorithm of the simulator is described in the following paragraphs.

The simulator classifies cards into the following categories:

nondollar (data) cards
: do not have a $ in column 1 and are never interpreted by the simulator. These cards usually are written to a file.

dollar cards
: have a $ in column 1 and are completely interpreted by the simulator. These cards also may be written to a file (e.g., file cards being written to the R* file in an execute activity).

non-GCOS dollar cards
: have a $ in column 1, but are not interpreted by the simulator. The cards are written to a file for interpretation by the slave program. These cards include the general loader, BMC, utility, and editor directive cards.

  Some of the non-GCOS dollar cards cause the simulator to take some action in addition to writing the card to a file. However, this action does not involve examining the option field of the card.

Notice that the descriptions of the non-GCOS dollar cards correspond to those in the supported release of native GCOS, while those of other dollar cards correspond to the supported release of native GCOS except where noted otherwise in the text of this section. This is because the non-GCOS dollar cards are interpreted by native GCOS slave software taken directly from the GCOS total system tape, while the other dollar cards are interpreted by the simulator.

For each activity delimiting card, there is a default nondollar file code. A group of nondollar cards encountered in the activity and not preceded by a $ DATA card is written using this default file code provided the default file code is not specified explicitly within the activity. (If the default file code is defined explicitly the cards are discarded and a single warning message is issued.) The first dollar card terminates the group of nondollar cards and subsequent cards are not written to the file.

Several dollar cards cause the nondollar file code to be set to a value other than the default. Among these are the $ DATA, $ OBJECT, $ DUMP, and $ UPDATE cards. Again, the first dollar card terminates the group of nondollar cards. Exceptions to this are the $ DATA card with the COPY option, which must be terminated by a $ ENDCOPY card, and the $ UPDATE card, which is terminated by the first dollar card that is other than a $ ALTER card.

The simulator refuses to write cards from the input deck onto a file that already exists in the activity (with the exception of writing file cards to the R* file in an execution activity). Such cards are discarded and a warning message is issued.

If a file code is used more than once in an activity (i.e., on several file cards), the last file card encountered is used. When a duplicate file code is encountered, the previous file is closed and the new one is opened. Thus, any errors on the earlier file card prevent the job from running, even though the earlier card was replaced.

Dollar cards that are completely unknown (stranger) to the simulator are printed on the execution report. If the -brief control argument is not given on the command line, these cards also are printed on the user's console during the first pass of input media conversion.

Dollar cards that are known to be valid GCOS control cards, but which are not supported by the simulator, cause a nonfatal error. Execution continues (the card is treated as an unknown card) after a nonfatal error only if the -continue control argument is given on the command line. If this argument is not given, the job is aborted after a nonfatal error.

Cards that are known to be BMC, utility, or editor directives that are encountered in the job deck in the wrong context cause a nonfatal error.

Errors in the operand field of a dollar card (including use of unsupported options) cause a fatal error. The job is always aborted after a fatal error. Errors of this type are detected during the allocation that precedes the activity and not before any activities have been run (as is the case in native GCOS).

CONTROL CARD DIFFERENCES

$ ABORT - is supported

$ ALGOL - is supported

$ ALTER - is supported as qualified
is processed by slave software, but if control card is encountered out of context, a warning is issued on the execution report.

$ ASCII - is not supported

$ ASSEM - is supported as qualified
is processed by slave software, but if control card is encountered out of context, a warning is issued on the execution report.

$ BREAK - is supported

$ CBL74 - is supported

$ CHANGE - is supported as qualified
is processed by slave software, but if control card is encountered out of context, a warning is issued on the execution report.

$ CIDS2 - is supported

$ COBOL - is supported

$ COMMENT - is supported as qualified
the message text is displayed on the user's terminal and not on the operator's console.

$ COMPILE - is supported as qualified
is processed by slave software, but if control card is encountered out of context, a warning is issued on the execution report.

$ CONVER - is supported

$ COPY - is supported as qualified
is processed by slave software, but if control card is encountered out of context, a warning is issued on the execution report.

$ DAC - is supported
processed as if it were a $ SYSOUT control card.

$ DATA - is supported as qualified
1. The saving of data files is not supported.
2. The TAKEc, SEQ, CKSEQ, NSEQ, GE225, and IBMPL options are not supported.
3. File concatenation is not supported.

$ DELETE - is supported as qualified
is processed by slave software, but if control card is encountered out of context, a warning is issued on the execution report.

$ DKEND - is supported

$ DUMMY - is supported as qualified
is processed by slave software, but if control card is encountered out of context, a warning is issued on the execution report.

$ DUMP - is supported

$ ENDCOPY - is supported as qualified
is processed by slave software, but if control card is encountered out of context, a warning is issued on the execution report.

$ ENDEDIT - is supported as qualified
is processed by slave software, but if control card is encountered out of context, a warning is issued on the execution report.

$ ENDJOB - is supported

$ ENDLD - is supported as qualified
is processed by slave software, but if control card is encountered out of context, a warning is issued on the execution report.

$ ENTRY - is supported

$ ENX - is not supported

***EOF - is supported

$ EQUATE - is supported

$ ETC - is supported as qualified
is processed by slave software, but if control card is encountered out of context, a warning is issued on the execution report.

$ EXECUTE - is supported

$ EXTEDIT - is not supported

$ EXTEND - is not supported

$ FFILE - is supported as qualified
is processed by slave software, but if control card is encountered out of context, a warning is issued on the execution report.

$ FILE -is supported as qualified
1. Since all mass storage I/O is simulated using Multics segments, device type options on the $ FILE card are ignored.
2. Creation of cataloged files or access to cataloged files via $ FILE is not supported.
3. File concatenation is not supported.

$ FILEDIT - is supported

$ FILGP - is not supported

$ FILSYS - is not supported

$ FORM - is supported as qualified
is processed by slave software, but if control card is encountered out of context, a warning is issued on the execution report.

$ FORTRAN - is supported

$ FORTY - is supported

$ FORT77 - is supported

$ FUTIL - is supported as qualified
is processed by slave software, but if control card is encountered out of context, a warning is issued on the execution report.

$ GET - is supported as qualified
is processed by slave software, but if control card is encountered out of context, a warning is issued on the execution report.

$ GETRWD - is supported

$ GMAP - is supported

$ GOTO - is supported

$ IDENT - is supported as qualified
no restrictions are enforced on the data in columns 16 through 72 and no accounting information is taken from the control card. Jobs submitted through the GCOS daemon must have the account_id registered. (See Appendix A for additional details.)

$ IDS - is supported

$ IDS2 - is supported as qualified
1. Concurrent access control is provided on a file basis for read and write access types for any IDS2 data base file initialized via $ PROGRAM Q2UTIL.
2. A lock control segment with the .CONCURRENCY suffix is created in the same directory as the IDS2 data base and must not be altered if concurrent access control is to be provided.

$ IF - is supported

$ INCLUDE - is supported as qualified
is processed by slave software, but if control card is encountered out of context, a warning is issued on the execution report.

$ INCODE - is supported as qualified
the GE225 and IBMPL options are not supported.

$ INPUT - is supported as qualified
is processed by slave software, but if control card is encountered out of context, a warning is issued on the execution report.

$ JOVIAL - is supported

$ Label - is supported

$ LIBRARY - is supported

$ LIMITS - is supported as qualified
1. A "P" preceding the print lines specification is ignored.
2. The I/O time field is not processed.
3. If NLSTOU is specified on the $ GMAP card, the default line count is allocated rather than 2500.

$ LINK - is supported

$ LIST - is supported as qualified
is processed by slave software, but if control card is encountered out of context, a warning is issued on the execution report.

