HEWLETT PACKARD

HP-UX Reference
Vol. 1B: Section 1

HP-UX Reference
Vol. 1B: Section 1 (M through Z)

for

HP Part Number 09000-90008

Hewlett-Packard Company
3404 East Harmony Road, Fort Collins, Colorado 80525

Printing History

New editions of this manual will incorporate all material updated since the previous edition. Update packages may be issued between editions and contain replacement and additional pages to be merged into the manual by the user. Each updated page will be indicated by a revision date at the bottom of the page. A vertical bar in the margin indicates the changes on each page. Note that pages which are rearranged due to changes on a previous page are not considered revised.

The manual printing date and part number indicate its current edition. The printing date changes when a new edition is printed. (Minor corrections and updates which are incorporated at reprint do not cause the date to change.) The manual part number changes when extensive technical changes are incorporated.

July 1985...Edition 1. This manual replaces HP-UX Reference Manual 09000-90007 and documents HP-UX Release 5.0 for Series 200, 300 and 500.

November 1985...Edition 2. Updated from Edition 1 to reflect Series 200/300 HP-UX Release 5.1 changes. Several omitted pages in Edition 1 were also added.

June 1986...Edition 3. Update 1 incorporated.

September 1986...Edition 3 Update 1. This update reflects additions and changes incorporated in Series 500 HP-UX Release 5.1. Added command *autobackup*(1M) and core files support (*core*(5)), changed blocksize limitations for SDF file formats, and fixed various bugs.

TABLE OF CONTENTS

1. Commands

intro(1) ..... introduction to Section 1
acctcom ..... search and print process accounting files
adb ..... debugger
adjust ..... simple text formatter
admin ..... create and administer SCCS files
ar ..... archive and library maintainer
arcv ..... convert archives to new format
as ..... assembler for MC68000
asa ..... interpret ASA carriage control characters
at ..... execute commands at a later time
aterm ..... general purpose asynchronous terminal emulation
atrans ..... translate assembly language
awk ..... text pattern scanning and processing language
banner ..... make posters in large letters
basename ..... extract portions of path names
bc ..... arbitrary-precision arithmetic language
bdiff ..... big diff
bfs ..... big file scanner
bifchmod ..... change mode of a BIF file
bifchown ..... change file owner or group
bifcp ..... copy to or from BIF files
bifdf ..... report number of free disc blocks
biffind ..... find files in a BIF system
biffsck ..... Bell file system consistency check and interactive repair
biffsdb ..... Bell file system debugger
bifls ..... list contents of BIF directories
bifmkdir ..... make a BIF directory
bifmkfs ..... construct a Bell file system
bifrm ..... remove BIF files or directories
bs ..... compiler/interpreter for modest-sized programs
cal ..... print calendar
calendar ..... reminder service
cat ..... concatenate, copy, and print files
cb ..... C program beautifier, formatter
cc ..... C compiler
cd ..... change working directory
cdb ..... C, FORTRAN, Pascal symbolic debugger
cdc ..... change the delta commentary of an SCCS delta
cflow ..... generate C flow graph
chatr ..... change program's internal attributes
chmod ..... change mode
chown ..... change file owner or group
chsh ..... change default login shell
clear ..... clear terminal screen
cmp ..... compare two files
col ..... filter reverse linefeeds and backspaces
comm ..... select/reject common lines of two files
compact ..... compress and uncompress files, and cat them
cp ..... copy, link or move files
cpio ..... copy file archives in and out
cpp ..... C language preprocessor
crontab ..... user crontab file
csh ..... C shell
ctags ..... create a tags file

cu ..... call another HP-UX system
cut ..... cut out selected fields of each line of a file
cxref ..... generate C program cross-reference
date ..... print and set the date
dc ..... desk calculator
dd ..... convert, reblock, translate, and copy a (tape) file
delta ..... make a delta (change) to an SCCS file
deroff ..... remove nroff/troff, tbl, and eqn constructs
diff ..... differential file comparator
diff3 ..... 3-way differential file comparison
diffmk ..... mark differences between files
dircmp ..... directory difference comparison
du ..... summarize disk usage
echo ..... echo (print) arguments
ed ..... text editor
edit ..... text editor (variant of ex for casual users)
enable ..... enable/disable LP printers
env ..... set environment for command execution
err ..... report error information on last failure
ex ..... text editor commands
expand ..... expand tabs to spaces, and vice versa
expr ..... evaluate arguments as an expression
f77 ..... see fc
factor ..... factor a number, generate large primes
fc ..... FORTRAN 77 compiler
file ..... determine file type
find ..... find files
findmsg ..... create message catalog file for modification
findstr ..... find strings for inclusion in message catalog
fixman ..... fix manual pages for faster viewing with man(1)
fold ..... fold long lines for finite-width output device
gencat ..... generate a formatted message-catalog file
get ..... get a version of an SCCS file
getopt ..... parse command options
getprivgrp ..... get special attributes for group
grep ..... search an ASCII file for a pattern
groups show group memberships
head ..... give first few lines of file
help ..... ask for help
hostname ..... set or print name of current host system
hp ..... handle special functions of HP 2640 and 2621 series terminals
hyphen ..... find hyphenated words
id ..... print user, group IDs and names
insertmsg ..... use findstring output to insert calls to getmsg
ipcrm ..... remove a message queue, semaphore set, or shared memory id
ipcs ..... report inter-process communication facilities status
join ..... relational database operator
kill ..... terminate a process
last ..... indicate last logins of users and teletypes
ld ..... link editor
leave ..... remind you when you have to leave
lex ..... generate programs for lexical analysis of text
lifcp ..... copy to or from LIF files
lifinit ..... write LIF volume header on file
lifls ..... list contents of LIF directory

lifrename ..... rename LIF files
lifrm ..... remove a LIF file
line ..... read one line from user input
linkinfo ..... object file link information utility
lint ..... a C program checker/verifier
lock ..... reserve a terminal
login ..... sign on
logname ..... get login name
lorder ..... find ordering relation for object library
lp ..... send or cancel requests to an LP line printer
lpstat ..... print LP status information
ls ..... list contents of directories
lsdev ..... list device drivers in the system
m4 ..... macro processor
machid ..... provide truth value about your processor type
mail ..... send mail to users or read mail
mailx ..... send and receive mail
make ..... maintain, update, recompile programs
man ..... on-line manual command
mediainit ..... initialize hard disc, flexible disc, or cartridge tape media
mesg ..... permit or deny messages to terminal
mkdir ..... make a directory
mkstr ..... extract error messages from C source into a file
mm ..... print documents formatted with MM macros
more ..... file perusal filter for crt viewing
mt ..... magnetic tape manipulating program
newgrp ..... log in to a new group
news ..... print news items
nice ..... run a command at low priority
nl ..... line numbering filter
nm ..... print name list (symbol table) of object file
nohup ..... run a command immune to hangups, logouts, and quits
nroff ..... format text
od ..... octal and hexadecimal dump
pack ..... compress and expand files
pam ..... Personal Applications Manager, a visual shell
passwd ..... change login password
paste ..... merge lines in one or more files
pc ..... Pascal compiler
pr ..... print files
prealloc ..... preallocate disc storage
prof ..... display profile data
prs ..... print and summarize an SCCS file
ps ..... report process status
ptx ..... create permuted index
pwd ..... working directory name
query ..... interactive IMAGE database access
ratfor ..... rational FORTRAN dialect
rev ..... reverse lines of a file
revision ..... get HP-UX revision information
rm ..... remove files or directories
rmdel ..... remove a delta from an SCCS file
rmnl ..... remove extra new-line characters from file
rtprio ..... execute process with real-time priority
sact ..... print current SCCS file editing activity

sccsdiff ..... compare two versions of SCCS file
sed ..... stream text editor
sh ..... shell, the standard command programming language
size ..... object file size
sleep ..... suspend execution for an interval
slp ..... set printer options
sort ..... sort and/or merge files
spell ..... find spelling errors
split ..... split a file into pieces
ssp ..... remove multiple line-feeds from output
strings ..... find printable strings in binary file
strip ..... remove symbols and relocation bits
stty ..... set the options for a terminal port
su ..... become another user
sum ..... print checksum and block count of a file
sync ..... update the super block
tabs ..... set tabs on a terminal
tail ..... deliver the last part of a file
tar ..... tape file archiver
tbl ..... format tables for nroff or troff
tcio ..... CS/80 Cartridge Tape utility
tee ..... pipe fitting
test ..... condition evaluation command
time ..... time a command
touch ..... update access/modification/change times of file
tput ..... query terminfo database
tr ..... translate characters
true ..... provide truth values
tset ..... terminal dependent initialization
tsort ..... topological sort
tty ..... get the terminal's name
ul ..... do underlining
umask ..... set file-creation mode mask
uname ..... print name of current HP-UX version
unget ..... undo a previous get of an SCCS file
uniq ..... report repeated lines in a file
units ..... unit conversion program
upm ..... unpack cpio archives from HP media
uucp ..... HP-UX to HP-UX copy; file transfer
uuls ..... list spooled uucp transactions grouped by transaction
uusnap ..... show snapshot of the UUCP system
uustat ..... uucp status inquiry and job control
uuto ..... public HP-UX-to-HP-UX file copy
uux ..... HP-UX to HP-UX command execution
val ..... validate SCCS file
vi ..... visual text editor
vis ..... make unprintable characters in a file visible or invisible
wait ..... await completion of process
wc ..... word, line, and character count
what ..... identify files for SCCS information
whereis ..... locate source, binary, and/or manual for program
who ..... which users are on the system
whoami ..... print effective current user id
write ..... interactively write (talk) to another user
xargs ..... construct argument list(s) and execute command

yacc ..... yet another compiler-compiler

1M. System Maintenance Utilities

accept ..... allow or prevent LP requests
acct ..... overview of accounting and miscellaneous accounting commands
acctcms ..... command summary from per-process accounting records
acctcon ..... connect-time accounting
acctmerg ..... merge or add total accounting files
acctprc ..... process accounting
acctsh ..... shell procedures for accounting
backup ..... backup or archive file system
brc ..... system initialization shell scripts
captoinfo ..... convert a termcap description into a terminfo description
catman ..... create the cat files for the manual
chroot ..... change root directory for a command
chsys ..... change to different operating system or version
clri ..... clear i-node
clrsvc ..... clear x.25 switched virtual circuit
config ..... configure an HP-UX system
cpset ..... install object files in binary directories
cron ..... clock daemon
devnm ..... device name
df ..... report number of free disk blocks
diskusg ..... generate disc accounting data by user ID
fsck ..... file system consistency check, interactive repair
fsclean ..... determine shutdown status of specified file system
fsdb ..... file system debugger
fwtmp ..... manipulate wtmp records
getty ..... set the modes of a terminal
getx25 ..... get x.25 line
init ..... process control initialization
install ..... install commands
kermit ..... KERMIT-protocol file transfer program
killall ..... send signal to all user processes
link ..... exercise link and unlink system calls
lpadmin ..... administer the LP spooling system
lpsched ..... start/stop the LP request scheduler and move requests
makekey ..... generate encryption key
mkdev ..... make device files
mkfs ..... construct a file system
mklp ..... configure the LP spooler system
mknod ..... create special, fifo, files
mount ..... mount and unmount file system
mvdir ..... move a directory
ncheck ..... generate names from i-numbers
newfs ..... construct a new file system
opx25 ..... execute HALGOL programs
osck ..... check integrity of OS in SDF boot area(s)
oscp ..... copy, create, append to, split operating system
osmark ..... mark SDF OS file as loadable/unloadable
osmgr ..... operating system manager package description
pwck ..... password/group file checkers
reboot ..... reboot the system

revck ..... check internal revision numbers of HP-UX files
rootmark ..... mark/unmark volume as HP-UX root volume
runacct ..... run daily accounting
sdfinit ..... initialize Structured Directory Format volume
setmnt ..... establish mnttab table
setprivgrp ..... set special attributes for group
shutdown ..... terminate all processing
stopsys ..... stop operating system with optional reboot
swapon ..... enable additional devices for swapping and paging
syncer ..... periodically sync for file system integrity
tic ..... terminfo compiler
tunefs ..... tune a file system
uconfig ..... system reconfiguration
umodem ..... XMODEM protocol file transfer program
untic ..... terminfo de-compiler
uucico ..... uucp copy in and copy out
uuclean ..... uucp spool directory clean-up
uusub ..... monitor uucp network
uuxqt ..... uucp command execution
wall ..... write to all users
whodo ..... which users are doing what

2. System Calls

access ..... determine accessibility of a file
alarm ..... set process's alarm clock
brk ..... change data segment space allocation
chdir ..... change working directory
chmod ..... change access mode of file
chown ..... change owner and group of a file
chroot ..... change root directory
close ..... close a file descriptor
creat ..... create new file, rewrite existing file
dup ..... duplicate an open file descriptor
dup2 ..... duplicate an open file descriptor
ems ..... Extended Memory System
errinfo ..... error indicator
errno ..... error indicator for system calls
exec ..... exccute a file
exit ..... terminate process
fcntl ..... file control
fork ..... create a new process
fsync ..... synchronize a file's in-core state with that on disc
ftime ..... get date and time more precisely
getgroups ..... get group access list
gethostname ..... get name of current host
getitimer ..... get/set value of interval timer
getpid ..... get process, process group, and parent process IDs
getprivgrp ..... get/set special attributes for group
gettimeofday ..... get/set date and time
getuid ..... get real/effective user, real/effective group IDs
ioctl ..... control device
kill ..... send signal to process(s)
link ..... link to a file

lockf ..... provide semaphores and record locking on files
lseek ..... move read/write file pointer; seek
memadvise ..... advise OS about segment reference patterns
memallc ..... allocate and free address space
memchmd ..... change memory segment access modes
memlck ..... lock/unlock process address space or segment
memvary ..... modify segment length
mkdir ..... create a directory file
mknod ..... make directory, special or ordinary file
mount ..... mount a file system
msgctl ..... message control operations
msgget ..... get message queue
msgop ..... message operations
nice ..... change priority of a process
open ..... open file for reading or writing
pause ..... suspend process until signal
pipe ..... create an inter-process channel
plock ..... lock process, text, or data in memory
prealloc ..... preallocate fast disc storage
profil ..... execution time profile
ptrace ..... process trace
read ..... read from file
reboot ..... reboot the system
rmdir ..... remove a directory file
rtprio ..... change or read real-time priority
select ..... synchronous I/O multiplexing
semctl ..... semaphore control operations
semget ..... get set of semaphores
semop ..... semaphore operations
setgroups ..... set group access list
sethostname ..... set name of host cpu
setpgrp ..... set process group ID
setuid ..... set user and group IDs
shmctl ..... shared memory control operations
shmget ..... get shared memory segment
shmop ..... shared memory operations
sigblock ..... block signals
signal ..... set up signal handling for program
sigpause ..... automatically release blocked signals and wait for interrupt
sigsetmask ..... set current signal mask
sigspace ..... assure sufficient signal stack space
sigvector ..... software signal facilities
stat ..... get file status
stime ..... set time and date
stty ..... control device
swapon ..... add a swap device for interleaved paging/signalling
sync ..... update the super block
time ..... get time
times ..... get process and child process times
trapno ..... hardware trap numbers
truncate ..... truncate a file to a specified length
ulimit ..... get and set user limits
umask ..... get and set file creation mask
umount ..... unmount a file system
uname ..... get name of current HP-UX system

unlink ..... remove directory entry; delete file
ustat ..... get file system statistics
utime ..... set file access and modification times
vfork ..... spawn new process in a virtual memory efficient way
vsadv ..... advise system about backing store usage
vson ..... advise OS about backing store devices
wait ..... wait for child process to terminate
write ..... write on a file

3. Subroutines

a64l ..... convert between long and base-64 ASCII
abort ..... generate an IOT fault
abs ..... integer absolute value
assert ..... program verification
atof ..... convert ASCII to numbers
bessel ..... bessel functions
bsearch ..... binary search on a sorted table
catread ..... MPE/RTE-style message catalog support
clock ..... report CPU time used
conv ..... character translation
crypt ..... DES encryption
ctermid ..... generate file name for terminal
ctime ..... convert date and time to ASCII
ctype ..... character classification
curses ..... CRT screen handling and optimization routines
cuserid ..... character login name of the user
dial ..... establish an out-going terminal line connection
directory ..... directory operations
drand48 ..... generate uniformly-distributed pseudo-random numbers
ecvt ..... output conversion
end ..... last locations in program
erf ..... error function and complementary error function
exp ..... exponential, logarithm, power, square root functions
fclose ..... close or flush a stream
ferror ..... stream file status inquiries
floor ..... absolute value, floor, ceiling, remainder functions
fopen ..... open or re-open a stream file; convert file to stream
fread ..... buffered binary input/output to a stream file
frexp ..... split into mantissa and exponent
fseek ..... reposition a stream
ftw ..... walk a file tree
gamma ..... log gamma function
getc ..... get character or word from stream file
getcwd ..... get pathname of current working directory
getenv ..... value for environment name
getfsent ..... get file system descriptor file entry
getgrent ..... get group file entry
getlogin ..... get login name
getmsg ..... get message from a catalog
getopt ..... get option letter from argv
getpass ..... read a password
getpw ..... get name from UID
getpwent ..... get password file entry

gets ..... get a string from a stream file
getut ..... access utmp file entry
gpio_get_status ..... return status lines of GPIO card
gpio_set_ctl ..... set control lines on GPIO card
hpib_abort ..... stop activity on specified HP-IB bus
hpib_bus_status ..... return status of HP-IB interface
hpib_card_ppoll_resp ..... control response to parallel poll on HP-IB
hpib_eoi_ctl ..... control EOI mode for HP-IB file
hpib_io ..... perform I/O with an HP-IB channel from buffers
hpib_pass_ctl ..... change active controllers on HP-IB
hpib_ppoll ..... conduct parallel poll on HP-IB bus
hpib_ppoll_resp_ctl ..... control response to parallel poll on HP-IB
hpib_ren_ctl ..... control the Remote Enable line on HP-IB
hpib_rqst_srvce ..... allow interface to enable SRQ line on HP-IB
hpib_send_cmnd ..... send command bytes over HP-IB
hpib_spoll ..... conduct a serial poll on HP-IB bus
hpib_status_wait ..... wait until the requested status condition becomes true
hpib_wait_on_ppoll ..... wait until a particular parallel poll value occurs
hsearch ..... manage hash search tables
hypot ..... Euclidean distance
initgroups ..... initialize group access list
intrapoff ..... disable/enable integer trap handler
io_burst ..... perform low-overhead I/O on an HP-IB channel
io_eol_ctl ..... set up read termination character on special file
io_get_term_reason ..... determine how last read terminated
io_interrupt_ctl ..... enable/disable interrupts for associated eid
io_on_interrupt ..... device interrupt (fault) control
io_reset ..... reset an I/O interface
io_speed_ctl ..... inform system of required transfer speed
io_timeout_ctl ..... establish time limit for I/O operations
io_width_ctl ..... set width of data path
l3tol ..... convert between 3-byte integers and long integers
langinfo ..... NLS native language information
logname ..... return login name of user
lsearch ..... linear search and update
malloc ..... main memory allocator
matherr ..... mathematical error handling
memory ..... memory operations
mktemp ..... make a unique file name
monitor ..... prepare execution profile
nl_conv ..... translate characters for use with NLS
nl_ctype ..... classify characters for use with NLS
nl_string ..... non-ASCII string collation used by NLS
nlist ..... get entries from name list
perror ..... system error messages
popen ..... initiate pipe I/O to/from a process
printf ..... output formatters
printmsg ..... print formatted output with numbered arguments
putc ..... put character or word on a stream
putenv ..... change or add value to environment
putpwent ..... write password file entry
puts ..... put a string on a stream file
qsort ..... quicker sort
rand ..... random number generator
regcmp ..... compile and execute regular expression

scanf ..... formatted input conversion, read from stream file
setbuf ..... assign buffering to a stream file
setjmp ..... non-local goto
sinh ..... hyperbolic functions
sleep ..... suspend execution for interval
sputl ..... access long integer data in machine-independent manner
ssignal ..... software signals
stdio ..... standard buffered input/output stream file package
stdipc ..... stamdard inter-process communication package
string ..... character string operations
strtod ..... convert string to double-precision integer
strtol ..... convert string to integer
swab ..... swap bytes
system ..... issue a shell command
termcap ..... access terminal capabilities in termcap(5)
tmpfile ..... create a temporary file
tmpnam ..... create a name for a temporary file
trig ..... trigonometric functions
tsearch ..... manage binary search trees
ttyname ..... find name of a terminal
ttyslot ..... find current user slot in utmp file
ungetc ..... push character back into input stream
vprintf ..... print formatted output from varargs argument list

4. Special Files

ct ..... CS/80 cartridge tape access
disc ..... direct disc access
graphics ..... information for crt graphics devices
hpib ..... hpib interface information
iomap ..... physical address mapping
lp ..... printer information
mem ..... core memory
modem ..... asynchronous serial modem line control
mt ..... magnetic tape interface and controls
null ..... null file ("bit bucket")
pty ..... pseudo-terminal driver
sttyv6 ..... version 6/PWD-compatibility terminal interface
termio ..... general terminal interface
tty ..... controlling terminal interface

5. File Formats

a.out ..... assembler and link editor output
acct ..... per-process accounting file format
ar ..... archive file format
bif ..... Bell Interchange Format file utilties
checklist ..... list of file systems processed by fsck
col_seq_8 ..... collating sequence tables for 8-bit NLS character sets
col_seq_16 ..... collating sequence tables for 16-bit NLS character sets
core ..... format of core image file
cpio ..... format of cpio archive
dialups ..... dialup security control

dir ..... SDF directory format
disktab ..... disc description file
errfile ..... system error logging file
fs ..... format of system volume
fspec ..... format specification in text files
gettydefs ..... speed and terminal settings used by getty(1M)
group ..... group file
inittab ..... control information for init(1M)
inode ..... format of an i-node
issue ..... issue identification file
lif ..... Logical Interchange Format description
magic ..... magic numbers for HP-UX implementations
master ..... master device information table
mknod ..... create a special file entry
mnttab ..... mounted file system table
model ..... HP-UX machine identification
nlist ..... nlist structure format
passwd ..... password file
privgrp ..... privileged values format
profile ..... set up user's environment at login time
ranlib ..... table of contents format for object libraries
sccsfile ..... format of SCCS file
term ..... compiled term file format
terminfo ..... terminal capability data base
ttytype ..... data base of terminal types by port
utmp ..... utmp and wtmp entry format

6. Games

No games are currently supported.

7. Miscellaneous Facilities

ascii ..... map of ASCII character set
environ ..... user environment
fcntl ..... file control options
hier ..... file system hierarchy
hpnls ..... Native Language Support model
kana8 ..... map of KANA8 character set used by NLS
langid ..... language identification variable used by NLS
man ..... macros for formatting entries in this manual
math ..... math functions and constants
mm ..... the MM macro package for formatting documents
regexp ..... regular expression compile and match routines
roman8 ..... ROMAN8 character set used by NLS
stat ..... data returned by stat/fstat system call
term ..... conventional device names
types ..... primitive system data types
values ..... machine-dependent values
varargs ..... handle-variable-argument list

9. Glossary

NAME

m4 - macro processor

SYNOPSIS

m4 [options] [files]

HP–UX COMPATIBILITY

Level: HP–UX/DEVELOPMENT

Origin: System V

DESCRIPTION

M4 is a macro processor intended as a front end for Ratfor, C, and other languages. Each of the argument files is processed in order; if there are no files, or if a file name is **-**, the standard input is read. The processed text is written on the standard output.

The options and their effects are as follows:

-e Operate interactively. Interrupts are ignored and the output is unbuffered. Using this mode requires a special state of mind.

-s Enable line sync output for the C preprocessor (#line ...)

-B*int* Change the size of the push–back and argument collection buffers from the default of 4,096.

-H*int* Change the size of the symbol table hash array from the default of 199. The size should be prime.

-S*int* Change the size of the call stack from the default of 100 slots. Macros take three slots, and non–macro arguments take one.

-T*int* Change the size of the token buffer from the default of 512 bytes.

To be effective, these flags must appear before any file names and before any **-D** or **-U** flags:

-D*name*[=*val*]
Defines *name* to *val* or to null in *val*'s absence.

-U*name*
undefines *name*.

Macro calls have the form:

name(arg1,arg2, ..., argn)

The (must immediately follow the name of the macro. If the name of a defined macro is not followed by a (, it is deemed to be a call of that macro with no arguments. Potential macro names consist of alphabetic letters, digits, and underscore _, where the first character is not a digit.

Leading unquoted blanks, tabs, and new–lines are ignored while collecting arguments. Left and right single quotes are used to quote strings. The value of a quoted string is the string stripped of the quotes.

When a macro name is recognized, its arguments are collected by searching for a matching right parenthesis. If fewer arguments are supplied than are in the macro definition, the trailing arguments are taken to be null. Macro evaluation proceeds normally during the collection of the arguments, and any commas or right parentheses which happen to turn up within the value of a nested call are as effective as those in the original input text. After argument collection, the value of the macro is pushed back onto the input stream and rescanned.

M4 makes available the following built–in macros. They may be redefined, but once this is done the original meaning is lost. Their values are null unless otherwise stated.

define the second argument is installed as the value of the macro whose name is the first argument. Each occurrence of **$***n* in the replacement text, where *n* is a digit, is

replaced by the *n*–th argument. Argument 0 is the name of the macro; missing arguments are replaced by the null string; **$#** is replaced by the number of arguments; **$*** is replaced by a list of all the arguments separated by commas; **$@** is like **$***, but each argument is quoted (with the current quotes). All **$** metavariable replacement ignores quoting; every occurrence of a **$** metavariable is replaced.

undefine removes the definition of the macro named in its argument.

defn returns the quoted definition of its argument(s). It is useful for renaming macros, especially built–ins.

pushdef like *define*, but saves any previous definition.

popdef removes current definition of its argument(s), exposing the previous one if any.

ifdef if the first argument is defined, the value is the second argument, otherwise the third. If there is no third argument, the value is null. The word *unix* is predefined on the HP–UX System versions of *m4*.

shift returns all but its first argument. The other arguments are quoted and pushed back with commas in between. The quoting nullifies the effect of the extra scan that will subsequently be performed.

changequote change quote symbols to the first and second arguments. The symbols may be up to five characters long. *Changequote* without arguments restores the original values (i.e., ` ´).

changecom change left and right comment markers from the default # and new–line. With no arguments, the comment mechanism is effectively disabled. With one argument, the left marker becomes the argument and the right marker becomes new–line. With two arguments, both markers are affected. Comment markers may be up to five characters long.

divert *m4* maintains 10 output streams, numbered 0–9. The final output is the concatenation of the streams in numerical order; initially stream 0 is the current stream. The *divert* macro changes the current output stream to its (digit–string) argument. Output diverted to a stream other than 0 through 9 is discarded.

undivert causes immediate output of text from diversions named as arguments, or all diversions if no argument. Text may be undiverted into another diversion. Undiverting discards the diverted text.

divnum returns the value of the current output stream.

dnl reads and discards characters up to and including the next new–line.

ifelse has three or more arguments. If the first argument is the same string as the second, then the value is the third argument. If not, and if there are more than four arguments, the process is repeated with arguments 4, 5, 6 and 7. Otherwise, the value is either the fourth string, or, if it is not present, null.

incr returns the value of its argument incremented by 1. The value of the argument is calculated by interpreting an initial digit–string as a decimal number. Overflow is not detected.

decr returns the value of its argument decremented by 1. Underflow is not detected.

eval evaluates its argument as an arithmetic expression, using 32–bit arithmetic. Operators include +, -, *, /, %, ** (exponentiation), bitwise &, |, ^, and ~; relationals; parentheses. Octal and hex numbers may be specified as in C. The second argument specifies the radix for the result; the default is 10. The third argument may be used to specify the minimum number of digits in the result. Overflow and underflow are not detected.

len	returns the number of characters in its argument.
index	returns the position in its first argument where the second argument begins (zero origin), or -1 if the second argument does not occur. A null (or missing) second argument causes index to return 0.
substr	returns a substring of its first argument. The second argument is a zero origin number selecting the first character; the third argument indicates the length of the substring. A missing third argument is taken to be large enough to extend to the end of the first string.
translit	transliterates the characters in its first argument from the set given by the second argument to the set given by the third. No abbreviations are permitted.
include	returns the contents of the file named in the argument.
sinclude	is identical to *include*, except that it says nothing if the file is inaccessible.
syscmd	executes the HP–UX System command given in the first argument. No value is returned.
sysval	is the return code from the last call to *syscmd*.
maketemp	calls mktemp(3) to fill in a string of XXXXX in its argument with the current process ID.
m4exit	causes immediate exit from *m4*. Argument 1, if given, is the exit code; the default is 0.
m4wrap	argument 1 will be pushed back at final EOF; example: m4wrap(`cleanup()')
errprint	prints its argument to *stderr*.
dumpdef	prints current names and definitions, for the named items, or for all if no arguments are given. Output goes to *stderr*.
traceon	with no arguments, turns on tracing for all macros (including built–ins). Otherwise, turns on tracing for named macros.
traceoff	turns off trace globally and for any macros specified. Macros specifically traced by *traceon* can be untraced only by specific calls to *traceoff*.

SEE ALSO

mktemp(3).

NAME

hp9000s200, hp9000s500, pdp11, u3b, u3b5, vax - provide truth value about your processor type

SYNOPSIS

hp9000s200

hp9000s500

pdp11

u3b

u3b5

vax

HP–UX COMPATIBILITY

Level: HP–UX/STANDARD

Origin: System V

DESCRIPTION

The following commands will return a true value (exit code of 0) if you are on a processor that the command name indicates.

hp9000s200
: True if you are on a Hewlett–Packard 9000 Series 200.

hp9000s500
: True if you are on a Hewlett–Packard 9000 Series 500.

pdp11
: True if you are on a PDP–11/45 or PDP–11/70.

u3b
: True if you are on a 3B 20S computer.

u3b5
: True if you are on a 3B 5 computer.

vax
: True if you are on a VAX–11/750 or VAX–11/780.

The commands that do not apply will return a false (non–zero) value. These commands are often used within *make*(1) makefiles and shell procedures to increase portability.

SEE ALSO

make(1), sh(1), test(1), true(1).

NAME

mail, rmail - send mail to users or read mail

SYNOPSIS

mail [**-epqr**] [**-f** file]

mail [**-t**] persons

rmail [**-t**] persons

HP-UX COMPATIBILITY

Level: HP-UX/STANDARD

Origin: System III

Native Language Support:
8-bit and 16-bit data.

DESCRIPTION

Note: An enhanced user mail interface is presented in *mailx*(1).

Mail without arguments prints a user's mail, message-by-message, in last-in, first-out order. For each message, the user is prompted with a **?**, and a line is read from the standard input to determine the disposition of the message:

<new-line>	Go on to next message.
+	Same as <new-line>.
d	Delete message and go on to next message.
p	Print message again.
-	Go back to previous message.
s [*files*]	Save message in the named *files* (**mbox** is default).
w [*files*]	Save message, without its header, in the named *files* (**mbox** is default).
m [*persons*]	Mail the message to the named *persons* (yourself is default).
q	Put undeleted mail back in the *mailfile* and stop.
EOT (control-d)	Same as **q**.
x	Put all mail back in the *mailfile* unchanged and stop.
!*command*	Escape to the shell to do *command*.
*****	Print a command summary.

The optional arguments alter the printing of the mail:

-e causes mail not to be printed. An exit value is returned:
0 = mail present
1 = no mail
2 = other error

-p causes all mail to be printed without prompting for disposition.

-q causes *mail* to terminate after interrupts. Normally an interrupt only causes the termination of the printing of the current message.

-r causes messages to be printed in first-in, first-out order.

-f*file* causes *mail* to use *file* (e.g., **mbox**) instead of the default *mailfile*.

When *persons* are named, *mail* takes the standard input up to an end-of-file (or up to a line consisting of just a **.**) and adds it to each *person*'s *mailfile*. The message is preceded by the sender's name and a postmark. Lines that look like postmarks in the message, (i.e., "From ...") are preceded with a >. The **-t** option causes the message to be preceded by all *persons* the *mail* is sent to. A *person* is usually a user name recognized by *login*(1). If a *person* being sent mail is not recognized, or if *mail* is interrupted during input, the file **dead.letter** will be saved to allow editing and resending. Note that this is regarded as a temporary file in that it is recreated every time needed, erasing the previous contents of **dead.letter**.

To denote a recipient on a remote system, prefix *person* by the system name and exclamation mark (see *uucp*(1C)). Everything after the first exclamation mark in *persons* is interpreted by

the remote system. In particular, if *persons* contains additional exclamation marks, it can denote a sequence of machines through which the message is to be sent on the way to its ultimate destination. For example, specifying **a!b!cde** as a recipient's name causes the message to be sent to user **b!cde** on system **a**. System **a** will interpret that destination as a request to send the message to user **cde** on system **b**. This might be useful, for instance, if the sending system can access system **a** but not system **b**, and system **a** has access to system **b**. *Mail* will not use *uucp* if the remote system is the local system name (i.e., localsystem!user).

The *mailfile* may be manipulated in two ways to alter the function of *mail*. The *other* permissions of the file may be read–write, read–only, or neither read nor write to allow different levels of privacy. If changed to other than the default, the file will be preserved even when empty to perpetuate the desired permissions. The file may also contain the first line:

Forward to *person*

which will cause all mail sent to the owner of the *mailfile* to be forwarded to *person*. This is especially useful to forward all of a person's mail to one machine in a multiple machine environment. In order for forwarding to work properly the *mailfile* should have "mail" as group ID, and the group permission should be read–write.

Rmail only permits the sending of mail; *uucp*(1C) uses *rmail* as a security precaution.

When a user logs in, the presence of mail, if any, is indicated. Also, notification is made if new mail arrives while using *mail*.

FILES

/etc/passwd	to identify sender and locate persons
/usr/mail/*user*	incoming mail for *user*; i.e., the *mailfile*
$HOME/mbox	saved mail
$MAIL	variable containing path name of *mailfile*
/tmp/ma*	temporary file
/usr/mail/*.lock	lock for mail directory
dead.letter	unmailable text

SEE ALSO

login(1), mailx(1), uucp(1C), write(1).

BUGS

Conditions sometimes result in a failure to remove a lock file.
After an interrupt, the next message may not be printed; printing may be forced by typing a **p**.

NAME

mailx - interactive message processing system

SYNOPSIS

mailx [*options*] [*name...*]

HP–UX COMPATIBILITY

Level: HP–UX/STANDARD

Origin: UCB

Remarks: Not supported on the Integral Personal Computer.

DESCRIPTION

The command *mailx* provides a comfortable, flexible environment for sending and receiving mes sages electronically. When reading mail, *mailx* provides commands to facilitate saving, deleting, and responding to messages. When sending mail, *mailx* allows editing, reviewing and other modification of the message as it is entered.

Incoming mail is stored in a standard file for each user, called the system *mailbox* for that user. When *mailx* is called to read messages, the *mailbox* is the default place to find them. As messages are read, they are marked to be moved to a secondary file for storage, unless specific action is taken, so that the messages need not be seen again. This secondary file is called the *mbox* and is normally located in the user's HOME directory (see "MBOX" (ENVIRONMENT VARIABLES) for a description of this file). Messages remain in this file until forcibly removed.

On the command line, *options* start with a dash (-) and any other arguments are taken to be des–tinations (recipients). If no recipients are specified, *mailx* will attempt to read messages from the *mailbox*. Command line options are:

-d	Turn on debugging output. Neither particularly interesting nor recom–mended.
-e	Test for presence of mail. *Mailx* prints nothing and exits with a success–ful return code if there is mail to read.
-f [*filename*]	Read messages from *filename* instead of *mailbox*. If no *filename* is specified, the *mbox* is used.
-F	Record the message in a file named after the first recipient. Overrides the "record" variable, if set (see ENVIRONMENT VARIABLES).
-h *number*	The number of network "hops" made so far. This is provided for network software to avoid infinite delivery loops.
-H	Print header summary only.
-i	Ignore interrupts. See also "ignore" (ENVIRONMENT VARIABLES).
-n	Do not initialize from the system default *Mailx.rc* file.
-N	Do not print initial header summary.
-r *address*	Pass *address* to network delivery software. All tilde commands are dis–abled.
-s *subject*	Set the Subject header field to *subject*.
-u *user*	Read *user*'s *mailbox*. This is only effective if *user*'s *mailbox* is not read protected.
-U	Convert *uucp* style addresses to internet standards. Overrides the "conv" environment variable.

When reading mail, *mailx* is in *command mode*. A header summary of the first several messages is displayed, followed by a prompt indicating *mailx* can accept regular commands (see COM–MANDS below). When sending mail, *mailx* is in *input mode*. If no subject is specified on the command line, a prompt for the subject is printed. As the message is typed, *mailx* will read the message and store it in a temporary file. Commands may be entered by beginning a line with the

tilde (~) escape character followed by a single command letter and optional arguments. See TILDE ESCAPES for a summary of these commands.

At any time, the behavior of *mailx* is governed by a set of *environment variables*. These are flags and valued parameters which are set and cleared via the **set** and **unset** commands. See ENVIRONMENT VARIABLES below for a summary of these parameters.

Recipients listed on the command line may be of three types: login names, shell commands, or alias groups. Login names may be any network address, including mixed network addressing. If the recipient name begins with a pipe symbol (|), the rest of the name is taken to be a shell command to pipe the message through. This provides an automatic interface with any program that reads the standard input, such as *lp*(1) for recording outgoing mail on paper. Alias groups are set by the **alias** command (see COMMANDS below) and are lists of recipients of any type.

Regular commands are of the form

[**command**] [*msglist*] [*arguments*]

If no command is specified in *command mode*, **print** is assumed. In *input mode*, commands are recognized by the escape character, and lines not treated as commands are taken as input for the message.

Each message is assigned a sequential number, and there is at any time the notion of a 'current' message, marked by a '>' in the header summary. Many commands take an optional list of messages (*msglist*) to operate on, which defaults to the current message. A *msglist* is a list of message specifications separated by spaces, which may include:

n	Message number **n**.
.	The current message.
^	The first undeleted message.
$	The last message.
*****	All messages.
n-m	An inclusive range of message numbers.
user	All messages from **user**.
/string	All messages with **string** in the subject line (case ignored).
:*c*	All messages of type *c*, where *c* is one of:

d	deleted messages
n	new messages
o	old messages
r	read messages
u	unread messages

Note that the context of the command determines whether this type of message specification makes sense.

Other arguments are usually arbitrary strings whose usage depends on the command involved. File names, where expected, are expanded via the normal shell conventions (see *sh*(1)). Special characters are recognized by certain commands and are documented with the commands below.

At start-up time, *mailx* reads commands from a system-wide file (**/usr/lib/mailx/mailx.rc**) to initialize certain parameters, then from a private start-up file (**$HOME/.mailrc**) for personalized variables. Most regular commands are legal inside start-up files, the most common use being to set up initial display options and alias lists. The following commands are not legal in the start-up file: **!**, **C**opy, **e**dit, **fo**llowup, **F**ollowup, **ho**ld, **m**ail, **pre**serve, **r**eply, **R**eply, **sh**ell, and **v**isual. Any errors in the start-up file cause the remaining lines in the file to be ignored.

COMMANDS

The following is a complete list of *mailx* commands:

!*shell-command*
Escape to the shell. See "SHELL" (ENVIRONMENT VARIABLES).

 # comment
Null command (comment). This may be useful in *.mailrc* files.

=
Print the current message number.

?
Prints a summary of commands.

alias *alias name ...*
group *alias name ...*
Declare an alias for the given names. The names will be substituted when *alias* is used as a recipient. Useful in the *.mailrc* file.

alternates *name ...*
Declares a list of alternate names for your login. When responding to a message, these names are removed from the list of recipients for the response. With no arguments, **alt**ernates prints the current list of alternate names. See also "allnet" (ENVIRONMENT VARIABLES).

cd [*directory*]
chdir [*directory*]
Change directory. If *directory* is not specified, $HOME is used.

copy [*filename*]
copy [*msglist*] *filename*
Copy messages to the file without marking the messages as saved. Otherwise equivalent to the **s**ave command.

Copy [*msglist*]
Save the specified messages in a file whose name is derived from the author of the message to be saved, without marking the messages as saved. Otherwise equivalent to the **S**ave command.

delete [*msglist*]
Delete messages from the *mailbox*. If "autoprint" is set, the next message after the last one deleted is printed (see ENVIRONMENT VARIABLES).

discard [*header-field ...*]
ignore [*header-field ...*]
Suppresses printing of the specified header fields when displaying messages on the screen. Examples of header fields to ignore are "status" and "cc." The fields are included when the message is saved. The **P**rint and **T**ype commands override this command.

dp [*msglist*]
dt [*msglist*]
Delete the specified messages from the *mailbox* and print the next message after the last one deleted. Roughly equivalent to a **d**elete command followed by a **p**rint command.

echo *string ...*
Echo the given strings (like *echo*(1)).

edit [*msglist*]
Edit the given messages. The messages are placed in a temporary file and the "EDITOR" variable is used to get the name of the editor (see ENVIRONMENT VARIABLES). Default editor is *ed*(1).

exit
xit
Exit from *mailx*, without changing the *mailbox*. No messages are saved in the *mbox* (see also **q**uit).

file [*filename*]
folder [*filename*]
Quit from the current file of messages and read in the specified file. Several special characters are recognized when used as file names, with the following substitutions:

%	the current *mailbox*.
%**user**	the *mailbox* for **user**.
#	the previous file.
&	the current *mbox*.