$ LODLIB - is not supported

$ LOWLOAD - is supported

$ MODIFY - is supported as qualified
is processed by slave software, but if control card is encountered out of context, a warning is issued on the execution report.

$ MSG1, $ MSG2, $ MSG3, - is supported as qualified
the message text is displayed on the user's terminal and not on the operator's console.

$ MULTI - is supported as qualified
is processed by slave software, but if control card is encountered out of context, a warning is issued on the execution report.

$ NEED - is supported as qualified
the SYS/n option is not allowed.

$ NLOAD - is supported

$ NOLIB - is supported

$ NTAPE - is not supported

$ OBJECT - is supported

$ OBJLIB - is not supported

$ OPTION - is supported

$ OUTPUT - is supported as qualified
is processed by slave software, but if control card is encountered out of context, a warning is issued on the execution report.

$ PARAM - is supported
the GCOS environment simulator support of the $ PARAM card allows some extensions to the GCOS support. These include up to 32 parameters (as opposed to 9 in GCOS), and "stranger options" (refer to General Comprehensive Operating Supervisor (GCOS) manual, Section 2). Parameters can also be

specified by using the -parameter control argument to the gcos command described in Section 3.

$ PATCH - is supported as qualified
is processed by slave software, but if control card is encountered out of context, a warning is issued on the execution report.

$ PL1 - is supported

$ PPS - is not supported

$ PPSRPT - is not supported

$ PPTP - is not supported

$ PPTR - is not supported

$ PRINT - is supported as qualified
it does not allocate a printer. Instead, an output collection file is used and is then processed as any SYSOUT file.

$ PRIVITY - is not supported

$ PRMFL - is supported as qualified
1. rec, r/c, w/c, t, t/c, c, and r/w/c permissions are not supported.
2. The following extensions are included:
   a. The $ USERID card is only necessary for jobs submitted via the GCOS daemon.
   b. A Multics lowercase pathname may be used rather than a GCOS catalog/filename string on the $ PRMFL card. The file string field is recognized as a GCOS catalog/filename string only if it contains a slash (/) and does not contain either a less-than (<) or greater-than (>) symbol. If the file string is a Multics pathname, a greater-than (>) symbol can be used to indicate that a $ ETC card follows. Also $ PARAM argument substitutions are not made in Multics pathnames.
3. Notice that the contents of the referenced file are assumed to be in GCOS format (not Multics ASCII) and that the -ascii argument in the comment field is not supported.
4. Refer to the gcos_create_file command as regards random file creation.
5. File concatenation is not supported.

$ PROGRAM - is supported as qualified
the EXTEND option however, is ignored if present.

$ PSM - is not supported

$ PUNCH (FILEDIT) - is supported as qualified
is processed by slave software, but if this control card is encountered out of context, a warning is issued on the execution report of the user's job and is interpreted as a request for allocation of a card punch.

$ PUNCH (GCOS) - is supported as qualified
the punch type field is ignored. Notice that punched output is automatically directed to a Multics segment for the user.

$ QUTIL - is supported as qualified
is processed by slave software, but if control card is encountered out of context, a warning is issued on the execution report.

$ READ - is not supported

$ RELADD - is not supported

$ RELCOM - is supported

$ REMOTE - is supported as qualified
all processing is as if the card were a $ SYSOUT control card.

$ REPORT - is not supported

$ REPTL - is not supported

$ REPTR - is not supported

$ RPG2 - is supported

$ S2PROG - is not supported

$ SELECT - is supported
the comment field (following the catalog/filename string) can contain the string -ascii to specify that the file being selected is a Multics ASCII segment.

$ SELECTD - is supported
the comment field (following the catalog/filename string) can contain the string -ascii to specify that the file being selected is a Multics ASCII segment.

$ SEQ - is supported as qualified
is processed by slave software, but if control card is encountered out of context, a warning is issued on the execution report.

$ SET - is supported

$ SETSQ - is supported as qualified
is processed by slave software, but if control card is encountered out of context, a warning is issued on the execution report.

$ SETSQ1 - is supported as qualified
is processed by slave software, but if control card is encountered out of context, a warning is issued on the execution report.

$ SNUMB - is supported
the urgency level is ignored except when processed by the GCOS daemon.

$ SOURCE - is supported

$ SRCLIB - is not supported

$ SYSEDIT - is supported as qualified
since the execution unit created is the same as native GCOS, SYSEDIT may not be used to create a Q* file from a file which is created in part by MME GESAVE.

$ SYSLD - is supported as qualified
is processed by slave software, but if control card is encountered out of context, a warning is issued on the execution report.

$ SYSNAME - is not supported

$ SYSOUT - is supported as qualified
the destination field is ignored.

$ TAPE, $ TAPE7, $ TAPE9 - is supported as qualified
1. Multireel tapes are currently not supported by the GCOS environment simulator; therefore the multireel indicator is ignored.
2. The class field is ignored.
3. File concatenation is not supported.
4. The tape mount message issued to the operator includes a request for a write ring unless the card contains "-noring" or -nr" as a comment. For example:

```
$ tape9 in,xld,,12345,,master_data,,den16 -nr
```

requests the operator to mount tape 12345 with no write ring inserted.

$ TAPE27, $ TAPE29 - is not supported

$ TYPE - is supported

$ UPDATE - is supported

$ USE - is supported

$ USERID - is supported as qualified
since Multics security is in effect for all GCOS environment simulator jobs, this control card is ignored, except for jobs submitted through the GCOS daemon.

$ UTILITY - is supported

$ UTL2 - is supported

$ WHEN - is supported

$ xxxPK - is not supported

$ 355MAP - is supported

$ 355SIM - is supported

SECTION 6

GCOS TIME SHARING ENVIRONMENT SIMULATOR

This section describes the GCOS time sharing environment simulator on Multics (GTSS). GTSS is intended to provide most of the functions available to a slave user of GCOS time sharing.

The gcos_tss command is a Multics command, invoked from Multics command level. It establishes the environment for the GCOS time sharing slave software and is executed as is (just as though it were executing on a native GCOS system).

The GTSS interfaces are nearly identical to those provided by native GCOS. Thus, the user can "log in" to GTSS by invoking the gcos_tss command and proceed as if actually logged in to native GCOS time sharing. There are some differences in the specific terminal interfaces and also in the file system. These are described later in this section.

The GTSS executive functions are performed by Multics programs, written to simulate the same functions provided by the native GCOS time sharing executive. Initially, the user is in GTSS command level, or build mode. When a command is entered, the executive calls in the GCOS subsystem to perform that function.

This is done by setting the object programs into execution in GCOS BAR mode. When the slave program executes a derail instruction for services by the GCOS time sharing executive, the derail is caught instead by the Multics operating system and passed to GTSS. GTSS interprets the derail request and simulates the same functions of the GCOS time sharing executive.

Although some derail functions are not implemented, most functions related to the building, editing, and compilation of a user's *SRC current file, with subsequent execution of the resulting object code, are provided.

GTSS SOFTWARE FILES

The GCOS software used by gtss resides in the gtss_fast_library_ file in the same directory as gtss. New or replacement GCOS software is added to this file with the gcos_tss_build_library command. When loading subsystems for execution, gtss searches the run-time subroutine libraries gtss_Lstar_ and gtss_starL_in the same directory. This last file is initially empty and can be loaded with site-specific subroutines with the gcos_reformat_syslib command. Links to these software files may replace the actual files in >unb.

Programs and commands intended to be loaded from the GCOS libraries CMDLIB and LIBRARY may be found in >udd>cmdlib>cmdlib and >udd>library>library. Files in these directories may be restored from a GCOS FILSYS save tape via the gfms command.

GTSS TERMINAL INTERFACE

GTSS uses existing Multics terminal interfaces wherever possible. It is felt that the GCOS time sharing interface is too restrictive (no echoplex, full duplex, canonicalization, type-ahead, etc.), and that users are willing and able to learn the Multics interfaces in order to get the Multics functions.

The following paragraphs describe the steps to be taken by a user wanting to use GTSS:

- User logs into Multics

  The user logs into Multics as a normal Multics user by invoking the login (1) command. As such the user is subject to normal Multics answering service controls as applied by the system and project administrators.

- GTSS is called

  To enter the GTSS facility, the user types the gcos_tss command.

  When the user types BYE under gcos_tss, the user is returned to Multics command level. For those individual users who do not plan to use the Multics command functions, a simple abbrev/exec_com can be set up to invoke GTSS and then automatically logout the user after a BYE.

- User ID and password input

  There is no additional password required of the user. Under certain modes of pathname mapping, the user must give the GCOS time sharing USERID. This value is used in the mapping of GCOS pathnames into Multics pathnames for applications brought over from native GCOS.

- Break key (functions normally, as on native GCOS)

  The use of the break key is processed the same as on native GCOS TSS, where it causes the currently executing subsystem/command to be reset. An option is provided for gcos_tss to override this and cause the user's process to go to Multics command level. See the description of the gcos_tss command (described later in this section) for additional details.

- System responds with line feed after each carriage return

  On native GCOS, the user types a CR to indicate the end of the input line. GCOS time sharing generally prints an NL and asterisk to indicate that it is ready to accept the next input line. On Multics, GTSS uses the standard terminal interface for line feed/carriage return processing. When GTSS receives another line of input (e.g., build mode), it outputs an asterisk without a line feed or carriage return to indicate that it is ready for another line of input.

- GCOS erase and kill characters are provided (Multics set_tty command option)

  The GCOS time sharing erase and kill characters are not set by the gcos_tss command. The standard Multics erase and kill characters are used to facilitate growth into native Multics. For those users who need the GCOS version, the set_tty command may be used to set these values. The system does not respond with "DEL" upon receipt of a line delete.

- Terminal input/output is like GCOS (with few exceptions)

  GTSS provides (if possible) additional detail for the user when reporting errors.

- Processing input lines with trailing white space

  Normally, Multics discards all trailing white space (blanks and horizontal tabs) on input from the terminal. This is incompatible with the input of trailing white space on native GCOS. The GTSS user has two options available to input lines with trailing white space:

  - Another character can be used in place of the trailing blank or horizontal tab. That character can then be replaced using the text editor.
  - The user can turn off canonicalization of the input lines (native GCOS does not provide this function) with the set_tty command.

Terminal Interface Differences

The following paragraphs describe the user terminal interface differences between the GTSS and GCOS time sharing.

USER SIGN-ON PROCEDURE

The sign-on procedure and system greeting message pertaining to Multics login (1) are described in the New Users' Introduction to Multics - Part I , (Order No. CH24).

The GCOS time sharing login procedure is described in the Time Sharing System Reference Manual, (Order No. DJ31).

INPUT LINE TRANSMISSION CONVENTION

The GCOS time sharing convention to indicate completion of a typed input line transmission is a carriage return (an ASCII RETURN character -- octal code 015).

The GTSS convention to indicate the completion of a typed input line is either the ASCII LINE FEED character (octal code 012), or a carriage return. The default is newline, but can be changed with the set_tty command to be the carriage return (set_tty -modes lfecho).

EDITING CONVENTION

Two editing capabilities on the typed line are available:

- The ability to delete the last character or characters entered
- The ability to delete the entire line

GTSS character-deletion and line-deletion control characters conform to the Multics editing convention. The two editing control characters for teleprinters (IBM 2741, and DATEL terminals) are:

Character	control function
# (number sign)	character deletion
@ (commercial at sign)	line deletion

The GTSS editing rules are:

1. Use of the character-delete control deletes from the current buffer line the character preceding the deletion character; use of n consecutive deletion characters deletes n preceding characters (including blanks) up to the beginning of the line. Although the delete character is a printable symbol, it is not retained as part of the line string.

   One character-delete symbol typed immediately after "white space" causes the entire white space to be erased. (White space is defined as any combination of spaces and horizontal tabs).

   The benefits of the white space concept are:

   a. Reduces the number of keystrokes necessary to remove white space.

   b. Eliminates the need for a user to remember how many spaces or horizontal tabs are typed on a line.

      In the following examples an S indicates a space and a T stands for horizontal tab:

      TheSSne###next

      or

      TheST#next

      produces "Thenext". As shown in this example, the user may have to reinsert an S to space the line to read "The next", if that is the desired result.

2. Use of the line-delete control deletes the entire line.

   For example:

   This is atekt@What is this

   produces:

   What is this

The Multics user terminal interface provides the user with the ability to define the characteristics and modes of a specific terminal associated with terminal input/output by using the set_tty command. With this command, the user can set various modes to effect certain terminal actions, e.g., specify the character-delete and line-delete symbols or "echo" a carriage return and/or line feed. For a detailed description of the set_tty command, (see the Commands manual).

ESCAPE CHARACTER CONVENTION

GTSS conforms to the established character escape convention of Multics, represented by the left slant (\). Escape conventions are provided for terminals

that do not have a full ASCII character set (see the Programmer's Reference manual).

USER PROGRAM INTERFACE

For those implemented derail functions, the slave programs experience the same interface as they do on GCOS time sharing.

- Programs use normal derail to obtain TSS services

  The derail fault is caught by Multics and passed to GTSS. The machine conditions are examined to determine the nature and validity of the fault. Legitimate derail requests are processed, the IC is adjusted to pass over the derail calling arguments, and control is returned to the user.

- Privileged code not accommodated

  The effort to accommodate privileged code is not justified at this time.

- Existing user subsystem, linked object files usable as is

  Users can directly execute programs and access files brought over from GCOS time sharing (i.e., recompilation and linking not required).

SECTION 7

OVERVIEW OF DERAIL PROCESSING

This section describes the differences in derail processing between native GCOS and GTSS. Complete documentation of the GCOS derails can be found in the Time Sharing System Reference Manual, (Order No. DJ31).

Intentional faults, called derails (DRLs), are mechanisms used by GTSS to request service functions instead of the MME functions used in the batch processing environment.

Every attempt is made by GTSS to process DRLs in the same manner as the native GCOS TSS. Since the Level 68 hardware is a superset of the Level 66 hardware, the hardware mechanisms for processing DRLs are identical. Some features of native GCOS TSS DRLs are not supported by GTSS DRLs; these differences are noted below.

ABORT - is supported

ABTJOB - is supported

ADDMEM - is supported

ATTRI - is not supported

CALLSS - is supported as qualified
the Multics virtual storage stacking facility is utilized when the current (calling) subsystem must be "swapped out." Note that there is no equivalent of the native GCOS swap file; four stack levels are used instead.

CGROUT - is not supported

CONSOL - is not supported

CORFIL - is supported

DEFIL - is supported

DIO - is supported

DRLDSC - is not supported

DRLIMT - is supported as qualified
this implementation does not update the processor time limit for the subsystem in ".limit" of the user status table (UST).

DRLSAV - is supported

FILACT - is supported

FILSP - is supported

GROW - is supported

GWAKE - is supported
time delay is in seconds

JOUT - is not supported

JSTS - is supported

KIN - is supported

KOTNOW - is supported as qualified
DRL KOTNOW and DRL KOUT are condensed into one derail module. Note that the derail processor utilizes the tally words which are modified on return to the subsystem. Therefore, they must be refreshed prior to each execution.

KOUT - is supported as qualified
see KOTNOW

KOUTN - is supported as qualified
the new line code (octal 12) transmitted by the Multics input manager is replaced by a carriage return code (octal 15).