Default file is the current *mailbox*.

folders
Print the names of the files in the directory set by the "folder" variable (see ENVIRONMENT VARIABLES).

followup [*message*]
Respond to a message, recording the response in a file whose name is derived from the author of the message. Overrides the "record" variable, if set. See also the **F**ollowup, **S**ave, and **C**opy commands and "outfolder" (ENVIRONMENT VARIABLES).

Followup [*msglist*]
Respond to the first message in the *msglist*, sending the message to the author of each message in the *msglist*. The subject line is taken from the first message and the response is recorded in a file whose name is derived from the author of the first message. See also the **fo**llowup, **S**ave, and **C**opy commands and "outfolder" (ENVIRONMENT VARIABLES).

from [*msglist*]
Prints the header summary for the specified messages.

group *alias name ...*
alias *alias name ...*
Declare an alias for the given names. The names will be substituted when *alias* is used as a recipient. Useful in the *.mailrc* file.

headers [*message*]
Prints the page of headers which includes the message specified. The "screen" variable sets the number of headers per page (see ENVIRONMENT VARIABLES). See also the **z** command.

help
Prints a summary of commands.

hold [*msglist*]
preserve [*msglist*]
Holds the specified messages in the *mailbox*.

if *s*|*r*
mail-commands
else
mail-commands
endif
Conditional execution, where *s* will execute following *mail-commands*, up to an **el**se or **en**dif, if the program is in *send* mode, and *r* causes the *mail-commands* to be executed only in *receive* mode. Useful in the *.mailrc* file.

ignore *header-field ...*
discard *header-field ...*
Suppresses printing of the specified header fields when displaying messages on the screen. Examples of header fields to ignore are "status" and "cc." All fields are included when the message is saved. The **P**rint and **T**ype commands override this command.

list
Prints all commands available. No explanation is given.

mail *name ...*
Mail a message to the specified users.

mbox [*msglist*]
Arrange for the given messages to end up in the standard *mbox* save file when *mailx* terminates normally. See "MBOX" (ENVIRONMENT VARIABLES) for a description of this file. See also the **ex**it and **q**uit commands.

next [*message*]
Go to next message matching *message*. A *msglist* may be specified, but in this case the first valid message in the list is the only one used. This is useful for jumping to the next message from a specific user, since the name would be taken as a command in the absence of a real command. See the discussion of *msglists* above for a description of possible message specifications.

pipe [*msglist*] [*shell-command*]
| [*msglist*] [*shell-command*]
Pipe the message through the given *shell-command*. The message is treated as if it were read. If no arguments are given, the current message is piped through the command specified by the value of the "cmd" variable. If the "page" variable is set, a form feed character is inserted after each message (see ENVIRONMENT VARIABLES).

preserve [*msglist*]
hold [*msglist*]
Preserve the specified messages in the *mailbox*.

Print [*msglist*]
Type [*msglist*]
Print the specified messages on the screen, including all header fields. Overrides suppression of fields by the **ig**nore command.

print [*msglist*]
type [*msglist*]
Print the specified messages. If "crt" is set, the messages longer than the number of lines specified by the "crt" variable are paged through the command specified by the "PAGER" variable. The default command is *pg*(1) (see ENVIRONMENT VARIABLES).

quit
Exit from *mailx*, storing messages that were read in *mbox* and unread messages in the *mailbox*. Messages that have been explicitly saved in a file are deleted.

Reply [*msglist*]
Respond [*msglist*]
Send a response to the author of each message in the *msglist*. The subject line is taken from the first message. If "record" is set to a filename, the response is saved at the end of that file (see ENVIRONMENT VARIABLES).

reply [*message*]
respond [*message*]
Reply to the specified message, including all other recipients of the message. If "record" is set to a filename, the response is saved at the end of that file (see ENVIRONMENT VARIABLES).

Save [*msglist*]
Save the specified messages in a file whose name is derived from the author of the first message. The name of the file is taken to be the author's name with all network addressing stripped off. See also the **C**opy, **fo**llowup, and **F**ollowup commands and "outfolder" (ENVIRONMENT VARIABLES).

save [*filename*]
save [*msglist*] *filename*
Save the specified messages in the given file. The file is created if it does not exist. The message is deleted from the *mailbox* when *mailx* terminates unless "keepsave" is set (see also ENVIRONMENT VARIABLES and the **ex**it and **q**uit commands).

set
set *name*
set *name=string*
set *name=number*
Define a variable called *name*. The variable may be given a null, string, or numeric value. **Se**t by itself prints all defined variables and their values. See ENVIRONMENT VARIABLES for detailed descriptions of the *mailx* variables.

shell
Invoke an interactive shell (see also "SHELL" (ENVIRONMENT VARIABLES)).

size [*msglist*]
Print the size in characters of the specified messages.

source *filename*
Read commands from the given file and return to command mode.

top [*msglist*]
Print the top few lines of the specified messages. If the "toplines" variable is set, it is taken as the number of lines to print (see ENVIRONMENT VARIABLES). The default is 5.

touch [*msglist*]
Touch the specified messages. If any message in *msglist* is not specifically saved in a file, it will be placed in the *mbox* upon normal termination. See **ex**it and **q**uit.

Type [*msglist*]
Print [*msglist*]
Print the specified messages on the screen, including all header fields. Overrides suppression of fields by the **ig**nore command.

type [*msglist*]
print [*msglist*]
Print the specified messages. If "crt" is set, the messages longer than the number of lines specified by the "crt" variable are paged through the command specified by the "PAGER" variable. The default command is *pg*(1) (see ENVIRONMENT VARIABLES).

undelete [*msglist*]
Restore the specified deleted messages. Will only restore messages deleted in the current mail session. If "autoprint" is set, the last message of those restored is printed (see ENVIRONMENT VARIABLES).

unset *name ...*
Causes the specified variables to be erased. If the variable was imported from the execution environment (i.e., a shell variable) then it cannot be erased.

version
Prints the current version and release date.

visual [*msglist*]
Edit the given messages with a screen editor. The messages are placed in a temporary file and the "VISUAL" variable is used to get the name of the editor (see ENVIRONMENT VARIABLES).

write [*msglist*] *filename*
Write the given messages on the specified file, minus the header and trailing blank line. Otherwise equivalent to the **s**ave command.

xit
exit
Exit from *mailx*, without changing the *mailbox*. No messages are saved in the *mbox* (see also **q**uit).

z[+|-]
Scroll the header display forward or backward one screen-full. The number of headers displayed is set by the "screen" variable (see ENVIRONMENT VARIABLES).

TILDE ESCAPES
The following commands may be entered only from *input mode*, by beginning a line with the tilde

escape character (~). See "escape" (ENVIRONMENT VARIABLES) for changing this special character.

~! *shell–command*
Escape to the shell.

~.
Simulate end of file (terminate message input).

~: *mail–command*
~_ *mail–command*
Perform the command–level request. Valid only when sending a message while reading mail.

~?
Print a summary of tilde escapes.

~A
Insert the autograph string "Sign" into the message (see ENVIRONMENT VARIABLES).

~a
Insert the autograph string "sign" into the message (see ENVIRONMENT VARIABLES).

~b *name ...*
Add the *name*s to the blind carbon copy (Bcc) list.

~c *name ...*
Add the *name*s to the carbon copy (Cc) list.

~d
Read in the *dead.letter* file. See "DEAD" (ENVIRONMENT VARIABLES) for a description of this file.

~e
Invoke the editor on the partial message. See also "EDITOR" (ENVIRONMENT VARIABLES).

~f [*msglist*]
Forward the specified messages. The messages are inserted into the message, without alteration.

~h
Prompt for Subject line and To, Cc, and Bcc lists. If the field is displayed with an initial value, it may be edited as if you had just typed it.

~i *string*
Insert the value of the named variable into the text of the message. For example, ~A is equivalent to '~i Sign.'

~m [*msglist*]
Insert the specified messages into the letter, shifting the new text to the right one tab stop. Valid only when sending a message while reading mail.

~**p**

Print the message being entered.

~**q**

Quit from input mode by simulating an interrupt. If the body of the message is not null, the partial message is saved in *dead.letter*. See "DEAD" (ENVIRONMENT VARIABLES) for a description of this file.

~**r** *filename*
~< *filename*
~< !*shell-command*

Read in the specified file. If the argument begins with an exclamation point (!), the rest of the string is taken as an arbitrary shell command and is executed, with the standard output inserted into the message.

~**s** *string ...*

Set the subject line to *string*.

~**t** *name ...*

Add the given *name*s to the To list.

~**v**

Invoke a preferred screen editor on the partial message. See also "VISUAL" (ENVIRONMENT VARIABLES).

~**w** *filename*

Write the partial message onto the given file, without the header.

~**x**

Exit as with ~**q** except the message is not saved in *dead.letter*.

~| *shell-command*

Pipe the body of the message through the given *shell-command*. If the *shell-command* returns a successful exit status, the output of the command replaces the message.

ENVIRONMENT VARIABLES

The following are environment variables taken from the execution environment and are not alterable within *mailx*.

HOME=*directory*

The user's base of operations.

MAILRC=*filename*

The name of the start-up file. Default is $HOME/.mailrc.

The following variables are internal *mailx* variables. They may be imported from the execution environment or set via the **se**t command at any time. The **uns**et command may be used to erase variables.

allnet

All network names whose last component (login name) match are treated as identical. This causes the *msglist* message specifications to behave similarly. Default is **noallnet**. See also the **alt**ernates command and the "metoo" variable.

append
Upon termination, append messages to the end of the *mbox* file instead of prepending them. Default is **noappend.**

askcc
Prompt for the Cc list after message is entered. Default is **noaskcc**.

asksub
Prompt for subject if it is not specified on the command line with the **-s** option. Enabled by default.

autoprint
Enable automatic printing of messages after **d**elete and **u**ndelete commands. Default is **noautoprint**.

bang
Enable the special–casing of exclamation points (!) in shell escape command lines as in *vi*(1). Default is **nobang**.

cmd=*shell–command*
Set the default command for the **pi**pe command. No default value.

conv=*conversion*
Convert uucp addresses to the specified address style. The only valid conversion now is *internet*, which requires a mail delivery program conforming to the RFC822 standard for electronic mail addressing. Conversion is disabled by default. See also "sendmail" and the **-U** command line option.

crt=*number*
Pipe messages having more than *number* lines through the command specified by the value of the "PAGER" variable (*pg*(1) by default). Disabled by default.

DEAD=*filename*
The name of the file in which to save partial letters in case of untimely interrupt or delivery errors. Default is $HOME/dead.letter.

debug
Enable verbose diagnostics for debugging. Messages are not delivered. Default is **node–bug**.

dot
Take a period on a line by itself during input from a terminal as end–of–file. Default is **nodot**.

EDITOR=*shell–command*
The command to run when the **e**dit or ~**e** command is used. Default is *ed*(1).

escape=*c*
Substitute *c* for the ~ escape character.

folder=*directory*
The directory for saving standard mail files. User specified file names beginning with a plus (+) are expanded by preceding the filename with this directory name to obtain the real filename. If *directory* does not start with a slash (/), $HOME is prepended to it. In

order to use the plus (+) construct on a *mailx* command line, "folder" must be an exported *sh* environment variable. There is no default for the "folder" variable. See also "outfolder" below.

header
Enable printing of the header summary when entering *mailx*. Enabled by default.

hold
Preserve all messages that are read in the *mailbox* instead of putting them in the standard *mbox* save file. Default is **nohold**.

ignore
Ignore interrupts while entering messages. Handy for noisy dial-up lines. Default is **noignore**.

ignoreeof
Ignore end-of-file during message input. Input must be terminated by a period (.) on a line by itself or by the **~.** command. Default is **noignoreeof**. See also "dot" above.

keep
When the *mailbox* is empty, truncate it to zero length instead of removing it. Disabled by default.

keepsave
Keep messages that have been saved in other files in the *mailbox* instead of deleting them. Default is **nokeepsave**.

MBOX=*filename*
The name of the file to save messages which have been read. The **x**it command overrides this function, as does saving the message explicitly in another file. Default is $HOME/mbox.

metoo
If your login appears as a recipient, do not delete it from the list. Default is **nometoo**.

LISTER=*shell-command*
The command (and options) to use when listing the contents of the "folder" directory. The default is *ls*(1).

onehop
When responding to a message that was originally sent to several recipients, the other recipient addresses are normally forced to be relative to the originating author's machine for the response. This flag disables alteration of the recipients' addresses, improving efficiency in a network where all machines can send directly to all other machines (i.e., one hop away).

outfolder
Causes the files used to record outgoing messages to be located in the directory specified by the "folder" variable unless the pathname is absolute. Default is **nooutfolder**. See "folder" above and the **S**ave, **C**opy, **fo**llowup, and **F**ollowup commands.

page
Used with the **pi**pe command to insert a form feed after each message sent through the pipe. Default is **nopage**.

PAGER=*shell–command*
The command to use as a filter for paginating output. This can also be used to specify the options to be used. Default is *pg*(1).

prompt=*string*
Set the *command mode* prompt to *string*. Default is "? ".

quiet
Refrain from printing the opening message and version when entering *mailx*. Default is **noquiet**.

record=*filename*
Record all outgoing mail in *filename*. Disabled by default. See also "outfolder" above.

save
Enable saving of messages in *dead.letter* on interrupt or delivery error. See "DEAD" for a description of this file. Enabled by default.

screen=*number*
Sets the number of lines in a screen-full of headers for the **h**eaders command.

sendmail=*shell–command*
Alternate command for delivering messages. Default is *mail*(1).

sendwait
Wait for background mailer to finish before returning. Default is **nosendwait**.

SHELL=*shell–command*
The name of a preferred command interpreter. Default is *sh*(1).

showto
When displaying the header summary and the message is from you, print the recipient's name instead of the author's name.

sign=*string*
The variable inserted into the text of a message when the ~**a** (autograph) command is given. No default (see also ~**i** (TILDE ESCAPES)).

Sign=*string*
The variable inserted into the text of a message when the ~**A** command is given. No default (see also ~**i** (TILDE ESCAPES)).

toplines=*number*
The number of lines of header to print with the **top** command. Default is 5.

VISUAL=*shell–command*
The name of a preferred screen editor. Default is *vi*(1).

FILES

$HOME/.mailrc	personal start–up file
$HOME/mbox	secondary storage file
/usr/mail/*	post office directory
/usr/lib/mailx/mailx.help*	help message files
/usr/lib/mailx/mailx.rc	global start–up file

/tmp/R[emqsx]* temporary files

SEE ALSO

mail(1), pg(1), ls(1).

BUGS

Where *shell command* is shown as valid, arguments are not always allowed. Experimentation is recommended.

Internal variables imported from the execution environment cannot be **unse**t.

The full internet addressing is not fully supported by *mailx*. The new standards need some time to settle down.

Attempts to send a message having a line consisting only of a "." are treated as the end of the message by *mail*(1) (the standard mail delivery program).

NAME

make - maintain, update, and regenerate groups of programs

SYNOPSIS

make [**-f** makefile] [**-p**] [**-i**] [**-k**] [**-s**] [**-r**] [**-n**] [**-b**] [**-e**] [**-t**] [**-d**] [**-q**] [names]

HP-UX COMPATIBILITY

Level: HP-UX/STANDARD

Origin: System III

DESCRIPTION

The following is a brief description of all options and some special names. Options may occur in any order.

-f *makefile* Description file name. *Makefile* is assumed to be the name of a description file. A file name of **-** denotes the standard input. The contents of *makefile* override the built-in rules if they are present. Note that the space between **-f** and *makefile* **must** be present.

-p Print out the complete set of macro definitions and target descriptions.

-i Ignore error codes returned by invoked commands. This mode is also entered if the fake target name **.IGNORE** appears in the description file.

-k When a command returns nonzero status, abandon work on the current entry, but continue on other branches that do not depend on that entry.

-s Silent mode. Do not print command lines before executing. This mode is also entered if the fake target name **.SILENT** appears in the description file.

-r Do not use the built-in rules.

-n No execute mode. Print commands, but do not execute them. Even lines beginning with an @ are printed.

-b Compatibility mode for old (Version 7) makefiles.

-e Environment variables override assignments within makefiles.

-t Touch the target files (causing them to be up-to-date) rather than issue the usual commands.

-d Debug mode. Print out detailed information on files and times examined. (This is intended for debugging the *make* command itself.)

-q Question. The *make* command returns a zero or non-zero status code depending on whether the target file is or is not up-to-date.

The "built-in" dependency targets are:

.DEFAULT
If a file must be made but there are no explicit commands or relevant built-in rules, the commands associated with the name **.DEFAULT** are used if it exists.

.PRECIOUS
Dependents of this target will not be removed when quit or interrupt are hit.

.SILENT
Same effect as the **-s** option.

.IGNORE
Same effect as the **-i** option.

Make executes commands in *makefile* to update one or more target *names*. *Name* is typically a program. If no **-f** option is present, **makefile**, **Makefile**, **s.makefile**, and **s.Makefile** are tried in order. If *makefile* is **-**, the standard input is taken. More than one **-f** *makefile* argument pair may

appear.

Make updates a target only if it depends on files that are newer than the target. All prerequisite files of a target are added recursively to the list of targets. Missing files are deemed to be out of date.

Makefile contains a sequence of entries that specify dependencies. The first line of an entry is a blank–separated, non–null list of targets, followed by a colon (**:**), followed by a (possibly null) list of prerequisite files or dependencies. Text following a **;** and all following lines that begin with a tab are shell commands to be executed to update the target. The first line that does not begin with a tab or # begins a new dependency or macro definition. Shell commands may be continued across lines with the <backslash><new–line> sequence. Everything printed by make (except the initial tab) is passed directly to the shell as is. Thus,

```
echo a\
b
```

will produce

```
ab
```

exactly the same as the shell would.

Sharp (#) and new–line surround comments.

The following *makefile* says that **pgm** depends on two files **a.o** and **b.o**, and that they in turn depend on their corresponding source files (**a.c** and **b.c**) and a common file **incl.h**:

```
pgm: a.o b.o
        cc a.o b.o -o pgm
a.o: incl.h a.c
        cc -c a.c
b.o: incl.h b.c
        cc -c b.c
```

Command lines are executed one at a time, each by its own shell. The first one or two characters in a command can be the following: –, @, –@, or @–. If @ is present, printing of the command is suppressed. If – is present, *make* ignores an error. A line is printed when it is executed unless the **-s** option is present, or the entry **.SILENT:** is in *makefile*, or unless the initial character sequence contains a @. The **-n** option specifies printing without execution; however, if the command line has the string **$(MAKE)** in it, the line is always executed (see discussion of the **MAKEFLAGS** macro under *Environment*). Note that this feature does not work if **MAKE** is enclosed in braces, as in **${MAKE}**. The **-t** (touch) option updates the modified date of a file without executing any commands.

Commands returning non–zero status normally terminate *make*. If the **-i** option is present, or the entry **.IGNORE:** appears in *makefile*, or the initial character sequence of the command contains –. the error is ignored. If the **-k** option is present, work is abandoned on the current entry, but continues on other branches that do not depend on that entry.

The **-b** option allows old makefiles (those written for the old version of *make*) to run without errors. The difference between the old version of *make* and this version is that this version requires all dependency lines to have a (possibly null or implicit) command associated with them. The previous version of *make* assumed, if no command was specified explicitly, that the command was null.

Interrupt and quit cause the target to be deleted unless the target depends on the special name **.PRECIOUS**.

Environment

The environment is read by *make*. All variables are assumed to be macro definitions and processed as such. The environment variables are processed before any makefile and after the

internal rules; thus, macro assignments in a makefile override environment variables. The **-e** option causes the environment to override the macro assignments in a makefile.

The **MAKEFLAGS** environment variable is processed by *make* as containing any legal input option (except **-f**, **-p**, and **-d**) defined for the command line. Further, upon invocation, *make* "invents" the variable if it is not in the environment, puts the current options into it, and passes it on to invocations of commands. Thus, **MAKEFLAGS** always contains the current input options. This proves very useful for "super–makes". In fact, as noted above, when the **-n** option is used, the command **$(MAKE)** is executed anyway; hence, one can perform a **make -n** recursively on a whole software system to see what would have been executed. This is because the **-n** is put in **MAKEFLAGS** and passed to further invocations of **$(MAKE)**. This is one way of debugging all of the makefiles for a software project without actually doing anything.

Macros

Entries of the form *string1* = *string2* are macro definitions. *String2* is defined as all characters up to a comment character or an unescaped new–line. Subsequent appearances of $(*string1*[:*subst1*=[*subst2*]]) are replaced by *string2*. The parentheses are optional if a single character macro name is used and there is no substitute sequence. The optional :*subst1*=*subst2* is a substitute sequence. If it is specified, all non–overlapping occurrences of *subst1* in the named macro are replaced by *subst2*. Strings (for the purposes of this type of substitution) are delimited by blanks, tabs, new–line characters, and beginnings of lines. An example of the use of the substitute sequence is shown under *Libraries*.

Internal Macros

There are five internally maintained macros which are useful for writing rules for building targets.

$*** The macro **$* stands for the file name part of the current dependent with the suffix deleted. It is evaluated only for inference rules.

$@** The **$@ macro stands for the full target name of the current target. It is evaluated only for explicitly named dependencies.

$<** The **$< macro is only evaluated for inference rules or the **.DEFAULT** rule. It is the module which is out–of–date with respect to the target (i.e., the "manufactured" dependent file name). Thus, in the **.c.o** rule, the **$<** macro would evaluate to the **.c** file. An example for making optimized **.o** files from **.c** files is:

```
.c.o:
        cc -c -O $*.c
```

or:

```
.c.o:
        cc -c -O $<
```

$?** The **$? macro is evaluated when explicit rules from the makefile are evaluated. It is the list of prerequisites that are out of date with respect to the target; essentially, those modules which must be rebuilt.

$% The **$%** macro is only evaluated when the target is an archive library member of the form **lib(file.o)**. In this case, **$@** evaluates to **lib** and **$%** evaluates to the library member, **file.o**.

Four of the five macros can have alternative forms. When an upper case **D** or **F** is appended to any of the four macros, the meaning is changed to "directory part" for **D** and "file part" for **F**. Thus, **$(@D)** refers to the directory part of the string **$@**. If there is no directory part, **./** is generated. The only macro excluded from this alternative form is **$?**. The reasons for this are debatable.

Suffixes

Certain names (for instance, those ending with **.o**) have inferable prerequisites such as **.c**, **.s**, etc. If no update commands for such a file appear in *makefile*, and if an inferable prerequisite exists, that prerequisite is compiled to make the target. In this case, *make* has inference rules which allow building files from other files by examining the suffixes and determining an appropriate inference rule to use. The current default inference rules are:

```
.c .c~ .sh .sh~ .c.o .c~.o .c~.c .s.o .s~.o .y.o .y~.o .l.o .l~.o
.y.c .y~.c .l.c .c.a .c~.a .s~.a .h~.h
```

To print out the rules compiled into the *make* on any machine in a form suitable for recompilation, the following command is used:

```
make -fp - 2>/dev/null </dev/null
```

The only peculiarity in this output is the (**null**) string which *printf*(3S) prints when handed a null string.

A tilde in the above rules refers to an SCCS file (see *sccsfile*(5)). Thus, the rule **.c~.o** would transform an SCCS C source file into an object file (**.o**). Because the **s.** of the SCCS files is a prefix, it is incompatible with *make*'s suffix point–of–view. Hence, the tilde is a way of changing any file reference into an SCCS file reference.

A rule with only one suffix (i.e., **.c:**) is the definition of how to build *x* from ***x*.c**. In effect, the other suffix is null. This is useful for building targets from only one source file (e.g., shell procedures, simple C programs).

Additional suffixes are given as the dependency list for **.SUFFIXES**. Order is significant; the first possible name for which both a file and a rule exist is inferred as a prerequisite.

The default list is:

```
.SUFFIXES: .o .c .y .l .s
```

Here again, the above command for printing the internal rules will display the list of suffixes implemented on the current machine. Multiple suffix lists accumulate; **.SUFFIXES:** with no dependencies clears the list of suffixes.

Inference Rules

The first example can be done more briefly:

```
pgm: a.o b.o
        cc a.o b.o -o pgm
a.o b.o: incl.h
```

This is because *make* has a set of internal rules for building files. The user may add rules to this list by simply putting them in the *makefile*.

Certain macros are used by the default inference rules to permit the inclusion of optional matter in any resulting commands. For example, **CFLAGS**, **LFLAGS**, and **YFLAGS** are used for compiler options to *cc*(1), *lex*(1), and *yacc*(1), respectively. Again, the previous method for examining the current rules is recommended.

The inference of prerequisites can be controlled. The rule to create a file with suffix **.o** from a file with suffix **.c** is specified as an entry with **.c.o:** as the target and no dependents. Shell commands associated with the target define the rule for making a **.o** file from a **.c** file. Any target that has

no slashes in it and starts with a dot is identified as a rule and not a true target.

Libraries

If a target or dependency name contains parentheses, it is assumed to be an archive library, the string within parentheses referring to a member within the library. Thus **lib(file.o)** and **$(LIB)(file.o)** both refer to an archive library which contains **file.o**. (This assumes the **LIB** macro has been previously defined.) The expression **$(LIB)(file1.o file2.o)** is not legal. Rules pertaining to archive libraries have the form **.*XX*.a** where the *XX* is the suffix from which the archive member is to be made. An unfortunate byproduct of the current implementation requires the *XX* to be different from the suffix of the archive member. Thus, one cannot have **lib(file.o)** depend upon **file.o** explicitly. The most common use of the archive interface follows. Here, we assume the source files are all C type source:

```
lib:    lib(file1.o) lib(file2.o) lib(file3.o)
        @echo lib is now up-to-date
.c.a:
        $(CC) -c $(CFLAGS) $<
        ar rv $@ $*.o
        rm -f $*.o
```

In fact, the **.c.a** rule listed above is built into *make* and is unnecessary in this example. A more interesting, but more limited example of an archive library maintenance construction follows:

```
lib:    lib(file1.o) lib(file2.o) lib(file3.o)
        $(CC) -c $(CFLAGS) $(?:.o=.c)
        ar rv lib $?
        rm $?   @echo lib is now up-to-date
.c.a:;
```

Here the substitution mode of the macro expansions is used. The **$?** list is defined to be the set of object file names (inside **lib**) whose C source files are out-of-date. The substitution mode translates the **.o** to **.c**. (Unfortunately, one cannot as yet transform to **.c~**; however, this may become possible in the future.) Note also, the disabling of the **.c.a:** rule, which would have created each object file, one by one. This particular construct speeds up archive library maintenance considerably. This type of construct becomes very cumbersome if the archive library contains a mix of assembly programs and C programs.

FILES

[Mm]akefile and s.[Mm]akefile

SEE ALSO

cc(1), cd(1), lex(1), sh(1), yacc(1).

WARNING

Be wary of any file (such as an include file) whose access, modification, and last change times cannot be altered by the *make*-ing process. For example, if a program depends on an include file which in turn depends on another include file, and if one or both of these files are out-of-date, *make* will try to update these files each time it is run, thus unnecessarily re-*make*ing up-to-date files dependent on the include file. The solution is to manually update these files with the *touch*(1) command before running *make*. (Note that it is generally a bad idea to include the *touch*(1) command in your makefile, because it can cause *make* to update a program that otherwise did not need to be updated.)

BUGS

Some commands return non-zero status inappropriately; use **-i** to overcome the difficulty.

File names with the characters = : @ will not work.

Commands that are directly executed by the shell, notably *cd*(1), are ineffectual across new-lines in *make*.

The syntax **lib(file1.o file2.o file3.o)** is illegal.

You cannot build **lib(file.o)** from **file.o**.

The macro **$(a:.o=.c~)** does not work.

There is a limit of 2500 characters, including the terminating new–line, for expanded dependency lines.

Make will not properly expand a macro within another macro when string substitution is involved.

NAME

man - find manual information by keywords; print out the manual

SYNOPSIS

man -k keyword ...
man -f file ...
man [**-**] [section] title ...

HP–UX COMPATIBILITY

Level: HP–UX/STANDARD

Origin: UCB

DESCRIPTION

Man is a program which gives information from the programmer's manual. It can be asked to form one line descriptions of commands specified by name, or for all commands whose description contains any of a set of keywords. It can also provide on–line access to the sections of the printed manual.

When given the option **-k** and a set of keywords, *man* prints out a one line synopsis of each manual section whose listing in the table of contents contains that keyword.

When given the option **-f** and a list of file names, *man* attempts to locate manual sections related to those files, printing out the table of contents lines for those sections.

When neither **-k** nor **-f** is specified, *man* formats a specified set of manual pages. If a section specifier is given *man* looks in that section of the manual for the given *titles*. *Section* is an arabic section number, i.e. 3, which may be followed by a single letter classifier, i.e. 1g indicating a graphics program in section 1. If *section* is omitted, *man* searches all sections of the manual, giving preference to commands over subroutines in system libraries, and printing the first section it finds, if any.

If the standard output is a teletype, or if the flag **-** is given, then *man* pipes its output through *rmnl*(1) to delete useless blank lines, *ul*(1) to create proper underlines for different terminals, and through *more*(1) to stop after each page. Hit a space to continue.

If the /usr/man/cat? directory is present and the file is not in it, but the file exists in /usr/man/man?, then the page is formatted and installed in /usr/man/cat? on first access. If only the /usr/man/cat? directories are present and/or nroff is not installed then only those pages which have been preformatted are displayable.

The Berkeley command *whatis*(1) is provided by **man –f**.

FILES

/usr/man/man?/*
/usr/man/cat?/*
/usr/local/man?/*
/usr/local/cat?/*
/usr/contrib/man?/*
/usr/contrib/cat?/*

SEE ALSO

rmnl(1), ul(1), more(1), whereis(1), catman(1M).

BUGS

The manual is supposed to be reproducible either on the phototypesetter or on a typewriter. However, on a typewriter some information is necessarily lost.

NAME

mediainit – initialize hard disc, flexible disc, or cartridge tape media

SYNOPSIS

mediainit [**–vr**] [**–f** fmt_optn] [**–i** interleave] pathname

HP-UX COMPATIBILITY

Level: HP-UX/NON-STANDARD

Origin: HP

Remarks: *Mediainit* is implemented on Series 200 only.

DESCRIPTION

Mediainit initializes mass storage media by formatting the media, writing and reading test patterns to verify media integrity, then sparing any defective blocks found. This process prepares the disc or tape for error-free operation.

The following command options are recognized. They can be specified in any order, but all must precede the pathname. Options without parameters can be listed individually or grouped together. Options with parameters must be listed individually, but white space between the option and its parameter is discretionary.

–v Normally, *mediainit* provides only fatal error messages, and they are directed to diagnostic output (stderr). The **v**erbose option sends device-specific error information related to low-level operation of *mediainit* to standard output (stdout). This option is most useful to trained service personnel because it usually requires detailed knowledge of device operation before the error information can be interpreted correctly.

–r The **r**e-certify option forces a complete tape certification whether or not the tape has been certified previously. All record of any previously spared blocks is discarded, so any bad blocks will have to be rediscovered. This option should be used only if: (a) it is suspected that numerous blocks on the tape have been spared which should not have been, or (b) it is necessary to destroy (overwrite) all previous data on the tape.

–f *fmt_optn*

The format option is a device-specific number in the range 0 through 239. It is intended solely for use with certain SS/80 devices that support multiple media formats (disregarding interleave factor). For example, certain microfloppy drives support 256, 512, and 1024-byte sectors. *Mediainit* passes any supplied format option directly through to the device. The device then either accepts it or rejects it if it is not supported. Refer to device operating manuals for additional information. The default format option is 0.

–i *interleave*

Interleave refers to relationship between sequential logical records and sequential physical records. It defines the number of physical records that lie between the beginning points of two consecutively numbered logical records. The choice of interleave factor can have a substantial impact on disc performance. For CS/80 and SS/80 devices, consult the appropriate operating manual for details. Acceptable interleave factors for non-CS/80 devices are as follows:

Device	Range	Default
HP9895 SS/DS	1 - 29	2
HP8290X	1 - 15	3
HP9121	1 - 15	2
HP913X_A	na	9
HP913X_B	na	9
HP913X_C	na	9

The following operand is required:

Pathname is the path name to the character (raw) device special file associated with the volume (or single-volume device) containing the media to be initialized. *Mediainit* aborts if you lack either read or write permission to the device special file, or if the volume has already been opened by any other process. Also, *mediainit* locks the volume during initialization so that no other processes can access it. Thus it is impossible to initialize any mounted volume; in particular, the root volume.

For CS/80 and SS/80 drives, a given device may contain multiple units or a given unit may contain multiple volumes. Any available volume can be initialized, even while other volumes are open. However, the initialization process tends to dominate device resources while initialization is in progress,thus causing delays in handling requests from other processes that need to access the same drive. In the case of non-CS/80 devices, the entire device is locked and no other processes can access the device until the target unit is completely initialized.

In general, *mediainit* attempts to carefully scrutinize any format or interleave options supplied, and aborts if an option is out of range or inappropriate for the media being initialized. Specifying an interleave factor or format option value of 0 has the same effect as not specifying the option at all.

For discs that support interleave factors, the acceptable range is usually 1 (no interleave) through N-1, where N is the number of sectors per track. With SS/80 hard discs, the optimum interleave factor is usually determined by the speed (normal or high) of the HP-IB interface card used and whether DMA is present in the system. The optimum interleave factor for SS/80 flexible disc drives is usually a constant (often 2), and is independent of the type of HP-IB interface used. The optimum interleave factor for CS/80 discs is usually 1, and is also usually not related to the type of HP-IB interface being used. In any case, refer to the appropriate device operating manual for recommended values.

If a disc being initialized requires an interleave factor but none is specified, *mediainit* provides an appropriate, though not necessarily optimum default. For CS/80 and SS/80 discs, *mediainit* uses whatever the device reports as its current interleave factor. SS/80 floppy drives report their minimum (usually best) interleave factor, if the currently installed media is unformatted.

When a given device supports format options, the allowable range of interleave factors may be related to the specified format option. In such instances, *mediainit* cannot range-check the interleave factor if one is specified.

Mediainit returns a value of 0 upon successful completion, a value of 1 if there was a device-related error, or a value of 2 if there was a syntax-related error.

EXAMPLES

The following example formats an HP 9122 SS/80 3 1/2″ flexible disc with an interleave factor of 2, 1024-byte sectors, double-sided HP format:

```
mediainit -i 2 -f 3 /dev/r9122
```

Using defaults, the next example initializes an HP 9135A non-CS/80 4.6-Mbyte Winchester hard disc with an interleave factor of 9, HP format, with the verbose mode switch invoked.

```
mediainit -v /dev/r9135A
```

SEE ALSO

mkfs(1M), lifinit(1).

DIAGNOSTICS

Appropriate error messages are given.

WARNINGS

Aborting *mediainit* is likely to leave the medium in a useless state, even if it was previously initialized.

BACKGROUND INFORMATION

Most types of mass storage media must be initialized before they can be used. HP hard discs, flexible discs, and cartridge tapes require some form of initialization, but 9-track tapes do not. Initialization usually involves formatting the media, writing and reading test patterns, then sparing any defective blocks. Depending upon the media and device type, none, some, or all of the initialization process may have been performed at the factory. *Mediainit* completes whatever steps are appropriate for the media to prepare it for error-free operation.

Most HP hard discs are formatted and exhaustively tested at the factory by use of a process more thorough but also more time-consuming than appropriate for *mediainit*. However, *mediainit* is still valuable for ensuring the integrity of the media after factory shipment, formatting with the correct interleave factor, and sparing any blocks which may have become defective since original factory testing was performed.

HP flexible discs are not usually formatted prior to shipment, so they must undergo the entire initialization process before they can be used.

All HP CS/80 cartridge tapes are formatted prior to shipment from the factory, but certification is optional at additional cost. If the tape is certified, it has been tested and sparing is already complete. *Mediainit* usually certifies a tape only if it has not been certified previously. On previously-certified tapes, *mediainit* usually re-organizes the tape's spare block table, retaining any previous spares, but optimizing their assignment for maximum performance under sequential access. Re-organizing the spare block table takes only a few seconds, whereas complete certification takes about a half-hour for 150-foot tapes, and over an hour for 600-foot tapes.

HP CS/80 cartridge tape drives have a feature called "auto-sparing", where if under normal usage the drive has trouble reading a block, the drive logs the fact then automatically spares out that block the next time data is written to it. Thus, as a tape is used, any marginal blocks that were not spared during certification are spared automatically if they cause problems. This sparing is automatic within the device, and is totally independent of *Mediainit*.

Reorganization of a tape's spare block table technically renders any existing data undefined, but the data is not usually destroyed by overwriting. To ensure that old tape data is destroyed (useful for security reasons among other things), complete tape re-certification can forced with the **-r** option.

Some applications may require that a file system be placed on the media before use. *Mediainit* does not create a file system; it only prepares media for writing and reading. Other utilities such as *newfs*(1m) or *lifinit*(1) must be invoked after running *mediainit*, if such a file system is required.

NAME

mesg - permit or deny messages to terminal

SYNOPSIS

mesg [**n**] [**y**]

HP–UX COMPATIBILITY

Level: HP–UX/STANDARD

Origin: System III

Remarks: Not supported on the Integral Personal Computer.

DESCRIPTION

Mesg with argument **n** forbids messages via *write*(1) by revoking non–user write permission on the user's terminal. *Mesg* with argument **y** reinstates permission. All by itself, *mesg* reports the current state without changing it.

FILES

/dev/tty*

SEE ALSO

write(1).

DIAGNOSTICS

Exit status is 0 if messages are receivable, 1 if not, 2 on error.

NAME

mkdir - make a directory

SYNOPSIS

mkdir dirname ...

HP-UX COMPATIBILITY

Level: HP-UX/NUCLEUS

Origin: System V

Native Language Support:
8-bit filenames.

DESCRIPTION

Mkdir creates specified directories in mode 777 (possibly altered by *umask*(1)). Standard entries, **.**, for the directory itself, and **..**, for its parent, are made automatically.

Mkdir requires write permission in the parent directory.

SEE ALSO

sh(1), rm(1), umask(1).

DIAGNOSTICS

Mkdir returns exit code 0 if all directories were successfully made; otherwise, it prints a diagnostic and returns non-zero.

NAME

mkstr - extract error messages from C source into a file

SYNOPSIS

mkstr [**-**] messagefile prefix file ...

HP-UX COMPATIBILITY

Level: HP-UX/DEVELOPMENT

Origin: UCB

DESCRIPTION

Mkstr examines a C program and creates a file containing error message strings used by the program. Programs with many error diagnostics can be made much smaller by referring to places in the file, and reduce system overhead in running the program.

Mkstr processes each of the specified *files*, placing a revised version of each in a file whose name consists of the specified *prefix* concatenated to the original name. A typical usage of *mkstr* would be

```
mkstr mystrings xx *.c
```

This command would cause all the error messages from the C source files in the current directory to be placed in the file *mystrings* and revised copies of the source for these files to be placed in files whose names are prefixed with *xx*.

To process the error messages in the source to the message file *mkstr* keys on the string **error(**" in the input file. Each time it is encountered, the C string starting after the '"' is placed in the message file followed by a null character and a new-line character; the null character terminates the message so it can be easily used when retrieved, and the new-line character makes it possible to sensibly *cat* the error message file to see its contents.

The new copy of the input file is the same as the original, except that each occurrence of a string that is in the error message file is replaced by an offset pointer usable by *lseek* to retrieve the message.

The optional **-** on the command line causes the error messages to be placed at the end of the specified message file instead of overwriting it. When many source files constitute a large *mkstr* ed program, this can be used to avoid reprocessing all the files.

All functions used by the original program whose names end in "error" and that can take a constant string as their first argument should be rewritten so that they search for the string in the error message file.

For example:

```
#include <stdio.h>
#include <sys/types.h>
#include <fcntl.h>

char errfile[] = "mystrings";

error(offset, a2, a3, a4)
int offset, a1, a2, a3;
{
    char msg[256];
    static int fd = -1;

        if (fd < 0)
        {
```

```
            fd = open(errfile, O_RDONLY);
            if (fd < 0)
            {
                perror(errfile);
                exit(1);
            }
        }

        if (lseek(fd, (off_t) offset, 0) || read(fd, msg, 256) <= 0)
        {
            printf("? Can't find error message in %s:\n", errfile);
            perror(errfile);
            exit(1);
        }

        printf(msg, a1, a2, a3);
    }
```

SEE ALSO

lseek(2), perror(3C), xstr(1)

BUGS

Strings in calls to functions whose names end in 'error', notably *perror*(3C), may be replaced with offsets.

Calls to error functions whose first argument is not a string constant are left unmodified without warning.

NAME

mm, osdd, checkmm - print/check documents formatted with the MM macros

SYNOPSIS

mm [options] [files]

osdd [options] [files]

checkmm [files]

HP-UX COMPATIBILITY

Level: HP-UX/EXTENDED

Origin: System III

DESCRIPTION

Mm can be used to type out documents using *nroff*(1) and the MM text-formatting macro package. It has options to specify preprocessing by *tbl*(1) and/or *neqn*(1) and postprocessing by various terminal-oriented output filters. The proper pipelines and the required arguments and flags for *nroff*(1) and MM are generated, depending on the options selected.

Osdd is equivalent to the command **mm -mosd**. For more information about the OSDD adapter macro package, see *mosd*(5).

Options for *mm* are given below. Any other arguments or flags (e.g., **-rC3**) are passed to *nroff*(1) or to MM, as appropriate. Such options can occur in any order, but they must appear before the *files* arguments. If no arguments are given, *mm* prints a list of its options.

-T*term* Specifies the type of output terminal; for a list of recognized values for *term*, type **help term2**. If this option is *not* used, *mm* will use the value of the shell variable **$TERM** from the environment (see *profile*(5) and *environ*(7)) as the value of *term*, if **$TERM** is set; otherwise, *mm* will use **450** as the value of *term*. If several terminal types are specified, the last one takes precedence.