MORLNK - is supported

NEWUSER - is not supported

OBJTIM - is supported as qualified
memory limit is initialized at 261120.

PART - is supported

PASAFT - is supported

PASDES - is supported

PASFLR - is supported

PASUST - is supported

PDIO - is supported

PRGDES - is supported

PSEUDO - is supported

RELMEM - is supported

RESTOR - is supported

RETFIL - is supported

RETURN - is supported

REW - is supported

RSTSWH - is supported

SETLNO - is supported

SETSWH - is supported

SNUMB - is not supported

SPAWN - is not supported

STATUS - is not supported

STOPPT - is supported

STPSYS - is not supported

SWITCH - is supported

SYSRET - is supported

TAPEIN - is supported as qualified
LUCID mode (8-bit input) is not supported

TASK - is not supported

TERM - is supported

TERMTP - is supported

TIME - is supported

T.CFIO - is supported

T.CMOV - is supported

T.CONN - is not supported

T.CRYP - is not supported

T.ERR - is supported

T.EXEC - is not supported

T.GOTO - is supported

T.LINL - is supported

T.MAIL - is not supported

T.MODR - is not supported

T.MODS - is not supported

T.PAGE - is not supported

T.RSCC - is not supported

T.STAT - is not supported

T.SYOT - is not supported

USERID - is not supported

SECTION 8

GTSS COMMAND LANGUAGE

GTSS consists of a time sharing executive, a number of independent processing subsystems which operate under the executive, and a common command language. Complete documentation for the TSS system can be found in the Time Sharing System Reference Manual. The following list identifies some of the major TSS subsystems.

ABACUS
: a desk calculator facility featuring complex algebraic capabilities such as functions, summation operations, and remembered variables.

BASIC
: an algebraic-language compiler/executor designed for the user with numerical calculations involving relatively small quantities of data.

Text EDITOR
: a facility for building, maintaining, and reformatting (RUNOFF) text files.

TSS ALGOL
: an ALGOL subsystem that provides the TSS user with the capabilities of the ALGOL language.

TSS JOVIAL
: a JOVIAL subsystem that provides the TSS user with the capabilities of the JOBIAL language processor.

TSS FORT
: a time-sharing-based FORTRAN subsystem.

The following subsystems and subroutines provide service and utility functions for GTSS:

ACCESS
: is a file system that allows the user to create, delete, and modify file system catalogs, subcatalogs, and named files. Only the file space, not the content, is affected by ACCESS.

Command Loader
: is a default subsystem which is invoked whenever an unrecognized command is given, either at system-selection level or in line-numbered mode. Input is assumed to be the catalog/file description of an H* file to be loaded and executed, or a series of responses for a command file application.

CONVER
: is a media-conversion tool, which provides the user with the ability to transliterate files from one media code to another.

FDUMP
: is a remote terminal, word-oriented file inspection and maintenance facility for files, regardless of their format.

HELP
permits a terminal user to obtain an explanation of a system error message.

LODS
provides a debugging environment for a specified TSS subsystem by loading the Debug Trace Package with the subsystem.

LODT
similar to LODS, except the debugging environment is provided for a user program resident on an H* file.

LODX
allows the user to load and execute a program resident on ah H* file.

SABT
retrieves specific locations of the ABRT file for printing (on either a user terminal or optionally on the central site printer). The file named ABRT must be created by the user and entered into the Available File Table(AFT).

The primary functions of the time sharing command language are:

1. Initiation of processing within a subsystem (e.g., LIST and RUN commands).
2. Storage, retrieval, and purge of permanent files (e.g., SAVE and OLD commands).
3. Request for operations on temporary time sharing files (e.g., NEW and RESEQUENCE commands).
4. Request for pertinent operations information (e.g., HELP and STATUS commands).
5. Direction of flow of control within the subsystem (e.g., DONE and BYE commands).

APPENDIX A

MULTICS GCOS DAEMON

The GCOS daemon is a facility designed to aid in the simulation of a GCOS environment on Multics. The daemon allows standard GCOS jobs to be submitted from either punched cards or IMCV magnetic tapes in GCOS standard system format.

DAEMON COMPOSITION

The GCOS daemon comprises the following routines and procedures:

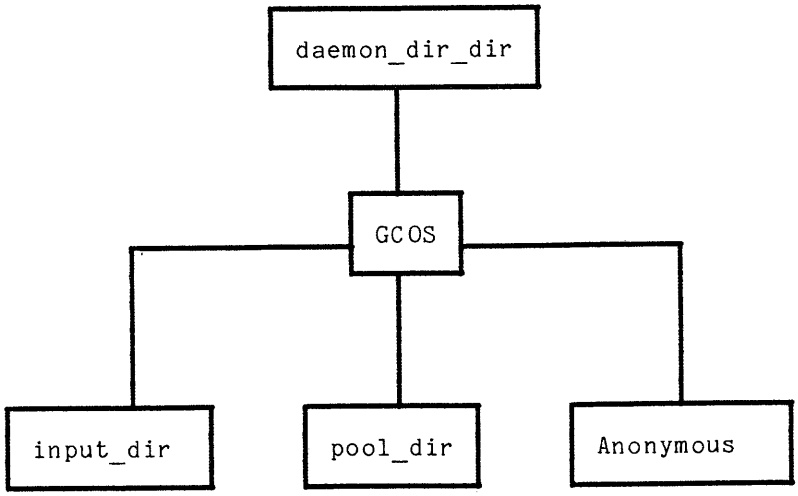
- Main routine
- Initialization routine
- Two procedures to prepare for reading: one for punched cards and one for IMCV magnetic tapes
- Generalized read routine
- Job queuing routine
- Procedure for converting GCOS time to Multics time

USER INTERFACE

Users submit GCOS job card decks and IMCV tapes to Multics operations personnel who activate the GCOS daemon to process the input jobs.

In order to use the GCOS daemon services, users must first be registered on Multics as a user and also be registered to use the GCOS daemon. The GCOS administrator for the site can do this. Specifically, an entry must be made in the gcos_user_table_ that defines the account_id to be specified on the $ IDENT card and the Multics user_id for whom the job is to be run. (Refer to Appendix B for more details.)

DIRECTORY STRUCTURE



FLOW OF INPUT AND OUTPUT

The simulator job stream is read into a segment named snumb.gcos in the directory input_dir. The snumb is taken from the $ SNUMB control card. When a $ IDENT card is encountered in the input stream, the account number field is used to do a table lookup of a person.project for which the daemon requests an absentee job be submitted.

NOTE: The term "segment" refers to either a segment or a multisegment file, as needed. A segment is a collection of instructions and/or data that is associated with a particular segment name (i.e., a file).

The daemon creates an absentee file in the directory gdd>Project>Person with the file name snumb.absin, where the snumb is taken from the $ SNUMB card. When the absentee job is executed, two output files are created: snumb.absout from the absentee job and output files from the GCOS job. Both of these output files are created in the user directories. The snumb.absout file is printed via the dprint option. The job output files are printed via the dprint option or punched on cards via the dpunch option, as specified by the GCOS job, for snumb.gcos.

JOB PRIORITIES

If an urgency level field is encountered on the $ SNUMB card, it is used to compute the priority level at which the absentee job and generation of results are queued.

- If the urgency number is 0-21, the third priority absentee queues are used.
- If the urgency number is 22-42, the second priority absentee queues are used.
- If the urgency number is 43-63, the first priority absentee queues are used.

The default is to use the third priority absentee queue. Usage of the priority absentee queues can be restricted by setting appropriate access on them. To prevent GCOS users from dominating absentee or printing facilities, the operator also can set the highest priority queue that all GCOS jobs can use.