-12 Indicates that the document is to be produced in 12-pitch. May be used when **$TERM** is set to one of **300**, **300s**, **450**, and **1620**. (The pitch switch on the DASI 300 and 300s terminals must be manually set to **12** if this option is used.)

-c Causes *mm* to invoke *col*(1); note that *col*(1) is invoked automatically by *mm* unless *term* is one of **300**, **300s**, **450**, **37**, **4000a**, **382**, **4014**, **tek**, **1620**, and **X**.

-e Causes *mm* to invoke *neqn*; also causes *neqn* to read the **/usr/pub/eqnchar** file (see *eqnchar*(5)).

-t Causes *mm* to invoke *tbl*(1).

-E Invokes the **-e** option of *nroff*.

-y Causes *mm* to use the non-compacted version of the macros (see *mm*(7)).

As an example (assuming that the shell variable **$TERM** is set in the environment to **450**), the two command lines below are equivalent:

```
mm -t -rC3 -12 ghh*
tbl ghh* | nroff -cm -T450-12 -h -rC3
```

Mm reads the standard input when **-** is specified instead of any file names. (Mentioning other files together with **-** leads to disaster.) This option allows *mm* to be used as a filter, e.g.:

```
cat dws | mm -
```

Checkmm is a program for checking the contents of the named *files* for errors in the use of the Memorandum Macros, missing or unbalanced *neqn* delimiters, and **.EQ/.EN** pairs. Note: The user need not use the *checkeq* program (see *eqn*(1)). Appropriate messages are produced. The program skips all directories, and if no file name is given, standard input is read.

HINTS

1. *Mm* invokes *nroff* with the **-h** flag. With this flag, *nroff* assumes that the terminal has tabs set every 8 character positions.

2. Use the **-o***list* option of *nroff* to specify ranges of pages to be output. Note, however, that *mm*, if invoked with one or more of the **-e**, **-t**, and **-** options, *together* with the **-o***list* option of *nroff* may cause a harmless "broken pipe" diagnostic if the last page of the document is not specified in *list*.
3. If you use the **-s** option of *nroff* (to stop between pages of output), use line–feed (rather than return or new–line) to restart the output. The **-s** option of *nroff* does not work with the **-c** option of *mm*, or if *mm* automatically invokes *col*(1) (see **-c** option above).
4. If you lie to *mm* about the kind of terminal its output will be printed on, you'll get (often subtle) garbage; however, if you are redirecting output into a file, use the **-T37** option, and then use the appropriate terminal filter when you actually print that file.

SEE ALSO

col(1), cw(1), env(1), eqn(1), greek(1), mmt(1), nroff(1), tbl(1), profile(5), mm(7), mosd(7), term(7).

MM-Memorandum Macros in *HP–UX Concepts and Tutorials*.

DIAGNOSTICS

mm "mm: no input file" if none of the arguments is a readable file and *mm* is not used as a filter.

checkmm "Cannot open *filename*" if file(s) is unreadable. The remaining output of the program is diagnostic of the source file.

NAME

more - file perusal filter for crt viewing

SYNOPSIS

more [**-cdflsu**] [-n] [+linenumber] [+/pattern] [name ...]
page [more options]

HP–UX COMPATIBILITY

Level: HP–UX/STANDARD

Origin: UCB

Native Language Support:
8-bit data.

DESCRIPTION

More is a filter which allows examination of continuous text, one screenful at a time, on a soft–copy terminal. It normally pauses after each screenful, printing **--More--** at the bottom of the screen. If the user then types a carriage return, one more line is displayed. If the user hits a space, another screenful is displayed. Other possibilites are enumerated later.

The command line options are:

-*n* An integer which is the size (in lines) of the window which *more* will use instead of the default.

-c *More* will draw each page by beginning at the top of the screen and erasing each line just before it draws on it. This avoids scrolling the screen, making it easier to read while *more* is writing. This option will be ignored if the terminal does not have the ability to clear to the end of a line.

-d *More* will prompt the user with the message "Hit space to continue, Rubout to abort" at the end of each screenful. This is useful if *more* is being used as a filter in some setting, such as a class, where many users may be unsophisticated.

-f This causes *more* to count logical lines, rather than screen lines. That is, long lines are not folded. This option is recommended if *nroff* output is being piped through *ul*, since the latter may generate escape sequences. These escape sequences contain characters which would ordinarily occupy screen postions, but which do not print when they are sent to the terminal as part of an escape sequence. Thus *more* may think that lines are longer than they actually are, and fold lines erroneously.

-l Do not treat ^L (form feed) specially. If this option is not given, *more* will pause after any line that contains a ^L, as if the end of a screenful had been reached. Also, if a file begins with a form feed, the screen will be cleared before the file is printed.

-s Squeeze multiple blank lines from the output, producing only one blank line. Especially helpful when viewing *nroff* output, this option maximizes the useful information present on the screen.

-u Normally, *more* will handle underlining and bold such as produced by *nroff* in a manner appropriate to the particular terminal: if the terminal can perform underlining or has a stand–out mode, *more* will output appropriate escape sequences to enable underlining, else stand–out mode, for underlined information in the source file. If the terminal can perform stand–out, more uses that mode for bold information. The **-u** option suppresses this processing, as do the "ul" and "os" terminfo flags.

+*linenumber*
Start up at *linenumber*.

+/*pattern*
Start up two lines before the line containing the regular expression *pattern*.

If the program is invoked as *page*, then the screen is cleared before each screenful is printed (but only if a full screen is being printed), and $k - 1$ rather than $k - 2$ lines are printed in each screenful, where k is the number of lines the terminal can display. *More* looks in the file */usr/lib/terminfo* to determine terminal characteristics, and to determine the default window size. On a terminal capable of displaying 24 lines, the default window size is 22 lines.

More looks in the environment variable *MORE* to pre-set any flags desired. For example, if you prefer to view files using the **-c** mode of operation, the shell command sequence *MORE='-c' ; export MORE* or the *csh* command *setenv MORE -c* would cause all invocations of *more*, including invocations by programs such as *man* and *msgs*, to use this mode. Normally, the user will place the command sequence which sets up the *MORE* environment variable in the *.profile* or *.cshrc* file.

If *more* is reading from a file, rather than a pipe, then a percentage is displayed along with the **--More--** prompt. This gives the fraction of the file (in characters, not lines) that has been read so far.

Other sequences which may be typed when *more* pauses, and their effects, are as follows (i is an optional integer argument, defaulting to 1) :

i<space>
: display i more lines, (or another screenful if no argument is given).

^D
: display 11 more lines (a "scroll"). If i is given, then the scroll size is set to i.

d
: same as ^D (control-D).

iz
: same as typing a space except that i, if present, becomes the new window size.

is
: skip i lines and print a screenful of lines.

if
: skip i screenfuls and print a screenful of lines.

q or Q
: Exit from *more*.

=
: Display the current line number.

v
: Start up the editor *vi* at the current line.

h
: Help command; give a description of all the *more* commands.

i/expr
: search for the i-th occurrence of the regular expression *expr*. If there are less than i occurrences of *expr*, and the input is a file (rather than a pipe), then the position in the file remains unchanged. Otherwise, a screenful is displayed, starting two lines before the place where the expression was found. The user's erase and kill characters may be used to edit the regular expression. Erasing back past the first column cancels the search command.

in
: search for the i-th occurrence of the last regular expression entered.

'
: (single quote) Go to the point from which the last search started. If no search has been performed in the current file, this command goes back to the beginning of the file.

!command
: invoke a shell with *command*. The characters "%" and "!" in "command" are replaced with the current file name and the previous shell command respectively. If there is no current file name, "%" is not expanded. The sequences "\%" and "\!" are replaced by "%" and "!" respectively.

i:n
: skip to the i-th next file given in the command line (skips to last file if n doesn't make sense).

i:p
: skip to the i-th previous file given in the command line. If this command is given in the middle of printing out a file, then *more* goes back to the beginning of the file. If i doesn't make sense, *more* skips back to the first file. If *more* is not reading from a file, the bell is

rung and nothing else happens.

:f display the current file name and line number.

:q or :Q
exit from *more* (same as q or Q).

. (dot) repeat the previous command.

The commands take effect immediately, i.e., it is not necessary to type a carriage return. Up to the time when the command character itself is given, the user may hit the line kill character to cancel the numerical argument being formed. In addition, the user may hit the erase character to redisplay the **--More--(*xx*%)**.

At any time when output is being sent to the terminal, the user can hit the quit key (normally control-\). *More* will stop sending output, and will display the usual **--More--** prompt. The user may then enter one of the above commands in the normal manner. Unfortunately, some output is lost when this is done, due to the fact that any characters waiting in the terminal's output queue are flushed when the quit signal occurs.

The terminal is set to *noecho* mode by this program so that the output can be continuous. What you type will thus not show on your terminal, except for the / and ! commands.

If the standard output is not a teletype, then *more* acts just like *cat,* except that a header is printed before each file (if there is more than one).

A sample usage of *more* in previewing *nroff* output would be

nroff -ms +2 doc.n | more –s

FILES

/usr/lib/terminfo/?/*	terminfo data base
/usr/lib/more.help	help file

VARIABLES

MORE Default paging mode.

SEE ALSO

csh(1), man(1), msgs(1), sh(1), terminfo(5), environ(5).

BUGS

Core will be dumped if *more* is invoked withtout TERM in the environment.

When invoked from a shell such that *stderr* is write–only, and if *stderr* is interactive (e.g., 2>*fname*), then input to *stderr* must be redirected in one of the following ways:

2<&0
2</dev/tty

NAME

mt - magnetic tape manipulating program

SYNOPSIS

mt [**-t** tapename] command [count]

HP–UX COMPATABILITY

Level: HP–UX/STANDARD

Origin: UCB

DESCRIPTION

Mt is used to give commands to the tape drive. If *tapename* is not specified, /dev/mt/0mn is used. If *count* is not specified, 1 is assumed.

Here are the commands:

eof	write *count* end–of–file marks
fsf	space forward *count* files
fsr	space forward *count* records
bsf	space backward *count* files
bsr	space backward *count* records
rew	rewind tape
offl	rewind tape and go offline.

EXAMPLE

The following command will rewind the default mag tape unit and take it off line:

```
mt offl
```

FILES

/dev/mt/*	Magnetic tape interface
/dev/rmt/*	Raw magnetic tape interface

/dev/rmt/0mn (or whatever drive is used) must be described as a Berkeley–compatibility mode tape drive (without rewind) for *mt* to operate as expected.

SEE ALSO

mt(4), dd(1).

NAME

newgrp - log in to a new group

SYNOPSIS

newgrp [**-**] [group]

HP-UX COMPATIBILITY

Level: HP-UX/STANDARD

Origin: System V

Native Language Support:
8-bit passwords.

DESCRIPTION

Newgrp changes a user's group identification. The user remains logged in and the current directory is unchanged, but calculations of access permissions to files are performed with respect to the new real and effective group IDs. The user is always given a new shell, replacing the current shell, by *newgrp*, regardless of whether it terminated successfully or due to an error condition (i.e., unknown group).

Exported variables retain their values after invoking *newgrp*; however, all unexported variables are either reset to their default value or set to null. System variables (such as PS1, PS2, PATH, MAIL, and HOME), unless exported by the system or explicitly exported by the user, are reset to default values. For example, a user has a primary prompt string (**PS1**) other than **$** (default) and has not exported **PS1**. After an invocation of *newgrp* , successful or not, their **PS1** will now be set to the default prompt string **$**. Note that the shell command *export* (see *sh*(1)) is the method to export variables so that they retain their assigned value when invoking new shells.

With no arguments, *newgrp* changes the group identification back to the group specified in the user's password file entry.

If the first argument to *newgrp* is a **-**, the environment is changed to what would be expected if the user actually logged in again.

A password is demanded if the group has a password and the user does not, or if the group has a password and the user is not listed in **/etc/group** as being a member of that group.

FILES

/etc/group	system's group file
/etc/passwd	system's password file

SEE ALSO

login(1), sh(1), group(5), passwd(5), environ(7).

DIAGNOSTICS

Sorry:	You didn't qualify as a group member.
Unknown group:	The group name was not in /etc/group.
Permission denied:	If a password must be given, it can only come from a teletype port. If the *stdin* is a non-tty file, this message is given.
You have no shell:	*Exec* of the shell failed.

BUGS

There is no convenient way to enter a password into **/etc/group**.
Use of group passwords is not encouraged, because, by their very nature, they encourage poor security practices. Group passwords may disappear in the future.
Shell variables which are not exported are lost.

NAME

news - print news items

SYNOPSIS

news [**-a**] [**-n**] [**-s**] [items]

HP-UX COMPATIBILITY

Level: HP-UX/STANDARD

Origin: System V

DESCRIPTION

News is used to keep the user informed of current events. By convention, these events are described by files in the directory **/usr/news**.

When invoked without arguments, *news* prints the contents of all current files in **/usr/news**, most recent first, with each preceded by an appropriate header. *News* stores the "currency" time as the modification date of a file named **.news_time** in the user's home directory (the identity of this directory is determined by the environment variable **$HOME**); only files more recent than this currency time are considered "current."

The following options can be used with *news*:

-a causes *news* to print all items, regardless of currency. In this case, the stored time is not changed.

-n causes *news* to report the names of the current items without printing their contents, and without changing the stored time.

-s causes *news* to report how many current items exist, without printing their names or contents, and without changing the stored time. It is useful to include such an invocation of *news* in one's **.profile** file, or in the system's **/etc/profile**.

Note that only one of the **-a**, **-n**, and **-s** options can be used at a time.

All other arguments are assumed to be specific news items that are to be printed.

If an interrupt is typed during the printing of a news item, printing stops and the next item is started. Another *delete* within one second of the first causes the program to terminate.

FILES

/etc/profile
/usr/news/*
$HOME/.news_time

SEE ALSO

mail(1), profile(5), environ(7).

NAME

nice - run a command at low priority

SYNOPSIS

nice [**-**increment] command [arguments]

HP–UX COMPATIBILITY

Level: HP–UX/STANDARD

Origin: System V

DESCRIPTION

Nice executes *command* with a lower CPU scheduling priority. If the *increment* argument (in the range 1–19) is given, it is used; if not, an increment of 10 is assumed.

The super–user may run commands with priority higher than normal by using a negative increment, e.g., **--10**.

SEE ALSO

nohup(1), nice(2).

DIAGNOSTICS

Nice returns the exit status of the subject command.

BUGS

An *increment* larger than 19 is equivalent to 19.

NAME

nl - line numbering filter

SYNOPSIS

nl [**-h**type] [**-b**type] [**-f**type] [**-v**start#] [**-i**incr] [**-p**] [**-l**num] [**-s**sep] [**-w**width] [**-n**format] [**-d**delim] file

HP-UX COMPATIBILITY

Level: HP-UX/STANDARD

Origin: System V

DESCRIPTION

Nl reads lines from the named *file* or the standard input if no *file* is named and reproduces the lines on the standard output. Lines are numbered on the left in accordance with the command options in effect.

Nl views the text it reads in terms of logical pages. Line numbering is reset at the start of each logical page. A logical page consists of a header, a body, and a footer section. Empty sections are valid. Different line numbering options are independently available for header, body, and footer (e.g., no numbering of header and footer lines while numbering blank lines only in the body).

The start of logical page sections are signaled by input lines containing nothing but the following delimiter character(s):

Line contents	*Start of*
\:\:\:	header
\:\:	body
\:	footer

Unless told otherwise, *nl* assumes the text being read is in a single logical page body.

Command options may appear in any order and may be intermingled with an optional file name. Only one file may be named. The options are:

-b*type* Specifies which logical page body lines are to be numbered. Recognized *types* and their meaning are:

a	number all lines;
t	number lines with printable text only;
n	no line numbering;
p*string*	number only lines that contain the regular expression specified in *string*.

The default *type* for logical page body is **t** (text lines numbered).

-h*type* Same as **-b***type* except for header. Default *type* for logical page header is **n** (no lines numbered).

-f*type* Same as **-b***type* except for footer. Default for logical page footer is **n** (no lines numbered).

-p Do not restart numbering at logical page delimiters.

-v*start#* *Start#* is the initial value used to number logical page lines. Default is **1**.

-i*incr* *Incr* is the increment value used to number logical page lines. Default is **1**.

-s*sep* *Sep* is the character(s) used in separating the line number and the corresponding text line. Default *sep* is a tab.

-w*width* *Width* is the number of characters to be used for the line number. Default *width* is **6**.

-n*format* *Format* is the line numbering format. Recognized values are:

ln left justified, leading zeroes suppressed;
rn right justified, leading zeroes supressed;
rz right justified, leading zeroes kept.

Default *format* is **rn** (right justified).

-l*num* *Num* is the number of blank lines to be considered as one. For example, **-l2** results in only the second adjacent blank being numbered (if the appropriate **-ha**, **-ba**, and/or **-fa** option is set). Default is **1**.

-d*xx* The delimiter characters specifying the start of a logical page section may be changed from the default characters (\:) to two user–specified characters. If only one character is entered, the second character remains the default character (:). No space should appear between the **-d** and the delimiter characters. To enter a backslash, use two backslashes.

EXAMPLE

The command:

```
nl -v10 -i10 -d!+ file1
```

will number file1 starting at line number 10 with an increment of ten. The logical page delimiters are ! and +.

SEE ALSO

pr(1).

NAME

nm - print name list (symbol table) of object file

SYNOPSIS

nm [**-gnoprsu**] [*filename* ...]

HP-UX COMPATIBILITY

Level: HP-UX/STANDARD

Origin: System III

Remarks: This manual page describes *nm* as implemented on the Series 200 computers. Refer to other *nm*(1) manual pages for information valid for other implementations.

DESCRIPTION

Nm prints the name list (symbol table) of each object file in the argument list. If an argument is an archive, a listing for each object file in the archive will be produced. If no *file* is given, the symbols in **a.out** are listed.

Each symbol name is preceded by its value printed in a representation appropriate to the architecture of the machine (blanks if undefined) and one of the letters **U** (undefined), **A** (absolute), **T** (text segment symbol), **D** (data segment symbol), **B** (bss segment symbol), **R** (register symbol), **F** (file symbol), or **C** (common symbol). If the symbol is local (non-external) the type letter is in lower case. The output is sorted alphabetically.

Options are:

-g Print only global (external) symbols.

-n Sort numerically rather than alphabetically.

-o Prepend file or archive element name to each output line rather than only once. This option can be used to make piping to *grep*(1) more meaningful.

-p Don't sort; print in symbol-table order.

-r Sort in reverse order.

-s Sort according to the size of the external symbol (computed from the difference between the value of the symbol and the value of the symbol with the next highest value). This difference is the value printed. This flag turns on **-g** and **-n** and turns off **-u** and **-p**.

-u Print only undefined symbols.

If the symbol was an align symbol, the letter **L** will be printed after the letter describing its type.

SEE ALSO

ar(1), a.out(5), ar(5).

NAME

nm - print name list (symbol table) of object file

SYNOPSIS

nm [**-gnopru**] [file ...]

HP–UX COMPATIBILITY

Level: HP–UX/STANDARD

Origin: System III

Remarks: This manual page describes *nm* as implemented on the Series 500 computers. Refer to other *nm* manual pages for information valid for other implementations.

DESCRIPTION

Nm prints the name list (symbol table) of each object *file* in the argument list. If an argument is an archive, a listing for each object file in the archive will be produced, preceded by the member name on a separate line. If no *file* is given, the symbols in **a.out** are listed.

Options are:

-g Print only global (external) symbols.

-n Sort numerically rather than alphabetically.

-o Prepend file or archive element name to each output line rather than only once. This option can be used to make piping to *grep*(1) more meaningful.

-p Don't sort; print in symbol–table order.

-r Sort in reverse order.

-u Print only undefined symbols.

The output from *nm* consists of five columns of data. The following is a portion of a typical output:

```
.  X  IDATA     00000108                  A__iob
.  X  IDATA     000002a0                  A__sctab
.  X  ICOMM     00000400 0 00000440       A__sibuf
.  X  ICOMM     00000400 0 00000840       A__sobuf
.  .  UDATA     00000c40                  Aallocs
.  X  FUNC      EDS c04 002a8 00000003    ___cleanup
.  X  DDATA     DR 00000098               ___ctype
.  X  FUNC      EDS c0c 00000 00000001    ___doscan
.  X  SYSTEM    EPP 004 0000e             ___exit
.  X  DDATA     DR 00000038               ___iob
.  X  DCOMM     00000004 000000b0         ___pfile
.  X  DDATA     DR 00000090               ___sctab
.  X  PTR       1 00000a 000000b4         ___sibuf
.  X  PTR       1 00000c 000000b8         ___sobuf
.  .  FILENAME  0000000a                  __exit.o
.  .  FILENAME  0000000f                  __print.o
```

>From left to right, the first column specifies whether the symbol is defined (.) or undefined (U). The second column specifies whether the symbol is non–external (.) or external (X). The third column gives the linker symbol type (as defined in *a.out.h* and described below). The fourth column lists the data associated with the specified symbol type. The fifth column gives the name of the system call, file, variable, array, common, etc., described by that entry in the symbol table.

Up to four data elements are reported in the fourth column. If they are not symbolic values (such as 'EDS' or 'DR'), then they are hexadecimal values. The number of data elements reported depends on the symbol type. Each symbol type has one to four parameters associated with it, whose values are given by the data elements in the fourth column. The symbol types and associated parameters are discussed below.

The following symbol types are supported:

ABS not currently generated; reserved for future use.

FUNC or **ENTRY** specifies that the entry refers to a function or procedure call. Four numbers, *ptr_type*, *segment*, *offset*, and *stt_index*, are associated. Their values are given in order, from left to right, by the data elements. *Ptr_type* consists of a single bit that is always cleared. It is symbolically represented by 'EDS'. *Ptr_type* is meaningful to the linker (see *ld*(1)), and specifies the storage format of the call in the symbol table. *Segment* specifies the code segment number (a number in the range 3073 to 4095, that indicates which code segment in the user's program space contains the desired code). *Offset* specifies the number of bytes from the beginning of the code segment where the function or procedure code begins. *Stt_index* is an indirect reference to the beginning of the function or procedure code.

SYSTEM specifies that the entry refers to a procedure call directly into the system kernel. Three numbers, *entry_type*, *segment*, and *stt*, are associated. Their values are given by the data elements. *Entry_type* consists of a single bit that is always set. Its value is symbolically represented by 'EPP'. *Entry_type* is meaningful to the linker, and specifies the storage format of the call in the symbol table. *Segment* specifies the system code segment number (the number of a code segment among those set aside for system use; typically in the range 0 to 64). *Stt* is an indirect pointer to the beginning of the procedure code.

LABEL specifies that the entry is the destination address for a branch instruction. Three numbers, *ptr_type*, *segment*, and *offset*, are associated. Their values are given by the data elements. *Ptr_type* consists of a single bit which is always cleared. Its value is symbolically represented by 'EDS'. *Ptr_type* is meaningful to the linker, and specifies the storage format of the address in the symbol table. *Segment* specifies the user code segment number. *Offset* specifies the number of bytes from the beginning of the code segment where the label begins.

DDATA specifies that the entry is a directly-addressable, initialized data structure (a variable, or the beginning of an array, common, structure, etc.). Two numbers, *base_reg* and *displacement*, are associated. Their values are given by the data elements. *Base_reg* is assigned one of nine possible symbolic values which describe the addressing scheme used to find the data structure. It is meaningful to the linker. The possible symbolic values are P+, P-, DB, DL, Q+, Q-, SB, S-, and DR. *Displacement* specifies the byte offset where the data structure is located. Note that this offset is measured relative to the beginning of the data space of the file for which the *nm* listing is made. The actual byte offset of the data structure in the executable **a.out** file could change.

IDATA or **UDATA** specifies that the entry refers to an indirectly-addressable, uninitialized array, or an indirectly-addressable, initialized common block. One number, *displacement*, is associated. Its value is given by the data element. It is identical to the *displacement* described above under **DDATA**.

DCOMM or **ICOMM**
: specifies that the entry is treated as a common block. Three numbers, *blocksize*, *needs_length_word, and displacement*, are associated. Their values are given by the data elements. *Blocksize* is the size, in bytes, of the common block. *Needs_length_word* is a boolean value which appears in a print–out as either 0 or 1. If its value is 1, the linker places the value of (*blocksize* - 4) in the first four bytes of the common block. This information is necessary when linking FORTRAN programs. *Displacement* is identical to that described under **DDATA** above.

PTR
: specifies that the entry is a pointer to an indirectly–addressable data structure (variable, array, common block, etc.). Three numbers, *ptr_to_common*, *target*, and *address*, are associated. Their values are given by the data elements. *Ptr_to_common* is an eight–bit boolean expression. Its value is given as 1 (true) or 0 (false). If true, then the entry is a pointer to a common block. If false, the entry is a pointer to some other type of data structure. *Target* is an index into the symbol table to the entry that describes the target of the data structure pointer. *Address* is a pointer to the data structure pointer; that is, an indirect pointer to the data structure.

SEGMENT
: not currently generated; reserved for future use.

FILENAME
: specifies that the entry is a file name. One number, *sequence*, is associated. Its value is given by the data element. *Sequence* reflects the order in which the linker encountered each file name.

SEE ALSO

ar(1), a.out(5), ar(5).

DIAGNOSTICS

Nm generates an error message for the following conditions:

invalid option
cannot open *file*
bad magic number
read error

NAME

nohup - run a command immune to hangups, logouts, and quits

SYNOPSIS

nohup command [arguments]

HP–UX COMPATIBILITY

Level: HP–UX/STANDARD

Origin: System V

DESCRIPTION

Nohup executes *command* with hangups and quits ignored. If output is not re–directed by the user, both standard output and standard error are sent to **nohup.out**. If **nohup.out** is not writable in the current directory, output is redirected to **$HOME/nohup.out**; otherwise, *nohup* will fail.

If output from *nohup* is redirected to a terminal, or is not redirected at all, the output is sent to **nohup.out**.

EXAMPLE

It is frequently desirable to apply *nohup* to pipelines or lists of commands. This can be done only by placing pipelines and command lists in a single file, called a shell procedure. One can then issue:

```
nohup sh file
```

and the *nohup* applies to everything in *file*. If the shell procedure *file* is to be executed often, then the need to type *sh* can be eliminated by giving *file* execute permission. Add an ampersand and the contents of *file* are run in the background with interrupts also ignored (see *sh*(1)):

```
nohup file &
```

An example of what the contents of *file* could be is:

```
tbl ofile | eqn | nroff > nfile
```

SEE ALSO

chmod(1), nice(1), sh(1), signal(2).

WARNINGS

nohup command1; command2 *nohup* applies only to *command1*
nohup (command1; command2) is syntactically incorrect.

Be careful of where standard error is redirected. The following command may put error messages on tape, making it unreadable:

```
nohup cpio -o <list >/dev/rmt/1m &
```

while

```
nohup cpio -o <list >/dev/rmt/1m 2>errors &
```

puts the error messages into file *errors*.

NAME

nroff - format text

SYNOPSIS

nroff [options] [files]

HP–UX COMPATIBILITY

Level: HP–UX/STANDARD

Origin: System III

DESCRIPTION

Nroff formats text contained in *files* (standard input by default) for printing on typewriter–like devices and line printers. Its capabilities are described in the *NROFF/TROFF User's Manual* cited below.

Nroff is best not used directly, but rather via macro packages such as *mm* or *ms* which provide a high–level interface to document processing, as opposed to the very low level one provided directly in *nroff*.

An argument consisting of a minus (**-**) is taken to be a file name corresponding to the standard input. The *options*, which may appear in any order, but must appear before the *files*, are:

-o*list* Print only pages whose page numbers appear in the *list* of numbers and ranges, separated by commas. A range *N*-*M* means pages *N* through *M*; an initial -*N* means from the beginning to page *N*; and a final *N*- means from *N* to the end. (See *BUGS* below.)

-n*N* Number first generated page *N*.

-s*N* Stop every *N* pages. *Nroff* will halt *after* every *N* pages (default *N*=1) to allow paper loading or changing, and will resume upon receipt of a line–feed or new–line (new–lines do not work in pipelines, e.g., with *mm*(1)). When *nroff* halts between pages, an ASCII **BEL** is sent to the terminal.

-r*aN* Set register *a* (which must have a one–character name) to *N*.

-i Read standard input after *files* are exhausted.

-q Invoke the simultaneous input–output mode of the **.rd** request.

-z Print only messages generated by **.tm** (terminal message) requests.

-m*name* Precede the input *files* with the non–compacted (ASCII text) macro file **/usr/lib/tmac/tmac.***name*.

-c*name* Precede the input *files* with the compacted macro files **/usr/lib/macros/cmp.[nt].[dt].***name* and **/usr/lib/macros/ucmp.[nt].***name*.

-k*name* Compact the macros used in this invocation of *nroff*, placing the output in files **[dt].***name* in the current directory (see the May 1979 Addendum to the *NROFF/TROFF User's Manual* for details of compacting macro files).

-T*name* Prepare output for specified terminal. Known *name*s are **37** for the (default) TELE–TYPE Model 37 terminal, **tn300** for the GE TermiNet 300 (or any terminal without half–line capability), **300s** for the DASI 300s, **300** for the DASI 300, **450** for the DASI 450, **lp** for a (generic) ASCII line printer, **382** for the DTC–382, **4000A** for the Trendata 4000A, **832** for the Anderson Jacobson 832, **X** for a (generic) EBCDIC printer, and **2631** for the Hewlett Packard 2631 line printer.

-e Produce equally–spaced words in adjusted lines, using the full resolution of the particular terminal.

-h Use output tabs during horizontal spacing to speed output and reduce output character count. Tab settings are assumed to be every 8 nominal character widths.

-u*n* Set the emboldening factor (number of character overstrikes) for the third font position (bold) to *n*, or to zero if *n* is missing.

HP–UX nroff differs from other versions of nroff as follows:

1) New facilities for handling of the revision bar (**.mc**) feature are provided:

The read/write number register *ic* holds the revision bar character (as an ordinal) that will be printed on the current line, if any. The number registers **.m** and **.e** contain respecitvely the revision bar character and offset as set by the **.mc** command, and are read only. These make it possible to deal with the revision bar information in a macro, thus making it possible for the printed result to properly reflect the revision information in the source.

2) The date and time used are the appropriate local time.

Series 500:

The **-c** and **-k** options are not currently supported.

FILES

/usr/lib/suftab	suffix hyphenation tables
/tmp/ta$#	temporary file
/usr/lib/tmac/tmac.*	standard macro files and pointers
/usr/lib/macros/*	standard macro files
/usr/lib/term/*	terminal driving tables for *nroff*

SEE ALSO

mm(1).

NROFF/TROFF User's Manual in *HP-UX: Selected Articles.*

BUGS

Nroff is keyed to Eastern Standard Time; as a result, depending on the time of the year and on your local time zone, the date that *nroff* generates may be off by one day from your current date. When *nroff* is used with the **-o***list* option inside a pipeline, it may cause a harmless "broken pipe" diagnostic if the last page of the document is not specified in *list*.

NAME

od, xd - octal and hexadecimal dump

SYNOPSIS

od [**-bcdox**] [file] [[**+**]offset[**.**][**b**]]
xd [**-bcdox**] [file] [[**+**]offset[**.**][**b**]]

HP-UX COMPATIBILITY

Level: HP-UX/STANDARD

Origin: System III and HP

DESCRIPTION

Od (*xd*) dumps *file* in one or more formats as selected by the first argument. If the first argument is missing, **-o** (**-x**) is the default. Each line is prepended with an offset field. For *od*, the offset is in octal, for *xd* the offset is in hexadecimal. The meanings of the format options are:

-b Interpret bytes in octal (hexadecimal).

-c Interpret bytes in ASCII. Certain non-graphic characters appear as C escapes: null=**\0**, backspace=**\b**, form-feed=**\f**, new-line=**\n**, return=**\r**, tab=**\t**; others appear as 3-digit octal numbers.

-d Interpret 16-bit words in decimal.

-o Interpret 16-bit words in octal.

-x Interpret 16-bit words in hexadecimal.

The *file* argument specifies which file is to be dumped. If no *file* argument is specified, the standard input is used.

The *offset* argument specifies the offset in the file where dumping is to commence. This argument is normally interpreted as octal bytes. If **.** is appended, *offset* is interpreted in decimal. If **0x** is prepended, *offset* is interpreted in hexadecimal. If **b** is appended, *offset* is interpreted in blocks of 512 bytes. If the file argument is omitted, *offset* must be preceded by **+**.

Dumping continues until end-of-file.

SEE ALSO

adb(1).

NAME

pack, pcat, unpack - compress and expand files

SYNOPSIS

pack [**-**] [**-f**] name ...

pcat name ...

unpack name ...

HP-UX COMPATIBILITY

Level: HP-UX/STANDARD

Origin: System V

Remarks: Not supported on the Integral Personal Computer.

DESCRIPTION

Pack attempts to store the specified files in a compressed form. Wherever possible (and useful), each input file *name* is replaced by a packed file *name*.**z** with the same access modes, access and modified dates, and owner as those of *name*. The **-f** option will force packing of *name*. This is useful for causing an entire directory to be packed even if some of the files will not benefit. If *pack* is successful, *name* will be removed. Packed files can be restored to their original form using *unpack* or *pcat*.

Pack uses Huffman (minimum redundancy) codes on a byte-by-byte basis. If the **-** argument is used, an internal flag is set that causes the number of times each byte is used, its relative frequency, and the code for the byte to be printed on the standard output. Additional occurrences of **-** in place of *name* will cause the internal flag to be set and reset.

The amount of compression obtained depends on the size of the input file and the character frequency distribution. Because a decoding tree forms the first part of each **.z** file, it is usually not worthwhile to pack files smaller than three blocks, unless the character frequency distribution is very skewed, which may occur with printer plots or pictures.

Typically, text files are reduced to 60-75% of their original size. Load modules, which use a larger character set and have a more uniform distribution of characters, show little compression, the packed versions being about 90% of the original size.

Pack returns a value that is the number of files that it failed to compress.

No packing will occur if:

the file appears to be already packed;
the file is all nulls;
the file name has more than 12 characters;
the file has links;
the file is a directory;
the file cannot be opened;
no disk storage blocks will be saved by packing;
a file called *name*.**z** already exists;
the **.z** file cannot be created;
an I/O error occurred during processing.

The last segment of the file name must contain no more than 12 characters to allow space for the appended **.z** extension. Directories cannot be compressed.

Pcat does for packed files what *cat*(1) does for ordinary files, except that *pcat* can not be used as a filter. The specified files are unpacked and written to the standard output. Thus to view a packed file named *name*.**z** use:

```
pcat name.z
```

or just:

pcat name

To make an unpacked copy, say *nnn*, of a packed file named *name*.**z** (without destroying *name*.**z**) use the command:

pcat name >nnn

Pcat returns the number of files it was unable to unpack. Failure may occur if:

the file name (exclusive of the **.z**) has more than 12 characters;
the file cannot be opened;
the file does not appear to be the output of *pack*.

Unpack expands files created by *pack*. For each file *name* specified in the command, a search is made for a file called *name*.**z** (or just *name*, if *name* ends in **.z**). If this file appears to be a packed file, it is replaced by its expanded version. The new file has the **.z** suffix stripped from its name, and has the same access modes, access and modification dates, and owner as those of the packed file.

Unpack returns a value that is the number of files it was unable to unpack. Failure may occur for the same reasons that it may in *pcat*, as well as for the following:

a file with the "unpacked" name already exists;
if the unpacked file cannot be created.

SEE ALSO

cat(1).

NAME

pam – Personal Applications Manager, a visual shell

SYNOPSIS

pam [**–c** args ...]

HP-UX COMPATIBILITY

Level: NON-STANDARD

Origin: HP

DESCRIPTION

Pam is a program that helps provide a friendlier, less intimidating means of communication between HP-UX and system users. It provides many of the traditional capabilities supported by other shell programs such as executing commands as foreground processes (where you must wait until one command has been completed before the system accepts the next command) or background processes (where the command runs in the background while you perform other tasks in the foreground). *Pam* also supports other useful capabilities such as using substituted files instead of standard input and standard output, pipelining several processes into a single command, and handling shell scripts and programs. *Pam* maintains a continuous display of the open folder (current directory), and makes use of windowing and mouse I/O facilities when they are available on the system.

Display

The *pam* display has two parts. The top two lines are called the command area, while the remainder of the display is the folder (directory) area. The first line in the command area displays messages (such as prompts and errors) from the system to the user, while the second line displays input commands and text from the user to the system. *Pam* maintains a buffer of 20 command lines. You can use shifted arrow keys, BACK SPACE, INSERT CHARACTER, and DELETE CHARACTER to access and edit any existing current or previous command line in the 20-line command buffer. For example, each time you press SHIFT–UP ARROW, the next previous command line in the buffer is displayed.

The folder (lower) area is used by *pam* to display those files that reside in the currently open folder; that is, the current directory. One of the file names displayed in the folder area is highlighted. This highlighted filename identifies which file in the folder is to be used as a filename parameter for *pam* commands that are invoked using the *pam* menu. The highlighted file name can be changed by using TAB, SHIFT–TAB and arrow keys.

Commands

A command is a sequence of non-blank words separated by blanks. In general, the first word is the name of the command, and the words that follow are passed as arguments to the invoked command. Two or more commands (together with their associated arguments, if any) separated by a vertical bar (|) form a pipeline. To provide a path for passing data between commands in a pipeline, the standard output from one command in the pipeline is connected to the standard input of the next command in the pipe.

To force *pam* to complete execution of the current command or pipeline before running another command, place a semicolon (;) at the end of the line. If you prefer to perform other tasks while the command or pipeline is being executed, run the first command as a "background" process by adding an ampersand (&) at the end of the command line. *Pam* then starts the command, and, without waiting for completion, returns for your next instruction.

In a windowed system, interactive command inputs are treated as background processes (&) unless a semicolon is present at the end of the line. In non-windowed systems, commands taken from a script or from interactive command inputs are run to completion before the next command is accepted for execution (;) unless an ampersand is present at the end of the command line.

Sequences of more than one command or pipeline can be joined on a single command line by placing a semicolon or ampersand (but not both) between each adjacent pair of commands/pipelines in the line. In such constructs, commands separated by ";" are executed in sequence (the first command is run to completion before the next is begun). Commands separated by "&" are executed simultaneously on a timesharing basis (this does not necessarily result in the most efficient use of computer resources due to timesharing overhead as sharing processes compete for processor time). Note that when ";" or "&" is used to separate commands, standard output from one command is not automatically connected to standard input for the next. Use the pipeline connector (|) instead when data must be passed between successive commands or programs, or redirect standard output and input to and from a specified file.

Pam runs commands based on the file type of the command name:

program (executable) - The command name is run (exec'ed) or, if it is a shell script, the commands in the script are run.

folder (directory) - The command name (folder) becomes the new open folder. This is equivalent to *cd* folder.

data (non-executable) - The command name (data file) is displayed one page at a time.

Standard Input, Output, and Error Files

The standard input, output and error of a command can be redirected using the following syntax:

< *name* Use the file *name* as standard input for the command.

> *name* Use the file *name* as standard output for the command.

>> *name* Use the file *name* as standard output for the command, but concatenate the output to the end of the file.

^ *name* Use the file *name* as standard error for the command.

^^ *name* Use the file *name* as standard error for the command, but concatenate the output to the current end of the file, if it exists.

name [For windowed systems only] Use the named window as standard input, output and error for the command. If the window doesn't exist then a window is created. Specific redirection of I/O with >, >>, <, |, ^, or ^^ overrides any redirection specified with "#".

I/O redirection is possible only with an associated command. Multiple redirections of standard input, output, and error associated with a command are not allowed. The I/O redirection can be placed anywhere in the command.

If a command is followed by "&" (background process), the default standard input for the command is the empty file "/dev/null".

Using Patterns to Represent Filenames

Each word in the command line (command name, parameter, redirection file name, window name) is scanned for the characters *, ?, and [. If one of these characters appears, the word is treated as a pattern that represents more than one filename. *Pam* replaces the pattern word with alphabetically sorted filenames corresponding to the pattern. If no file name is found that matches the pattern, the word is left unchanged. A period character (.) at the start of a filename or immediately following a /, as well as the character / itself, must be matched explicitly.

* Matches any string, including the null string.

? Matches any single character.

[...] Matches any one of the enclosed characters. A pair of characters separated by - matches any character lexically between the pair, inclusive. A NOT operator, !, can be specified immediately following the left bracket to match any single character not enclosed in the brackets.

Quoting

Characters can be quoted on the *pam* command line to prevent *pam* from performing special processing on the characters (such as <, >, #, space, |, ;, &, *, ?, [). A pair of double (") or single (') quote characters can be used to enclose the character string being quoted. The \ character quotes only the single following character.

Scripts

A script is an executable file containing command lines and comments. A comment is a line that begins with "!" or "#"; comments are ignored by *pam*. The command lines in the script file are executed in sequence (unless non-sequential execution is explicitely specified using the "&" character).

Script arguments that are specified when a script is run can be accessed by script commands using the notation "$1" for the first argument, "$2" for the second argument, etc. All arguments can be accessed at once using "$*". The name of the script can be accessed using "$0".

Autost

If a file named *Autost* exists in the open folder when *pam* is started, it is automatically processed as a command. If *Autost* is a script file, it is run as **source** *Autost* (see Built-In Commands described later). Otherwise, it is processed as if it were entered as a command (for example, if *Autost* is a data file, it is viewed). *Pam* does not process any command input until processing of the *Autost* file is complete.

Environment

The *pam* environment is set up when *pam* is run, and can be reset at any time by using the command *getenv*. The environment variables are read from a file and are not sorted or checked for syntax by *pam*. *Pam* passes the current environment to commands that it starts and uses the following environment variables in running commands:

ACTION — The ACTION variable specifies a command name (corresponding to an executable file), and is used whenever a data file is specified as a command. The ACTION command is run in this case and the data file is passed as the first argument. The default value for ACTION is "view".

HOME — The HOME variable specifies a folder and is used whenever the "cd" command is run without an argument. The HOME folder specifies the directory to change to in this case. The default value for HOME is "/".

LANG — The LANG variable is added by *pam* to its environment file if it is not already there. This happens when *pam* is initialized or when the getenv command is done. The value of the LANG variable is set to match the language that the system is localized for.

PATH — The PATH variable specifies a list of folders. When *pam* runs a command it looks for it in the folders in the PATH list.

SCRSHELL — The SCRSHELL variable specifies the shell to be used by *pam* in running scripts. If the specified name does not contain a "/" then *pam* searches for the shell using the PATH environment variable. If the SCRSHELL is undefined or the specified shell does not exist, the script

is processed by *pam*.

Menu

The *pam* menu displays the following softkey menu labels corresponding to the indicated function keys:

[function key 1] **open**, **view**, or **start** (a program), or **reread**
[f2] **echo**
[f3] **send** or **arrow** (toggle key)
[f4] **move**
[f5] **copy**
[f6] **rename**
[f7] **delete**
[f8] **close**

The command associated with a menu item is run whenever the item is selected by pressing the corresponding function key. The highlighted file name in the folder area of the display is used as a parameter for the command.

The menu item associated with **f3** is used to toggle the semantics of the arrow keys and is available only in non-windowed systems. **f3** is initially set to use the arrow keys for manipulating the position of the file highlight in the folder area. **f3** alternately controls movement of the cursor on the command line.

Built-In Commands

Several commands are executed directly by *pam*:

cd [name] — Make the named folder (directory) the open folder (current directory). If no folder is specified, the HOME environment variable is used to determine which folder to open.

close — Closes the open folder and displays the parent folder.

copy*name1* [*name2*]
copy *name1* [*name2* ...] *folder_name* — Copy *name1* to *name2*; if exists and is not a folder, it is overwritten. If only *name1* is specified, the copy is completed with the command *to*name2. If the last parameter specified is a folder, all the specified files are copied into that folder.

delete *name1* [*name2* ...] — The named files and folders (if empty) are deleted.

echo [arg ...] — Arguments are written to standard output. The echo menu item (menu item 2 and/or function key 2) writes the full pathname of the highlighted file in the folder area of the display to the command line.

getenv *name* — The named file is read in and used as the active environment.

makefolder *name1* [*name2* ...] — Folders are created and given the specified name(s).

move *name1* [*name2*]
move *name1* [*name2*] *folder_name* — Rename file or folder *name1* to *name2*. If *name2* exists and is a file, it is overwritten. If only *name1* is specified, use the command *to name2* to complete the move. If the last parameter is a folder, all the specified files are moved

into that folder.

netunam *pathname* [*string*]
: Initiate a network connection to the specified system (as indicated by *pathname*) using the specified login (as indicated by *string*). If *string* is omitted, the network connection to the specified system, if currently active, is disconnected.

print *name1* [*name2* ...]
: Print the specified files on the designated system printer.

rename *name1* [*name2*]
rename *file_name1* [*file_name2* ...] *folder_name*
: Same as **move.**

reread
: Reread the open folder and update the display. The keystroke CONTROL–L also does a reread.

send
: Send the full *pathname* of the highlighted *filename* in the folder area to an application as if it were typed from the keyboard (windowed systems only).

source *name* [*arg* ...]
: Read command lines from the named script file and execute them. A shell is NOT forked to execute the commands. Parameter substitution for the arguments (*arg* ...) is handled the same way as during regular script execution.

stopprint
: Stop current printing activity if it was started by the **print** command.

to [*name*]
: Complete a pending copy, move or rename. The file or folder *name* identifies the destination for a preceding **copy**, **move**, or **rename** command that had no destination specified. If *name* is omitted, the destination defaults to the current open folder.

view *name1* [*name2* ...]
: Copy the specified file(s) to standard output. If standard output is the screen (default), the file is displayed one page at a time.

Signals

Pam ignores INTERRUPT and QUIT signals if the command is followed by an "&"; otherwise *pam* uses default signal handling when running commands.

Invoking *Pam*

Pam can be invoked as a keyboard command or from a program. When *pam* is invoked without **-c** as the first argument, *pam* acts as an interactive, display-oriented command interpreter.

When *pam* is invoked with **-c** as the first argument (either from the keyboard or from a running program), the remaining arguments in the command are interpreted as command inputs intended for processing by *pam*. The list of arguments intended as commands for *pam* must not exceed a total of 160 characters. When the **-c** option is used, *pam* executes the list of command arguments (built-in commands, redirection, pipes, and most other *pam* facilities can be used), then exits.

Exiting from *Pam*

To terminate *pam* and return to normal HP-UX operation, press CONTROL–D or CONTROL–C.

HARDWARE DEPENDENCIES

With a non-windowed *pam* running on certain terminals the shifted right and left arrow keys cannot be used to move the cursor on the command line.

IPC: *pam* runs commands with the SIGHUP signal ignored.
The *netunam* built-in command is not supported.

Series 300/500: The built-in commands *print*, *stopprint*, and *send* are not supported.
The environment variable LANG is not set up by *pam*.

FILES

/rom/PAM
/rom/.environ
/tmp/Plock
Autost
/rom/PAMmsg
/usr/lib/nls/n-computer/pam.cat
/dev/null

NAME

passwd - change login password

SYNOPSIS

passwd [name]

HP–UX COMPATIBILITY

Level: HP–UX/STANDARD

Origin: System V

Native Language Support:
8-bit passwords.

Remarks: Not supported on the Integral Personal Computer.

DESCRIPTION

This command changes or installs a password associated with the login *name*. If *name* is omitted, it defaults to *getlogin*(3) name.

Ordinary users may change only the password which corresponds to their login *name*.

Passwd prompts ordinary users for their old password, if any. It then prompts for the new password twice. The first time the new password is entered *passwd* checks to see if the old password has "aged" sufficiently. If "aging" is insufficient the new password is rejected and *passwd* terminates; see *passwd*(4).

Assuming "aging" is sufficient, a check is made to insure that the new password meets construction requirements. When the new password is entered a second time the two copies of the new password are compared. If the two copies are not identical the cycle of prompting for the new password is repeated for at most two more times.

Passwords must be constructed to meet the following requirements:

Each password must have at least six characters. Only the first eight characters are significant.

Each password must contain at least two alphabetic characters and at least one numeric or special character. In this case, "alphabetic" means upper and lower case letters.

Each password must differ from the user's login *name* and any reverse or circular shift of that login *name*. For comparison purposes, an upper case letter and its corresponding lower case letter are equivalent.

New passwords must differ from the old by at least three characters. For comparison purposes, an upper case letter and its corresponding lower case letter are equivalent.

One whose effective user ID is zero is called a super–user; see *id*(1), and *su*(1). Super–users may change any password; hence, *passwd* does not prompt super–users for the old password. Super–users are not forced to comply with password aging and password construction requirements. A super–user can create a null password by entering a carriage return in response to the prompt for a new password.

FILES

/etc/passwd

SEE ALSO

login(1), id(1), su(1), crypt(3C), passwd(5).

DIAGNOSTICS

Permission denied.
name is not in password file, or you are not user *name* or the super–user.

Sorry. the old password does not match.

Sorry: <x weeks since the last change
password aging is in effect, and it is too soon to change yours.

You may not change this password
the super–user has made it impossible to change your password.

Too short
passwords must be at least 4 characters long.

Please use at least one non–numeric character
your new password does not utilize a sufficiently varied selection of characters. You can override this rule by re–entering your new password 2 more times.

Please use a longer password.
your new password is not long enough to be sufficiently secure. You can override this rule by re–entering your new password 2 more times.

They don't match, try again.
the two entries of your new password are not identical.

Temporary file busy; try again later
only one user can change his password at a time.

Cannot create temporary file
see the super–user.

Cannot unlink 'filename'
see the super–user.

cannot link 'file' to 'file'.
see the super–user.

cannot recover 'file'.
see the super–user.

Password unchanged.
the new and old passwords are the same.

NAME

paste - merge same lines of several files or subsequent lines of one file

SYNOPSIS

paste file1 file2 ...
paste -d list file1 file2 ...
paste -s [**-d** list] file1 file2 ...

HP–UX COMPATIBILITY

Level: HP–UX/STANDARD

Origin: System V

DESCRIPTION

In the first two forms, *paste* concatenates corresponding lines of the given input files *file1*, *file2*, etc. It treats each file as a column or columns of a table and pastes them together horizontally (parallel merging). If you will, it is the counterpart of *cat*(1) which concatenates vertically, i.e., one file after the other. In the last form above, *paste* replaces the function of an older command with the same name by combining subsequent lines of the input file (serial merging). In all cases, lines are glued together with the *tab* character, or with characters from an optionally specified *list*. Output is to the standard output, so it can be used as the start of a pipe, or as a filter, if **-** is used in place of a file name.

The meanings of the options are:

-d Without this option, the new–line characters of each but the last file (or last line in case of the **-s** option) are replaced by a *tab* character. This option allows replacing the *tab* character by one or more alternate characters (see below).

list One or more characters immediately following **-d** replace the default *tab* as the line concatenation character. The list is used circularly, i.e., when exhausted, it is reused. In parallel merging (i.e., no **-s** option), the lines from the last file are always terminated with a new–line character, not from the *list*. The list may contain the special escape sequences: **\n** (new–line), **\t** (tab), **** (backslash), and **\0** (empty string, not a null character). Quoting may be necessary, if characters have special meaning to the shell (e.g., to get one backslash, use *-d"\\\\"*).

-s Merge subsequent lines rather than one from each input file. Use *tab* for concatenation, unless a *list* is specified with **-d** option. Regardless of the *list*, the very last character of the file is forced to be a new–line.

- May be used in place of any file name, to read a line from the standard input. (There is no prompting).

EXAMPLES

ls | paste -d" " -	list directory in one column
ls | paste - - - -	list directory in four columns
paste -s -d"\t\n" file	combine pairs of lines into lines

SEE ALSO

grep(1), cut(1),
pr(1): **pr -t -m**... works similarly, but creates extra blanks, tabs and new–lines for a nice page layout.

DIAGNOSTICS

line too long	Output lines are restricted to 511 characters.
too many files	Except for **-s** option, no more than 12 input files may be specified.

NAME

pc - Pascal compiler

SYNOPSIS

pc [options] files

HP–UX COMPATIBILITY

Level: HP–UX/STANDARD

Origin: HP

Native Language Support:
8-bit and 16-bit characters in strings and comments.

Remarks: This manual page describes the generic HP Pascal compiler; implementation dependencies for different machines are noted as needed. Currently the Series 300 and Series 500 computers support HP Pascal under HP–UX, as described in this document.

DESCRIPTION

Pc is the HP standard Pascal compiler. It accepts several types of file arguments:

(1) Arguments whose names end with **.p** are taken to be Pascal source files. They are each compiled, and each corresponding archived object program or module(s) is left in the current directory in a file whose name is that of the source, with **.a** substituted for **.p**. Note that the Series 500 does not produce archived object (**.a**) files, instead it produces simple object (**.o**) files (where the **.p** is replaced with **.o**). The **.a** (or **.o**) file will not be created if only a single source is compiled and linked, if the **-C** option is specified, or if the source fails to compile correctly.

(2) All other file arguments, including those whose names end with **.o** or **.a** are passed on to the linker (*ld*(1)) to be linked into the final program.

Arguments can be passed to the compiler through the PCOPTS environment variable as well as on the command line. The compiler picks up the value of PCOPTS and places its contents before any arguments on the command line. For example (in *sh*(1) notation),

```
$ PCOPTS=–v
$ export PCOPTS
$ pc –L prog.p
```

is equivalent to

```
$ pc –v –L prog.p
```

The following options are recognized:

-A Produce warnings for the use of non ANSI–Pascal features. (same as **$ANSI ON$**).

-C Suppress code generation - no **.a** (or **.o**) files will be created and linking will be suppressed. This is effectively a request for syntax/semantic checking only (same as **$CODE OFF$**).

-c Suppress linking and only produce object (**.a** or **.o**) files from source files.

-g Generate additional information needed by the Pascal Symbolic Debugger (pdb), and ensure that the program is linked as required by pdb. See the appropriate implementation reference manual for more information on symbolic debugging support.

-L Write a program listing to *stdout* (or for the Series 300 only, to the file given in the **$LIST filename$** option in the source).

-l*x* Cause the linker to search the library named either **/lib/lib***x***.a** (tried first) or **/usr/lib/lib***x***.a**. (See *ld*(1).)

-N Cause the output file from the linker to be marked as unshareable (see **-n**). For details and system defaults, refer to the linker documentation (*ld*(1)).

-n Cause the output file from the linker to be marked as shareable (see **-N**). For details and system defaults, refer to the linker documentation (*ld*(1)).

-o *outfile* Name the output file from the linker *outfile* instead of *a.out*.

-P *lines* Specifies the number of lines (including any header or trailer) which should be listed per page of generated listing (same as **$LINES n$**).

-Q Cause the output file from the linker to be marked as not demand loadable (see **-q**). For details and system defaults, refer to the linker documentation (*ld*(1)).

-q Cause the output file from the linker to be marked as demand loadable (see **-Q**). For details and system defaults, refer to the linker documentation (*ld*(1)).

-s Cause the output of the linker to be *stripped* of symbol table information (see *ld*(1) and *strip*(1)). (This option is incompatible with symbolic debugging).

-t *c,name* Substitute or insert subprocess *c* with *name* where *c* is one or more of a set of identifiers indicating the subprocess(es). This option works in two modes: 1) if *c* is a single identifier, *name* represents the full path name of the new subprocess; 2) if *c* is a set of (more than one) identifiers, *name* represents a prefix to which the standard suffixes are concatenated to construct the full path name of the new subprocesses.

The values *c* can take on are:

c compiler body (standard suffix is *pascomp*)
0 same as *c*
l linker (standard suffix is *ld*)

-v Enable verbose mode, producing a step–by–step description of the compilation process on *stderr*.

-w Suppress warning messages (same as **$WARN OFF$**);

-W *c,arg1[,arg2,...,argN]*

Cause *arg1* through *argN* to be handed off to subprocess *c*. The *argi* are of the form *-argoption[,argvalue]*, where *argoption* is the name of an option recognized by the subprocess and *argvalue* is a separate argument to *argoption* where necessary. The values that *c* can assume are those listed under the **-t** option, as well as the value *d* (driver program) which has a special meaning explained below.

For example, the specification to pass the –r (preserve relocation information) option to the linker would be:

–W l,–r

The **-W d** option specification allows additional, implementation–specific options to be recognized and passed through the compiler driver to the appropriate compiler subprocesses. For example,

–W d,–U (Series 500)

will send the option *-U* to the driver and compiler. Furthermore, a shorthand notation for this mechanism can be used by prepending + to the option name; as in

+U

which is equivalent to the previous option expression. Note that for simplicity this shorthand is applied to each implementation–specific option individually, and that the *argvalue* is no longer separated from the *argoption* by a comma (see **-W**).

-Y Enable 16-bit Native Language Support when parsing string literals and comments (same as **$HP16 ON$**). Note that 8-bit parsing is always supported.

The implementation–specific options on the Series 500 are:

+E Cause the program to be linked with the library **/lib/libpcesc.a** which transforms all execution errors (HP–UX signals, Pascal run–time errors, Pascal I/O errors and HP–UX errors) into escapes. This differs from the default library **/lib/libpccat.a** which prints the appropriate error message and aborts the program.

+F Cause the compiler to generate information which is used by various program analyzers.

+H [*bytes*] Display (if *bytes* is omitted) or set a Pascal program's maximum heap size. *Bytes* is the maximum number of bytes in the heap.

+Q *dfile* Cause *dfile* to be read before compilation of each source file. *Dfile* may only contain compiler options.

+U Cause the compiler to upshift externally visible names, default is lower case (same as **$UPSHIFT_LEVEL1 ON$**).

+W [*bytes*] Display (if *bytes* is omitted) or set a Pascal program's working set size. *Bytes* is the number of bytes in the program's working set;

FILES

file.p	input file (Pascal source file)
file.a	generated object archive file (except for the Series 500) or any archive file to be searched at link time
file.o	object file to be relocated at link time (or generated object file for the Series 500 only)
a.out	linked executable output file
/bin/pc	compiler and linker driver program
/usr/lib/pascomp	compiler
/usr/lib/paserrs	compiler error message file
/usr/lib/escerrs	Pascal escape codes (Series 300 only)
/usr/lib/syserrs	HP–UX system messages (Series 300 only)
/usr/lib/ioerrs	Pascal I/O results (Series 300 only)
/lib/crt0.o	runtime startup (Series 300 only)
/lib/prt0.o	runtime startup (Series 500 only)
/lib/libpc.a	Pascal run–time library
/lib/libm.a	HP-UX math library (Series 300 only)
/lib/libpccat.a	Pascal run–time library, reports errors and aborts program (Series 500 only)
/lib/libpcesc.a	Pascal run–time library, translates errors into escapes (Series 500 only)
/usr/tmp/*	temporary files used by the compiler; names are created by *tmpnam*(3S).

SEE ALSO

HP Pascal Language Reference, HP Part No. 98680–90015 (Series 300).
Pascal/9000 Language Reference Manual, HP Part No. 97082–90001 (Series 500).
Programming in Pascal with Hewlett–Packard Pascal, by Peter Grogono (Series 300/500).

DIAGNOSTICS

The diagnostics produced by *pc* are intended to be self–explanatory. Occasional messages may be produced by the linker.

A list of all compiler errors may be found in /usr/lib/paserrs.

If a listing is requested (**-L** option), errors are written to the listing file (*stdout*). If a listing is requested and either or both of *stdout/stderr* has been redirected to something other than a terminal, errors will also be written to *stderr*. If no listing is requested (no **-L** option), errors are written to *stderr*. This effectively guarantees that *stderr* will always recieve error messages, unless that would result in duplication of error messages printed on the terminal.

HARDWARE DEPENDENCIES

Series 500:

The following options are no longer supported: **-b**, **-e**, and **-f**.

The following options must now be specified with the **-W d,...** option or the + shorthand: **-E**, **-F**, **-H** [*bytes*], and **-W** [*bytes*].

To use the **+H** or **+W** options on an executable file other than **a.out**, the file to be examined (modified) must be specified with the **-o** option. To set the heap of program foo to 1000000

use:

pc +H 1000000 –o foo

do not use:

pc +H 1000000 –W l,–o,foo

NAME

pr - print files

SYNOPSIS

pr [options] [files]

HP–UX COMPATIBILITY

Level: HP–UX/STANDARD

Origin: System V

Native Language Support:
8-bit data, customs, messages.

DESCRIPTION

Pr prints the named files on the standard output. If *file* is **-**, or if no files are specified, the standard input is assumed. By default, the listing is separated into pages, each headed by the page number, a date and time, and the name of the file.

By default, columns are of equal width, separated by at least one space; lines which do not fit are truncated. If the **-s** option is used, lines are not truncated and columns are separated by the separation character.

If the standard output is associated with a terminal, error messages are withheld until *pr* has completed printing.

The below *options* may appear singly or be combined in any order:

+*k* Begin printing with page *k* (default is 1).

-*k* Produce *k*–column output (default is 1). The options **-e** and **-i** are assumed for multi–column output.

-a Print multi–column output across the page.

-m Merge and print all files simultaneously, one per column (overrides the **-***k*, and **-a** options).

-d Double–space the output.

-e*ck* Expand *input* tabs to character positions *k*+1, 2**k*+1, 3**k*+1, etc. If *k* is 0 or is omitted, default tab settings at every eighth position are assumed. Tab characters in the input are expanded into the appropriate number of spaces. If *c* (any non–digit character) is given, it is treated as the input tab character (default for *c* is the tab character).

-i*ck* In *output*, replace white space wherever possible by inserting tabs to character positions *k*+1, 2**k*+1, 3**k*+1, etc. If *k* is 0 or is omitted, default tab settings at every eighth position are assumed. If *c* (any non–digit character) is given, it is treated as the output tab character (default for *c* is the tab character).

-n*ck* Provide *k*–digit line numbering (default for *k* is 5). The number occupies the first *k*+1 character positions of each column of normal output or each line of **-m** output. If *c* (any non–digit character) is given, it is appended to the line number to separate it from whatever follows (default for *c* is a tab).

-w*k* Set the width of a line to *k* character positions (default is 72 for equal–width multi–column output, no limit otherwise).

-o*k* Offset each line by *k* character positions (default is 0). The number of character positions per line is the sum of the width and offset.

-l*k* Set the length of a page to *k* lines (default is 66).

-h Use the next argument as the header to be printed instead of the file name.

-p Pause before beginning each page if the output is directed to a terminal (*pr* will ring the bell at the terminal and wait for a carriage return).

-f Use form–feed character for new pages (default is to use a sequence of line–feeds). Pause before beginning the first page if the standard output is associated with a terminal.

-r Print no diagnostic reports on failure to open files.

-t Print neither the five–line identifying header nor the five–line trailer normally supplied for each page. Quit printing after the last line of each file without spacing to the end of the page.

-s*c* Separate columns by the single character *c* instead of by the appropriate number of spaces (default for *c* is a tab).

EXAMPLES

Print **file1** and **file2** as a double–spaced, three–column listing headed by "file list":

```
pr -3dh "file list" file1 file2
```

Write **file1** on **file2**, expanding tabs to columns 10, 19, 28, 37, . . . :

```
pr -e9 -t <file1 >file2
```

FILES

/dev/tty* to suspend messages

SEE ALSO

cat(1), lp(1), ul(1).

NAME

prealloc - preallocate disk storage

SYNOPSIS

prealloc name size

HP–UX COMPATIBILITY

Level: HP–UX/STANDARD

Origin: HP

DESCRIPTION

Prealloc will preallocate at least *size* bytes of disk space for an ordinary file *name* of zero length. It will create the file if it does not already exist. The space will be allocated in an implementation dependent fashion for fast sequential reads and writes for the file.

Prealloc will fail and no disk space will be allocated if *name* already exists and is not an ordinary file of zero length, if there is not enough space left on disk, or if *size* exceeds the maximum file size or the process' file size limit (see *ulimit*(2)). The EOF is left at the end of the preallocated area. The current file pointer is left at zero. The file is zero–filled.

DIAGNOSTICS

Upon successful completion, *prealloc* exits with a 0 status. Exit status is 1 if *name* already exists and is not an ordinary file of zero length, 2 if there is not enough room on disk, or 3 if *size* exceeds file size limits.

SEE ALSO

prealloc(2), ulimit(2)

BUGS

The allocation of the file space is highly dependent on the current disk usage. A successful return does not tell you how fragmented the file actually might be if the disk is reaching its capacity.

NAME

prof - display profile data

SYNOPSIS

prof [**-a**] [**-l**] [file]

HP-UX COMPATIBILITY

Level: HP-UX/STANDARD

Origin: System III

Remarks: *Prof*(1) is implemented on the Series 200 only.

DESCRIPTION

Prof interprets the file **mon.out** produced by the *monitor*(3C) subroutine. Under default modes, the symbol table in the named object file (**a.out** default) is read and correlated with the **mon.out** profile file. For each external symbol, the percentage of time spent executing between that symbol and the next is printed (in decreasing order), together with the number of times that routine was called and the number of milliseconds per call.

If the **-a** option is used, all symbols are reported rather than just external symbols. If the **-l** option is used, the output is listed by symbol value rather than decreasing percentage.

In order for the number of calls to a routine to be tallied, the **-p** option of *cc* must have been given when the file containing the routine was compiled. This option also arranges for the **mon.out** file to be produced automatically.

FILES

mon.out for profile
a.out for namelist

SEE ALSO

cc(1), profil(2), monitor(3C).

BUGS

Beware of quantization errors.

NAME

prs – print and summarize an SCCS file

SYNOPSIS

prs [–**d**[dataspec]] [–**r**[SID]] [–**e**] [–**l**] [–**c**[date-time]] [–**a**] files

HP-UX COMPATIBILITY

Level: HP-UX/STANDARD

Origin: System V

DESCRIPTION

Prs prints, on the standard output, parts or all of an SCCS file (see *sccsfile*(5)) in a user-supplied format. If a directory is named, *prs* behaves as though each file in the directory were specified as a named file, except that non-SCCS files (last component of the path name does not begin with **s.**), and unreadable files are silently ignored. If a name of **-** is given, the standard input is read; each line of the standard input is taken to be the name of an SCCS file to be processed; non-SCCS files and unreadable files are silently ignored.

Arguments to *prs*, which can appear in any order, consist of *keyletter* arguments and file names.

All the described *keyletter* arguments apply independently to each named file:

–**d**[dataspec] Used to specify the output data specification. The *dataspec* is a string consisting of SCCS file data keywords (see Data Keywords below) interspersed with optional user-supplied text.

–**r**[SID] Used to specify the *S*CCS *ID*entification (SID) string of a delta for which information is desired. If no SID is specified, the SID of the most recently created delta is assumed. If an SID is specified, it must agree exactly with an SID in the file (i.e., the SID structure used by *get*(1) does not work here).

–**e** Requests information for all deltas created *earlier* than and including the delta designated via the –**r** keyletter or the date given by the –**c** option.

–**l** Requests information for all deltas created *later* than and including the delta designated via the –**r** keyletter or the date given by the –**c** option.

–**c**[cutoff] Cutoff date-time, in the form

YY[MM[DD[HH[MM[SS]]]]]

Units omitted from the date-time default to their maximum possible values. Thus, –**c7502** is equivalent to –**c750228235959**. One or more non-numeric characters can be used to separate the various 2-digit segments of the cutoff date (for example –**c77/2/2 9:22:25**).

–**a** Requests printing of information for both removed; i.e., delta type = *R*, (see *rmdel*(1)) and existing; i.e., delta type = *D*, deltas. If the –**a** keyletter is not specified, information for existing deltas only is provided.

If no option letters (or only –**a**) are given, *prs* prints the file name using the default *dataspec* and the –**e** option. This produces information on all deltas.

Data Keywords

Data keywords specify which parts of an SCCS file are to be retrieved and output. All parts of an SCCS file (see *sccsfile*(5)) have an associated data keyword. There is no limit on the number of times a data keyword can appear in a *dataspec*.

The information printed by *prs* consists of: (1) the user supplied text; and (2) appropriate values (extracted from the SCCS file) substituted for recognized data keywords in their order of appearance in the *dataspec*. The format of a data keyword value is either *Simple* (S), where keyword substitution is direct, or *Multi-line* (M), where keyword substitution is followed by a carriage return.

User-supplied text is any text other than recognized data keywords. Escapes can be used as follows:

tab	\t
new-line	\n
colon	\:
backspace	\b
carriage-return	\r
form feed	\f
backslash	\\
single quote	\'

The default *dataspec* is:

:Dt:\t:DL:\n:MRs:\n:MR:COMMENTS:\n:C:

TABLE 1. SCCS Files Data Keywords

Keyword	*Data Item*	*File Section*	*Value*	*Format*
:Dt:	Delta information	Delta Table	See below*	S
:DL:	Delta line statistics	"	:Li:/:Ld:/:Lu:	S
:Li:	Lines inserted by Delta	"	nnnnn	S
:Ld:	Lines deleted by Delta	"	nnnnn	S
:Lu:	Lines unchanged by Delta	"	nnnnn	S
:DT:	Delta type	"	*D* or *R*	S
:I:	SCCS ID string (SID)	"	:R:.:L:.:B:.:S:	S
:R:	Release number	"	nnnn	S
:L:	Level number	"	nnnn	S
:B:	Branch number	"	nnnn	S
:S:	Sequence number	"	nnnn	S
:D:	Date Delta created	"	:Dy:/:Dm:/:Dd:	S
:Dy:	Year Delta created	"	nn	S
:Dm:	Month Delta created	"	nn	S
:Dd:	Day Delta created	"	nn	S
:T:	Time Delta created	"	:Th::Tm::Ts:	S
:Th:	Hour Delta created	"	nn	S
:Tm:	Minutes Delta created	"	nn	S
:Ts:	Seconds Delta created	"	nn	S
:P:	Programmer who created Delta	"	logname	S
:DS:	Delta sequence number	"	nnnn	S
:DP:	Predecessor Delta seq-no.	"	nnnn	S
:DI:	Seq-no. of deltas incl., excl., ignored	"	:Dn:/:Dx:/:Dg:	S
:Dn:	Deltas included (seq #)	"	:DS: :DS:...	S
:Dx:	Deltas excluded (seq #)	"	:DS: :DS:...	S
:Dg:	Deltas ignored (seq #)	"	:DS: :DS:...	S
:MR:	MR numbers for delta	"	text	M
:C:	Comments for delta	"	text	M
:UN:	User names	User Names	text	M
:FL:	Flag list	Flags	text	M
:Y:	Module type flag	"	text	S
:MF:	MR validation flag	"	*yes* or *no*	S
:MP:	MR validation pgm name	"	text	S
:KF:	Keyword error/warning flag	"	*yes* or *no*	S

TABLE 1 (continued). SCCS Files Data Keywords

Keyword	*Data Item*	*File Section*	*Value*	*Forma*
:KV:	Keyword validation string	"	text	S
:BF:	Branch flag	"	*yes* or *no*	S
:J:	Joint edit flag	"	*yes* or *no*	S
:LK:	Locked releases	"	:R:...	S
:Q:	User defined keyword	"	text	S
:M:	Module name	"	text	S
:FB:	Floor boundary	"	:R:	S
:CB:	Ceiling boundary	"	:R:	S
:Ds:	Default SID	"	:I:	S
:ND:	Null delta flag	"	*yes* or *no*	S
:FD:	File descriptive text	Comments	text	M
:BD:	Body	Body	text	M
:GB:	Gotten body	"	text	M
:W:	A form of *what*(1) string	N/A	:Z::M:\t:I:	S
:A:	A form of *what*(1) string	N/A	:Z::Y: :M: :I::Z:	S
:Z:	*what*(1) string delimiter	N/A	@(#)	S
:F:	SCCS file name	N/A	text	S
:PN:	SCCS file path name	N/A	text	S

* :Dt: = :DT: :I: :D: :T: :P: :DS: :DP:

EXAMPLES

prs –d"Users and/or user IDs for :F: are:\n:UN:" s.file

produces text similar to the following on the standard output:

Users and/or user IDs for s.file are:
xyz
131
abc

prs –d"Newest delta for pgm :M:: :I: Created :D: By :P:" –r s.file

produces text similar to the following on the standard output:

Newest delta for pgm main.c: 3.7 Created 77/12/1 By cas

As a special case (when no **–d** keyletter is given):

prs s.file

produces text similar to the following on the standard output:

D 1.1 77/12/1 00:00:00 cas 1 000000/00000/00000
MRs:
bl78-12345
bl79-54321
COMMENTS:
this is the comment line for s.file initial delta

for each delta table entry of the "D" type.

FILES

/tmp/pr????? {temp files exist only while psr is active.}

SEE ALSO

admin(1), delta(1), get(1), help(1), sccsfile(5).

Source Code Control System User's Guide in *HP-UX Concepts and Tutorials.*

DIAGNOSTICS

Use *help*(1) for explanations.

NAME

ps - report process status

SYNOPSIS

ps [**-edafl**] [**-c** corefile] [**-s** swapdev] [**-n** namelist] [**-t** tlist] [**-p** plist] [**-u** ulist] [**-g** glist]

HP–UX COMPATIBILITY

Level: HP–UX/NUCLEUS

Origin: System V and HP

DESCRIPTION

Ps prints certain information about active processes. Without *options*, information is printed about processes associated with the current terminal. The output consists of a short listing containing only the process ID, terminal identifier, cumulative execution time, and the command name. Otherwise, the information that is displayed is controlled by the selection of *options*.

Options using lists as arguments can have the list specified in one of two forms: a list of identifiers separated from one another by a comma, or a list of identifiers enclosed in double quotes and separated from one another by a comma and/or one or more spaces.

The *options* are:

-e Print information about all processes.

-d Print information about all processes, except process group leaders.

-a Print information about all processes, except process group leaders and processes not associated with a terminal.

-f Generate a *full* listing. (See below for meaning of columns in a full listing.)

-l Generate a *long* listing. (See below.)

-c *corefile* Use the file *corefile* in place of **/dev/mem**.

-s *swapdev* Use the file *swapdev* in place of **/dev/swap**. This is useful when examining a *corefile*; a *swapdev* of **/dev/null** will cause the user block to be zeroed out.

-n *namelist* The argument will be taken as the name of an alternate system *namelist* file in place of **/hp–ux**.

-t *termlist* Restrict listing to data about the processes associated with the terminals given in *termlist*. Terminal identifiers may be specified in one of two forms: the device's file name (e.g., **tty04**) or if the device's file name starts with **tty**, just the digit identifier (e.g., **04**).

-p *proclist* Restrict listing to data about processes whose process ID numbers are given in *proclist*.

-u *uidlist* Restrict listing to data about processes whose user ID numbers or login names are given in *uidlist*. In the listing, the numerical user ID will be printed unless the **-f** option is used, in which case the login name will be printed.

-g *grplist* Restrict listing to data about processes whose process group leaders are given in *grplist*.

The column headings and the meaning of the columns in a *ps* listing are given below; the letters **f** and **l** indicate the option (*full* or *long*) that causes the corresponding heading to appear. **All** means that the heading always appears. Note that these two options determine only what information is provided for a process; they do *not* determine which processes will be listed.

F (l) Flags (octal and additive) associated with the process:

0	swapped;
1	in core;
2	system process;
4	locked in core (e.g., for physical I/O);
10	being swapped;
20	being traced by another process;

40 another tracing flag;

S (l) The state of the process:
- 0 non–existent;
- S sleeping;
- W waiting;
- R running;
- I intermediate;
- Z terminated;
- T stopped;
- X growing.

UID (f,l) The user ID number of the process owner; the login name is printed under the **-f** option.

PID (all) The process ID of the process; it is possible to kill a process if you know this datum.

PPID (f,l) The process ID of the parent process.

C (f,l) Processor utilization for scheduling.

PRI (l) The priority of the process; higher numbers mean lower priority.

NI (l) Nice value; used in priority computation.

ADDR (l) The memory address of the process, if resident; otherwise, the disk address.

SZ (l) The size in blocks of the core image of the process.

WCHAN (l) The event for which the process is waiting or sleeping; if blank, the process is running.

STIME (f) Starting time of the process.

TTY (all) The controlling terminal for the process (without the initial "tty", if any).

TIME (all) The cumulative execution time for the process (reported in the form "min:sec").

CMD (all) The command name; the full command name and its arguments are printed under the **-f** option.

A process that has exited and has a parent, but has not yet been waited for by the parent, is marked **<defunct>** (see "zombie process" in *exit*(2)).

Under the **-f** option, *ps* tries to determine the command name and arguments given when the process was created by examining memory or the swap area. Failing this, the command name, as it would appear without the **-f** option, is printed in square brackets.

To make *ps* output safer to display and easier to read, all control characters in the CMD field are mapped to "visible" equivalents. These are of the form ^*C* where the original character was in the range 0 – 037 and **c** is that value plus 040.

HARDWARE DEPENDENCIES

Series 500:

The **F** field is always 01.
In the **S** field, **I** means "waiting for input from terminal".
In the **S** field, the **P** (paused) state is added.
In the **S** field, the **T** state is not currently supported.
In the **S** field, **L** means "waiting on a file lock via *lockf*(2)".
In the **S** field, the **B** (blocked) state means "blocked via an IPC system call such as *semop*(2), *msgrcv*(2), or *msgsnd*(2)".
The **C** field is always zero.
The **ADDR** field reports the partition number.
In the **SZ** field, the block size is 1K bytes.
The **WCHAN** field is always blank.
The **CMD** field is renamed **COMMAND** except when the **-fl** option is specified.

The definition of **STIME** is as follows:

The time when the process was forked, *not* the time when it was modified by

exec; the date is included only if the elapsed time is greater than 24 hours.

The **s**, **n**, and **c** options are not currently supported. A diagnostic is printed if they are used.

The files */dev/mem, /dev/swap, /etc/ps_data*, and */hp-ux* do not exist.

FILES

/hp-ux system namelist
/dev/mem
memory
/dev/swap
the default swap device
/etc/passwd
supplies UID information
/etc/ps_data
internal data structure
/dev searched to find terminal ("tty") names

SEE ALSO

acctcom(1), kill(1), nice(1), exec(2), exit(2) lockf(2), msgop(2), semop(2).

BUGS

Things can change while *ps* is running; the picture it gives is only a snapshot in time. Some data printed for defunct processes are irrelevant.

If two special files for terminals are located at the same select code, they are reported in the order in which they appear in **/dev**, not in alphabetical order.

NAME

ptx - permuted index

SYNOPSIS

ptx [options] [input [output]]

HP–UX COMPATIBILITY

Level: HP–UX/STANDARD

Origin: System V

Remarks: Not supported on the Integral Personal Computer.

DESCRIPTION

Ptx generates the file *output* that can be processed with a text formatter to produce a permuted index of file *input* (standard input and output default). It has three phases: the first does the permutation, generating one line for each keyword in an input line. The keyword is rotated to the front. The permuted file is then sorted. Finally, the sorted lines are rotated so the keyword comes at the middle of each line. *Ptx* output is in the form:

.xx "tail" "before keyword" "keyword and after" "head"

where **.xx** is assumed to be an *nroff* or *troff*(1) macro provided by the user, or provided by the *mptx*(7) macro package. The *before keyword* and *keyword and after* fields incorporate as much of the line as will fit around the keyword when it is printed. *Tail* and *head*, at least one of which is always the empty string, are wrapped–around pieces small enough to fit in the unused space at the opposite end of the line.

The following *options* can be applied:

-f Fold upper and lower case letters for sorting.

-t Prepare the output for the phototypesetter by using a line length of 100.

-w *n* Use the next argument, *n*, as the length of the output line. The default line length is 72 characters for *nroff* and 100 for *troff*.

-g *n* Use the next argument, *n*, as the number of characters that *ptx* will reserve in its calculations for each gap among the four parts of the line as finally printed. The default gap is 3.

-o *only* Use as keywords only the words given in the *only* file.

-i *ignore* Do not use as keywords any words given in the *ignore* file. If the **-i** and **-o** options are missing, use **/usr/lib/eign** as the *ignore* file.

-b *break* Use the characters in the *break* file to separate words. Tab, new–line, and space characters are *always* used as break characters. Punctuation characters are treated as part of the word in the absence of this option.

-r Take any leading non–blank characters of each input line to be a reference identifier (as to a page or chapter), separate from the text of the line. Attach that identifier as a 5th field on each output line.

The index for this manual was generated using *ptx*.

FILES

/bin/sort
/usr/lib/eign
/usr/lib/tmac/tmac.ptx

SEE ALSO

nroff(1), troff(1), mm(7), mptx(7).

BUGS

Line length counts do not account for overstriking or proportional spacing.

Lines that contain tildes (~) are botched, because *ptx* uses that character internally.

NAME

pwd - working directory name

SYNOPSIS

pwd

HP-UX COMPATIBILITY

Level: HP-UX/STANDARD

Origin: System V

DESCRIPTION

Pwd prints the path name of the working (current) directory.

SEE ALSO

cd(1).

DIAGNOSTICS

"Cannot open .." and "Read error in .." indicate possible file system trouble and should be referred to the system manager.

NAME

query - interactive IMAGE database access

SYNOPSIS

query

HP–UX COMPATIBILITY

Level: HP–UX/NON–STANDARD

Origin: HP

Remarks: *Query* is implemented on the Series 500 only. IMAGE must be previously installed on the system for *query* to function.

DESCRIPTION

Query is an interactive, command driven program to simplify IMAGE database access. It can be used to generate reports from the database, add information to the database, change information in the database, and aid in developing programs that access databases using IMAGE library subroutines.

Consistent with the HP–UX environment in which it operates, *query* is initiated by simply typing its name. There are no options or parameters. Input and output redirection can be done at the shell level (< >) although more convenient methods are available via *query* commands.

A list of the available commands:

data–base=	help	exit	!
list	form	find	xeq
update a	update d	update r	report

Query accepts these commands in upper– or lower–case. Special care must be taken in the case of set names, item names, and item values since these are case sensitive. That is, Setname, setname, and SETNAME are three unique sets.

All *query* commands must be followed by a semicolon. *Query* waits silently for a semicolon or a zero–length record before processing a command. A zero–length record is entered as a solitary carriage return. This method of signaling the end of a command line enables you to enter commands which are several lines long. Line length is limited to 256 characters. At any point in a line, you may type a carriage return and continue the command line, thus improving the readability of long command lines.

Once initiated, *query* identifies itself and gives the prompt:

NEXT?

Whenever this prompt appears, you may enter any of the *query* commands, which are described below.

DATA–BASE=

The **data–base=** command opens a database. You can type:

data–base=data_base_name;

where *data_base_name* is the name of a database. If you are presently in the directory where the database exists, you need only give the database name. If the database is in another directory, you need to supply a partial or complete path name.

Some examples are:

data–base=/users/fred/inventory;

specifies a database called "inventory" in the directory /users/fred.

data–base=equipment;

specifies a database called "equipment" in the current directory.

As part of the **data–base=** command, *query* asks for a password for that database with the prompt:

PASSWORD?

The password is not echoed on the terminal as you type it. As usual, the password must be followed by a semicolon. If no password is required, simply press RETURN.