OPERATOR INTERFACE

The daemon is activated at the command level by the operator, who enters the daemon name GCOS.SysDaemon via the daemon console. The daemon then types the message ENTER REQUEST on the console.

Procedures differ slightly depending upon whether input is via punched cards or an IMCV magnetic tape.

- If input is via punched cards, the operator enters the attach command to attach a card reader, and enters the read command to issue a read request to the card reader.

  The daemon then goes blocked waiting for placement of a card deck in the card reader. When a deck is read, the daemon processes it and again types ENTER REQUEST.

  When all cards have been read in, or if the card reader is needed for another use, the operator enters the command "detach" to detach the device.

  The operator then enters "logout" when processing is completed.

- If input is via an IMCV magnetic tape, the operator enters the "imcv reel_id" command for nine-track tape, or "imcv7 reel_id" for seven-track tape. This command is followed by an argument or arguments specifying the jobs on the tape to be executed.

  The daemon attaches an appropriate tape drive, reads the tape, automatically detaches the tape drive, and processes the appropriate jobs.

  The operator logs out the daemon by entering "logout" via the daemon console.

  NOTE: Jobs on magnetic tape can be read in while the card reader is not attached.

DAEMON COMMANDS

The daemon executes any of the following requests when at command level:

.
: invokes a self-identify request.

?
: prints a list of daemon requests (same as the help request).

abort
: terminates the current daemon job (usually as a result of a quit signal being issued). The effect is analogous to the release command at Multics command level in that the daemon is returned to request level after cleaning up from the job in progress.

admin
: enters admin mode (requires password). (The admin mode is a means

of entering the standard Multics command mode under the authority of the daemon.)

attach
attaches the card reader.

detach
detaches the card reader.

imcv reel_id-option
imcv7 reel_id-option
request for reading in a job or jobs from an IMCV magnetic tape: where "imcv reel_id" indicates a nine-track tape is being used, and "imcv7 reel_id" indicates a seven-track tape is being used. The system replies with mounting instructions, or with a message indicating the tape unit is not available. One of the following options must be included as an argument (option):

-all
all jobs on the IMCV magnetic tape are accepted.

-deleteb sssss...sssss
all jobs except those specified are read in from the tape. From one to five snumbs are allowed. After the tape is read, it is detached. Those jobs read from tape are queued as absentee jobs.

-startb sssss
beginning with the indicated snumb, all jobs through end of file are read in.

-startb sssss...sssss
beginning with the first snumb, all jobs up to but not including the second snumb are read in.

-takebb sssss...sssss
only those snumbs specified for execution are read in. From one to five snumbs are allowed.

NOTE: The hyphen preceding these command options is optional. Also, each b represents one blank space and is not a character entry.

help
prints a list of daemon requests (same as the "?" request).

logout
logs out the daemon.

read
issues a read request to the card reader.

re_init
reinitializes the daemon.

set_maximum_queue_level
smq_level
sets the maximum priority at which jobs can be queued.

start
resumes processing from the quit signal.

RECORD QUOTA OVERFLOWS

If a record quota overflow occurs while a job is being read in, that portion read in is deleted and an appropriate message is printed on the daemon console.

Record quota overflows occurring while in the absentee process cause the absentee process to signal the daemon to print a message on the console so that the operator can take some appropriate action (e.g., suspend absentee processes). The GCOS routines are responsible for "cleaning up" the gcos_pool directory.

MAINTENANCE OF THE GCOS USER TABLE

The GCOS daemon uses the account number field from the $ IDENT control card image of a GCOS job to determine the Multics person and project identifications for the job to be run. This correspondence between the GCOS account number and the Multics person-project identifications is maintained in a table (gcos_user_table_) in the GCOS daemon's home directory.

A number of commands (described in the following paragraphs) are provided to maintain this table. These commands are:

```
gcos_user$add
gcos_user$delete
gcos_user$init_table
gcos_user$print
```

Creating GCOS User Table

The table is created by using the Multics create command:

```
cwd >ddd>GCOS
cr gcos_user_table_
```

Initializing the User Table

The following command initializes all header data in the gcos_user_table_ segment:

```
gcos_user$init_table
```

NOTE: The gcos_user_table_ bit count must be zero before initialization is allowed. If an existing table is to be re-initialized, the bit count must be set to zero.

Adding Users to Table

The following command adds a user to the table:

```
gcos_user$add
```

The command prompts the caller to enter needed data. (The entries below, preceded by the exclamation mark (!), indicate user response.)

```
    GCOS account ID?
!   (GCOS account identification)
    Multics Person Name?
!   (Multics person identification)
    Multics Project ID?
!   (Multics project identification)
```

Examining Table Contents

The following command allows the caller to examine the contents of the GCOS user table:

```
gcos_user$print {-control_args}
```

where control_args are:

-gcos_account_id ID1 {ID2...IDn}, -gaid ID1 {ID2...IDn}
: each IDn is another argument specifying a GCOS account number. The command prints the corresponding Multics person-project identifications.

-header, -hdr
: causes the command to print status information about the table. There are no additional arguments.

NOTE: The -gcos_account_id and -header control arguments may be mixed on a command line, and are processed in the order in which they are specified. Multiple occurrences of either, or both control arguments are permitted.

Deleting From Table

The following command allows the caller to delete entries from the table:

```
gcos_user$delete gcos_account_id_ {...  gcos_account_idn}
```

where gcos_account_id1 is the first account number in a string to be deleted and gcos_account_idn is the last account number in a string to be deleted.

INSTALLING THE GCOS DAEMON

In addition to having the GCOS SysDaemon procedures installed, the SysDaemon must be registered and a number of directories and segments must be created to support operation of the SysDaemon. Creation of the segments can be delegated to a GCOS project administrator.

Registering the SysDaemon

1. The daemon should be registered in the SysDaemon project under the name GCOS. A password and normal daemon attributes should be included. The project master file entry must include:

   ```
   homedir:  >ddd>GCOS;

   initproc:  >pathname>gcos_daemon;
   ```

2. A GCOS project must be created and the user "Anonymous" registered with a password and normal attributes. The project master file entry must include:

   ```
   homedir:      >ddd>GCOS>Anonymous;
   initproc:     process_overseer_;
   ring:         5,5;
   ```

3. A GCOS directory must be created under the daemon_directory_directory (ddd) and include the input_dir, pool_dir, and Anonymous subdirectories (see the directory structure diagram in Appendix A).

   The ACLs for the subdirectories must include:

```
sma GCOS.SysDaemon.*
sma Anonymous.GCOS.*
sma Administrator.Project.*

NOTE:  *.SysDaemon.* includes GCOS.*SysDaemon.*.
```

   The project administrator should have at least s (status) access on >ddd>GCOS.

   The initial ACLs for the subdirectories must include:

```
rw Anonymous.GCOS.*
```

   and should include:

```
rw Administrator.Project.*
```

CREATING THE GCOS SYSDAEMON USER DIRECTORY STRUCTURE

The directory gcos_directory_directory (gdd) must be created to contain the user directories and segments referenced by the SysDaemon. The structure of this directory is analogous to that of the user_directory_directory (udd). Each project.person using the SysDaemon must have a directory under >gdd (see the Multics storage system hierarchy in Figure 1-1).

The ACLs for each person directory must include:

```
sma     GCOS.SysDaemon.*
sma     Anonymous.GCOS.*
sma     Person.Project.*
sma     Administrator.Project.*
```

The initial ACLs must include:

```
rw      Anonymous.GCOS.*
```

and should include:

```
rw      Person.Project.*
rw      Administrator.Project.*
```

The administrator needs sma access in the superior directories to create the project and person directories. Note that the project administrator also needs access rights to the gcos_user_table_ to maintain the table.