Provided the database name and password are valid, the database is opened with "modify shared" access (DBOPEN mode 1). The command prompt "NEXT?" appears.

HELP

The **help** command provides a syntax model, a brief description, and examples of itself or any other *query* command. It can be invoked in the following ways:

help [command];
or
? [command];

where *command* is any *query* command. If no command is supplied, **help** describes itself and gives a list of the commands for which help is available.

EXIT The **exit** command provides you with a graceful way to terminate the *query* program. It is entered thus:

exit;

Query can also be terminated by hitting the BREAK key in response to a command prompt.

!

At any time, in response to a "NEXT?" prompt, you may wish to execute a shell command without leaving the *query* program. This is useful when debugging report procedure files from within *query*, or routing output files to a printer during a *query* session. For example:

!pr filename | lpr

runs the formatter/printer *pr* on a file called *filename*, and pipes the output into the *lpr* program.

A shell command following an exclamation mark is executed, and *query* is suspended until that command is completed. *Query* processing can be continued when the "[Hit return to continue QUERY]" message appears.

LIST The **list** command is a convenient method of redirecting output from within the *query* program. *Query* sends output to stdout (the input device) by default. To send output elsewhere, type:

list=filename;

Output is sent to *filename* in the current directory. It may be sent to a file in another directory by specifying the desired pathname.

An example is:

list=/usr/spool/uucppublic/report;

which specifies that the output file is */usr/spool/uucppublic/report.*

If *query* finds that the named file already exists, it prints the message:

FILE ALREADY EXISTS. (O)VERWRITE IT, (A)PPEND TO IT,
OR (N)EITHER ?

You type **o**, **a**, or **n** to select an option. If you choose **n**, *query* prompts:

NEW FILE NAME=

in response to which you provide a (presumably) different file name. Otherwise, *query* overwrites or appends to the selected file, as instructed. Output directed to a file is properly formatted for direct submission to *lpr*(1). At any time during a *query* session, you may return output to the terminal by typing:

list;

This can be repeated as often as necessary, using the same file or many different files for output. When the **list** command appears in an XEQ file, no choices are offered. The specified file is silently opened and any output is appended to it.

FORM

The **form** command outputs a *schema* description for the open database. It lists each data set name, its type (automatic master, manual master, or detail), the set capacity, and the current number of entries in the set. With each data set, each item is listed including its name, type (alphanumeric, integer, or real), length in bytes, and number of elements in the item. *Query* also identifies key items, sort items (the sort item name may be truncated), and indicates whether you have write access for the item. It is initiated by typing:

form;

Form's output is directed to the file specified by the **list** command, or stdout by default. Its output can be terminated by hitting the BREAK key. Within a few seconds, output stops and a command prompt (NEXT?) appears.

FIND

A major use of any *query* program is to search a database for an arbitrary group of entries meeting some criteria. **Find** is used in conjunction with the **update** or **report** commands, providing "victims" for the update or report. It is entered by typing:

find retrieve_procedure end;

where *retrieve_procedure* is a group of data item names, data item values, and relational operators joined together by logical connectors.

A retrieve procedure defines a relationship between a data item and a data item value, and the **find** command collects entries which satisfy that relationship for later use by **update** or **report**. A typical retrieve procedure looks like this:

[setname.]itemname operator "value"

where *setname* is the name of a data set which contains the data item. It is always accepted, but is not necessary when the item name exists in only one data set, or when the set name has been previously established in the retrieve procedure. *Itemname* is simply the name of a data item. For compound items, only the first element is used. *Operator* is one of the following relational operators:

is, **ie**	equal to
isnot, **ine**	not equal to
ilt	less than
inlt	not less than
igt	greater than
ingt	not greater than

Value is enclosed in quotation marks (" ") and is compared to the value of the named item for each entry in the specified (or implied) set. It should be appropriate for the data item type.

Two or more retrieve procedures can be joined by the logical operators **and** and **or** to make a more complex procedure. Parentheses are not allowed in **find** procedures, so care should be taken in ordering statements in a compound retrieve procedure.

Some examples are:

find inventory.quantity is "324" end;

searches all entries in the "inventory" set for a "quantity" equal to "324". This would be appropriate for a quantity of any type (alphanumeric, integer, or real).

find part_description ie "widgit" end;

searches all entries in the set which contains the item "part_description" for a value of "widgit". The value would obviously be inappropriate for an item type of integer or real. This example generates an error if "part_description" exists in more than one set in the database.

find inventory.quantity igt "324" and part_description isnot "widgit" end;

searches all entries in the set "inventory", collecting those that show a quantity greater than 324, excluding widgits. The items "quantity" and "part_description" must both be items contained in the set "inventory".

XEQ The **xeq** command allows any number of commands to be read from a file created by any of the HP-UX editors. Commands must appear in the file exactly as they would be entered interactively, one command per line. The **xeq** command is entered:

xeq=filename;

The *filename* may be an absolute pathname, if necessary. *Query* reads commands from that file until it encounters either an end-of-file or another **xeq** command. When end-of-file is reached, *query* returns to an interactive state. When an **xeq** command is encountered within an **xeq** file, *query* closes the current **xeq** file and begins reading commands from the new one. The old one is not re-opened. **Xeq** files can be nested up to 10 deep.

A few commands behave differently when they occur in an **xeq** file. The "**list=**file" command silently opens the specified file and appends data to it. The update mode of the **update r** command can be terminated only by a lone semicolon. (In interactive use, **update r** can be terminated by a semicolon or a zero-length record.)

UPDATE A

Update a (**a**dd) adds entries to a data set. It is the only update which does not require a preceding **find**. The **update a**dd command is entered:

update a, setname;
or
update add, setname;

Query checks the validity of the set name, and then prompts for item values one at a time. The item name is displayed followed by an "=". The value to be assigned to that item should then be entered, enclosed in quotes, and followed by a semicolon. *Query* then prompts for the next item value. Only one prompt is given for compound data items; the item values should be entered each in quotes, separated by commas. Null values may be entered by a lone carriage return in response to a prompt, but *query* insists on valid values for key items. Addition of detail entries requires that values for key items already exist in the corresponding master set(s).

The BREAK key can be used to abort an **update a** command. No update takes place, and a command prompt appears.

UPDATE D

Update d (**d**elete) deletes entries from a data set. It requires a preceding **find**, and complains if not satisfied. The command is entered:

update d;
or

```
update delete;
```

As a safety check, *query* asks "OK TO DELETE?(YES/NO)". Upon receipt of a **y** or **n**, *query* proceeds as directed. It refuses to delete master set entries which contain chain heads with non–empty chains (i.e. connected detail sets), and displays a message to that effect.

UPDATE R

Update r (**r**eplace) also must be preceded by a **find**. It is a means of changing item values in an existing entry. It is invoked by typing:

```
update r;
   or
update replace;
```

Query responds with the prompt "ITEM =". You then enter:

```
item_name="value";
```

where *item_name* is an item name which exists in the entries in the select file. *Value* is a value appropriate to the item type (alpha, integer, or real) enclosed in double quotes.

When you have finished entering the changes desired, a lone carriage return or a lone semi–colon exits this update mode, and *query* executes the changes and returns to the command level. (In an **xeq** file, only the semicolon suffices.) The new value(s) are inserted into all entries collected in the select file. Updates are refused for key items and sort items in master sets. Updating key or sort items in detail sets causes that record to be deleted and re–entered with the new values. A report following such an update may give an "EMPTY RECORD" error message. Don't panic. The record may be found at its new location by a **find** command.

The BREAK key can be used to abort an **update r** command. No update takes place, and a command prompt appears.

REPORT

The **report** command provides many features to display information about the entries in the select file. The information is sent to the list device (the input device, by default). **Report**'s output can be terminated by using the BREAK key, which yields a command prompt (NEXT?). You can request the name and value of each data item for all the data entries specified in the select file, or request the data item values for all of the data entries without printing the data item name. Also you may create output formats complete with page headings, page numbers, column headings, space and page control, and selectivity in item value display. **Report** can be invoked in one of three ways:

```
report all [,character];
   or
report name=procedure_name;
   or
report;
body
end;
```

where *character* is any ASCII printing character which determines the printing of certain optional information.

Procedure_name is the name of a file (specified as a relative or absolute pathname) which contains **report** commands stored via a system editor, such as *ed* or *vi*.

Body consists of header, detail, edit, and sort commands as outlined below.

The three forms of the report command are described below.

REPORT ALL [,CHARACTER];
prints the entire data item and all elements of a compound item. This report form prints the item name, followed by "=", followed by the item value. The optional *character* causes *query* to print only the item value, without the item name and "=". All item values appear left justified, and numbers are stripped of insignificant zeros. Real numbers may appear in decimal form or scientific notation, as necessary. *This is the only report form which shows all values for compound data items.*

REPORT NAME=PROCEDURE_NAME;
gets header, detail, edit, and sort commands from a file, reading commands until an "end;" or an error is encountered. The contents of a procedure file are identical to the "body" in the next form of the **report** command. It should be noted that the use of the shell escape (!) is a valuable aid in the development of procedure files. It enables you to invoke an editor, modify a file, exit it, and return to the same point in *query* to test the file, without having to re–define the database or re–establish a select file.

REPORT;
BODY
END; accepts **report** commands from the user, scanning each line as it is entered for syntax errors. The entry of an "end;" command initiates the execution of the commands, producing a report. The body is a collection of the following commands:

Header	Prints title, column headings, and page numbers at the top of each report page.
Detail	Prints data item values in the column position specified.
Edit	Describes the number of decimal places to be displayed for real numbers.
Sort	Sorts data entries based upon the value of a specified data item.

Report Formatting

The above commands can be formatted using the following parameters. (Note: these are *parameters* to the **report** commands, not commands themselves.)

print position
Specifies the ending column for an item value or heading.

space and space control
Causes line skips between item values or heading lines.

skip and skip control
Causes page skips between item values or heading lines.

edit
Specifies edit commands to which output item values should conform.

These parameters are described below.

print position
This parameter is an integer between 1 and 132 which indicates the column number in which the last character of an item value should appear in a header or detail line. It is your responsibility to avoid overlap between fields on the same line. However, in most cases *query* replaces an overlapping value with asterisks to indicate an error.

space and space control
This parameter outputs blank lines either before or after the printing of a header string or detail line value. The keyword **space** should be followed by either an **a** or **b**, indicating where the blank line should appear - **a**fter or **b**efore the line to be printed. The **a** or **b** may be followed by an integer in the range 1 - 5, to skip

multiple lines. Absence of the integer causes *query* to skip 1 line. These may appear more than once in a command, as in spacing before and after a line:

```
h1,"page",35,space b2,space a3;
```

This generates two blank lines before printing "page", and three blank lines afterward.

skip and skip control

Similar to **space**, **skip** yields page feeds either before or after the printing of a line. Unlike **space**, **skip** can only be used with a "detail" command. The keyword **skip** is followed by an **a** or **b** to direct where the page feed should be placed (see "space and space control" above). Normally, *query* prints 54 lines per page before skipping to a new page.

edit

The **edit** parameter is the letter **e** followed by an integer in the range 0 - 9. This number corresponds to a numbered edit command which specifies the number of decimal places (for real numbers) or the number of characters (for alphanumeric strings) to be printed.

Report Commands

h (**h**eader command)

The **header** command is used to print heading information of your choice at the top of each page of the report. A maximum of five lines of header information can appear at the top of each report page. The format of the header command is:

```
hnumber,data_type,print_position [,space space-control];
```

where *number* is an integer from 1 to 5 specifying on which header line (out of five possible lines) the information is to appear. Header information in a header command labeled "h1" appears in the first line, "h2" appears in the second line, etc.

Data_type is either an ASCII character string enclosed in double quotes, or the word **pageno** (without quotes). If **pageno** appears in the header command, *query* prints the page number of the report in the position specified by *print_position*. *Query* increments the page number automatically for each page printed.

Print_position, *space*, and *space-control* are parameters defined in the section on report formatting.

An example is:

```
h1,"PAGE",70,space b2;
h1,pageno,76;
h2,"DAILY REPORT",50,space a3;
```

which prints the word "PAGE" with the letter "E" in column 70, on the second line from the top of a page (via the "space b2" parameter). On the same line, the page number is printed ending in column 76. The next line contains "DAILY REPORT" ending in column 50, followed by three blank lines.

d (**d**etail command)

The **detail** command indicates which data items of a data entry specified in the select file are to be printed in the report. Data items can be printed on up to 10 different lines. *Query* prints only the values of data items which appear in a detail command.

If an ASCII value length exceeds the distance between a preceding value on the same line or the left margin, it is silently truncated on the left. If a numeric value overlaps in the same manner, it is replaced by a series of asterisks, indicating the error.

Detail commands without an edit parameter print numeric values in whatever format necessary to give maximum accuracy.

The format of a detail command is:

d[n],data_type,print_position[,space space-control][,skip skip-control][,edit];

where *n* is an integer from 1 to 9. Each number specifies a different line on which the data items are printed. If the number is omitted, the unnumbered detail item is printed on a separate line above any numbered detail item lines. The lowest numbered command is printed first and all others follow in numeric order. Detail commands with the same number are printed on the same line.

Data_type is either an ASCII character string enclosed in double quotes, or the name of a data item contained in the data entries specified in the select file. NOTE: the **report** command processor expects the data item name *by itself.* Preceding the item name with a set name generates an error.

Print_position, *space*, *space-control*, *skip*, *skip-control*, and *edit* are parameters defined in the section on report formatting.

e (**e**dit command)

The **edit** command is used to format the printing of real and/or alphanumeric item values. Up to ten edit commands, labeled from e0 to e9, can be used in a report. To edit output from a detail command, you include the label of the desired edit command. *Query* refers to the labeled edit command to edit the value printed by the detail command. The same edit command can be referenced by more than one detail command in the same report, and each edit command must be referenced at least once in the report body. The format of the edit command is:

enumber,"places,format";

where *number* is an integer from 0 to 9, identifying the edit command. An integer cannot be used to identify more than one edit command.

Places is an integer indicating the number of digits to follow the decimal point (for real numbers) or the number of characters to be printed (for alphanumeric strings).

Format is one of the following single letters:

f indicating that the number should be formatted in decimal form, with the specified number of digits following the decimal point. Numbers accurate to more than the specified number of places are rounded.

e indicating that the number should be formatted in scientific notation with the specified number of digits following the decimal point.

s indicating that the data item is an alphanumeric string. The number of characters printed is specified by the accompanying integer. If you specify 10 characters for a data item 40 characters in length, the leftmost 10 characters are printed. If you specify 100 characters for the same data item, only 40 characters are printed.

Here are some examples:

```
e1,"6f";
d1,real_number,40,e1;
```

might yield such numbers as:

```
2.340000
25487.123456
```

```
1.000000
```

and

```
e1,"4e";
d1,realnum,40,e1;
```

might yield

```
2.3400e+0
3.2549e+5
1.0000e+0
```

and

```
e1,"15s";
d1,String,40,e1;
```

might yield

```
Smith, Jame
truncate ri
Walker, Mau
Doe, John
```

Finally, the difference between the edit *command* and the edit *parameter* should be emphasized. For example,

```
e1,"7f";
d1,Any_real,30,e1;
end;
```

The first line is an edit command, specifying a format for real numbers. In the second line, the "e1" is a parameter, indicating that the real number(s) "Any_real" should be printed according to the format shown in the "e1" command.

s (sort command)

The **sort** command specifies an item upon which you want the entries in the select file sorted. The format of the sort command is:

```
s,itemname;
```

where *itemname* is the name of a data item which appears in entries currently stored in the select file.

A sort item value may not exceed 80 bytes in length. In the case of a compound data item, sort uses only the first value in that item. After a "find", the entries appear in the select file in the order the find command encounters them. The sort command will rearrange those entries in ascending alphabetic or numeric order, depending on the sort item.

Report Example

Assuming that *Emp_name* is a 20–byte alphanumeric item, *Emp_age* is a 2–byte integer, and *Emp_wage* is a 4–byte real:

```
H1,"EMPLOYEE REPORT",34,space b3;
h1,"PAGE",52;
h1,pageno,56,space a2;
h2,"NAME",7;
H2,"AGE",27;
H2,"HOURLY WAGE",52;
e1,"2f";
s,Emp_name;
d1,Emp_name,20;
```

```
d1,Emp_age,27;
d1,"$",44;
d1,Emp_wage,50,e1,space a;
end;
```

This report might yield:

EMPLOYEE REPORT PAGE 1

NAME	AGE	HOURLY WAGE
Anderson,Richard	32	$ 14.75
Carr,Elaine	21	$ 11.50
Wilson,Kathy	42	$ 17.25

.
.
.

Summary

Although the commands appear throughout this document in lower–case, *query* accepts them in upper–case also. This is helpful when working with databases ported from Series 500 BASIC, in which database names, data set names, and data item names are frequently in upper–case.

It should be stressed that all commands must end with a semicolon or zero–length entry. If *query* seems to have "gone away", be sure that a semicolon followed tha last command entered. If this is not the case, an extra carriage return serves to terminate the command and prompt *query* into action.

Query sometimes appears to be "eating" report commands and doing nothing about them, other than supplying the "NEXT?" prompt. This is usually the result of having used the "list" command to re–route output earlier in the session, and having forgotten about it. Typing

```
list;
```

re–routes output to the terminal again.

Abnormal termination of *query* leaves files in */usr/tmp*. It is your responsibility to remove these files or they may accumulate and use up large amounts of memory. The files can be identified by the owner id, shown by typing "ll /usr/tmp". Do not attempt to remove files belonging to anyone else.

FILES

/usr/lbin/query	
/usr/lbin/querysort	*query*'s own sort routine
/usr/lib/query.help	help file
/usr/tmp/*	temporary files

NAME

ratfor - rational Fortran dialect

SYNOPSIS

ratfor [options] [files]

HP–UX COMPATIBILITY

Level: HP–UX/STANDARD

Origin: System V

Remarks: Not supported on the Integral Personal Computer.

DESCRIPTION

Ratfor converts a rational dialect of Fortran into ordinary irrational Fortran. *Ratfor* provides control flow constructs essentially identical to those in C:

statement grouping:

```
{ statement; statement; statement }
```

decision–making:

```
if (condition) statement [ else statement ]
switch (integer value) {
        case integer:    statement
        ...
        [ default: ]     statement
}
```

loops:

```
while (condition) statement
for (expression; condition; expression) statement
do limits statement
repeat statement [ until (condition) ]
break
next
```

and some syntactic sugar to make programs easier to read and write:

free form input:

multiple statements/line; automatic continuation

comments:

```
# this is a comment.
```

compiler directives:

directives beginning with a dollar sign (**$**) in column one are passed through to the compiler unchanged.

translation of relationals:

>, >=, etc., become **.GT.**, **.GE.**, etc.

return expression to caller from function:

return (expression)

define:

define *name replacement*

include:

include *file*

The *options* are as follows:

-h causes quoted strings to be turned into Hollerith constructs.

-C copies comments to the output and attempts to format it neatly.

-6*c* Normally, continuation lines are marked with an **&** in column 1; the option **-6***c* makes the continuation character *c* and places it in column 6.

Ratfor is best used with *fc*(1).

-d compatibility mode for earlier versions of *ratfor*.

HARDWARE DEPENDENCIES

Series 500:

Fc(1) does not recognize **ratfor.r** files. Therefore, *ratfor* must be called directly.

The **-h** option should not be used.

The **-6x** option must be used for successful automatic continuation.

SEE ALSO

efl(1), fc(1).

B. W. Kernighan and P. J. Plauger, *Software Tools*, Addison–Wesley, 1976.

NAME

rev - reverse lines of a file

SYNOPSIS

rev [file] ...

HP–UX COMPATIBILITY

Level: HP–UX/STANDARD

Origin: UCB

Remarks: Not supported on the Integral Personal Computer.

DESCRIPTION

Rev copies the named files to the standard output, reversing the order of characters in every line. If no file is specified, the standard input is copied.

NAME

revision - get HP–UX revision information

SYNOPSIS

/usr/bin/revision

HP–UX COMPATIBILITY

Level: HP–UX/NON–STANDARD

Origin: HP

Remarks: *Revision* is implemented on the Series 500 only.

DESCRIPTION

This command prints six lines to standard output. Those six lines consist of the six data items output by *uname*(2), which give information on the kernel.

The following is a sample output from a machine whose loader chip was not programmed with a serial number:

```
System:     HP–UX
Release:    TEST23#
Version:    B
Machine:
Identity:   HP–UX
Nodename:   hpfcla
```

SEE ALSO

uname(2).

NAME

rm, rmdir - remove files or directories

SYNOPSIS

rm [**-fri**] file ...

rmdir dir ...

HP–UX COMPATIBILITY

Level: HP–UX/STANDARD

Origin: System V

DESCRIPTION

Rm removes the entries for one or more files from a directory. If an entry was the last link to the file, the file is destroyed. Removal of a file requires write permission in its directory, but neither read nor write permission on the file itself.

If a file has no write permission and the standard input is a terminal, its permissions are printed and a line is read from the standard input. If that line begins with **y** the file is deleted, otherwise the file remains. No questions are asked when the **-f** option is given or if the standard input is not a terminal.

If a designated file is a directory, an error comment is printed unless the optional argument **-r** has been used. In that case, *rm* recursively deletes the entire contents of the specified directory, and the directory itself. (Note that *rm* can recursively remove a maximum of 17 directory levels.)

If the **-i** (interactive) option is in effect, *rm* asks whether to delete each file, and, under **-r**, whether to examine each directory.

Rmdir removes entries for the named directories, which must be empty and have execute permission for the user trying to remove them.

SEE ALSO

unlink(2).

DIAGNOSTICS

Generally self–explanatory. It is forbidden to remove the file **..** merely to avoid the consequences of inadvertently doing something like:

```
rm -r .*
```

NAME

rmdel - remove a delta from an SCCS file

SYNOPSIS

rmdel -rSID files

HP-UX COMPATIBILITY

Level: HP-UX/STANDARD

Origin: System III

DESCRIPTION

Rmdel removes the delta specified by the *SID* from each named SCCS file. The delta to be removed must be the newest (most recent) delta in its branch in the delta chain of each named SCCS file. In addition, the SID specified must *not* be that of a version being edited for the purpose of making a delta (i. e., if a *p-file* (see *get*(1)) exists for the named SCCS file, the SID specified must *not* appear in any entry of the *p-file*).

If a directory is named, *rmdel* behaves as though each file in the directory were specified as a named file, except that non-SCCS files (last component of the path name does not begin with **s.**) and unreadable files are silently ignored. If a name of **-** is given, the standard input is read; each line of the standard input is taken to be the name of an SCCS file to be processed; non-SCCS files and unreadable files are silently ignored.

The exact permissions necessary to remove a delta are documented in the *Source Code Control System User's Guide*. Simply stated, they are either (1) if you make a delta you can remove it; or (2) if you own the file and directory you can remove a delta.

FILES

x.file (see *delta*(1))
z.file (see *delta*(1))

SEE ALSO

delta(1), get(1), help(1), prs(1), sccsfile(5).
Source Code Control System User's Guide in *HP-UX Concepts and Tutoials.*

DIAGNOSTICS

Use *help*(1) for explanations.

NAME

rmnl – remove extra new–line characters from file

SYNOPSIS

rmnl

HP–UX COMPATIBILITY

Level: HP–UX/STANDARD

Origin: UCB

DESCRIPTION

Rmnl is useful for removing excess white space from files for display on a crt terminal. Groups of more than one \n character are compressed to one \n character, effectively removing all blank lines. This is used by the *man* command.

Ssp(1) can be used to remove redundant blank lines, rather than all blank lines.

SEE ALSO

ssp(1), man(1).

NAME

rtprio - execute process with realtime priority

SYNOPSIS

rtprio priority command [arguments]
rtprio priority -pid

HP-UX COMPATIBILITY

Level: HP-UX/STANDARD

Origin: HP

DESCRIPTION

Rtprio executes *command* with a realtime priority, or changes the realtime priority of currently executing process *pid*. Realtime priorities range from one (highest) to 127 (lowest), a priority of zero indicates normal unix process scheduling which effectively makes the process non-realtime. Realtime processes are not subject to priority degradation and are all of greater (scheduling) importance than non-realtime processes. See *rtprio*(2) for more details.

Command will not be scheduled, or *pid*'s realtime priority will not be changed, if the user is not a member of a group having PRIV_RTPRIO access or is not the super-user. When changing the realtime priority of a currently executing process, the effective user ID of the calling process must be superuser, or the real or effective user ID must match the real or effective user ID of the process to be modified.

SEE ALSO

rtprio(2), getprivgrp(2)

DIAGNOSTICS

Rtprio returns exit status 0 if *command* is successfully scheduled or if *pid*'s realtime priority is successfully changed, 1 if *command* is not executable or *pid* does not exist, 2 if *command (pid)* lacks realtime capability, or the invoker's effective user ID is not superuser or effective user ID does not match the real or effective user ID of the process to be changed.

NAME

sact - print current SCCS file editing activity

SYNOPSIS

sact files

HP-UX COMPATIBILITY

Level: HP-UX/STANDARD

Origin: System III

DESCRIPTION

Sact informs the user of any impending deltas to a named SCCS file. This situation occurs when *get*(1) with the **-e** option has been previously executed without a subsequent execution of *delta*(1). If a directory is named on the command line, *sact* behaves as though each file in the directory were specified as a named file, except that non-SCCS files (last component of path name does not begin with **s.**) and unreadable files are silently ignored. If a name of **-** is given, the standard input is read with each line being taken as the name of an SCCS file to be processed.

The output for each named file consists of five fields separated by spaces.

Field 1	specifies the SID of a delta that currently exists in the SCCS file to which changes will be made to make the new delta.
Field 2	specifies the SID for the new delta to be created.
Field 3	contains the logname of the user who will make the delta (i.e., executed a *get* for editing).
Field 4	contains the date that **get -e** was executed.
Field 5	contains the time that **get -e** was executed.

SEE ALSO

delta(1), get(1), unget(1).

DIAGNOSTICS

Use *help*(1) for explanations.

NAME

sccsdiff - compare two versions of an SCCS file

SYNOPSIS

sccsdiff -rSID1 **-r**SID2 [**-p**] [**-s**n] files

HP-UX COMPATIBILITY

Level: HP-UX/STANDARD

Origin: System III

DESCRIPTION

Sccsdiff compares two versions of an SCCS file and generates the differences between the two versions. Any number of SCCS files may be specified, but arguments apply to all files.

-r*SID?* *SID1* and *SID2* specify the deltas of an SCCS file that are to be compared. Versions are passed to *bdiff*(1) in the order given. The SID's accepted, and the corresponding version retrieved for the comparison are the same as for *get*(1).

-p pipe output for each file through *pr*(1).

-s*n* *n* is the file segment size that *bdiff* will pass to *diff*(1). This is useful when *diff* fails due to a high system load.

FILES

/tmp/get????? Temporary files

SEE ALSO

bdiff(1), diff(1), get(1), help(1), pr(1).
Source Code Control System User's Guide in *HP-UX: Selected Articles.*

DIAGNOSTICS

"*file*: No differences" if the two versions are the same.
Use *help*(1) for explanations.

NAME

sed – stream text editor

SYNOPSIS

sed [**–n**] [**–e** script] [**–f** sfile] [files]

HP-UX COMPATIBILITY

Level: HP-UX/STANDARD

Origin: System V

DESCRIPTION

Sed copies the named *files* (standard input default) to the standard output, edited according to a script of commands. The **–f** option causes the script to be taken from file *sfile*; these options accumulate. If there is just one **–e** option and no **–f** options, the flag **–e** may be omitted. The **–n** option suppresses the default output. A script consists of editing commands, one per line, of the following form:

[address [, address]] function [arguments]

In normal operation, *sed* cyclically copies a line of input into a *pattern space* (unless there is something left after a **D** command), applies in sequence all commands whose *addresses* select that pattern space, and at the end of the script copies the pattern space to the standard output (except under **–n**) and deletes the pattern space.

Some of the commands use a *hold space* to save all or part of the *pattern space* for subsequent retrieval.

An *address* is either a decimal number that counts input lines cumulatively across files, a **$** that addresses the last line of input, or a context address, i.e., a */regular expression/* in the style of *ed*(1) modified thus:

In a context address, the construction *****?*regular expression*?, where *?* is any character, is identical to */regular expression/*. Note that in the context address **\xabc\xdefx**, the second **x** stands for itself, so that the regular expression is **abcxdef**.

The escape sequence **\n** matches a new-line *embedded* in the pattern space.

A period **.** matches any character except the *terminal* new-line of the pattern space.

A command line with no addresses selects every pattern space.

A command line with one address selects each pattern space that matches the address.

A command line with two addresses selects the inclusive range from the first pattern space that matches the first address through the next pattern space that matches the second. (If the second address is a number less than or equal to the line number first selected, only one line is selected.) Thereafter the process is repeated, looking again for the first address.

Editing commands can be applied only to non-selected pattern spaces by use of the negation function **!** (below).

In the following list of functions the maximum number of permissible addresses for each function is indicated in parentheses.

The *text* argument consists of one or more lines, all but the last of which end with **** to hide the new-line. Backslashes in text are treated like backslashes in the replacement string of an **s** command, and may be used to protect initial blanks and tabs against the stripping that is done on every script line. The *rfile* or *wfile* argument must terminate the command line and must be preceded by exactly one blank. Each *wfile* is created before processing begins. There can be at most

10 distinct *wfile* arguments.

(1) **a**
text Append. Place *text* on the output before reading the next input line.

(2) **b** *label* Branch to the **:** command bearing the *label*. If *label* is empty, branch to the end of the script.

(2) **c**
text Change. Delete the pattern space. With 0 or 1 address or at the end of a 2-address range, place *text* on the output. Start the next cycle. Ignores any subsequent operations related to the deleted space.

(2) **d** Delete the pattern space. Start the next cycle. Ignores any subsequent operations related to the deleted pattern space.

(2) **D** Delete the initial segment of the pattern space through the first new-line. Start the next cycle.

(2) **g** Replace the contents of the pattern space by the contents of the hold space.

(2) **G** Append the contents of the hold space to the pattern space.

(2) **h** Replace the contents of the hold space by the contents of the pattern space.

(2) **H** Append the contents of the pattern space to the hold space.

(1) **i**
text Insert. Place *text* on the standard output.

(2) **l** List the pattern space on the standard output in an unambiguous form. Non-printing characters are spelled in two-digit ASCII and long lines are folded.

(2) **n** Copy the pattern space to the standard output. Replace the pattern space with the next line of input.

(2) **N** Append the next line of input to the pattern space with an embedded new-line. (The current line number changes.)

(2) **p** Print. Copy the pattern space to the standard output.

(2) **P** Copy the initial segment of the pattern space through the first new-line to the standard output.

(1) **q** Quit. Branch to the end of the script. Do not start a new cycle.

(2) **r** *rfile* Read the contents of *rfile*. Place them on the output before reading the next input line.

(2) **s**/*regular expression*/*replacement*/*flags*
Substitute the *replacement* string for instances of the *regular expression* in the pattern space. Any character may be used instead of /. For a fuller description see *ed*(1). *Flags* is zero or more of:

- **n** n= 1 - 512. Substitute for just the n th occurrence of the *regular expression.*
- **g** Global. Substitute for all nonoverlapping instances of the *regular expression* rather than just the first one.
- **p** Print the pattern space if a replacement was made.
- **w** *wfile* Write. Append the pattern space to *wfile* if a replacement was made.

(2) **t** *label* Test. Branch to the **:** command bearing the *label* if any substitutions have been made since the most recent reading of an input line or execution of a **t**. If *label* is empty, branch to the end of the script.

(2) **w** *wfile* Write. Append the pattern space to *wfile*.

(2) **x** Exchange the contents of the pattern and hold spaces.

(2) **y**/*string1*/*string2*/
Transform. Replace all occurrences of characters in *string1* with the corresponding character in *string2*. The lengths of *string1* and *string2* must be equal.

(2)**!** *function*
Don't. Apply the *function* (or group, if *function* is {) only to lines *not* selected by the address(es).

(0) **:** *label* This command does nothing; it bears a *label* for **b** and **t** commands to branch to.

(1) **=** Place the current line number on the standard output as a line.

(2) **{** Execute the following commands through a matching **}** only when the pattern space is selected. The syntax is:

```
{ cmd1
cmd2
cmd3
.
.
.
}
```

(0) An empty command is ignored.

(0) **#** If a # appears as the first character on the first line of a script file, then that entire line is treated as a comment, with one exception. If the character after the # is an 'n', then the default output will be suppressed. The rest of the line after #n is also ignored. A script file must contain at least one non-comment line.

SEE ALSO

awk(1), ed(1), grep(1).

BUGS

There is a limit of 100 commands in the script.

NAME

sh, rsh - shell, the standard/restricted command programming language

SYNOPSIS

sh [**-acefhiknrstuvx**] [args]
rsh [**-acefhiknrstuvx**] [args]

HP-UX COMPATIBILITY

Level: HP-UX/NUCLEUS

Origin: System V

Native Language Support:
8-bit filenames.

DESCRIPTION

Sh is a command programming language that executes commands read from a terminal or a file. *Rsh* is a restricted version of the standard command interpreter *sh*; it is used to set up login names and execution environments whose capabilities are more controlled than those of the standard shell. See *Invocation* below for the meaning of arguments to the shell.

Definitions

A *blank* is a tab or a space. A *name* is a sequence of letters, digits, or underscores beginning with a letter or underscore. A *parameter* is a name, a digit, or any of the characters *, @, #, ?, -, $, and !.

Commands

A *simple-command* is a sequence of non-blank *words* separated by *blanks*. The first word specifies the name of the command to be executed. Except as specified below, the remaining words are passed as arguments to the invoked command. The command name is passed as argument 0 (see *exec*(2)). The *value* of a simple-command is its exit status if it terminates normally, or (octal) 200+*status* if it terminates abnormally (see *signal*(2) for a list of status values).

A *pipeline* is a sequence of one or more *commands* separated by **|** (or, for historical compatibility, by ^). The standard output of each command but the last is connected by a *pipe*(2) to the standard input of the next command. Each command is run as a separate process; the shell waits for the last command to terminate. The exit status of a pipeline is the exit status of the last command.

A *list* is a sequence of one or more pipelines separated by **;**, **&**, **&&**, or **||**, and optionally terminated by **;** or **&**. Of these four symbols, **;** and **&** have equal precedence, which is lower than that of **&&** and **||**. The symbols **&&** and **||** also have equal precedence. A semicolon (**;**) causes sequential execution of the preceding pipeline; an ampersand (**&**) causes asynchronous execution of the preceding pipeline (i.e., the shell does *not* wait for that pipeline to finish). The symbol **&&** (**||**) causes the *list* following it to be executed only if the preceding pipeline returns a zero (non-zero) exit status. An arbitrary number of new-lines may appear in a *list*, instead of semicolons, to delimit commands.

A *command* is either a simple-command or one of the following. Unless otherwise stated, the value returned by a command is that of the last simple-command executed in the command.

for *name* [**in** *word* ...] **do** *list* **done**
: Each time a **for** command is executed, *name* is set to the next *word* taken from the **in** *word* list. If **in** *word* ... is omitted, then the **for** command executes the **do** *list* once for each positional parameter that is set (see *Parameter Substitution* below). Execution ends when there are no more words in the list.

case *word* **in** [*pattern* [| *pattern*] ...) *list* **;;**] ... **esac**
: A **case** command executes the *list* associated with the first *pattern* that matches *word*. The form of the patterns is the same as that used for file-name generation (see *File Name Generation*) except that a slash, a leading dot, or a dot immediately following a slash

need not be matched explicitly.

if *list* **then** *list* [**elif** *list* **then** *list*] ... [**else** *list*] **fi**

The *list* following **if** is executed and, if it returns a zero exit status, the *list* following the first **then** is executed. Otherwise, the *list* following **elif** is executed and, if its value is zero, the *list* following the next **then** is executed. Failing that, the **else** *list* is executed. If no **else** *list* or **then** *list* is executed, then the **if** command returns a zero exit status.

while *list* **do** *list* **done**

A **while** command repeatedly executes the **while** *list* and, if the exit status of the last command in the list is zero, executes the **do** *list*; otherwise the loop terminates. If no commands in the **do** *list* are executed, then the **while** command returns a zero exit status; **until** may be used in place of **while** to negate the loop termination test.

(*list*)

Execute *list* in a sub–shell.

{*list*;}

list is simply executed.

name () {*list*;}

Define a function which is referenced by *name*. The body of the function is the *list* of commands between { and }. Execution of functions is described below (see *Execution*).

The following words are only recognized as the first word of a command and when not quoted:

if then else elif fi case esac for while until do done { }

Comments

A word beginning with # causes that word and all the following characters up to a new–line to be ignored.

Command Substitution

The standard output from a command enclosed in a pair of grave accents (` `) may be used as part or all of a word; trailing new–lines are removed.

Parameter Substitution

The character **$** is used to introduce substitutable *parameters*. There are two types of parameters, positional and keyword. If *parameter* is a digit, it is a positional parameter. Positional parameters may be assigned values by **set**. Keyword parameters (also known as variables) may be assigned values by writing:

name=*value* [*name*=*value*] ...

Pattern–matching is not performed on *value*. There cannot be a function and a variable with the same *name*.

${*parameter***}**

The value, if any, of the parameter is substituted. The braces are required only when *parameter* is followed by a letter, digit, or underscore that is not to be interpreted as part of its name. If *parameter* is * or @, all the positional parameters, starting with **$1**, are substituted (separated by spaces). Parameter **$0** is set from argument zero when the shell is invoked.

${*parameter***:-***word***}**

If *parameter* is set and is non–null, substitute its value; otherwise substitute *word*.

${*parameter***:=***word***}**

If *parameter* is not set or is null set it to *word*; the value of the parameter is then substituted. Positional parameters may not be assigned to in this way.

${*parameter***:?***word***}**

If *parameter* is set and is non–null, substitute its value; otherwise, print *word* and exit from the shell. If *word* is omitted, then the message "parameter null or not set" is printed.

${*parameter***:+***word***}**
If *parameter* is set and is non–null then substitute *word*; otherwise substitute nothing.

In the above, *word* is not evaluated unless it is to be used as the substituted string, so that, in the following example, **pwd** is executed only if **d** is not set or is null:

```
echo ${d:-`pwd`}
```

If the colon (**:**) is omitted from the above expressions, then the shell only checks whether *parameter* is set or not.

The following parameters are automatically set by the shell:

#	The number of positional parameters in decimal.
-	Flags supplied to the shell on invocation or by the **set** command.
?	The decimal value returned by the last synchronously executed command.
$	The process number of this shell.
!	The process number of the last background command invoked.

The following parameters are used by the shell:

HOME The default argument (home directory) for the *cd* command.

PATH The search path for commands (see *Execution* below). The user may not change **PATH** if executing under *rsh*.

CDPATH
The search path for the *cd* command.

MAIL If this parameter is set to the name of a mail file *and* the **MAILPATH** parameter is not set, the shell informs the user of the arrival of mail in the specified file.

MAILCHECK
This parameter specifies how often (in seconds) the shell will check for the arrival of mail in the files specified by the **MAILPATH** or **MAIL** parameters. The default value is 600 seconds (10 minutes). If set to 0, the shell will check before each prompt.

MAILPATH
A colon (**:**) separated list of file names. If this parameter is set, the shell informs the user of the arrival of mail in any of the specified files. Each file name can be followed by **%** and a message that will be printed when the modification time changes. The default message is *you have mail*.

PS1 Primary prompt string, by default "**$** ".

PS2 Secondary prompt string, by default "> ".

IFS Internal field separators, normally **space**, **tab**, and **new–line**.

SHACCT
If this parameter is set to the name of a file writable by the user, the shell will write an accounting record in the file for each shell procedure executed. Accounting routines such as *acctcom*(1) and *acctcms*(1M) can be used to analyze the data collected.

SHELL
When the shell is invoked, it scans the environment (see *Environment* below) for this name. If it is found and there is an 'r' in the file name part of its value, the shell becomes a restricted shell. **SHELL** is also used by some to run.

The shell gives default values to **PATH**, **PS1**, **PS2**, **MAILCHECK** and **IFS**. **HOME** and **MAIL** are set by *login*(1).

Blank Interpretation

After parameter and command substitution, the results of substitution are scanned for internal field separator characters (those found in **IFS**) and split into distinct arguments where such characters are found. Explicit null arguments ("" or '') are retained. Implicit null arguments (those resulting from *parameters* that have no values) are removed.

File Name Generation

Following substitution, each command *word* is scanned for the characters *****, **?**, and **[**. If one of these characters appears then the word is regarded as a *pattern*. The word is replaced with alphabetically sorted file names that match the pattern. If no file name is found that matches the pattern, then the word is left unchanged. The character **.** at the start of a file name or immediately following a **/**, as well as the character **/** itself, must be matched explicitly.

*****	Matches any string, including the null string.
?	Matches any single character.
[...]	Matches any one of the enclosed characters. A pair of characters separated by **-** matches any character lexically between the pair, inclusive. If the first character following the opening ``[´´ is a **"!"** any character not enclosed is matched.

Quoting

The following characters have a special meaning to the shell and cause termination of a word unless quoted:

; & ( ) | ^ < > new–line space tab

A character may be *quoted* (i.e., made to stand for itself) by preceding it with a ****. The pair **\new–line** is ignored. All characters enclosed between a pair of single quote marks (´ ´), except a single quote, are quoted. Inside double quote marks (˝˝), parameter and command substitution occurs and **** quotes the characters ****, **`**, **˝**, and **$**. ˝**$***˝ is equivalent to ˝**$1 $2** ...˝, whereas ˝**$@**˝ is equivalent to ˝**$1**˝ ˝**$2**˝ ....

Prompting

When used interactively, the shell prompts with the value of **PS1** before reading a command. If at any time a new–line is typed and further input is needed to complete a command, then the secondary prompt (i.e., the value of **PS2**) is issued.

Input/Output

Before a command is executed, its input and output may be redirected using a special notation interpreted by the shell. The following may appear anywhere in a simple–command or may precede or follow a *command* and are *not* passed on to the invoked command; substitution occurs before *word* or *digit* is used:

<word	Use file *word* as standard input (file descriptor 0).
>word	Use file *word* as standard output (file descriptor 1). If the file does not exist then it is created; otherwise, it is truncated to zero length.
≫word	Use file *word* as standard output. If the file exists then output is appended to it (by first seeking to the end–of–file); otherwise, the file is created.
≪[-]word	The shell input is read up to a line that is the same as *word*, or to an end–of–file. The resulting document becomes the standard input. If any character of *word* is quoted, then no interpretation is placed upon the characters of the document; otherwise, parameter and command substitution occurs, (unescaped) **\new–line** is ignored, and **** must be used to quote the characters ****, **$**, **`**, and the first character of *word*. If **-** is appended to **≪**, then all leading tabs are stripped from *word* and from the document.
<&digit	Use the file associated with file descriptor *digit* as standard input. Similarly for the standard output using **>&digit**. (See *dup*(2)).
<&-	The standard input is closed. Similarly for the standard output using **>&-**.

If any of the above is preceded by a digit, then the file descriptor which will be associated with the file is that specified by the digit (instead of the default 0 or 1). For example:

... 2>&1

associates file descriptor 2 with the file currently associated with file descriptor 1. Note that this type of I/O redirection is necessary if you want to *synchronously* collect stdout and stderr output in the same file. Redirecting stdout and stderr separately will cause asynchronous collection of

data at the destination (i.e. things written to stdout can subsequently be over–written by things written to stderr, and vice–versa).

The order in which redirections are specified is significant. The shell evaluates redirections left–to–right. For example:

... 1>*xxx* 2>&1

first associates file descriptor 1 with file *xxx*. It associates file descriptor 2 with the file associated with file descriptor 1 (i.e. *xxx*). If the order of redirections were reversed, file descriptor 2 would be associated with the terminal (assuming file descriptor 1 had been) and file descriptor 1 would be associated with file *xxx*.

If a command is followed by **&** then the default standard input for the command is the empty file **/dev/null**. Otherwise, the environment for the execution of a command contains the file descriptors of the invoking shell as modified by input/output specifications.

Redirection of output is not allowed in the restricted shell.

Environment

The *environment* (see *environ*(7)) is a list of name–value pairs that is passed to an executed program in the same way as a normal argument list. The shell interacts with the environment in several ways. On invocation, the shell scans the environment and creates a parameter for each name found, giving it the corresponding value. Executed commands inherit the same environment. If the user modifies the value of any of these parameters or creates new parameters, none of these affects the environment unless the **export** command is used to bind the shell's parameter to the environment (see also **set –a**). A parameter may be removed from the environment with the **unset**command. The environment seen by any executed command is thus composed of any unmodified name–value pairs originally inherited by the shell, minus any pairs removed by **unset**, plus any modifications or additions, all of which must be noted in **export** commands.

The environment for any *simple–command* may be augmented by prefixing it with one or more assignments to parameters. Thus:

```
TERM=450 cmd                          and
(export TERM; TERM=450; cmd)
```

are equivalent (as far as the execution of *cmd* is concerned).

If the **-k** flag is set, *all* keyword arguments are placed in the environment, even if they occur after the command name. The following first prints **a=b c** and then **c**:

```
echo a=b c
set -k
echo a=b c
```

Signals

The INTERRUPT and QUIT signals for an invoked command are ignored if the command is followed by **&**; otherwise signals have the values inherited by the shell from its parent, with the exception of signal 11 (but see also the **trap** command below).

Execution

Each time a command is executed, the above substitutions are carried out. If the command name matches one of the *Special Commands* listed below, it is executed in the shell process. If the command name does not match a *Special Command*, but matches the name of a defined function, the function is executed in the shell process (note how this differs from the execution of shell procedures). The positional parameters **$1**, **$2**, ... are set to the arguments of the function. If the command name matches neither a *Special Command* nor the name of a defined function, a new process is created and an attempt is made to execute the command via *exec*(2).

The shell parameter **PATH** defines the search path for the directory containing the command. Alternative directory names are separated by a colon (**:**). The default path is **:/bin:/usr/bin**

(specifying the current directory, **/bin**, and **/usr/bin**, in that order). Note that the current directory is specified by a null path name, which can appear immediately after the equal sign or between the colon delimiters anywhere else in the path list. If the command name contains a / then the search path is not used; such commands will not be executed by the restricted shell. Otherwise, each directory in the path is searched for an executable file. If the file has execute permission but is not an **a.out** file, it is assumed to be a file containing shell commands. A sub–shell (i.e., a separate process) is spawned to read it. A parenthesized command is also executed in a sub–shell.

The location in the search path where a command was found is remembered by the shell (to help avoid unnecessary *execs* later). If the command was found in a relative directory, its location must be re–determined whenever the current directory changes. The shell forgets all remembered locations whenever the **PATH** variable is changed or the **hash −r** command is executed (see below).

Special Commands

The following commands are executed in the shell process. Input/output redirection is permitted for these commands. File descriptor 1 is the default output location.

: No effect; the command does nothing. A zero exit code is returned.

. *file* Read and execute commands from *file* and return. The search path specified by **PATH** is used to find the directory containing *file*. Note that this command does not spawn another shell to execute *file*, and thus differs in behavior and output from executing *file* as a shell script.

break [*n*]
Exit from the enclosing **for** or **while** loop, if any. If *n* is specified then break *n* levels.

continue [*n*]
Resume the next iteration of the enclosing **for** or **while** loop. If *n* is specified then resume at the *n*–th enclosing loop.

cd [*arg*]
Change the current directory to *arg*. The shell parameter **HOME** is the default *arg*. The shell parameter **CDPATH** defines the search path for the directory containing *arg*. Alternative directory names are separated by a colon (:). The default path is **<null>** (specifying the current directory). Note that the current directory is specified by a null path name, which can appear immediately after the equal sign or between the colon delimiters anywhere else in the path list. If *arg* begins with a / the search path is not used. Otherwise, each directory in the path is searched for *arg*. The *cd* command may not be executed by *rsh*.

echo [*arg* ...]
Echo arguments. See *echo*(1) for usage and description.

eval [*arg* ...]
The arguments are read as input to the shell and the resulting command(s) executed.

exec [*arg* ...]
The command specified by the arguments is executed in place of this shell without creating a new process. Input/output arguments may appear and, if no other arguments are given, cause the shell input/output to be modified.

exit [*n*]
Causes a shell to exit with the exit status specified by *n*. If *n* is omitted then the exit status is that of the last command executed (an end–of–file will also cause the shell to exit.)

export [*name* ...]
The given *name*s are marked for automatic export to the *environment* of subsequently–executed commands. If no arguments are given, then a list of all names that are exported in this shell is printed. Function names may *not* be exported.

hash [**-r**] [*name* ...]
For each *name*, the location in the search path of the command specified by *name* is determined and remembered by the shell. The **–r** option causes the shell to forget all remembered locations. If no arguments are given, information about remembered commands is presented. *Hits* is the number of times a command has been invoked by the shell process. *Cost* is a measure of the work required to locate a command in the search path. There are certain situations which require that the stored location of a command be recalculated. Commands for which this will be done are indicated by an asterisk (*****) adjacent to the *hits* information. *Cost* will be incremented when the recalculation is done.

newgrp [*arg* ...]
Equivalent to **exec newgrp** *arg* .... See *newgrp*(1) for usage and description.

pwd Print the current working directory. See *pwd*(1) for usage and description.

read [*name* ...]
One line is read from the standard input and the first word is assigned to the first *name*, the second word to the second *name*, etc., with leftover words assigned to the last *name*. The return code is 0 unless an end–of–file is encountered.

readonly [*name* ...]
The given *name*s are marked *readonly* and the values of the these *name*s may not be changed by subsequent assignment. If no arguments are given, then a list of all *readonly* names is printed.

return [*n*]
Causes a function to exit with the return value specified by *n*. If *n* is omitted, the return status is that of the last command executed.

set [**--aefhkntuvx** [*arg* ...]]
- **-a** Mark variables which are modified or created for export.
- **-e** Exit immediately if a command exits with a non–zero exit status.
- **-f** Disable file name generation
- **-h** Locate and remember function commands as functions are defined (function commands are normally located when the function is executed).
- **-k** All keyword arguments are placed in the environment for a command, not just those that precede the command name.
- **-n** Read commands but do not execute them.
- **-t** Exit after reading and executing one command.
- **-u** Treat unset variables as an error when substituting.
- **-v** Print shell input lines as they are read.
- **-x** Print commands and their arguments as they are executed.
- **--** Do not change any of the flags; useful in setting **$1** to **-**.

Using **+** rather than **-** causes these flags to be turned off. These flags can also be used upon invocation of the shell. The current set of flags may be found in **$-**. The remaining arguments are positional parameters and are assigned, in order, to **$1**, **$2**, ... If no arguments are given then the values of all names are printed.

shift [*n*]
The positional parameters from **$n+1** ... are renamed **$1** .... If *n* is not given, it is assumed to be 1.

test
Evaluate conditional expressions. See *test*(1) for usage and description. Note that "[...]" in an **if** *list* is interpreted the same as "**test** ...". There must be blanks around the brackets.

times
Print the accumulated user and system times for processes run from the shell.

trap [*arg*] [*n*] ...
The command *arg* is a command to be read and executed when the shell receives signal(s) *n*. (Note that *arg* is scanned once when the trap is set and once when the trap is taken.) Trap commands are executed in order of signal number. Any attempt to set a trap on a

signal that was ignored on entry to the current shell is ineffective. An attempt to trap on signal 11 (memory fault) produces an error. If *arg* is absent then all trap(s) *n* are reset to their original values. If *arg* is the null string then this signal is ignored by the shell and by the commands it invokes. If *n* is 0 then the command *arg* is executed on exit from the shell. The **trap** command with no arguments prints a list of commands associated with each signal number.

type [*name* ...]
For each *name*, indicate how it would be interpreted if used as a command name.

ulimit [**-fp**] [*n*]
imposes a size limit of *n*
-f imposes a size limit of *n* blocks on files written by child processes (files of any size may be read). With no argument, the current limit is printed.
If no option is given, **-f** is assumed.

umask [*nnn*]
The user file–creation mask is set to *nnn* (see *umask*(2)). If *nnn* is omitted, the current value of the mask is printed.

unset [*name* ...]
For each *name*, remove the corresponding variable or function. The variables **PATH**, **PS1**, **PS2**, **MAILCHECK** and **IFS** cannot be unset.

wait [*n*]
Wait for the specified process and report its termination status. If *n* is not given, all currently active child processes are waited for and the return code is zero.

Invocation

If the shell is invoked through *exec*(2) and the first character of argument zero is **-**, commands are initially read from **/etc/profile** and then from **$HOME/.profile**, if such files exist. Thereafter, commands are read as described below, which is also the case when the shell is invoked as **/bin/sh**. The flags below are interpreted by the shell on invocation only; Note that unless the **-c** or **-s** flag is specified, the first argument is assumed to be the name of a file containing commands, and the remaining arguments are passed as positional parameters to that command file:

-c *string* If the **-c** flag is present then commands are read from *string*.
-s If the **-s** flag is present or if no arguments remain then commands are read from the standard input. Any remaining arguments specify the positional parameters. Shell output (except for *Special Commands*) is written to file descriptor 2.
-i If the **-i** flag is present or if the shell input and output are attached to a terminal, then this shell is *interactive*. In this case TERMINATE is ignored (so that **kill 0** does not kill an interactive shell) and INTERRUPT is caught and ignored (so that **wait** is interruptible). In all cases, QUIT is ignored by the shell.
-r If the **-r** flag is present the shell is a restricted shell.

The remaining flags and arguments are described under the **set** command above.

Rsh Only

Rsh is used to set up login names and execution environments whose capabilities are more controlled than those of the standard shell. The actions of *rsh* are identical to those of *sh*, except that the following are disallowed:
changing directory (see *cd*(1)),
setting the value of **$PATH,**
specifying path or command names containing /,
redirecting output (> and >>).

The restrictions above are enforced after **.profile** is interpreted.

When a command to be executed is found to be a shell procedure, *rsh* invokes *sh* to execute it. Thus, it is possible to provide to the end–user shell procedures that have access to the full power of the standard shell, while imposing a limited menu of commands; this scheme assumes that the

end–user does not have write and execute permissions in the same directory.

The net effect of these rules is that the writer of the **.profile** has complete control over user actions, by performing guaranteed setup actions and leaving the user in an appropriate directory (probably *not* the login directory).

The system administrator often sets up a directory of commands (i.e., **/usr/rbin**) that can be safely invoked by *rsh*. Some systems also provide a restricted editor *red*.

FILES

/etc/profile
$HOME/.profile
/tmp/sh*
/dev/null

RETURN VALUE

The error codes returned by the shell are:

0 - success;
1 - a built–in command failure (see *Special Commands*);
2 - syntax error;
3 - signal received that is not trapped.

If the shell is non–interactive, it will terminate and pass one of the above as its exit status. If it is interactive, it will not terminate, but $? will be set to one of the above values.

Whenever a child process of the shell dies due to a signal, the shell returns an exit status of 80 hexadecimal + the number of the signal.

SEE ALSO

acctcms(1M), acctcom(1), cd(1), echo(1), env(1), login(1), newgrp(1), pwd(1), test(1), umask(1), dup(2), exec(2), fork(2), pipe(2), signal(2), ulimit(2), umask(2), wait(2), a.out(5), profile(5), environ(7).

CAVEATS

If a command is executed, and a command with the same name is installed in a directory in the search path before the directory where the original command was found, the shell will continue to *exec* the original command. Use the **hash** command to correct this situation.

The shell assumes it is talking to terminals that only process the least significant seven bits of a character. If your terminal uses all eight bits, you may see some strange strings.

When the shell encounters >>, it does not open the file in append mode. Instead, it opens the file for writing and seeks to the end. If you move the current directory or one above it, **pwd** may not give the correct response. Use the **cd** command with a full path name to correct this situation.

The command **readonly** (without arguments) produces the same output as the command **export**.

NAME

size - print section sizes of common object files

SYNOPSIS

size [**-o**] [**-x**] files

HP–UX COMPATIBILITY

Level: HP–UX/STANDARD

Origin: System V

DESCRIPTION

The *size* command produces section size information for each section in common object files. The size of the text, data and bss (uninitialized data) sections are printed along with the total size of the object file. If an archive file is input to the *size* command the information for all archive members is displayed.

By default, numbers are printed in decimal, unless altered by the options described below.

-o print numbers in octal.

-x print numbers in hexadecimal.

HARDWARE DEPENDENCIES

Series 500:

The *text* size shown is the sum of the sizes of all code segments.

The *data* size shown is the sum of the initialized portions of the ddata and idata segments (which may be one or two data segments).

The *bss* size shown is the sum of the uninitialized portions of the ddata and idata segments.

If *size* is run on any commands shipped with HP–UX, the *text* size does not include any shared library segments referenced by the command.

SEE ALSO

as(1), cc(1), ld(1), a.out(5), ar(5).

DIAGNOSTICS

size: name: cannot open
if *name* cannot be read.

size: name: bad magic
if *name* is not an appropriate common object file.

NAME

sleep - suspend execution for an interval

SYNOPSIS

sleep time

HP–UX COMPATIBILITY

Level: HP–UX/STANDARD

Origin: System V

DESCRIPTION

Sleep suspends execution for *time* seconds. It is used to execute a command after a certain amount of time, as in:

(sleep 105; *command*)&

or to execute a command every so often, as in:

```
while true
do
        command
        sleep 37
done
```

SEE ALSO

alarm(2), sleep(3C).

BUGS

Time must be less than 65536 seconds.

NAME

slp - set the options for a printer

SYNOPSIS

slp [**-a**] [**-c** cols] [**-d**] [**-i** indent] [**-l** lines] [**-n**]

HP-UX COMPATIBILITY

Level: HP-UX/NON-STANDARD

Origin: HP

Remarks: *Slp* is implemented on the Series 200 only.

DESCRIPTION

Slp sets certain printer options for the device that is the current standard output.

The meanings of the options are:

-a Reports all of the option settings.

-c *cols* The number of columns which are to be printed is indicated by the *cols* argument. Characters beyond the last specified column will be truncated.

-d Resets options to the defaults for the device. (This action is not taken until the next open occurs on the device.)

-i *indent* *Indent* selects the number of columns to indent before the first printed column.

-l *lines* The number of lines per page is set to *lines*. The last new-line character of each page will be changed to a form-feed.

-n Set the page size to infinity. (Since the last new-line of the page is never encountered, no new-line characters will be changed to form-feeds.)

EXAMPLES

A typical case is to set the printer to 80 columns, no indentation, and no form-feeds between pages:

```
slp -c80 -i0 -n >/dev/lp
```

HARDWARE DEPENDENCIES

Series 200:

The value of *cols* will be forced into the range of 1 to 227, the value of *indent* from 0 to 227, and the value of *lines* from 1 to MAXSHORT.

SEE ALSO

ioctl(2), lp(4).

NAME

sort - sort and/or merge files

SYNOPSIS

sort [[**-cmu**] [**-o**output] [**-y**kmem] [**-z**recsz] [**-dfilkMnr** [**-L**language] [**-K**table]] [**-bt**x] [+pos1 [-pos2]]] [files]

HP-UX COMPATIBILITY

Level: HP-UX/STANDARD

Origin: System V

Native Language Support:
8-bit and 16-bit data, customs, messages

DESCRIPTION

Sort sorts lines of all the named files together and writes the result on the standard output. The standard input is read if **-** is used as a file name or no input files are named.

Comparisons are based on one or more sort keys extracted from each line of input. By default, there is one sort key, the entire input line, and ordering is lexicographic by bytes in machine collating sequence.

The following options alter the default behavior:

-c Check that the input file is sorted according to the ordering rules; give no output unless the file is out of sort.

-m Merge only, the input files are already sorted.

-u Unique: suppress all but one in each set of lines having equal keys.

-o*output*
The argument given is the name of an output file to use instead of the standard output. This file may be the same as one of the inputs. There may be optional blanks between **-o** and *output*.

-y*kmem*
The amount of main memory used by the sort has a large impact on its performance. Sorting a small file in a large amount of memory is a waste. If this option is omitted, *sort* begins using a system default memory size, and continues to use more space as needed. If this option is presented with a value, *kmem*, *sort* will start using that number of kilobytes of memory, unless the administrative minimum or maximum is violated, in which case the corresponding extremum will be used. Thus, **-y**0 is guaranteed to start with minimum memory. By convention, **-y** (with no argument) starts with maximum memory.

-z*recsz*
The size of the longest line read is recorded in the sort phase so buffers can be allocated during the merge phase. If the sort phase is omitted via the **-c** or **-m** options, a popular system default size will be used. Lines longer than the buffer size will cause *sort* to terminate abnormally. Supplying the actual number of bytes in the longest line to be merged (or some larger value) will prevent abnormal termination.

The following options override the default ordering rules.

-d "Dictionary" order: only letters, digits and blanks (spaces and tabs) are significant in comparisons.

-f Fold lower case into upper case, for purposes of comparison.

-i Ignore characters outside the ASCII range 040–0176 in non-numeric comparisons.

-M Compare as months. The first three non-blank characters of the field are folded to upper case and compared so that "JAN" < "FEB" < ... < "DEC". Invalid field compare low to

"JAN". The **-M** option implies the **-b** option (see below).

-n An initial numeric string, consisting of optional blanks, optional minus sign, and zero or more digits with optional decimal point, is sorted by arithmetic value. The **-n** option implies the **-b** option (see below). Note that the **-b** option is only effective when restricted sort key specifications are in effect.

-r Reverse the sense of comparisons.

The following two options apply to extended ASCII character sets.

-l Collate characters using the table associated with the user's *LANG* variable (See *environ*(7)).

-L*language*
Allow the user to specify his/her local language. The **–f** and **–l** options are to behave in a language dependent fashion, the **–d, –i, –M** and **–n** options are not. The *language* overrides the *LANG* variable.

The following two options apply to data containing 16–bit characters.

-k 16–bit characters data appears in canonical 8–bit form. This option implies –b. If **-K** is not specified native machine collation is used.

-K*table* Name of a file which contains a 16–bit character set collation table as specified in *nl_col_seq_16*(5).

When ordering options appear before restricted sort key specifications, the requested ordering rules are applied globally to all sort keys. When attached to a specific sort key (described below), the specified ordering options override all global ordering options for that key.

The notation +*pos1* -*pos2* restricts a sort key to one beginning at *pos1* and ending at *pos2*. The characters at positions *pos1* and *pos2* are included in the sort key (provided that *pos2* does not precede *pos1*). A missing -*pos2* means the end of the line.

Specifying *pos1* and *pos2* involves the notion of a field, a minimal sequence of characters followed by a field separator or a new–line. By default, the first blank (space or tab) of a sequence of blanks acts as the field separator. All blanks in a sequence of blanks are considered to be part of the next field; for example, all blanks at the beginning of a line are considered to be part of the first field. The treatment of field separators can be altered using the options:

-t*x* Use *x* as the field separator character; *x* is not considered to be part of a field (although it may be included in a sort key). Each occurrence of *x* is significant (e.g., *xx* delimits an empty field).

-b Ignore leading blanks when determining the starting and ending positions of a restricted sort key. If the **-b** option is specified before the first +*pos1* argument, it will be applied to all +*pos1* arguments. Otherwise, the **b** flag may be attached independently to each +*pos1* or -*pos2* argument (see below).

Pos1 and *pos2* each have the form *m.n* optionally followed by one or more of the flags **bdfinr**. A starting position specified by +*m.n* is interpreted to mean the *n*+1st character in the *m*+1st field. A missing *.n* means .0, indicating the first character of the *m*+1st field. If the **b** flag is in effect *n* is counted from the first non–blank in the *m*+1st field; +*m*.0**b** refers to the first non–blank character in the *m*+1st field.

A last position specified by -*m.n* is interpreted to mean the *n*th character (including separators) after the last character of the *m th* field. A missing *.n* means .0, indicating the last character of the *m*th field. If the **b** flag is in effect *n* is counted from the last leading blank in the *m*+1st field; -*m*.1**b** refers to the first non–blank in the *m*+1st field.

When there are multiple sort keys, later keys are compared only after all earlier keys compare equal. Lines that otherwise compare equal are ordered with all bytes significant.

HARDWARE DEPENDENCIES

Series 200/500: 16-bit collation currently is not supported.

EXAMPLES

Sort the contents of *infile* with the second field as the sort key:

```
sort +1 -2 infile
```

Sort, in reverse order, the contents of *infile1* and *infile2*, placing the output in *outfile* and using the first character of the second field as the sort key:

```
sort -r -o outfile +1.0 -1.2 infile1 infile2
```

Sort, in reverse order, the contents of *infile1* and *infile2* using the first non–blank character of the second field as the sort key:

```
sort -r +1.0b -1.1b infile1 infile2
```

Print the password file (*passwd*(5)) sorted by the numeric user ID (the third colon–separated field):

```
sort -t: +2n -3 /etc/passwd
```

Print the lines of the already sorted file *infile*, suppressing all but the first occurrence of lines having the same third field (the options **-um** with just one input file make the choice of a unique representative from a set of equal lines predictable):

```
sort -um +2 -3 infile
```

FILES

/usr/tmp/stm???

SEE ALSO

comm(1), join(1), uniq(1), nl_col_seq_16(5), nl_col_seq_8(5), environ(7), hpnls(7), langid(7).

DIAGNOSTICS

Comments and exits with non–zero status for various trouble conditions (e.g., when input lines are too long), and for disorder discovered under the **-c** option. When the last line of an input file is missing a **new–line** character, *sort* appends one, prints a warning message, and continues.

BUGS

Sort does not understand "missing" fields. For example, consider a file with the following contents:

Doe,John	mailman	17550	8
Spencer,Joe	plumber		4
Johns,Ann	secretary	15950	
Malley,Dean	engineer	26750	4

You may get unexpected results if you try to sort on the third or fourth fields (all names and associated data are fictitious).

Sort does not expand tabs when counting characters to locate a field.

Sort does not perform "–fl" in which it would fold lower to upper, then compare in language–dependent sequence. Instead, it performs "–fl" as though it were "–l".

NAME

spell, spellin, spellout - find spelling errors

SYNOPSIS

spell [options] [files]

/usr/lib/spell/spellin [list]

/usr/lib/spell/spellout [**-d**] list

HP–UX COMPATIBILITY

Level: HP–UX/STANDARD

Origin: System III

DESCRIPTION

Spell collects words from the named *files* and looks them up in a spelling list. Words that neither occur among nor are derivable (by applying certain inflections, prefixes, and/or suffixes) from words in the spelling list are printed on the standard output. If no *files* are named, words are collected from the standard input.

Spell ignores most *nroff*(1), and *tbl* constructions.

The options are:

-v all words not literally in the spelling list are printed, and plausible derivations from the words in the spelling list are indicated.

-b British spelling is checked. Besides preferring *centre*, *colour*, *speciality*, *travelled*, etc., this option insists upon *–ise* in words like *standardise*, Fowler and the OED to the contrary notwithstanding.

-x every plausible stem is printed with = for each word.

The spelling list is based on many sources, and while more haphazard than an ordinary dictionary, is also more effective with respect to proper names and popular technical words. Coverage of the specialized vocabularies of biology, medicine, and chemistry is light.

Pertinent auxiliary files may be specified by name arguments, indicated below with their default settings. Copies of all output are accumulated in the history file. The stop list filters out misspellings (e.g., thier=thy-y+ier) that would otherwise pass.

Two routines help maintain the hash lists used by *spell* (both expect a list of words, one per line, from the standard input):

spellin adds the words on the standard input to the pre–existing *list* and places a new list on the standard output. If no *list* is specified, the new list is created from scratch.

Spellout looks up each word read from the standard input, and prints on the standard output those that are missing from (or, with the **-d** option, present in) the hash list.

HARDWARE DEPENDENCIES

Series 200/500:

eqn, *typo* and *troff* are not currently supported.

FILES

D_SPELL=/usr/lib/spell/hlist[ab]	hashed spelling lists, American & British
S_SPELL=/usr/lib/spell/hstop	hashed stop list
H_SPELL=/usr/lib/spell/spellhist	history file
/tmp/spell.$$	temporary
/usr/lib/spell/spellprog	program

SEE ALSO

deroff(1), nroff(1), sed(1), sort(1), tbl(1), tee(1).

BUGS

The spelling list's coverage is uneven; new installations will probably wish to monitor the output for several months to gather local additions; typically, these are kept in a separate local dictionary that is added to the hashed *list* via *spellin*.

British spelling was done by an American.

NAME

split - split a file into pieces

SYNOPSIS

split [**-***n*] [file [name]]

HP–UX COMPATIBILITY

Level: HP–UX/STANDARD

Origin: System V

DESCRIPTION

Split reads *file* and writes it in *n*–line pieces (default 1000 lines) onto a set of output files. The name of the first output file is *name* with **aa** appended, and so on lexicographically, up to **zz** (a maximum of 676 files). *Name* cannot be longer than 12 characters. If no output name is given, **x** is default.

If no input file is given, or if **-** is given instead, then the standard input file is used.

SEE ALSO

bfs(1), csplit(1).

NAME

ssp - remove multiple line–feeds from output

SYNOPSIS

ssp

HP–UX COMPATIBILITY

Level: HP–UX/STANDARD

Origin: UCB

Remarks: Not supported on the Integral Personal Computer.

DESCRIPTION

Ssp (single–space) removes redundant blank lines from the standard input and sends the result to the standard output. It is typically used in pipelines like

```
nroff -ms file1 | ssp
```

Ssp is equivalent the the 4.2BSD **cat** **–s** command.

SEE ALSO

cat(1), rmnl(1).

NAME

strings - find the printable strings in a object, or other binary, file

SYNOPSIS

strings [**-a**] [**-o**] [*-number*] [file] ...

HP–UX COMPATIBILITY

Level: HP–UX/STANDARD

Origin: UCB

Remarks: Not supported on the Integral Personal Computer.

DESCRIPTION

Strings looks for ascii strings in a file. If no *files* are specified, **stdin** is used. A string is any sequence of 4 or more printing characters ending with a new–line or a null.

The following flags are defined.

a By default, *strings* only looks in the initialized data space of object files (as recognised by their magic numbers). If this flag is used, the whole file is inspected. This flag is always set if stdin is being read or the file is not recognized as an object file. For backward compatability, **-** is taken as a synonym for **-a**.

o Each string is preceded by its offset in the file (in octal).

number *number* is used as the minimum string length rather than 4.

Strings is useful for identifying random object files and many other things.

SEE ALSO

od(1)

BUGS

The algorithm for identifying strings is extremely primitive.

NAME

strip - remove symbols and debug information

SYNOPSIS

strip name ...

HP-UX COMPATIBILITY

Level: HP-UX/DEVELOPMENT

Origin: System V

DESCRIPTION

Strip removes the symbol table and debug information from an executable object file. Once this is done, no symbolic debugging access will be available for that file; therefore this command is normally run only on production modules that have been debugged and tested. The effect is the same as use of the **-s** option of *ld*.

If *name* is a relocatable file, *strip* removes the debug information.

If the *strip* command is executed on an archive file (see *ar* (5)) the archive symbol table will be removed. The archive symbol table must be restored by executing the *ar* (1) command with the **s** option before the archive can be link-edited by the *ld* (1) command. Also, strip removes the debug information from any *a.out* file it finds in the archive.

The purpose of this command is to reduce the file storage overhead taken by the object file.

HARDWARE DEPENDENCIES

Series 200:

If *name* is a relocatable file, *strip* will remove the local symbols from it. If *name* is an archive file, *strip* will remove the local symbols from any *a.out* format files it finds in the archive. Certain libraries, such as those residing in **/lib**, have no need for local symbols. By deleting them, the size of the archive is decreased and link editing performance is increased.

FILES

/tmp/s* temporary files

SEE ALSO

ar(1), ld(1), ar(5), a.out(5).

NAME

stty - set the options for a terminal port

SYNOPSIS

stty [**-a**] [**-g**] [options]

HP-UX COMPATIBILITY

Level: HP-UX/NUCLEUS

Origin: System III

DESCRIPTION

Stty sets certain terminal I/O options for the device that is the current standard input; without arguments, it reports the settings of certain options; with the **-a** option, it reports all of the option settings; with the **-g** option, it reports current settings in a form that can be used as an argument to another *stty* command. Detailed information about the modes listed in the first five groups below may be found in *termio*(4) for asynchronous lines. Options in the last group are implemented using options in the previous groups. Note that many combinations of options make no sense, but no sanity checking is performed. The options are selected from the following:

Control Modes

parenb (**-parenb**) enable (disable) parity generation and detection.

parodd (**-parodd**) select odd (even) parity.

cs5 cs6 cs7 cs8 select character size (see *termio*(4)).

0 hang up phone line immediately.

50 75 110 134.5 150 200 300 600 900 1200 1800 2400 3600 4800 7200 9600 19200 38400 exta extb
Set terminal baud rate to the number given, if possible. (All speeds are not supported by all hardware interfaces.)

hupcl (**-hupcl**) hang up (do not hang up) modem connection on last close.

hup (**-hup**) same as **hupcl** (**-hupcl**).

cstopb (**-cstopb**) use two (one) stop bits per character.

cread (**-cread**) enable (disable) the receiver.

crts (**-crts**) enable (disable) request-to-send.

clocal (**-clocal**) assume a line without (with) modem control.

Input Modes

ignbrk (**-ignbrk**) ignore (do not ignore) break on input.

ienqak (**-ienqak**) enable (disable) ENQ-ACK handshaking.

brkint (**-brkint**) signal (do not signal) INTR on break.

ignpar (**-ignpar**) ignore (do not ignore) parity errors.

parmrk (**-parmrk**) mark (do not mark) parity errors (see *termio*(4)).

inpck (**-inpck**) enable (disable) input parity checking.

istrip (**-istrip**) strip (do not strip) input characters to seven bits.

inlcr (**-inlcr**) map (do not map) NL to CR on input.

igncr (**-igncr**) ignore (do not ignore) CR on input.

icrnl (**-icrnl**) map (do not map) CR to NL on input.

iuclc (**-iuclc**) map (do not map) upper-case alphabetics to lower case on input.

ixon (**-ixon**) enable (disable) START/STOP output control. Output is stopped by sending an ASCII DC3 and started by sending an ASCII DC1.

ixany (**-ixany**) allow any character (only DC1) to restart output.

ixoff (**-ixoff**) request that the system send (not send) START/STOP characters when the input queue is nearly empty/full.

Output Modes

opost (**-opost**) post–process output (do not post–process output; ignore all other output modes).

olcuc (**-olcuc**) map (do not map) lower–case alphabetics to upper case on output.

onlcr (**-onlcr**) map (do not map) NL to CR–NL on output.

ocrnl (**-ocrnl**) map (do not map) CR to NL on output.

onocr (**-onocr**) do not (do) output CRs at column zero.

onlret (**-onlret**) on the terminal NL performs (does not perform) the CR function.

ofill (**-ofill**) use fill characters (use timing) for delays.

ofdel (**-ofdel**) fill characters are DELs (NULs).

cr0 cr1 cr2 cr3 select style of delay for carriage returns (see *termio*(4)).

nl0 nl1 select style of delay for line–feeds (see *termio*(4)).

tab0 tab1 tab2 tab3 select style of delay for horizontal tabs (see *termio*(4).

bs0 bs1 select style of delay for backspaces (see *termio*(4)).

ff0 ff1 select style of delay for form–feeds (see *termio*(4)).

vt0 vt1 select style of delay for vertical tabs (see *termio*(4)).

Local Modes

isig (**-isig**) enable (disable) the checking of characters against the special control characters INTR and QUIT.

icanon (**-icanon**) enable (disable) canonical input (ERASE and KILL processing).

xcase (**-xcase**) canonical (unprocessed) upper/lower–case presentation.

echo (**-echo**) echo back (do not echo back) every character typed.

echoe (**-echoe**) echo (do not echo) ERASE character as a backspace–space–backspace string. Note: this mode will erase the ERASEed character on many CRT terminals; however, it does *not* keep track of column position and, as a result, may be confusing on escaped characters, tabs, and backspaces.

echok (**-echok**) echo (do not echo) NL after KILL character.

lfkc (**-lfkc**) the same as **echok** (**-echok**); obsolete.

echonl (**-echonl**) echo (do not echo) NL.

noflsh (**-noflsh**) disable (enable) flush after INTR or QUIT.

Control Assignments

control–character c set *control–character* to *c*, where *control–character* is **erase**, **kill**, **intr**, **quit**, **swtch**, **eof**, **min**, or **time** (**min** and **time** are used with **-icanon**; see *termio*(4)). If *c* is preceded by an (escaped from the shell) caret (^), then the value used is the corresponding CTRL character (e.g., "^**d**" is a **CTRL–d**); "^**?**" is interpreted as DEL and "^**-**" is interpreted as undefined.

line *i* set line discipline to *i* ($0 < i < 127$). (See *termio*(4)).

Combination Modes

evenp or **parity** enable **parenb** and **cs7**.

oddp enable **parenb**, **cs7**, and **parodd**.

-parity, **-evenp**, or **-oddp** disable **parenb**, and set **cs8**.

raw (**-raw** or **cooked**) enable (disable) raw input and output (no ERASE, KILL, INTR, QUIT, EOT, or output post processing).

nl (**-nl**) unset (set) **icrnl**, **onlcr**. In addition **-nl** unsets **inlcr**, **igncr**, **ocrnl**, and **onlret**.

lcase (**-lcase**) set (unset) **xcase**, **iuclc**, and **olcuc**.

LCASE (**-LCASE**) same as **lcase** (**-lcase**).

tabs (**-tabs** or **tab3**) preserve (expand to spaces) tabs when printing.

ek reset ERASE and KILL characters back to normal # and @.

sane resets all modes to some reasonable values.

term set all modes suitable for the terminal type *term*, where *term* is one of **tty33**, **tty37**, **vt05**, **tn300**, **ti700**, **hp**,

HARDWARE DEPENDENCIES

Series 200/500:

Refer to *termio*(4) for a description of the capabilities that are not supported.

SEE ALSO

tabs(1), ioctl(2), termio(4).

NAME

su - become super–user or another user

SYNOPSIS

su [**-**] [name [arg ...]]

HP–UX COMPATIBILITY

Level: HP–UX/STANDARD

Origin: System V

DESCRIPTION

Su allows one to become another user without logging off. The default user *name* is **root** (i.e., super–user).

To use *su*, the appropriate password must be supplied (unless one is already **root**). If the password is correct, *su* will execute a new shell with the real and effective user ID, real and effective group ID, and group access list set to that of the specified user. The new shell will be the optional program named in the shell field of the specified user's password file entry (see *passwd*(4)), or **/bin/sh** if none is specified (see *sh*(1)). To restore normal user ID privileges, type an **EOF** to the new shell.

Any additional arguments given on the command line are passed to the program invoked as the shell, permitting the super–user to run shell procedures with restricted privileges. When using programs like *sh*(1), an *arg* of the form **-c** *string* executes *string* via the shell and an arg of **-r** will give the user a restricted shell.

The following statements are true only if the optional program named in the shell field of the specified user's password file entry is like *sh*(1). If the first argument to *su* is a **-**, the environment will be changed to what would be expected if the user actually logged in as the specified user. This is done by invoking the program used as the shell with an *arg0* value whose first character is **-**, thus causing first the system's profile (**/etc/profile**) and then the specified user's profile (**.profile** in the new HOME directory) to be executed. Otherwise, the environment is passed along unchanged, except that **$PATH**, is unconditionally set to **/bin:/etc:/usr/bin** for **root**. Note that if the optional program used as the shell is **/bin/sh**, the user's **.profile** can check *arg0* for **-sh** or **-su** to determine if it was invoked by *login*(1) or *su*(1), respectively. If the user's program is other than **/bin/sh**, then **.profile** is invoked with an *arg0* of *–program* by both *login*(1) and *su*(1).

The **-** option always resets **$PATH** to **/bin:/etc:/usr/bin** for the super–user, and **/bin:/usr/bin** for all others. However, the files */etc/profile* and *.profile* are normally executed anyway, thus restoring the intended value of **$PATH**.

All attempts to become another user are logged in /usr/adm/sulog, including failures. Successful attempts are flagged with "+", failures with "–".

EXAMPLES

To become user **bin** while retaining your previously exported environment, execute:

```
su bin
```

To become user **bin** but change the environment to what would be expected if **bin** had originally logged in, execute:

```
su – bin
```

To execute *command* with the temporary environment and permissions of user **bin**, type:

```
su – bin –c "command args"
```

FILES

/etc/passwd	system's password file
/etc/logingroup	system's default group access list file
/etc/profile	system's profile
$HOME/.profile	user's profile
/usr/adm/sulog	log of all attempts

VARIABLES

HOME	the user's home directory
LOGNAME	the user's login name
PATH	the command name search path
PS1	the default prompt
SHELL	the name of the user's shell

HARDWARE DEPENDENCIES

Multiple groups are not currently supported on Series 500 and the Integral PC.

SEE ALSO

env(1), login(1), sh(1), initgroups(3C), group(5), passwd(5), profile(5), environ(7).

NAME

sum - print checksum and block count of a file

SYNOPSIS

sum [**-r**] [file]

HP–UX COMPATIBILITY

Level: HP–UX/STANDARD

Origin: System V

Remarks: Not supported on the Integral Personal Computer.

DESCRIPTION

Sum calculates and prints a 16–bit checksum for the named file, and also prints the number of 512-byte blocks in the file. **Stdin** is used if no file names are given. *Sum* is typically used to look for bad spots, or to validate a file communicated over some transmission line. The option **-r** causes an alternate algorithm to be used in computing the checksum.

SEE ALSO

wc(1).

DIAGNOSTICS

"Read error" is indistinguishable from end of file on most devices; check the block count.

NAME

sync - update the super block

SYNOPSIS

sync

HP–UX COMPATIBILITY

Level: HP–UX/NUCLEUS

Origin: System V

Remarks: Not supported on the Integral Personal Computer.

DESCRIPTION

Sync executes the *sync* system intrinsic. If the system is to be stopped, *sync* must be called to insure file system integrity. It will flush all previously unwritten system buffers out to disk, thus assuring that all file modifications up to that point will be saved. See *sync*(2) for details.

SEE ALSO

sync(2).

REDUCED SEMANTICS

If sync(2) is a no–op and the system is RUN ONLY or NUCLEUS, the sync command need not be present.

NAME

tabs - set tabs on a terminal

SYNOPSIS

tabs [tabspec] [**+m**n] [**-T**type]

HP-UX COMPATIBILITY

Level: HP-UX/STANDARD

Origin: System III

DESCRIPTION

Tabs sets the tab stops on the user's terminal according to the tab specification *tabspec*, after clearing any previous settings. The user's terminal must have remotely-settable hardware tabs.

If you are using a non-HP terminal, you should keep in mind that behavior will vary for some tab settings.

Four types of tab specification are accepted for *tabspec*: "canned," repetitive, arbitrary, and file. If no *tabspec* is given, the default value is **-8**, i.e., HP-UX "standard" tabs. The lowest column number is 1. Note that for *tabs*, column 1 always refers to the leftmost column on a terminal, even one whose column markers begin at 0.