APPENDIX B

INSTALLATION AND MAINTENANCE OF GCOS SOFTWARE

COMMANDS

The following commands are used to create and update the software and subroutine libraries required by the gcos and gcos_tss commands. The input files for these maintainance commands may be either files copied from the GCOS total system tape or files produced by these commands, or both. These commands allow a site to install PSP software or updated releases of GCOS.

gcos_build_library

Name: gcos_build_library, gcbl

This command builds a GCOS simulator software library from a native GCOS system-loadable file and optionally appends a second software file. A catalog (or load table) is built from information on the input file and the optional append file, and occupies the first 1000 words of the output file. This process is analogous to building the "GECALL TABLE" by GCOS, at startup, and exists for the same reason (i.e., to facilitate loading of the individual modules).

Usage

gcbl input_path output_path {-control_args}

where:

1. input_path
   is the pathname of the input file. This file may or may not already have a catalog. This file may have been created by this procedure, in which case it has a catalog, or by the gcos_extract_module or gcos_pull_tape, in which case it would not have a catalog.

2. output_path
   is the pathname of the file into which the modules in the input file (plus those in the optional append file) are to be copied, preceded by a catalog. If this file already exists, it is overwritten, with no warning.

3. control_args
   can be selected from the following:

   -append append_path
   append_path is the pathname of a file whose contents are to be appended to those of the input file (i.e., copied to the end of the output file) before the new catalog is built.

   -brief, bf
   suppresses printing of the catalog (load name and offset) of output_path.

Notes

The catalog occupies the first 1000 words of the output file, and has space for 499 entries.

Existing modules are not replaced by the appending function. The only way to replace an existing module with a different version of itself is to extract from the existing library, into a new file, all the modules except the one(s) to be replaced, extract the new versions of those modules into the new file (they are appended to those previously extracted into the file), and then build a catalog for the new file.

This command cannot be used to manipulate a random system-loadable file since the catalog created differs from the format of the catalog produced by SYSEDIT (commonly used for ** files).

This command is not used to convert the GCOS subroutine libraries (the SOFTW-SYSLIB files) to a form usable by the GCOS simulator. To do this, the gcos_reformat_syslib command is used.

Example

Assume the software_1.gcos and software_2.gcos files have just been read from a GCOS Total System Tape with the gcos_pull_tapefile command. A single GCOS simulator library named gcos_system_software_ is to be built which contains the software on these two files. The command to do this is:

```
gcbl software_1.gcos gcos_system_software_ -append software_2.gcos
```

The gcos_system_software_ file can now be used by the gcos or gtss command and the software_1.gcos, and software_2.gcos files may be deleted.

Name: gcos_extract_module, gcem

This command extracts one or more modules from a (tape format) GCOS system-loadable file, or a simulator-format software library, placing them in an output file. The names of the modules to be extracted are given in a control file.

Usage

gcem control_path input_path output_path {-control_arg}

where:

1. control_path
   is the pathname of a segment containing the names of the modules to be extracted (in lowercase ASCII), each on a separate line. This segment can be created by the gcos_library_summary command, using the -brief control argument and preceding it by the file_output command, and then editing the resulting segment to remove the names of unwanted modules; or the segment can be created directly by any of the Multics editors.

2. input_path
   is the pathname of a file containing GCOS modules. It can be a file copied from a GCOS total system tape with gcos_pull_tapefile, or it may be a simulator-format library produced by gcos_build_library.

3. output_path
   is the pathname of the output file into which the extracted modules are to be written. If the file already exists, it is appended to, enabling the construction of a single library from several smaller ones.

4. control_arg
   can be -brief which inhibits printing the name of each module copied.

Notes

The names of each module copied are printed on user_output, unless the -brief control argument is given. The names of any requested modules that are not found in the input file are printed on error_output.

This command cannot be used to manipulate a random system-loadable file (GCOS format **, Q* or H* file).

gcos_library_summary

Name: gcos_library_summary, gcls

This command prints a summary of the contents of a GCOS tape-format system-loadable file or a simulator-format software library.

Usage

gcls path {-control_args}

where:

1. path
   is the pathname of either a GCOS format system-loadable file or a simulator-format software library file.

2. control_args
   can be selected from the following:

   -brief, -bf
   prints only the names of the modules, omitting any preface or catalog information.

   -gcos, -gc
   identifies the input file as a GCOS ** file.

   -preface
   prints the preface record information found at the beginning of each module.

Notes

If path is a GCOS format file as copied from a Total System Tape, then gcls lists the modules and the preface information.

If path is a GCOS simulator-format library, then gcls lists the module names and the catalog offset.

If path is a native GCOS ** file and -gcos is specified, then the modules on the file are listed.

gcos_pull_tapefile

Name: gcos_pull_tapefile, gcpt

This command copies files from a GCOS tape into the Multics storage system. Since it was designed to read files from a total system tape, from which the gcos simulator libraries are built, it accepts records longer than 320 words. It reads the header label preceding each file, prints the name and file sequence number, and asks the user if the file is to be copied. If the answer is yes, it asks for a pathname, and copies the file into it. Then it continues with the next file. This process is terminated either by a reply of "quit" to the copy question, or an end-of-reel label or partial label on the tape.

Usage

gcpt {reel_number} {-control_args}

where:

1. reel_number
   is the number of the tape reel to be read. It need not match any field in the tape labels. (See -attached control argument.)

2. control_args
   can be selected from the following, and can be in any order:

   NOTE: The following list identifies all control arguments grouped by function.

   ATTACHING TAPE
   -attached
   -block
   -density
   -detach
   -gsr
   -retain

   PROGRAM CONTROL
   -brief
   -long
   -long_brief
   -skip

   -attached, -att
   may be given in place of reel_number to indicate that the tape has been retained from a previous use of this command and is already mounted. (See -retain control argument)

   -block N
   N specifies the tape buffer size (any value up to 4096) needed to read a GCOS total system tape. This argument is not required in order to read a total system tape, but is needed for tapes with records larger than 3842 words. (Default is 3842)

   -brief, -bf
   specifies that certain warning and informative messages NOT be displayed.

-density N, -den N
N specifies tape density. (Default is 1600)

-detach, -det
specifies detaching and dismounting of the tape before any attempt is made to attach the currently specified tape. If this is the only argument given, the tape is detached and no other processing takes place.

-gsr
specifies standard size records (320 words or less) to be copied into the Multics storage system in a format readable by other Multics tools that manipulate GCOS files. Records shorter than 320 words are padded out to 320 words, allowing a read of 320 words to obtain exactly one record. This argument should not be used when reading a total system tape.

-long, -lg
specifies that hardware status, labels, and length and first word of all data records be displayed.

-long_brief, -lb
specifies that hardware status and labels be displayed, but length and first word of data records NOT be displayed.

-retain, -ret
specifies that the tape remain mounted when processing by this command is completed. This saves work for the Multics operator if several attempts to read the tape are to be made.

-skip N
N specifies the files to be skipped over before the user is asked if files are to be copied. (Questioning starts with N+1st file.)