-*code* Gives the name of one of a set of "canned" tabs. The legal *code*s and their meanings are as follows:

-a 1,10,16,36,72
Assembler, IBM S/370, first format

-a2 1,10,16,40,72
Assembler, IBM S/370, second format

-c 1,8,12,16,20,55
COBOL, normal format

-c2 1,6,10,14,49
COBOL compact format (columns 1-6 omitted). Using this code, the first typed character corresponds to card column 7, one space gets you to column 8, and a tab reaches column 12. Files using this tab setup should include a format specification as follows:

<:t-c2 m6 s66 d:>

-c3 1,6,10,14,18,22,26,30,34,38,42,46,50,54,58,62,67
COBOL compact format (columns 1-6 omitted), with more tabs than **-c2.** This is the recommended format for COBOL. The appropriate format specification is:

<:t-c3 m6 s66 d:>

-f 1,7,11,15,19,23
FORTRAN

-p 1,5,9,13,17,21,25,29,33,37,41,45,49,53,57,61
PL/I

-s 1,10,55
SNOBOL

-u 1,12,20,44
UNIVAC 1100 Assembler

In addition to these "canned" formats, three other types exist:

-*n* A repetitive specification requests tabs at columns 1+*n*, 1+2**n*, etc. Of particular importance is the value **-8**: this represents the HP-UX "standard" tab setting, and is the most likely tab setting to be found at a terminal. It is required for use with the *nroff*(1) **-h** option for high-speed output. Another special case is the value **-0**, implying no tabs at all.

n1,*n2*,... The arbitrary format permits the user to type any chosen set of numbers, separated by commas, in ascending order. Up to 40 numbers are allowed. If any number (except the

first one) is preceded by a plus sign, it is taken as an increment to be added to the previous value. Thus, the tab lists 1,10,20,30 and 1,10,+10,+10 are considered identical.

--*file* If the name of a file is given, *tabs* reads the first line of the file, searching for a format specification. If it finds one there, it sets the tab stops according to it, otherwise it sets them as **-8**. This type of specification may be used to make sure that a tabbed file is printed with correct tab settings, and would be used with the *pr*(1) command:

tabs -- file; pr file

Any of the following may be used also; if a given flag occurs more than once, the last value given takes effect:

-T*type* *Tabs* usually needs to know the type of terminal in order to set tabs and always needs to know the type to set margins. *Type* is a name listed in *term*(7). If no **-T** flag is supplied, *tabs* searches for the **$TERM** value in the *environment* (see *environ*(7)). If no *type* can be found, *tabs* tries a sequence that will work for many terminals.

+m*n* The margin argument may be used for some terminals. It causes all tabs to be moved over *n* columns by making column *n+1* the left margin. If **+m** is given without a value of *n*, the value assumed is 10. The normal (leftmost) margin on most terminals is obtained by **+m0**. The margin for most terminals is reset only when the **+m** flag is given explicitly.

Tab and margin setting is performed via the standard output.

DIAGNOSTICS

illegal tabs	when arbitrary tabs are ordered incorrectly.
illegal increment	when a zero or missing increment is found in an arbitrary specification.
unknown tab code	when a "canned" code cannot be found.
can't open	if **--***file* option used, and file can't be opened.
file indirection	if **--***file* option used and the specification in that file points to yet another file. Indirection of this form is not permitted.

SEE ALSO

pr(1), nroff(1), tset(1), environ(7), term(7).

BUGS

There is no consistency among different terminals regarding ways of clearing tabs and setting the left margin.

It is generally impossible to usefully change the left margin without also setting tabs.

Tabs clears only 20 tabs (on terminals requiring a long sequence), but is willing to set 64.

NAME

tail - deliver the last part of a file

SYNOPSIS

tail [±[number][**lbc**[**f**]]] [file]

HP-UX COMPATIBILITY

Level: HP-UX/STANDARD

Origin: System V

DESCRIPTION

Tail copies the named file to the standard output beginning at a designated place. If no file is named, the standard input is used.

Copying begins at distance +*number* from the beginning, or -*number* from the end of the input (if *number* is null, the value 10 is assumed). *Number* is counted in units of lines, blocks, or characters, according to the appended option **l**, **b**, or **c**. When no units are specified, counting is by lines.

With the **-f** ("follow") option, if the input file is not a pipe, the program will not terminate after the line of the input file has been copied, but will enter an endless loop, wherein it sleeps for a second and then attempts to read and copy further records from the input file. Thus it may be used to monitor the growth of a file that is being written by some other process.

EXAMPLES

Tail accepts at most two arguments: the first consists of specified options, and the second specifies the file of interest. If the *number* and **f** options are both desired, they must be concatenated to create a single option argument, as follows:

```
tail -3lf john
```

This example prints the last three lines in the file *john* to the standard output, and leaves *tail* in "follow" mode.

If only the **f** option is desired, it must be preceded by a **-**, as follows:

```
tail -f fred
```

This example prints the last ten lines of the file *fred*, followed by any lines that are appended to *fred* between the time *tail* is initiated and killed. Note that this output may build up very quickly for rapidly changing input files, perhaps too fast to read on a CRT.

As another example, the command:

```
tail -15cf fred
```

will print the last 15 characters of the file *fred*, followed by any lines that are appended to *fred* between the time *tail* is initiated and killed.

The + option starts at the number indicated from the beginning of the file (rather than skipping the number of units indicated and then starting). For example:

```
tail +1b fred
```

prints the entire contents of the file *fred.*

SEE ALSO

head(1).

BUGS

Tails relative to the end of the file are stored in a buffer, and thus are limited in length. Thus, be wary of the results when piping output from other commands into *tail.*

Various kinds of anomalous behavior may happen with character special files.

Tail can pick up a maximum of 4K bytes of data from the specified file.

NAME

tar - tape file archiver

SYNOPSIS

tar [*key*] [[*file* | –C *directory*] ...]

HP–UX COMPATIBILITY

Level: HP–UX/DEVELOPMENT

Origin: System III and UCB

Native Language Support:
8-bit filenames.

DESCRIPTION

Tar saves and restores files on magnetic tape or flexible disc. Its actions are controlled by the *key* argument. The *key* is a string of characters containing at most one function letter and possibly one or more function modifiers. The *key* string may be preceded by a dash (-) (similar to the way options are specified in other HP–UX commands), but it is not necessary. Other arguments to the command are *files* (or directory names) specifying which files are to be dumped or restored. In all cases, appearance of a directory name refers to the files and (recursively) subdirectories of that directory.

The function portion of the *key* is specified by one of the following letters:

r The named *files* are added to the end of the archive. The **c** function implies this function.

x The named *files* are extracted from the archive. If a named file matches a directory whose contents had been written onto the archive, this directory is (recursively) extracted. If a named file on tape does not exist on the system, the file is created as follows:

The user, group, and other protections are restored from the tape.
The modification time is restored from the tape unless the **m** option is specified.
The file owner and group owner are normally that of the restoring process.
The set–user–ID, set–group–ID and sticky bits are normally not set. The **o** and **p** options control the restoration of protection; see below for more details.

If the files exist, their modes are not changed except that the set–user–ID, set–group–ID and sticky bits are cleared. If no *files* argument is given, the entire content of the archive is extracted. Note that if several files with the same name are on the archive, the last one overwrites all earlier ones.

t The names of all the files on the archive are listed. Adding the **v** option will expand this listing to include the file modes and owner numbers. The names of all files are listed each time that they occur on the tape.

u The named *files* are added to the archive if they are not already there, or have been modified since last written on that archive.

c Create a new archive; writing begins at the beginning of the archive, instead of after the last file. This command implies the **r** function.

The following function modifiers may be used in addition to the function letters listed above:

#s Where # is a tape drive number (**0,...,7**), and **s** is the density (**l** – low (800 bpi), **m** – medium (1600 bpi), or **h** – high (6250 bpi)). This modifier selects the drive on which the 9 track tape is mounted. The default is **0m**.

v Normally, *tar* does its work silently. The **v** (verbose) option causes it to type the name of each file it treats, preceded by the function letter. With the **t** function, **v** gives more information about the tape entries than just the name.

w Causes *tar* to print the action to be taken, followed by the name of the file, and then wait for the user's confirmation. If a word beginning with **y** is given, the action is performed. Any other input means "no".

f Causes *tar* to use the next argument as the name of the archive instead of **/dev/rmt/??**. If the name of the file is **-**, *tar* writes to the standard output or reads from the standard input, whichever is appropriate. Thus, *tar* can be used as the head or tail of a pipeline. *Tar* can also be used to move hierarchies with the command:

cd fromdir; tar cf - . | (cd todir; tar xf -)

b Causes *tar* to use the next argument as the blocking factor for archive records. If both **f** and **b** modifiers are specified, their arguments must match the order in which they are specified. This option should only be used when a blocking factor other than 1 is desired. The default is 1 (512 bytes) and the maximum is 20. The block size is determined automatically when reading 9 track tapes (key letters **x** and **t**). The blocking factor must be specified when reading flexible discs and cartridge tapes if they were written with a blocking factor different than the default.

l Tells *tar* to complain if it cannot resolve all of the links to the files being dumped. If **l** is not specified, no error messages are printed.

m Tells *tar* to not restore the modification time written on the archive. The modification time of the file will be the time of extraction.

h Forces *tar* to follow symbolic links as if they were normal files or directories. Normally, *tar* does not follow symbolic links. Not all HP-UX systems support symbolic links.

o For writing:

This option suppresses writing certain directory information that older versions of tar cannot handle on input. Tar normally writes information specifying owners and modes of directories in the archive. Former versions of tar, when encountering this information, will give error message of the form

"<name>/: cannot create".

This option will suppress writing that information.

For reading:

Causes extracted files to take on the user and group identifier of the user running the program rather than those on the tape. This is the default for the ordinary user, and may be overridden, to the extent the system protections allow, by the **p** option.

p This option causes files to be restored to the original modes and ownerships written on the archive, if possible. This is the default for the super–user, and may be overridden by the **o** option. For the ordinary used, if the system protections forbid the *chown*(2) operation needed to do this, the error will be ignored, and the ownership left with the restoring process. Set–user–ID, set–group–ID and sticky information will be restored as allowed by the protections defined by *chmod*(2), if the *chown* operation above succeeded.

The following option may be included in the file list:

-C*directory* *tar* will perform a *chdir*(2) to *directory*. This allows multiple directories not related by a close common parent to be archived using short relative path names.

If a 9 track tape drive is used as the output device, it must be configured in Berkeley compatability mode; see *mt*(4).

EXAMPLES

tar cvf /dev/rfd.0 file1 file2

This example creates a new archive on /dev/rfd.0 and copies file1 and file2 onto it, using a blocking factor of 20. The *key* is made up of one function letter (**c**) and two function modifiers (**v**, and **f**).

tar cv –C /usr include –C / etc

This example archives files from /usr/include and from /etc.

FILES

/dev/rmt/*

/dev/rfd.*
/tmp/tar*

SEE ALSO

ar(1), cpio(1), mt(4).

DIAGNOSTICS

Complaints about bad key characters and tape read/write errors.
Complaints if enough memory is not available to hold the link tables.

BUGS

There is no way to ask for the *n*–th occurrence of a file.
Tape errors are handled ungracefully.
The **u** option can be slow.
If the archive is on a flexible disc or cartridge tape, and if the blocking factor specified on output was not the default, the same blocking factor must be specified on input. This is because the blocking factor is not explicitly stored on the archive. Not following this rule and updating the archive can destroy it.
The current limit on file–name length is 100 characters.
Some previous versions of tar have claimed to support selective listing of file names using the **t** option with a list. To our knowledge this was an error in the documentation and does not appear in the original source code.
There is no way to restore an absolute path name to a relative position.
Archives should never be created on block special devices (e.g., */dev/fd.o*).
Tar always pads information written to an archive up to the next multiple of the block size. Therefore, if you are creating a small archive and write out one block of information, *tar* reports that one block was written, but the actual size of the archive may be larger if the **b** option was used.
Note that **tar c0m** is not the same as **tar cm0**.

NAME

tbl – format tables for nroff

SYNOPSIS

tbl [**–TX**] [files]

HP-UX COMPATIBILITY

Level: HP-UX/STANDARD

Origin: System V

DESCRIPTION

Tbl is a preprocessor that formats tables for *nroff*(1). The input files are copied to the standard output, except for lines between **.TS** and **.TE** command lines, which are assumed to describe tables and are re-formatted by *tbl*. (The **.TS** and **.TE** command lines are not altered by *tbl*).

.TS is followed by global options. The available global options are:

center center the table (default is left-adjust);
expand make the table as wide as the current line length;
box enclose the table in a box;
doublebox enclose the table in a double box;
allbox enclose each item of the table in a box;
tab (*x*) use the character *x* instead of a tab to separate items in a line of input data.

The global options, if any, are terminated with a semi-colon (;).

Next come lines describing the format of each line of the table. Each such format line describes one line of the actual table, except that the last format line (which must end with a period) describes *all* remaining lines of the table. Each column of each line of the table is described by a single key-letter, optionally followed by specifiers that determine the font and point size of the corresponding item, that indicate where vertical bars are to appear between columns, that determine column width, inter-column spacing, etc. The available key-letters are:

c center item within the column;
r right-adjust item within the column;
l left-adjust item within the column;
n numerically adjust item in the column: units positions of numbers are aligned vertically;
s span previous item on the left into this column;
a center longest line in this column and then left-adjust all other lines in this column with respect to that centered line;
^ span down previous entry in this column;
_ replace this entry with a horizontal line;
= replace this entry with a double horizontal line.

The characters **B** and **I** stand for the bold and italic fonts, respectively; the character | indicates a vertical line between columns.

The format lines are followed by lines containing the actual data for the table, followed finally by **.TE**. Within such data lines, data items are normally separated by tab characters.

If a data line consists of only _ or =, a single or double line, respectively, is drawn across the table at that point; if a *single item* in a data line consists of only _ or =, then that item is replaced by a single or double line.

Full details of all these and other features of *tbl* are given in the reference manual cited below.

The **–TX** option forces *tbl* to use only full vertical line motions, making the output more suitable for devices that cannot generate partial vertical line motions (e.g., line printers).

If no file names are given as arguments (or if – is specified as the last argument), *tbl* reads the standard input, so it may be used as a filter. When it is used with *neqn*(1), *neqn*, *tbl* should come first to minimize the volume of data passed through pipes.

EXAMPLE

If we let <tab> represent a tab (which should be typed as a genuine tab), then the input:

```
.TS
center box ;
cB s s
cI | cI s
^ | c c
l | n n .
Household Population
_
Town<tab>Households
<tab>Number<tab>Size
=
Bedminster<tab>789<tab>3.26
Bernards Twp.<tab>3087<tab>3.74
Bernardsville<tab>2018<tab>3.30
Bound Brook<tab>3425<tab>3.04
Bridgewater<tab>7897<tab>3.81
Far Hills<tab>240<tab>3.19
.TE
```

yields:

Household Population		
Town	*Households*	
	Number	Size
Bedminster	789	3.26
Bernards Twp.	3087	3.74
Bernardsville	2018	3.30
Bound Brook	3425	3.04
Bridgewater	7897	3.81
Far Hills	240	3.19

SEE ALSO

tbl Tutorial in the *HP-UX Concepts and Tutorials*.
cw(1), eqn(1), mm(1), mmt(1), neqn(1), nroff(1), mm(5), mv(5), troff(1).

BUGS

See *BUGS* under *nroff*(1).

NAME

tcio – Command Set 80 Cartridge Tape Utility

SYNOPSIS

tcio **–o[dervV]** [**–S** buffersize] [**–l** number] [**–n** limit] filename
tcio **–i[drv]** [**–S** buffersize] [**–l** number] [**–n** limit] filename
tcio **–u[rvV]** [**–m** blocknumber] [**–l** number] filename

HP-UX COMPATIBILITY

Level: HP-UX/NON-STANDARD

Origin: HP

Remarks: Not supported on the Integral Personal Computer.

DESCRIPTION

Tcio is an effective utility for optimizing the data transfer rate between a cartridge tape unit and the host processor. When used in conjunction with other utilities such as *cpio*, a significant improvement in transfer efficiency results while simultaneously reducing wear and tear on tape cartridges. When used with autochanger mechanisms, *tcio* can load a specified cartridge or automatically switch to successive cartridges as needed. The *tcio* utility option supports functions that are unique to cartridge tape operations.

Tcio –o (copy out) reads the standard input and writes the data to the raw CS/80 Cartridge Tape Unit associated with the special file *filename*.

Tcio –i (copy in) reads from the CS/80 Cartridge Tape Unit associated with the special file *filename* in raw mode and writes the data to the standard output.

Tcio –u (utility option) performs utility functions on the cartridge tape, such as release, mark, and/or verify the cartridge.

In all cases, *filename* **must** refer to a character (raw) special file associated with a CS/80 cartridge tape unit.

Tcio input and output (read/write) operations enable *immediate report* which can yield a significant increase in the tape performance (see *BACKGROUND INFORMATION* in this section for details). In addition, *tcio* puts a tape mark in the first block on each tape to prevent the tape from being image restored over a disc. A flag is also placed in the last block on each tape to identify whether the tape is the last tape in a multiple-tape sequence.

The following command options are recognized. At least one option, **–o**, **–i**, or **–u** **must** be specified. Additional options can be specified in any order, but all must precede the file name. Options without parameters can be listed individually or grouped together. Options with parameters require the parameter, and must be listed individually. Available modifiers and their meanings are:

–v Verbose mode; prints information and error messages to *stderr*.

–d Prints a checksum to *stderr*. The checksum is a 32-bit unsigned addition of the bytes in the data stream, thus providing an extra data integrity check in addition to tape verification. The checksum value is reported to the user, but it is not written on the media. It is up to the user to manually record and verify it. The checksum is valid **only** if the same number of bytes are read from the tape as were written to it. This option is independent of the verbose modifier.

–e Applies only to output operation, and causes a tape mark to be written on the nearest 1024-byte boundary following the end of the data. When a tape containing an end-of-data tape mark is read back, the read terminates upon encountering the tape mark. When this option is used, checksums generated by corresponding input and output operations always agree.

–V This option disables tape verification. Newer cartridge tape units using the HP 9144 drive provide hardware for verifying output data as it is written on the tape (called *read-while-*

write). This improves the integrity of recorded data as it is written on tape. HP 7908/11/12/14 drives do not have *read-while-write* verification hardware, so they require a separate verification operation after writing. This option should not be used with such units.

–r Releases the tape from the drive. On autochanger units, the tape is returned to the magazine.

–S [*buffersize*]
Enables specification of buffer size. This option forces the allocation of a block of memory to be used in reading or writing the tape. The buffer size in bytes is 1024 times the value specified for *buffersize*. A *buffersize* less than 4 or greater than 512 causes the program to terminate. If *buffersize* is not specified, *tcio* allocates a 64-Kbyte buffer. On tape units such as an HP 9144A or 7942/6 that support *immediate report,* a significant performance increase can sometimes be obtained by using smaller buffer sizes. For instance, if *tcio* is used as part of a pipeline, an appropriate buffer size is 8 (see *BACKGROUND INFORMATION* for further details).

–m [blocknumber]
This option writes a tape mark on a tape at the specified block. If a tape is created by some means other than *tcio*, a tape mark in block 0 of the tape prevents it from being image restored to a disc.

–l [number]
Autochanger mode: This option is intended solely for autochanger type tape units (such as the HP 35401). During input or output operations (**–i** or **–o**) the autochanger option automatically selects the target cartridge from the magazine and begins the transfer. When utility function are used instead, (**–u** option) *tcio* loads the specified cartridge into the drive, but performs no read/write operations. (Note: the autochanger must be in *selective mode* for the autochanger option to work properly.)

–n [limit]
Applies only to autochanger units, and must be preceded by the **–l** option. This option specifies the maximum number of cartridges to be used in a multi-tape transfer. Thus, **–l** starts the transfer by loading cartridge *number* and uses at most *limit* cartridges. If **–l** is specified without **–n**, *tcio* quietly assumes the remaining cartridges (in ascending order) from the magazine.

HARDWARE DEPENDENCIES

Due to I/O software architecture, a *buffersize* greater than 64 provides no increase in performance, but merely ties up system memory. Thus, the default *buffersize* is 64. If *buffersize* is specified greater than 64, it is silently truncated back to 64.

BACKGROUND INFORMATION

Tcio enables *immediate report* on cartridge tape units that support this mode (such as the HP 9144A, HP 7942/6, and HP 35401). Thus, the drive is able to return status before it completes writing data to the tape, allowing the host to send (or receive) data to the drive's buffer while the drive is simultaneously transferring data from the drive to the tape. If the host is able to transfer data to (or from) the tape unit fast enough (so the drive does not need to reposition the tape between requests) the tape is said to *stream*. *Streaming* dramatically increases tape drive read/write data rates.

Two basic strategies are available to the user. If the **–S** option is not specified, *tcio* uses as large a buffer as possible. The tape streams within the large transaction, but must reposition itself between successive requests (this strategy is recommended for the HP 7942/6 when transferring data between disk and tape). An alternate strategy is to use small buffers which can allow *continuous* streaming. Continuous streaming is more efficient, provided the drive's timing requirements are met. Use of small buffers minimizes the time necessary for the host to refill the drive's buffer. When reading or writing more than 8 Kbytes to a pipe, a small additional overhead is incurred. Thus, when *tcio* is used as part of a pipeline, a buffersize of 8 is recommended. This

alternate strategy for continuous streaming is sensitive to competing HP-IB activity.

EXAMPLES

The first example below copies the contents of a directory into an archive; the second restores it:

```
ls | cpio –o | tcio –o /dev/rct
tcio –i /dev/rct | cpio –i
```

To unload the cartridge from the drive without verifying the tape, execute:

```
tcio -urvV /dev/rct
```

This example copies all files in the current directory (by executing *find*) to the tape specified by the device file */dev/rct*. A checksum (option **-d**) is performed to verify that the data was written correctly on the tape, and verbose mode (**-v**) is used so you can see the names of all files being copied. In addition, a buffer size (option **-S)** of 8 blocks (i.e. 8 Kbytes) is specified:

```
find . -print | cpio -o | tcio -odv -S 8 /dev/rct
```

The next example assumes that the cartridge tape unit is an autochanger, with 8 slots in the magazine. The device has a *read after write* head, so *verify* is disabled. The *tcio* operation will start writing with cartridge 3, and will use at most 4 cartridges before prompting for additional media:

```
find usr -print | cpio -o | tcio -oVv -l 3 -n 4 /dev/rhp35401
```

The following example copies all the files and directories from the tape (specified by */dev/rct*) to the current directory. Data is transferred through an 8-block buffer.

```
tcio -ivr -S 8 /dev/rct | cpio -icdv
```

The **r** flag in this example releases the tape *after* upon completion of the transfer.

SEE ALSO

cpio(1).

BUGS

If the cartridge drive cannot read the manufacturer's block on the tape, the cartridge is locked in the drive and it cannot be extracted without removing power from the disc/tape drive. This failure is usually caused by faulty tape or a dirty drive mechanism.

NAME

tcio – Command Set 80 Cartridge Tape Utility

SYNOPSIS

/etc/tcio –o [**dervSVC**] [buffersize] filename
/etc/tcio –i [**drvS**] [buffersize] filename
/etc/tcio –u [**cmrvV**] [blocknumber] [**save** | **restore**] filename [disc_filename]

HP-UX COMPATIBILITY

Level: HP-UX/NON-STANDARD

Origin: HP

Remarks: This manual page describes the Series 500 implementation only. See other manual page for Series 200/300 implementation. Not supported on the Integral Personal Computer.

DESCRIPTION

Tcio –o (copy out) reads the standard input and writes the data to the raw Command Set 80 Cartridge Tape Unit specified by *filename.*

Tcio –i (copy in) reads the Command Set 80 Cartridge Tape Unit specified by *filename* in raw mode and writes the data to the standard output.

Tcio –u (utility) performs utility functions on the cartridge tape, such as image backup and restore, release, mark, and/or verify cartridge.

In all cases, *filename* MUST refer to a character special file associated with a Command Set 80 cartridge tape unit.

With the output and input operations, *tcio* utilizes a large buffer to transfer data to/from the cartridge tape, yielding a significant increase in performance, as well as a savings in wear and tear on the media and the mechanism. In addition, *tcio* puts a tape mark in the first block on each tape to prevent the tape from being image restored over a disc; it also utilizes the last block on each tape to flag whether the tape is the last tape in a multi-tape sequence or not.

With the utility operation, *tcio* provides functions that are unique to cartridge tapes.

One of the options **o**, **i**, or **u** must be specified. The meanings of the available modifiers are:

v Verbose mode; prints information and error messages to *stderr.*

d Prints a checksum to *stderr.* The checksum is a 32-bit unsigned addition of the bytes, providing an extra check of the validity of the tape in addition to tape verification. The value is only reported to the user and is not written on the media; thus, it's left up to the user to manually record and check it. The checksum is valid only for the **i** and **o** operations, and if the same number of bytes are read from the tape as were written to it. This option is independent of the verbose modifier.

e Applies only to the output operation, and causes a tape mark to be written on the nearest 1024-byte boundary following the end of the data. When a tape containing an end-of-data tape mark is read back, the read will terminate upon encountering the tape mark. Thus, with the use of this option, the checksums generated by the input and output operations are guaranteed to agree.

S Enables specification of buffer size. This option forces the allocation of a block of memory to be used in reading or writing the tape. The size in bytes of the buffer is 1024 times the value specified for *buffersize.* A *buffersize* less than 32 or greater than 512 will cause the program to terminate. If *buffersize* is not specified, *tcio* will attempt to allocate buffer sizes in powers of 2 from 512 down to 64, taking the largest one possible. The primary uses of this option are to allow buffer sizes smaller than 64 Kbytes, and to allow the user to pick a buffer size that is most suitable for his application.

V This option turns off tape verification. It is suggested that this option not be used, for the sake of the integrity of the data output to tape.

m This option writes a tape mark on a tape at the specified block. If a tape is created by some other means than *tcio*, a tape mark in block 0 of the tape will prevent it from being image restored to a disc. Note that *blocknumber* must be specified.

r Releases the tape from the mechanism, unlocking the door.

c Image copy option. Provides the same capability as the push-button save and restore available in the HP 79XX single controller drive. The **save** and **restore** keywords are the same as the labels on those switches. **Save** implies disc to tape; **restore** implies tape to disc. Currently only single controller disc/tape units can be backed up in this way.

C Check read option. Provides a measure of data security not found in the tape verification or check digit options. Check read requires two I/O buffers of the size indicated by *buffersize*, one for writing and one for reading. The data in the first buffer is written to the tape. Then the tape is backspaced and read into the second buffer. The two buffers are then compared. If a difference occurs, *tcio* reports the error and terminates. This option forces no tape verification. Note that this option promotes wear and tear on both the media and the drive, and should only be used when complete assurance of the data's integrity is required.

HARDWARE DEPENDENCIES

In general, tapes which have any tape marks other than in the first or the last block cannot be read successfully.

The **e** option is not supported, and because of the above restriction, tapes which have been written under the **e** option cannot be read successfully.

EXAMPLES

The first example below copies the contents of a directory into an archive; the second restores it:

```
ls | cpio –o | tcio –o /dev/rct
tcio –i /dev/rct | cpio –i
```

The next example copies all files in the current directory (via executing *find*)*to* the tape specified by the device file */dev/rct*; a checksum (option **-d**) is performed to verify that the write to tape was performed correctly; verbose mode (**-v**) is used so that you can see the file names of files being copied; in addition, a buffer size (option **-S**) is specified at 128 memory blocks:

```
find . -print | cpio -o | tcio -odvS 128 /dev/rct
```

The following example copies all the files and directories from the tape (specified by */dev/rct*) to the current directory; the data is transfered through a 128-block buffer. Note that with the *cpio* command, the wildcard character * is inclosed in double quotes "*"; this must be done so that the shell doesn't expand the * to all the files in the current directory--i.e., you want the * to be interpreted as all the files on the tape, not your current directory. Here is the command:

```
tcio -ivS 128 /dev/rct |cpio -icdvu "*"
```

SEE ALSO

cpio(1).

WARNING

To be able to use the save/restore facility, the following two conditions must be met:

your system must be in single-user mode;

you must never have used networking on your system. If networking has been used on your system, you must reboot the system before using the save/restore facility.

Tcio can tie up substantial portions of memory, creating a situation where progress on other processes (including those processes feeding *tcio*) is hindered. If this should occur, it is best to kill

tcio and re-execute using a smaller *buffersize*. This problem is especially acute when using the **C** option, because two buffers are required.

BUGS

If the cartridge drive cannot read the manufacturer's block on the tape, the cartridge is locked in the drive and cannot be extracted without turning off the disc/tape drive. This failure is usually the result of faulty tapes or a dirty drive mechanism.

NAME

tee - pipe fitting

SYNOPSIS

tee [**-i**] [**-a**] [file] ...

HP-UX COMPATIBILITY

Level: HP-UX/NUCLEUS

Origin: System V

Native Language Support:
8-bit data.

DESCRIPTION

Tee transcribes the standard input to the standard output and makes copies in the *files*. The **-i** option ignores interrupts; the **-a** option causes the output to be appended to the *files* rather than overwriting them.

NAME

test, [- condition evaluation command

SYNOPSIS

test expr
[expr]

HP–UX COMPATIBILITY

Level: HP–UX/NUCLEUS

Origin: System III

DESCRIPTION

Test evaluates the expression *expr* and, if its value is true, returns a zero (true) exit status; otherwise, a non–zero (false) exit status is returned; *test* also returns a non–zero exit status if there are no arguments. The following primitives are used to construct *expr*:

-r *file* true if *file* exists and is readable.

-w *file* true if *file* exists and is writable.

-x *file* true if *file* exists and is executable.

-f *file* true if *file* exists and is a regular file.

-d *file* true if *file* exists and is a directory.

-c *file* true if *file* exists and is a character special file.

-b *file* true if *file* exists and is a block special file.

-p *file* true if *file* exists and is a named pipe (fifo).

-u *file* true if *file* exists and its set–user–ID bit is set.

-g *file* true if *file* exists and its set–group–ID bit is set.

-k *file* true if *file* exists and its sticky bit is set.

-s *file* true if *file* exists and has a size greater than zero.

-t [*fildes*] true if the open file whose file descriptor number is *fildes* (1 by default) is associated with a terminal device.

-z *s1* true if the length of string *s1* is zero.

-n *s1* true if the length of the string *s1* is non–zero.

s1 **=** *s2* true if strings *s1* and *s2* are identical.

s1 **!=** *s2* true if strings *s1* and *s2* are *not* identical.

s1 true if *s1* is *not* the null string.

n1 **-eq** *n2* true if the integers *n1* and *n2* are algebraically equal. Any of the comparisons **-ne**, **-gt**, **-ge**, **-lt**, and **-le** may be used in place of **-eq**.

These primaries may be combined with the following operators:

! unary negation operator.

-a binary *and* operator.

-o binary *or* operator (**-a** has higher precedence than **-o**).

(expr) parentheses for grouping.

Notice that all the operators and flags are separate arguments to *test*. Notice also that parentheses are meaningful to the shell and, therefore, must be escaped.

Test is directly interpreted by the shell.

SEE ALSO

eval(1) (see *sh*(1)), find(1), sh(1).

WARNING

In the second form of the command (i.e., the one that uses [ ], rather than the word *test*), the square brackets must be *delimited* by blanks.

The second form of the command ([ ]) is not recognized by *csh*.

NAME

time - time a command

SYNOPSIS

time command

HP–UX COMPATIBILITY

Level: HP–UX/STANDARD

Origin: System V

DESCRIPTION

The *command* is executed; after it is complete, *time* prints the elapsed time during the command, the time spent in the system, and the time spent in execution of the command. Times are reported in seconds.

The execution time can depend on the performance of the memory in which the program is running.

The times are printed on standard error.

HARDWARE DEPENDENCIES

Series 500:

For those computers with multiple CPU's, the child CPU times listed may be greater than the actual real elapsed time, since CPU time is counted on a per–CPU basis. Thus, if three CPUs are executing, the times listed are obtained by adding the execution times of each CPU.

SEE ALSO

times command in sh(1), timex(1), times(2).

NAME

touch - update access, modification, and/or change times of file

SYNOPSIS

touch [**-amc**] [mmddhhmm[yy]] files

HP–UX COMPATIBILITY

Level: HP–UX/NUCLEUS

Origin: System V

DESCRIPTION

Touch causes the access, modification, and last change times of each argument to be updated. The file name is created if it does not exist. If no time is specified (see *date*(1)) the current time is used. The **-a** and **-m** options cause touch to update only the access or modification times respectively (default is **-am**). The **-c** option silently prevents *touch* from creating the file if it did not previously exist.

The return code from *touch* is the number of files for which the times could not be successfully modified (including files that did not exist and were not created).

SEE ALSO

date(1), utime(2).

NAME

tput - query terminfo database

SYNOPSIS

tput [–**T**type] capname

HP–UX COMPATIBILITY

Level: HP–UX/STANDARD

Origin: System V

DESCRIPTION

Tput uses the *terminfo(5)* database to make terminal–dependent capabilities and information available to the shell. *Tput* outputs a string if the attribute (**cap**ability **name**) is of type string, or an integer if the attribute is of type integer. If the attribute is of type boolean, tput simply sets the exit code (0 for TRUE, 1 for FALSE), and does no output.

–**T***type* indicates the type of terminal. Normally this flag is unnecessary, as the default is taken from the environment variable **$TERM.**

Capname indicates the attribute from the *terminfo* database. See *terminfo(5).*

EXAMPLES

tput clear Echo clear–screen sequence for the current terminal.

tput cols Print the number of columns for the current terminal.

tput –Thp2623 cols
Print the number of columns for the hp2623 terminal.

bold='tput smso' Set shell variable "bold" to stand–out mode sequence for current terminal. This might be followed by a prompt:
echo "${bold}Please type in your name: \c"

tput hc Set exit code to indicate if current terminal is a hardcopy terminal.

FILES

/usr/lib/terminfo/?/*	Terminal descriptor files
/usr/include/term.h	Definition files
/usr/include/curses.h	

DIAGNOSTICS

Tput prints error messages and returns the following error codes on error:

–**1**	Usage error.
–**2**	Bad terminal type.
–**3**	Bad capname.

In addition, if a capname is requested for a terminal that has no value for that capname (e.g., **tput –T450 lines**), –**1** is printed.

SEE ALSO

stty(1), terminfo(5).

NAME

tr - translate characters

SYNOPSIS

tr [**-cds**] [string1 [string2]]

HP–UX COMPATIBILITY

Level: HP–UX/STANDARD

Origin: System V

DESCRIPTION

Tr copies the standard input to the standard output with substitution or deletion of selected characters. Input characters found in *string1* are mapped into the corresponding characters of *string2*. Any combination of the options **-cds** may be used:

-c Complements the set of characters in *string1* with respect to the universe of characters whose ASCII codes are 001 through 377 octal.

-d Deletes all input characters in *string1*.

-s Squeezes all strings of repeated output characters that are in *string2* to single characters.

The following abbreviation conventions may be used to introduce ranges of characters or repeated characters into the strings:

[a-z] Stands for the string of characters whose ASCII codes run from character **a** to character **z**, inclusive.

[a*n] Stands for *n* repetitions of **a**. If the first digit of *n* is **0**, *n* is considered octal; otherwise, *n* is taken to be decimal. A zero or missing *n* is taken to be huge; this facility is useful for padding *string2*.

The escape character \ may be used as in the shell to remove special meaning from any character in a string. In addition, \ followed by 1, 2, or 3 octal digits stands for the character whose ASCII code is given by those digits.

EXAMPLE

The following creates a list of all the words in *file1* one per line in *file2*, where a word is taken to be a maximal string of alphabetics. The strings are quoted to protect the special characters from interpretation by the shell; 012 is the ASCII code for newline.

```
tr -cs "[A-Z][a-z]" "[\012*]" <file1 >file2
```

SEE ALSO

ed(1), sh(1), ascii(7).

BUGS

Will not handle ASCII **NUL** in *string1* or *string2*; always deletes **NUL** from input.

NAME

true, false - provide truth values

SYNOPSIS

true

false

HP–UX COMPATIBILITY

Level: HP–UX/NUCLEUS

Origin: System V

DESCRIPTION

True does nothing, successfully. *False* does nothing, unsuccessfully. They are typically used in input to *sh*(1) such as:

```
while true
do
        command
done
```

SEE ALSO

machid(1), sh(1).

DIAGNOSTICS

True has exit status zero, *false* nonzero.

NAME

tset – terminal dependent initialization

SYNOPSIS

tset [options] [**–m** [ident] [test baudrate] :type] ... [type]

reset ...

HP-UX COMPATIBILITY

Level: HP-UX/STANDARD

Origin: UCB

Tset sets up your terminal when you first log in to an HP-UX system. It does terminal-dependent processing such as setting erase and kill characters, setting or resetting delays, and sending any sequences needed to properly initialize the terminal. It first determines the *type* of terminal involved, then performs the necessary initializations and mode settings. The type of terminal attached to each HP-UX port is specified in the */etc/ttytype* data base. Type names for terminals are stored in the */usr/lib/terminfo/?/** data base. If a port is not hardwired directly to a specific terminal (thus considered a modem connection), it is given an appropriate generic identifier, such as *dialup*.

In the case where no arguments are specified, *tset* simply reads the terminal type out of the environment variable TERM and re-initializes the terminal. The rest of this manual entry concerns itself with mode and environment initialization, typically done once at login, and options used at initialization time to determine the terminal type and set up terminal modes.

When used in a startup script (*.profile* for *sh*(1) users, or *.login* for *csh*(1) users) it is desirable to give information about the type of terminal you will usually use on ports which are not hardwired. These ports are identified in */etc/ttytype* as *dialup* or *plugboard*, etc. To specify what terminal type you usually use on these ports, the **–m** (map) option flag is followed by the appropriate port type identifier, an optional baud rate specification, and the terminal type. (The effect is to "map" from some conditions to a terminal type, that is, to tell *tset*, "If I'm on this kind of port, then I'll probably be on this kind of terminal".) If more than one mapping is specified, the first applicable mapping prevails. A missing port type identifier matches all identifiers. A *baudrate* is specified as with *stty*(1), and is compared with the speed of the diagnostic output (which should be the control terminal). The baud rate *test* may be any combination of **>**, **=**, **<**, **@**, and **!**; **@** is a synonym for **=** and **!** inverts the sense of the test. To avoid problems with metacharacters, it is best to place the entire argument to **–m** within single quotes; users of *csh*(1) must also put a "\" before any "!" used.

Thus,

tset –m 'dialup>300:2622' –m 'dialup:2624' –m 'plugboard:?2623'

causes the terminal type to be set to an HP 2622 if the port in use is a dialup at a speed greater than 300 baud, or to an HP 2624 if the port is otherwise a dialup (i.e. at 300 baud or less). If the *type* finally determined by *tset* begins with a question mark, the user is asked if he or she really wants that type. A null response means to use that type; otherwise, another type can be entered. Thus, in the above case, if the user is on a plugboard port, he or she will be asked whether or not he or she is actually using an HP 2623.

If no mapping applies and a final *type* option, not preceded by a **–m**, is given on the command line, then that type is used. Otherwise, the identifier found in the */etc/ttytype* data base will be taken to be the terminal type. The latter should always be the case for hardwired ports.

It is usually desirable to return the terminal type, as finally determined by *tset*, and information about the terminal's capabilities to a shell's environment. This can be done using the **-s** option. Using the Bourne shell (*sh*(1)), the command

eval `tset –s *options...*`

or using the C shell, *csh*(1):

set noglob; eval `tset –s *options...*`

These commands cause *tset* to generate as output a sequence of shell commands which place the variable TERM in the environment; see *environ*(5).

Once the terminal type is known, *tset* engages in terminal mode setting. This normally involves sending an initialization sequence to the terminal, setting the single character erase (and optionally the full line erase or line-kill) characters, and setting special character delays. Tab and newline expansion are turned off during transmission of the terminal initialization sequence.

On terminals that can backspace but not overstrike (such as a CRT), and when the erase character is the default erase character ("#" on standard systems), the erase character is changed to BACKSPACE (^H).

The options are:

–e*c* sets the erase character to be the named character *c*; *c* defaults to ^H (BACKSPACE). The character *c* can either be typed directly, or entered using the "hat" notation used here (e.g. the "hat" notation for control-H is ^H; in *sh*(1), the ^ character should be escaped (\^)).

–k*c* sets the kill character to *c*. The default *c* is ^X. If *c* is not specified, the kill character will remain unchanged unless the original value of the kill character is null. In this case, the kill character is set to an "at" sign (@).

– report terminal type. Whatever type is decided on is reported. If no other flags are given, the only effect is to write the terminal type on the standard output.

–s generates appropriate commands (depending on your SHELL environment variable) to set TERM.

–I suppresses transmitting terminal initialization strings.

–Q suppresses printing the "Erase set to" and "Kill set to" messages.

–A asks the user for the TERM type.

–S Outputs the string that would be assigned to TERM in the environment rather than generating commands for a shell. In *sh*(1), the following is an alternate way of setting TERM.

```
set -- `tset –S ...`
TERM=$1
```

For compatibility with earlier versions of *tset*, the following flags are accepted, but their use is discouraged:

–r report to the user in addition to other flags.

–E*c* sets the erase character to *c* only if the terminal can backspace. *C* defaults to ^H.

EXAMPLES

These examples all assume the Bourne shell (*sh*(1)). Note that a typical use of *tset* in a *.profile* will also use the **–e** and **–k** options, and often the **–n** or **–Q** options as well. These options have not been included here to keep the examples small.