Examples

Assume a 1600-bpi GCOS Total System Tape, reel number U812, contains the following software files:

```
SOFTW-SYSLIB
GCOS-PART1
GCOS-PART2
TSS-SUB-SYS
SOFTW-PART1
SOFTW-PART2
```

The gcos_pull_tapefile command is used to copy the softw-syslib, softw-part1, and softw-part2 tape files into Multics segments. The path names used for the input copy files may be any valid Multics name.

```
! gcpt u812
  Tape u812,den=1600,block=3842 will be mounted with no write ring.
  Tape u812,den=1600,block=3842 mounted on drive tapa_03 with no write ring.
  File 1. is "softw-syslib"

! Copy?   yes

! Pathname of file to copy to:  softw_syslib.gcos
  998 tape records read, 313 Multics records (pages) written
  End of File
  File 2. is "gcos-part1"
```

```
! Copy?  no
  End of File
  File 3. is "gcos-part2"

! Copy?  no
  End of File
  File 4. is "tss-sub-sys"

! Copy?  no
  End of File
  File 5. is "softw-part1"

! Copy?  yes

! Pathname of file to copy to:  software_1.gcos
  410 tape records read, 442 Multics records (pages) written
  End of File
  File 6. is "softw-part2"

! Copy?  yes

! Pathname of file to copy to:  software_2.gcos
  896 tape records read, 1382 Multics records (pages) written
  End of File
  Partial label (end of tape).
```

The following example shows how a single file, tss-sub-sys, may be copied.

```
! gcpt u812 -skip 3 -bf
  Tape u812,den=1600,block=3842 will be mounted with no write ring.
  Tape u812,den=1600,block=3842 mounted on drive tapa_03 with no write ring.
  End of File
  End of File
  End of File
  File 4. is "tss-sub-sys"

! Copy?  yes

! Pathname of file to copy to:  gcos_tss_software_update_8
  188 tape records read, 316 Multics records (pages) written
  End of File
  File 5. is "softw-part1"

! Copy?  quit
```

gcos_reformat_syslib

Name: gcos_reformat_syslib, gcrs

This command reformats the softw-syslib file (the library used by GELOAD) from the total system tape, so that it can be read by the simulator.

Usage

```
gcrs input_path {output_path} {record_length}
```

where:

1. input_path
   is the pathname of the file pulled from a GCOS total systems tape. Typically, this is the file for "softw-syslib."

2. output_path
   is the pathname for the reformatted data.

   If output_path is not given, the modifications are made to the input file. Since a temporary file is not used, quitting and releasing while updating the input file leaves the file in an inconsistent state, from which recovery is almost impossible.

3. record_length
   is the number of words from each input record to be written to the output file after the first data word of each input record is discarded.

   If record_length is not given, the default is 321. Record_length is distinguished from output_path in that it must be numeric. It may precede or follow output_path, and output_path need not be given when record_length is given.

Notes

As read from the total system tape, the file contains 321-word blocks. The first is a block control word (BCW), and the next 320 make up a block that GELOAD interprets itself. The BCW of each block is discarded, and the remaining 320-word blocks are packed together. If read from a 7-track tape, the word count is rounded up to the next even number (322), so a word at the end must also be discarded. The number of words to be read is therefore an optional argument, with the default being 321. The first word is discarded, and the next 320 words are always retained.

The first word (the BCW) is checked for sequential block serial numbers (BSN), and block lengths of 320 words or fewer, to verify that the correct record length was used for reading, and that the words being discarded are really BCWs.

Name: gcos_tss_build_library, gtbl

This command builds or updates the GCOS software library, gtss_fast_library_, used by gcos_tss, and updates the program descriptors required to access the added software.

Usage

```
gtbl input_libs {-control_args}
```

where:

1. input_libs
   are the pathnames of one or more GCOS software libraries. The input libraries are either in native GCOS format (as copied from a GCOS total system tape) or in GCOS environment simulator format (as output from gcos_build_library).

2. control_args
   can be one or more of the following:

   -brief, -bf
   suppresses printing of information concerning updated modules and program descriptors and the catalog, even if requested by the -print_catalog control argument.

   -module_file path, -mf path
   path specifies the segment containing a list of modules to be moved from the input libraries to the output library. These names are added to those specified by the -module control argument.

   -modules STR1 {...STRn}, -mods STR1 {...STRn}
   STRi specifies the name of the module to be moved from an input library to the output library. These names are added to those specified by the -module_file control argument.

   -output_library path, -olb path
   specifies the path of the gtss_fast_library_ output library. If the library already exists, it is appended to.

   -print_catalog, -prcat
   prints the catalog (modules and offsets) of an input library if the library is in simulator format.

   -program_descriptors path, -prgdes path
   path specifies the gtss program descriptors (a copy of gtss_prgdes_alm_.incl.alm) to be updated.

Notes

If no output library path is specified, then only a list of the modules found on the input libraries is printed.

To use the added modules in gtss, the -prgdes control argument must be used. Then gtss_prgdes_ext_.alm must be reassembled with the gtss_prgdes_alm_.incl.alm specified by -prgdes. Following this, gtss must be rebound with the new gtss_prgdes_ext_, and the updated output library must be copied to gtss_fast_library_ in the directory containing gtss (or a link established).

This is the only procedure required to add PSP or user software to the gcos_tss library for which there are program descriptors.

If the -modules and -module_ file control arguments are not specified, then all modules found on the input libraries are copied to the output library. The only exception is when the -prgdes control argument is specified. In this case, only modules on the input libraries, which are actually needed by the program descriptor file, are copied.

Example

Assume that the segment pcf_softw exists, having been copied from a GCOS tape with the gcos_pull_tapefile command and that gtss_prgdes_alm_.incl.alm has been copied to the working directory. The command to add pcf_softw to the known gtss subsystems is:

```
gtbl pcf_softw -olb >unb>gtss_fast_library_ -prgdes gtss_prgdes_alm_.incl.alm
```

When finished, the command prints:

```
Updated program descriptors will be labeled "11/30/83  1113.4".
Added module ".PCPCF" from >udd>ges>4js3>pcf_softw
Added program descriptor 106 for ".PCPCF".
```

This shows that the single module ".PCPCF" from the input library pcf_softw has been appended to the output library gtss_fast_library_. Program descriptor 106 in gtss_prgdes_alm_.incl.alm has been updated with the load information for the module and the date/time it was added.

When the gtss modules referencing the prgdes include file are recompiled and gtss is rebound, the new software in gtss_fast_library_ is accessible.

STEPS TO BUILD GCOS SOFTWARE AND LIBRARY FILES

The following steps describe how to use the maintenance commands (above) to build or update the three kinds of GCOS software files required by gcos and gtss. The GCOS software files shipped with gcos and gtss include all the most recent bundled GCOS software available prior to the Multics release. These files may be used as shipped, unless a site needs to install a subsequent GCOS release or any unbundled GCOS software.

Building the GCOS Software Load Files

Follow these steps to build the gcos_system_software_ and gcos_second_software_ files and any other GCOS load file from a GCOS system tape.

1. Use the gcos_pull_tapefile command to pull each desired file off of the GCOS Total System Tape (TST) and into a separate segment (or MSF).

2. Use the gcos_build_library command to convert these files into the gcos_system_software_ file (for bundled software) and gcos_second_software_ (for unbundled software). The files read from a GCOS tape can be concatenated into a single software file, but a separate run of gcos_build_library is required for each file added. The file created by gcos_build_library is limited to 500 load modules.

3. Once the gcos_system_software_ and gcos_second_software_ files are created, their input files (the ones read by gcos_pull_tapefile) can be deleted.

4. Use the gcos_library_summary command to produce a list of the load modules on the gcos_system_software_ and gcos_second_software_ files.

5. When copied into the >unb directory (see Note), gcos_system_software_ and gcos_second_software_ files are immediately available to the gcos command. The gcos_second_software_ file is searched only when the -userlib control arg is used.

Building the GCOS and GTSS Software Subroutine Libraries

Follow these steps to build subroutine library files from the SOFTW-SYSLIB tape file.

1. Use gcos_pull_tapefile to read the SOFTW-SYSLIB file off the TST or unbundled software tape. (This file has the same name regardless of its content. Some sites may only get a single, customized version.)

2. Use the gcos_reformat_syslib command to restructure the syslib file into one usable by the simulator. Name it gcos_library_subroutines_ (or gcos_altlib_subroutines_ if it only contains unbundled software.)

3. Add the name gcos_Lstar_ to gcos_library_subroutines_ and the name gcos_starL_ to gcos_altlib_subroutines_.

4. When moved to >unb (see Note), the gcos_library_subroutines_ file is available to the gcos command. The altlib file is accessed via the "$ LIBRARY *L..." control card with Multics path given.

5. To access gcos_library_subroutines from gtss, make the primary name gtss_Lstar_ (note the uppercase L). Now run the ec in >unb to add the necessary gtss file attributes:

   ec gtss_file_attributes gtss_Lstar_ r

   The r argument is required.

6. To access gcos_altlib_subroutines from gtss, make the primary name gtss_starL_. Run the ec as in step 5.

   ec gtss_file_attributes gtss_starL_ r

7. Both gtss_Lstar_ and gtss_starL_ are immediately available to gtss when moved to >unb and these add names are the PRIMARY names.

Building the GTSS Command File

Follow these steps to build the gtss_fast_library_ file from the TSS-SUB-SYS tape file.

1. Use the gcos_pull_tapefile command to read the appropriate TSS files from the TST tape. These include SOFTW-PART1, SOFTW-PART2, and TSS-SUB-SYS (some sites may have others).

2. OPTIONAL STEP FOR SITES WITH UNUSUAL TSS INSTALLATIONS.

   a. Use gcos_card_utility to read GCOS GMAP source of module TSSA from GC2 source tape. Write to Multics ascii text segment "tssa.ascii".

   b. Compare tssa.ascii to gtss_prgdes_alm_.incl.alm and make the required load module NAME changes in the include file. You may have to convert tssa.ascii to ALM.

3. Use the gcos_tss_build_library command to convert the GCOS files read from the TST tape in Step 1 into the gtss_fast_library_ file. This command also updates the gtss_prgdes_alm_.incl.alm include file.

4. Reassemble gtss_prgdes_ext_.alm with the updated gtss_prgdes_alm_.incl.alm. (If the NUMBER of program descriptors was changed in step 2, then make that change in gtss_primitives_.incl.pl1 and gtss_prgdes_.incl.pl1 and recompile ALL gtss pl1 modules.)

5. Rebind gtss.

6. The new gtss_fast_library_ is available to gtss when it and the newly bound gtss are moved into >unb (see Note).

7. Steps 1 through 6 can be repeated to add new TSS modules to gtss_fast_library_.

Note: A user working directory can be substituted for >unb for all of these procedures for test and verification, but this directory must contain or have links to all of gcos, gtss, and the related gcos software files.

APPENDIX C

GCOS ENVIRONMENT SIMULATOR AND NATIVE GCOS DIFFERENCES

Detailed differences in GCOS environment simulation and native GCOS functions or responses are noted in the sections of this document that describe the specific functions, especially in those sections describing MMEs and control cards. This is a brief discussion regarding some of the basic reasons these differences arise.

The GCOS environment simulator is not an operating system. It is a Multics subsystem whose purpose is to run GCOS jobs with native GCOS software (i.e., compilers and utilities), thereby eliminating any differences in job results. To this end, the environment simulates the GCOS functions of intrajob scheduling of activities and the calling of required software and service functions.

All other operating system functions (i.e., file management, resource allocation, dispatching, input/output, etc.) are performed by the Multics operating system. Therefore, the GCOS environment does not simulate these GCOS operating system functions.

Further, the GCOS environment need not and does not simulate in detail the intrajob scheduling function of GCOS. (However, since both are time sequential, they have the same general form.) In part, this is because of the simplifying restriction of having one GCOS job per Multics process.

The environment simulates GCOS by:

- Reading the job stream
- Interpreting the job control cards to obtain system resources from the Multics operating system and to control job activity execution
- Initiating, monitoring, and terminating job activities
- Interpreting slave job requests for GCOS operating system services and making equivalent requests to the Multics operating system

The environment must work through the Multics input/output (and peripheral allocation) system and the Multics file structure.

Differences arise when an equivalent function cannot be performed under Multics, or when the equivalent function is not meaningfully expressed in GCOS terms; for example, GCOS file code assignment via the system configuration table number, or file sizes expressed in links or llinks.

Differences also arise when a GCOS function is simulated functionally, but not in detail.

Finally, usage statistics - where they are available - come from the Multics system accounting procedures and may not correspond to GCOS usage statistics.

If these basic structural differences are kept in mind, it may help in understanding those detailed differences that are noted elsewhere in this document.

GCOS ENVIRONMENT IN GCOS TERMS

Another approach to describing the GCOS environment is to consider it, in GCOS terms, as a special GCOS system operating from disk to disk with offline input and output (with the exception of data tapes).

From this viewpoint, the GCOS daemon provides offline local batch input from punched cards or IMCV magnetic tapes to a disk buffer file. This is equivalent to input entered via GCOS remote system input (.MRGIN). The GCOS daemon also assumes the system scheduler function by queuing the job in the Multics absentee job facility.

Alternatively, jobs can be entered offline via a Multics interactive terminal in a manner that is equivalent to GCOS time sharing.

The GCOS environment reads the job stream from the disk file through the equivalent of the .MGENB module (Mass Store Device Input) of system input (.MGEIN). Output is roughly equivalent to SYSOUT that is purged to disk rather than to magnetic tape. Unpurging, however, is via a BCD-to-ASCII translation and then via the offline Multics I/O daemon to a printer or card punch.

A virtual machine, corresponding to the GCOS environment comprises:

- One central processor
- 256K of memory (one Multics segment)
- One Input/Output Multiplexor (IOM) with magnetic tape and disk subsystems, but with no unit record devices
- One Front-End Network Processor (FNP) with one remote terminal (the interactive user's terminal for Multics, and the direct-access output-then-input terminal for GCOS -- this terminal also receives GCOS console output messages).

INDEX

CUT ALONG LINE

HONEYWELL INFORMATION SYSTEMS
Technical Publications Remarks Form

TITLE: MULTICS
GCOS ENVIRONMENT SIMULATOR

ORDER NO. AN05-02A

DATED DECEMBER 1985

ERRORS IN PUBLICATION

SUGGESTIONS FOR IMPROVEMENT TO PUBLICATION

Your comments will be investigated by appropriate technical personnel and action will be taken as required. Receipt of all forms will be acknowledged; however, if you require a detailed reply, check here. ☐

FROM: NAME _____ DATE _____

TITLE _____

COMPANY _____

ADDRESS _____

_____

PLEASE FOLD AND TAPE—
NOTE: U. S. Postal Service will not deliver stapled forms

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 39531 WALTHAM, MA02154

POSTAGE WILL BE PAID BY ADDRESSEE

HONEYWELL INFORMATION SYSTEMS
200 SMITH STREET
WALTHAM, MA 02154

ATTN: PUBLICATIONS, MS486

FOLD ALONG LINE

FOLD ALONG LINE

CUT ALONG LINE

Honeywell

Together, we can find the answers.

Honeywell

Honeywell Information Systems
U.S.A.: 200 Smith St., MS 486, Waltham, MA 02154
Canada: 155 Gordon Baker Rd., Willowdale, ON M2H 3N7
U.K.: Great West Rd., Brentford, Middlesex TW8 9DH **Italy**: 32 Via Pirelli, 20124 Milano
Mexico: Avenida Nuevo Leon 250, Mexico 11, D.F. **Japan**: 2-2 Kanda Jimbo-cho Chiyoda-ku, Tokyo
Australia: 124 Walker St., North Sydney, N.S.W. 2060 **S.E. Asia**: Mandarin Plaza, Tsimshatsui East, H.K.

39402, 5C1283, Printed in U.S.A.

AN05-02