Assume, for the moment, that you are on an HP 2622. This is suitable for typing by hand but not for a *.profile*, unless you are *always* on a 2622.

export TERM; TERM=`tset – 2622`

Now, you have an HP 2623 at home which you dial up on, but your office terminal is hardwired and known in */etc/ttytype*.

```
export TERM; TERM=`tset - -m dialup:2623`
```

You have a switch which connects everything to everything, making it nearly impossible to key on what port you are coming in on. You use an HP 2622 in your office at 9600 baud, and dial up to switch ports at 1200 baud from home on an HP 2623. Sometimes you use someone else's terminal at work, so you want it to ask you to make sure what terminal type you have at high speeds, but at 1200 baud you are always on a 2623. Note the placement of the question mark, and the quotes to protect the > and ? from interpretation by the shell.

```
export TERM; TERM=`tset - -m 'switch>1200:?2622' -m 'switch<=1200:2623'`
```

All of the above entries will fall back on the terminal type specified in */etc/ttytype* if none of the conditions hold. The following entry is appropriate if you always dial up, always at the same baud rate, on many different kinds of terminals. Your most common terminal is an HP 2622. It always asks you what kind of terminal you are on, defaulting to 2622.

```
export TERM; TERM=`tset - ?2622`
```

If the file */etc/ttytype* is not properly installed and you want to key entirely on the baud rate, the following can be used:

```
export TERM; TERM=`tset - -m '>1200:2624' 2622`
```

FILES

/etc/ttytype	port name to terminal type mapping data base;
/usr/lib/terminfo/?/*	terminal information data base.

VARIABLES

SHELL	if "csh" then generate *csh*(1) commands, otherwise generate *sh*(1) commands.
TERM	the (canonical) terminal name.

SEE ALSO

csh(1), sh(1), stty(1), ttytype(5), terminfo(5), environ(7).

NAME

tsort - topological sort

SYNOPSIS

tsort [file]

HP-UX COMPATIBILITY

Level: HP-UX/STANDARD

Origin: System V

DESCRIPTION

Tsort produces on the standard output a totally ordered list of items consistent with a partial ordering of items mentioned in the input *file*. If no *file* is specified, the standard input is understood.

The input consists of pairs of items (nonempty strings) separated by blanks. Pairs of different items indicate ordering. Pairs of identical items indicate presence, but not ordering.

SEE ALSO

lorder(1).

DIAGNOSTICS

Odd data: there is an odd number of fields in the input file.

BUGS

Uses a quadratic algorithm; not worth fixing for the typical use of ordering a library archive file.

NAME

tty - get the name of the terminal

SYNOPSIS

tty [**-s**]

HP-UX COMPATIBILITY

Level: HP-UX/STANDARD

Origin: System III

DESCRIPTION

Tty prints the path name of the user's terminal. The **-s** option inhibits printing of the terminal path name, allowing one to test just the exit code.

RETURN VALUE

0 if standard input is a terminal,
1 otherwise.
2 if invalid options were specified,

DIAGNOSTICS

"not a tty" if the standard input is not a terminal and **-s** is not specified.

NAME

ul - do underlining

SYNOPSIS

ul [**-i**] [**-t** terminal] [name ...]

HP–UX COMPATIBILITY

Level: HP–UX/STANDARD

Origin: UCB

DESCRIPTION

Ul reads the named files (or standard input if none are given) and translates occurences of underscores to the sequence which indicates underlining for the terminal in use, as specified by the environment variable **TERM**. The **-t** option overrides the terminal kind specified in the environment. The *terminfo*(5) file corresponding to **TERM** is read to determine the appropriate sequences for underlining. If the terminal is incapable of underlining, but is capable of a standout, mode then that is used instead. If the terminal can overstrike, or handles underlining automatically, *ul* degenerates to *cat*(1). If the terminal cannot underline, underlining is ignored.

The **-i** option causes *ul* to indicate underlining on a separate line containing appropriate dashes '-'; this is useful when you want to look at the underlining which is present in an *nroff* output stream on a CRT.

FILES

/usr/lib/terminfo/?/* terminal capability files

SEE ALSO

man(1), nroff(1).

BUGS

Nroff usually outputs a series of backspaces and underlines intermixed with the text to indicate underlining. No attempt is made to optimize the backward motion.

NAME

umask - set file–creation mode mask

SYNOPSIS

umask [ooo]

HP–UX COMPATIBILITY

Level: HP–UX/NUCLEUS

Origin: System III

Remarks: Not supported on the Integral Personal Computer.

DESCRIPTION

The user file–creation mode mask is set to *ooo*. The three octal digits refer to read/write/execute permissions for *owner*, *group*, and *others*, respectively (see *chmod*(2) and *umask*(2)). The value of each specified digit is subtracted from the corresponding "digit" specified by the system for the creation of a file (see *creat*(2)). For example, **umask 022** causes files to be created without write permission for group or other. (files normally created with mode **777** become mode **755**; files created with mode **666** become mode **644**). Existing files are not affected.

If *ooo* is omitted, the current value of the mask is printed with four octal digits. The first digit, a zero, specifies that the output is expressed in octal.

Umask is recognized and executed by the shell.

Note that the file creation mask does not affect the set–user–ID, set–group–ID, or "sticky" bits.

SEE ALSO

chmod(1), sh(1), chmod(2), creat(2), umask(2).

NAME

uname - print name of current HP–UX version

SYNOPSIS

uname [**-snrvmia**]

HP–UX COMPATIBILITY

Level: HP–UX/NUCLEUS

Origin: System V

DESCRIPTION

Uname prints the current system name of the HP–UX system on the standard output file. It is mainly useful to determine which system one is using. The options cause selected information returned by *uname*(2) to be printed:

-s print the system name (default).

-n print the nodename (the nodename may be a name that the system is known by on a communications network). (e.g. *uucp*).

-r print the operating system release.

-v print the operating system version.

-m print the machine hardware name.

-i print the machine identification number.

-a print all the above information.

SEE ALSO

hostname(1), gethostname(2), sethostname(2), uname(2).

BUGS

The **–i** option reports the nodename if and only if the machine has no promid.

NAME

unget - undo a previous get of an SCCS file

SYNOPSIS

unget [**-r**SID] [**-s**] [**-n**] files

HP–UX COMPATIBILITY

Level: HP–UX/STANDARD

Origin: System III

DESCRIPTION

Unget undoes the effect of a **get -e** done prior to creating the intended new delta. If a directory is named, *unget* behaves as though each file in the directory were specified as a named file, except that non–SCCS files and unreadable files are silently ignored. If a name of **-** is given, the standard input is read with each line being taken as the name of an SCCS file to be processed. Refer to *sact*(**1**), which describes how to determine what deltas are currently binding for an s–file.

Keyletter arguments apply independently to each named file.

-r*SID* Uniquely identifies which delta is no longer intended. (This would have been specified by *get* as the "new delta"). The use of this keyletter is necessary only if two or more outstanding *get*s for editing on the same SCCS file were done by the same person (login name). A diagnostic results if the specified *SID* is ambiguous, or if it is necessary and omitted on the command line (see *sact*(1)).

-s Suppresses the printout, on the standard output, of the intended delta's *SID*.

-n Causes the retention of the gotten file which would normally be removed from the current directory.

Note: *unget* can only be executed by the user who did the corresponding **get -e**. If a system administrator needs to *unget* a **get -e** done by another user, he must either use *su*(1) to change into that user, or edit the p–file directly (which can be done either by the s–file owner of the super–user).

FILES

p–file see *delta*(1).

g–file see *delta*(1).

SEE ALSO

delta(1), get(1), help(1), sact(1).

DIAGNOSTICS

Use *help*(1) for explanations.

NAME

uniq - report repeated lines in a file

SYNOPSIS

uniq [**-udc** [+n] [-n]] [input [output]]

HP–UX COMPATIBILITY

Level: HP–UX/STANDARD

Origin: System V

Native Language Support:
8-bit data, customs, messages.

DESCRIPTION

Uniq reads the input file comparing adjacent lines. In the normal case, the second and succeeding copies of repeated lines are removed; the remainder is written on the output file. *Input* and *output* should always be different. Note that repeated lines must be adjacent in order to be found; see *sort*(1). If the **-u** flag is used, just the lines that are not repeated in the original file are output. The **-d** option specifies that one copy of just the repeated lines is to be written. The normal mode output is the union of the **-u** and **-d** mode outputs.

The **-c** option supersedes **-u** and **-d** and generates an output report in default style but with each line preceded by a count of the number of times it occurred.

The *n* arguments specify skipping an initial portion of each line in the comparison:

-*n* The first *n* fields together with any blanks before each are ignored. A field is defined as a string of non–space, non–tab characters separated by tabs and spaces from its neighbors.

+*n* The first *n* characters are ignored. Fields are skipped before characters.

SEE ALSO

comm(1), sort(1).

NAME

units - conversion program

SYNOPSIS

units

HP–UX COMPATIBILITY

Level: HP–UX/STANDARD

Origin: System V

Remarks: Not supported on the Integral Personal Computer.

DESCRIPTION

Units converts quantities expressed in various standard scales to their equivalents in other scales. It works interactively in this fashion:

```
You have: inch
You want: cm
        * 2.540000e+00
        / 3.937008e-01
```

A quantity is specified as a multiplicative combination of units optionally preceded by a numeric multiplier. Powers are indicated by suffixed positive integers, and division by the usual sign:

```
You have: 15 lbs force/in2
You want: atm
        * 1.020689e+00
        / 9.797299e-01
```

Units only does multiplicative scale changes; thus it can convert Kelvin to Rankine, but not Celsius to Fahrenheit. Most familiar units, abbreviations, and metric prefixes are recognized, together with a generous leavening of exotica and a few constants of nature including:

pi	ratio of circumference to diameter
c	speed of light
e	charge on an electron
g	acceleration of gravity
force	same as **g**,
mole	Avogadro's number,
water	pressure head per unit height of water,
au	astronomical unit.

Pound is not recognized as a unit of mass; **lb** is. Compound names are run together, (e.g., **lightyear**). British units that differ from their U.S. counterparts are prefixed thus: **brgallon**. For a complete list of units, type:

```
cat /usr/lib/unittab
```

FILES

/usr/lib/unittab

BUGS

units(1) expects the singular form of each unit to be entered, but will attempt to form the singular by deleting a trailing "s" if the singular form is not found. Thus, some plurals will be correctly recognized, while others will not.

NAME

upm - unpack cpio archives from HP media

SYNOPSIS

upm -E [**cdmtuvx**] pathname chardevice [patterns]

upm -iM [**cdmtuvx**] [patterns] </dev/rmf?

HP-UX COMPATIBILITY

Level: HP-UX/NON-STANDARD

Origin: HP

Remarks: *Upm* is implemented on the Series 500 only.

DESCRIPTION

Upm is similar to *cpio*(1), and is included to enable you to restore files from 88140L/S tape cartridges or 5.25-inch flexible discs more efficiently.

Upm -E (copy in from tape cartridge) extracts all files specified by *patterns* from the file named by *pathname* (assumed to be the product of a previous *cpio -o*). *Patterns* is a series of zero or more blank-separated character strings given in the name-generating notation of *sh*(1). Note that the metacharacters **?**, *****, and [...] match the slash (/) when used in *patterns*. The default *pattern* is '*', which selects all files. *Chardevice* identifies the character special device file describing the volume containing *pathname*. (Note that, if this volume is not the root, it must be mounted at the time *upm* is used, and *pathname* must include the directory name on which the volume is mounted.)

Upm -iM (copy in) extracts all files selected by zero or more of the specified *patterns* (see above for a description of *patterns*). The files are extracted from the standard input, which is redirected from a raw miniature flexible disc device /dev/rmf?. The resulting standard input is assumed to be the product of a previous *cpio -o*.

Any other options specified must be concatenated with the initial *E* or *iM* options. The options have the following meanings:

- **c** read header information which was previously written in ASCII character form for portability;
- **d** directories are to be created as needed;
- **m** retain previous file modification time. This option is ineffective on directories that are being copied;
- **t** print a table of contents of the input; no files are created;
- **u** copy unconditionally (normally, an older file will not replace a newer file with the same name);
- **v** verbose; causes a list of file names to be printed. When used with the **t** option, the table of contents looks like the output of an *ls -l* command (see *ls*(1));
- **x** restore device special files; *mknod*(2) is used to recreate these files, and thus **-Ex** or **-iMx** can only be used by the super-user. Restoring device files onto a different system can be very dangerous. This is intended for backup use;

When the end of a volume is reached, *upm* will prompt the user for the next flexible disc and continue.

The number of blocks reported by *upm* is always in units of 512-byte blocks, regardless of the block size of the initialized media.

SEE ALSO

cpio(1), tcio(1), mknod(2).

WARNING

The **-B** option must not be used when performing raw I/O using the HP 9130K miniature flexible disc drive.

BUGS

Only the super–user can copy special files.

If /dev/tty is not accessible, *upm* issues a complaint, or refuses to work.

The **-Edr** and **-iMdr** options will not make empty directories.

NAME

uucp, uulog, uuname - HP–UX system to HP–UX system copy

SYNOPSIS

uucp [options] source–files destination–file

uulog [options]

uuname [**-l**] [**-v**]

HP–UX COMPATIBILITY

Level: HP–UX/STANDARD

Origin: System V

DESCRIPTION

Uucp.

Uucp copies files named by the *source–file* arguments to the *destination–file* argument. A file name may be a path name on your machine, or may have the form:

system–name!path–name

where *system–name* is taken from a list of system names which *uucp* knows about. The *system–name* may also be a list of names such as

system–name!system–name!...!system–name!path–name

in which case an attempt is made to send the file via the specified route, and only to a destination in PUBDIR (see below). Care should be taken to insure that intermediate nodes in the route are willing to foward information.

The shell metacharacters **?**, ***** and **[...]** appearing in *path–name* will be expanded on the appropriate system.

Path names may be one of:

(1) a full path name;

(2) a path name preceded by ~*user* where *user* is a login name on the specified system and is replaced by that user's login directory;

(3) a path name preceded by ~/*user* where *user* is a login name on the specified system and is replaced by that user's directory under PUBDIR (see FILES);

(4) anything else is prefixed by the current directory.

The local and remote system access to the path name is specified in the USERFILE. If the result is an erroneous path name for the remote system the copy will fail. If the *destination–file* is a directory, the last part of the *source–file* name is used. The accessibility of the file or path name is specified in USERFILE.

Uucp preserves execute permissions across the transmission and gives 0666 read and write permissions (see *chmod*(2)).

The following options are interpreted by *uucp*:

-d Make all necessary directories for the file copy (default).

-f Do not make intermediate directories for the file copy.

-c Use the source file when copying out rather than copying the file to the spool directory (default).

-C Copy the source file to the spool directory immediately and use the copy.

-m*file* Report status of the transfer in *file*. If *file* is omitted, send mail to the requester when the copy is completed.

-n*user* Notify *user* on the remote system that a file was sent.

-e*sys* Send the *uucp* command to system *sys* to be executed there. (Note: this will only be successful if the remote machine allows the *uucp* command to be executed by **/usr/lib/uucp/uuxqt**.)

-g*grade*
Request *grade* as a priority for the work sequencing. Grades are specified in the order A - Z, a - z, with **A** specifying that the work should be done first, and **z** specifying that the work should be done last. All other grades specify a sequence somewhere in between. The default is **n**.

-r Queue job but do not start the file transfer process. By default a file transfer process is started each time uucp is evoked.

-j Control writing of the *uucp* job number to standard output (see below).

Uucp associates a job number with each request. This job number can be used by *uustat* to obtain status or terminate the job.

The environment variable **JOBNO** and the **–j** option are used to control the listing of the *uucp* job number on standard output. If the environment variable **JOBNO** is undefined or set to **OFF**, the job number will not be listed (default). If *uucp* is then invoked with the **–j** option, the job number will be listed. If the environment variable **JOBNO** is set to **ON** and is exported, a job number will be written to standard output each time uucp is invoked. In this case, the **–j** option will supress output of the job number.

Uulog
Uulog queries a summary log of *uucp* and *uux*(1C) transactions in the file **/usr/spool/uucp/LOGFILE.**

The options cause *uulog* to print logging information:

-s*sys* Print information about work involving system *sys*. If *sys* is not specified, then logging information for all systems will be printed.

-u*user* Print information about work done for the specified *user*. If *user* is not specified then logging information for all users will be printed.

Uuname.
Uuname lists the uucp names of known systems. Duplicate lines are not shown, but blank lines are. The **-l** option returns the local system name. The **-v** option will print additional information about each system. A description will be printed for each system that has a line of information in **/usr/lib/uucp/ADMIN**. The format of ADMIN is:

sysname tab *description* tab.

FILES

/usr/spool/uucp	spool directory
/usr/spool/uucppublic	public directory for receiving and sending (PUBDIR)
/usr/lib/uucp/*	other data and program files

SEE ALSO
mail(1), uux(1C), chmod(2).

WARNING
The domain of remotely accessible files can (and for obvious security reasons, usually should) be severely restricted. You will very likely not be able to fetch files by path name; ask a responsible person on the remote system to send them to you. For the same reasons, you will probably not be able to send files to arbitrary path names. As distributed, the remotely accessible files are those whose names begin **/usr/spool/uucppublic** (equivalent to ~**uucp** or just ~). Note that, if */etc/passwd* contains a blank line, a null user entry, or an entry for ~**uucp**, then ~ and ~**uucp** will not expand properly. Because of this, the *uuto* script will not send files to the proper directory.

NOTES

In order to send files that begin with a dot (e.g., .profile) the files must by qualified with a dot. For example: .profile, .prof*, .profil? are correct; whereas *prof*, ?profile are incorrect.

Uucp will not generate a job number for a strictly local transaction.

BUGS

All files received by *uucp* will be owned by *uucp*.
The **-m** option will only work sending files or receiving a single file. Receiving multiple files specified by special shell characters **? * [...]** will not activate the **-m** option.
The **-m** option will not work if all transactions are local or if **uucp** is executed remotely via the **-e** option.
The **-n** option will function only when the source and destination are not on the same machine. Any excess characters are ignored.

If *uulog* is issued with no parameters when a *uucp* process is writing to a temporary logfile, some log information (that information written after the **LOG.*** files are unlinked) may be lost.

Uucp, when used to copy files locally, will create the new file with mode 644 instead of 666.

NAME

uuls - list spooled uucp transactions grouped by transaction

SYNOPSIS

uuls [**-m**] [*directories...*]
uuls -s [**-m**] [*directories...*]
uuls -k [**-m**] [*directories...*]

HP-UX COMPATIBILITY

Level: HP-UX/STANDARD

Origin: HP

Remarks: Not supported on the Integral PC.

DESCRIPTION

This command lists the contents of uucp spool directories (default "/usr/spool/uucp") with the files grouped into three categories:

Transactions:
Each line starts with a transaction control filename and includes the name of each local (same-directory) subfile referenced by the control file (see below). Each is possibly followed by the total size in bytes (**-s** option) or Kbytes (**-k** option) in the transaction (see below). The **-m** (meanings) option replaces the subfile names with nodename, user, and commandline information (see below).

Orphans:
All subfiles not referenced by any control file.

Others:
All other files in the directory (all files not listed under one of the above categories).

Filenames are columnated so there may be more than one file per line. If a transaction has more subfiles than fit on one line, it is followed by continuation lines which are indented further.

The **-s** (size in bytes) and **-k** (Kbytes) options cause the command to follow each transaction in the *Transactions* section with a total size for all stat-able, sendable files in that transaction. This includes "D.*" files only, not "C.*" or "X.*" files. It does include stat-able files outside the spool directory which are indirectly referenced by "C.*" files. Sizes are either in bytes or rounded to the nearest Kbyte (1024 bytes), respectively. A totals line is also added at the end of the *Transactions* section.

The **-m** (meanings) option causes the command to follow "C.*" and "X.*" files with a "*nodename*!*username commandline*" line, instead of subfilenames. For "C" files, one line is printed per remote execution ("D*X*") subfile it references. *Nodename* is truncated at seven characters, *username* at eight, and *commandline* at however much fits on one line.

If **-m** is given, for each "C" file with no remote execution files, the command instead shows the meaning of the "C" file itself on one or more lines. Each line consists of a username, then "R" (receive) or "S" (send), then the name of the file to be transferred. See below for details.

Filenames are listed in alphabetical order within each section, except that the first section is only sorted by the control filename. Every file in the directory except "**.**" and "**..**" appears exactly once in the entire list, unless **-m** is used.

Details
Transaction files are those whose names start with "C." or "X.". Subfilenames, which usually start with "D.", are gleaned from control file lines, at most one per line, from blank-separated fields, as follows:

C.*: R <remotefrom> <localto> <user> -<options>
C.*: S <localfrom> <remoteto> <user> -<options> <subfile> <mode>
X.*: F <subfile>

Lines that don't begin with the appropriate character ('R', 'S', or 'F') are ignored.

In the "R" (receive) case, <remotefrom> is used to print the "C"–file meaning, and its transaction size is taken as zero (unknown).

In the "S" (send) case, if <subfile> is "D.0", <localfrom> is a file not in the spool directory, resulting from a typical **uucp** call without the **-C** (copy) option. In this case <localfrom> is used for the transaction size, if stat–able, and to print the "C"–file meaning.

uucp -C and **uux** both set <subfile> to a true (spooled) subfile name.

Orphan files are those whose names start with "D." and which are not referenced by any control files.

This algorithm extracts from control files the names of all subfiles which should exist in the spool directory when the transaction is not being actively processed. It is not unusual to see "missing subfiles" and "orphans" if you **uuls** a spool directory while **uucico**, **uucp**, **uux**, or **uuxqt** is active.

Meanings information is gotten by reading each "D*X*" subfile referenced by each "C.*" file, and by reading "X*X*" files. *Nodename*!*username* is taken from the last line in the file which is of the form:

U <username> <nodename>

Likewise, *commandline* is taken from the last line of the form:

C <commandline>

If a subfile name is referenced more than once, references after the first show the subfile as missing. If a subfile name appears in a (corrupt) directory more than once, the name is only found once, but then it is listed again under *Orphans*.

HARDWARE DEPENDENCIES

Uuls uses *directory*(3x), but it truncates filenames to 14 characters. This should cause no problems, even if longer names are allowed, due to the **uucp** file naming conventions.

SEE ALSO

mail(1), uucp(1), uuto(1), uux(1), uuxqt(1), stat(2).

DIAGNOSTICS

The program writes an appropriate message to standard error if it has any problems dealing with a specified file (directory), including failure to get heap space. It always returns zero as its exit value.

If a control file is unopenable (wrong permissions or it disappeared while **uuls** was running), its name is preceded by a "*" and the size of the transaction is zero. If a subfile is missing (filename not found in the directory being listed) or un–stat–able (if required for **-s** or **-k**), its name is preceded by a "*" and it contributes zero bytes to the size of the transaction.

If **-m** is specified and a "D*X*" file is missing or unreadable, its name is given with a "*" prepended, as usual.

BUGS

This command uses *chdir*(2) to change to each directory in turn. If more than one is specified, the second through last directories must be absolute (not relative) pathnames, or the chdir() may fail. (This could be fixed, but might result in the program running slower.)

NAME

uusnap - show snapshot of the UUCP system

SYNOPSIS

uusnap

HP-UX COMPATIBILITY

Level: HP–UX/STANDARD

Origin: UCB

DESCRIPTION

Uusnap displays in tabular format a synopsis of the current UUCP situation. The format of each line is as follows:

site N Cmds N Data N Xqts Message

Where "site" is the name of the site with work, "N" is a count of each of the three possible types of work (command, data, or remote execute), and "Message" is the current status message for that site as found in the STST file.

Included in "Message" may be the time left before UUCP can re–try the call, and the count of the number of times that UUCP has tried to reach the site. The process ID of a UUCICO or other demon may also be shown if it is in a "Talking of LOCKED" state.

SEE ALSO

uucp(1C), *Serial Network Communications Guide*.

NAME

uustat - uucp status inquiry and job control

SYNOPSIS

uustat [options]

HP–UX COMPATIBILITY

Level: HP–UX/STANDARD

Origin: System V

DESCRIPTION

Uustat will display the status of, or cancel, previously specified *uucp* commands, or provide general status on *uucp* connections to other systems. At most one of the following options may be specified:

-j*jobn* Report the status of the *uucp* request *jobn*. If **all** is used for *jobn*, the status of all *uucp* requests is reported. An argument must be supplied otherwise the usage message will be printed and the request will fail.

-k*jobn* Kill the *uucp* request whose job number is *jobn*. The killed *uucp* request must belong to the person issuing the *uustat* command unless one is the super–user.

-r*jobn* Rejuvenate *jobn*. That is, *jobn* is touched so that its modification time is set to the current time. This prevents *uuclean* from deleting the job until the jobs modification time reaches the limit imposed by *uuclean*.

-c*hour* Remove the status entries which are older than *hour* hours. This administrative option can only be initiated by the user **uucp** or the super–user.

-m*mch* Report the status of accessibility of machine *mch*. If *mch* is specified as **all**, then the status of all machines known to the local *uucp* are provided.

-M*mch* This is the same as the *-m* option except that two times are printed. The time that the last status was obtained and the time that the last successful transfer to that system occurred.

If none of the above options are specified, any or all of the following options may appear:

-u*user* Report the status of all *uucp* requests issued by *user*.

-s*sys* Report the status of all *uucp* requests which communicate with remote system *sys*.

-o*hour* Report the status of all *uucp* requests which are older than *hour* hours.

-y*hour* Report the status of all *uucp* requests which are younger than *hour* hours.

-O Report the *uucp* status using the octal status codes listed below. If this option is not specified, the verbose description is printed with each *uucp* request.

-q List the number of jobs and other control files queued for each machine and the time of the oldest and youngest file queued for each machine. If a lock file exists for that system, its date of creation is listed.

When no options are given, *uustat* outputs the status of all *uucp* requests issued by the current user.

For example, the command:

uustat -uhdc -smhtsa -y72

will print the status of all *uucp* requests that were issued by user *hdc* to communicate with system *mhtsa* within the last 72 hours. The meanings of the job request status are:

job–number user remote–system command–time status–time status

where the *status* may be either an octal number or a verbose description. The octal code corresponds to the following description:

OCTAL	STATUS
000001	the copy failed, but the reason cannot be determined
000002	permission to access local file is denied
000004	permission to access remote file is denied
000010	bad *uucp* command is generated
000020	remote system cannot create temporary file
000040	cannot copy to remote directory
000100	cannot copy to local directory
000200	local system cannot create temporary file
000400	cannot execute *uucp*
001000	copy (partially) succeeded
002000	copy finished, job deleted
004000	job is queued
010000	job killed (incomplete)
020000	job killed (complete)

The meanings of the machine accessibility status are:

system–name time status

where *time* is the latest status time and *status* is a self–explanatory description of the machine status.

FILES

/usr/spool/uucp	spool directory
/usr/lib/uucp/L_stat	system status file
/usr/lib/uucp/R_stat	request status file

SEE ALSO

uucp(1C).

NAME

uuto, uupick - public HP–UX–to–HP–UX system file copy

SYNOPSIS

uuto [options] source–files destination
uupick [**-s** system]

HP–UX COMPATIBILITY

Level: HP–UX/STANDARD

Origin: System V

DESCRIPTION

Uuto sends *source–files* to *destination*. *Uuto* uses the *uucp*(1C) facility to send files, while it allows the local system to control the file access. A source–file name is a path name on your machine. Destination has the form:

system**!***user*

where *system* is taken from a list of system names that *uucp* knows about (see *uuname*(1C)). *Logname* is the login name of someone on the specified system.

Two *options* are available:

-p Copy the source file into the spool directory immediately, and send the copy.
-m Send mail to the requester when the copy is complete.

The files (or sub–trees if directories are specified) are sent to PUBDIR on *system*, where PUBDIR is the *uucp* public directory (/usr/spool/uucppublic). Specifically the files are sent to

PUBDIR/receive/*user*/*mysystem*/files.

The recipient is notified by *mail*(1) of the arrival of files.

Uupick accepts or rejects the files transmitted to the recipient. Specifically, *uupick* searches PUBDIR for files destined for the user. For each entry (file or directory) found, the following message is printed on the standard output:

from *system***:** [file *file–name*] [dir *dirname*] **?**

Uupick then reads a line from the standard input to determine the disposition of the file:

<new–line>	Go on to next entry.
d	Delete the entry.
m [*dir*]	Move the entry to named directory *dir* (current directory is default). Note that, if the current working directory is desired for *dir*, you should **not** specify any parameter with **m**. A construction like **m.** is erroneous, and results in loss of data.
a [*dir*]	Same as **m** except move all the files sent from *system*.
p	Print the contents of the file.
q	Stop.
EOT (control–d)	Same as **q**.
!*command*	Escape to the shell to do *command*.
*****	Print a command summary.

Uupick invoked with the **-s***system* option will only search the PUBDIR for files sent from *system*.

FILES

PUBDIR/usr/spool/uucppublic public directory

NOTES

In order to send files that begin with a dot (e.g., .profile) the files must by qualified with a dot. For example: .profile, .prof*, .profil? are correct; whereas *prof*, ?profile are incorrect.

SEE ALSO

mail(1), uuclean(1M), uucp(1C), uulog(1C), uuname(1C), uustat(1C), uux(1C).

NAME

uux - HP–UX–to–HP–UX system command execution

SYNOPSIS

uux [options] command–string

HP–UX COMPATIBILITY

Level: HP–UX/STANDARD

Origin: System V

DESCRIPTION

Uux will gather zero or more files from various systems, execute a command on a specified system and then send standard output to a file on a specified system. Note that, for security reasons, many installations will limit the list of commands executable on behalf of an incoming request from *uux*. Many sites will permit little more than the receipt of mail (see *mail*(1)) via *uux*.

The *command–string* is made up of one or more arguments that look like a Shell command line, except that the command and file names may be prefixed by *system–name*!. A null *system–name* is interpreted as the local system.

File names may be one of

(1) a full path name;

(2) a path name preceded by ~*xxx* where *xxx* is a login name on the specified system and is replaced by that user's login directory;

(3) anything else is prefixed by the current directory.

As an example, the command

uux "!diff usg!/usr/dan/f1 pwba!/a4/dan/f1 > !f1.diff"

will get the **f1** files from the "usg" and "pwba" machines, execute a *diff* command and put the results in **f1.diff** in the local directory.

Any special shell characters such as **<>;|** should be quoted either by quoting the entire *command–string*, or quoting the special characters as individual arguments.

Uux will attempt to get all files to the execution system. For files which are output files, the file name must be escaped using parentheses. For example, the command

uux a!uucp b!/usr/file \(c!/usr/file\)

will send a *uucp* command to system "a" to get **/usr/file** from system "b" and send it to system "c".

Uux will notify you if the requested command on the remote system was disallowed. The response comes by remote mail from the remote machine. The amount of mail notification can be reduced with the **-z** option, which notifies the remote system only if the command failed. Notification can be disabled totally with the **-n** option. Executable commands are listed in **/usr/lib/uucp/L.cmds** on the remote system. The format of the **L.cmds** file is:

cmd,machine1,machine2,...

If no machines are specified, then any machine can execute **cmd**. If machines are specified, only the listed machines can execute **cmd**. If the desired command is not listed in **L.sys** then no machine can execute that command.

Redirection of standard input and output is usually restricted to files in PUBDIR. Directories into which redirection is allowed must be specified in **/usr/lib/uucp/USERFILE** by the system administrator. See the *HP–UX System Administrator Guide* for details.

The following *options* are interpreted by *uux*:

- The standard input to *uux* is made the standard input to the *command–string*.

-n Send no notification to user.

-z Send notification only of failures to user.

-m*file* Report status of the transfer in *file*. If *file* is omitted, send mail to the requester when the copy is completed.

-j Control writing of the *uucp* job number to standard output.

-r Queue job but do not start the file transfer process. By default a file transfer process is started each time uux is evoked.

Uux associates a job number with each request. This job number can be used by *uustat* to obtain status or terminate the job.

The environment variable **JOBNO** and the **-j** option are used to control the listing of the *uux* job number on standard output. If the environment variable **JOBNO** is undefined of set to **OFF**, the job number will not be listed (default). If *uuco* is then invoked with the **-j** option, the job number will be listed. If the environment variable **JOBNO** is set to **ON** and is exported, a job number will be written to standard output each time *uux* is invoked. In this case, the **-j** option will suppress output of the job number.

FILES

/usr/spool/uucp	spool directory
/usr/spool/uucppublic	public directory (PUBDIR)
/usr/lib/uucp/*	other data and programs

SEE ALSO

mail(1), uuclean(1M), uucp(1C).

BUGS

Only the first command of a shell pipeline may have a *system–name*!. All other commands are executed on the system of the first command.
The use of the shell metacharacter * will probably not do what you want it to do. The shell tokens << and >> are not implemented.
Any excess characters are ignored.

NAME

val - validate SCCS file

SYNOPSIS

val -
val [**-s**] [**-r**SID] [**-m**name] [**-y**type] files

HP-UX COMPATIBILITY

Level: HP-UX/STANDARD

Origin: System III

DESCRIPTION

Val determines if the specified *file* is an SCCS file meeting the characteristics specified by the optional argument list. Arguments to *val* may appear in any order. The arguments consist of keyletter arguments, which begin with a **-**, and named files.

Val has a special argument, **-**, which causes reading of the standard input until an end-of-file condition is detected. Each line read is independently processed as if it were a command line argument list.

Val generates diagnostic messages on the standard output for each command line and file processed, and also returns a single 8-bit code upon exit as described below.

The keyletter arguments are defined as follows. The effects of any keyletter argument apply independently to each named file on the command line.

-s The presence of this argument silences the diagnostic message normally generated on the standard output for any error that is detected while processing each named file on a given command line.

-r*SID* The argument value *SID* (*S*CCS *ID*entification String) is an SCCS delta number. A check is made to determine if the *SID* is ambiguous (e. g., **r**1 is ambiguous because it physically does not exist but implies 1.1, 1.2, etc., which may exist) or invalid (e. g., **r**1.0 or **r**1.1.0 are invalid because neither case can exist as a valid delta number). If the *SID* is valid and not ambiguous, a check is made to determine if it actually exists.

-m*name* The argument value *name* is compared with the SCCS %M% keyword in *file*.

-y*type* The argument value *type* is compared with the SCCS %Y% keyword in *file*.

The 8-bit code returned by *val* is a disjunction of the possible errors, i. e., can be interpreted as a bit string where (moving from left to right) set bits are interpreted as follows:

bit 0 = missing file argument;
bit 1 = unknown or duplicate keyletter argument;
bit 2 = corrupted SCCS file;
bit 3 = cannot open file or file not SCCS;
bit 4 = *SID* is invalid or ambiguous;
bit 5 = *SID* does not exist;
bit 6 = %Y%, **-y** mismatch;
bit 7 = %M%, **-m** mismatch;

Note that *val* can process two or more files on a given command line and in turn can process multiple command lines (when reading the standard input). In these cases an aggregate code is

returned - a logical **OR** of the codes generated for each command line and file processed.

SEE ALSO

admin(1), delta(1), get(1), help(1), prs(1).

DIAGNOSTICS

Use *help*(1) for explanations.

BUGS

Val can process up to 50 files on a single command line. Any number above 50 will produce a fatal error.

NAME

vi - screen–oriented (visual) display editor based on ex

SYNOPSIS

vi [**-t** *tag*] [**-r** *file*] [**-l**] [**-w***n*] [**-R**] [+*command*] name ...
view [**-t** *tag*] [**-r** *file*] [**-l**] [**-w***n*] [**-R**] [+*command*] name ...
vedit [**-t** *tag*] [**-r** *file*] [**-l**] [**-w***n*] [**-R**] [+*command*] name ...

HP–UX COMPATIBILITY

Level: HP–UX/STANDARD

Origin: System V

Remarks: The decryption facilities provided by this software are under control by the United States Government and cannot be exported without special licenses. These capabilities are considered an HP–UX/OPTIONAL feature, and can be sold only to domestic customers at this time.

DESCRIPTION

Vi (visual) is a display–oriented text editor based on an underlying line editor *ex*(1). It is possible to use the command mode of *ex* from within *vi* and vice–versa.

When using *vi*, changes you make to the file are reflected in what you see on your terminal screen. The position of the cursor on the screen indicates the position within the file. The *Vi Quick Reference* card, the *Introduction to Display Editing with Vi* and the *Ex Reference Manual* provide full details on using *vi*.

INVOCATION

The following invocation options are interpreted by *vi*:

-t *tag* — Edit the file containing the *tag* and position the editor at its definition.

-r*file* — Recover *file* after an editor or system crash. If *file* is not specified a list of all saved files will be printed.

-l — **LISP** mode; indents appropriately for lisp code, the () {} [[and]] commands in *vi* and *open* are modified to have meaning for *lisp* .

-w*n* — Set the default window size to *n*. This is useful when using the editor over a slow speed line.

-R — Read only mode; the **readonly** flag is set, preventing accidental overwriting of the file.

+*command* — The specified *ex* command is interpreted before editing begins.

The *name* argument indicates files to be edited.

The *view* invocation is the same as *vi* except that the **readonly** flag is set.

The *vedit* invocation is intended for beginners. The **report** flag is set to 1, and the **showmode** and **novice** flags are set. These defaults make it easier to get started learning the editor.

"VI MODES"

Command — Normal and initial mode. Other modes return to command mode upon completion. ESC (escape) is used to cancel a partial command.

Input — Entered by **a i A I o O c C s S R**. Arbitrary text may then be entered. Input mode is normally terminated with ESC character, or abnormally with interrupt.

Last line — Reading input for **: / ?** or **!**; terminate with CR to execute, interrupt to cancel.

COMMAND SUMMARY

Sample commands

← **↓** **↑** **→** — arrow keys move the cursor
h j k l — same as arrow keys

i*abc*ESC	insert text *abc*
cw*new*ESC	change word to *new*
ea*s*ESC	pluralize word
x	delete a character
dw	delete a word
dd	delete a line
3dd	delete 3 lines
u	undo previous change
ZZ	exit vi, saving changes
:q!CR	quit, discarding changes
/*text*CR	search for *text*
^U ^D	scroll up or down
:*ex cmd***CR**	any ex or ed command

Counts before vi commands

Numbers may be typed as a prefix to some commands. They are interpreted in one of these ways.

line/column number	**z G |**
scroll amount	**^D ^U**
repeat effect	most of the rest

Interrupting, canceling

ESC	end insert or incomplete cmd
^?	(delete or rubout) interrupts
^L	reprint screen if **^?** scrambles it
^R	reprint screen if ^L is → key

File manipulation

:wCR	write back changes
:qCR	quit
:q!CR	quit, discard changes
:e *name***CR**	edit file *name*
:e!CR	reedit, discard changes
:e + *name***CR**	edit, starting at end
:e +*n***CR**	edit starting at line *n*
:e #CR	edit alternate file
	synonym for **:e #**
:w *name***CR**	write file *name*
:w! *name***CR**	overwrite file *name*
:shCR	run shell, then return
:!*cmd***CR**	run *cmd*, then return
:nCR	edit next file in arglist
:n *args***CR**	specify new arglist
^G	show current file and line
:ta *tag***CR**	to tag file entry *tag*
^]	**:ta**, following word is *tag*

In general, any *ex* or *ed* command (such as *substitute* or *global*) may be typed, preceded by a colon and followed by a **CR**.

Positioning within file

^F	forward screen
^B	backward screen
^D	scroll down half screen
^U	scroll up half screen
G	go to specified line (end default)
/pat	next line matching *pat*
?*pat*	prev line matching *pat*
n	repeat last / or ?
N	reverse last / or ?
/pat/+n	noth line after *pat*
?*pat***?-***n*	noth line before *pat*
]]	next section/function
[[	previous section/function
(	beginning of sentence
)	end of sentence
{	beginning of paragraph
}	end of paragraph
%	find matching ( ) { or }

Adjusting the screen

^L	clear and redraw
^R	retype, eliminate @ lines
zCR	redraw, current at window top
z-CR	... at bottom
z .CR	... at center
*/pat/z-CR**	*pat* line at bottom
z*n***.CR**	use *n* line window
^E	scroll window down 1 line
^Y	scroll window up 1 line

Marking and returning

``	move cursor to previous context
''	... at first non–white in line
m*x*	mark current position with letter *x*
`*x*	move cursor to mark *x*
'*x*	... at first non–white in line

Line positioning

H	top line on screen
L	last line on screen
M	middle line on screen
+	next line, at first non–white
-	previous line, at first non–white
CR	return, same as +
↓ or **j**	next line, same column
↑ or **k**	previous line, same column

Character positioning

^	first non white
0	beginning of line
$	end of line
h or →	forward
l or ←	backwards
^**H**	same as ←
space	same as →
f*x*	find *x* forward
F*x*	**f** backward
t*x*	upto *x* forward
T*x*	back upto *x*
;	repeat last **f F t** or **T**
,	inverse of **;**
|	to specified column
%	find matching ({) or }

Words, sentences, paragraphs

w	word forward
b	back word
e	end of word
)	to next sentence
}	to next paragraph
(	back sentence
{	back paragraph
W	blank delimited word
B	back **W**
E	to end of **W**

Commands for LISP Mode

)	Forward s–expression
}	... but do not stop at atoms
(	Back s–expression
{	... but do not stop at atoms

Corrections during insert

^**H**	erase last character
^**W**	erase last word
erase	your erase, same as ^**H**
kill	your kill, erase input this line
****	quotes ^**H**, your erase and kill
ESC	ends insertion, back to command
^**?**	interrupt, terminates insert
^**D**	backtab over *autoindent*
↑^**D**	kill *autoindent*, save for next
0^**D**	... but at margin next also
^**V**	quote non–printing character

Insert and replace

a	append after cursor
i	insert before cursor
A	append at end of line
I	insert before first non–blank
o	open line below
O	open above
r*x*	replace single char with *x*

R*text***ESC** replace characters

Operators

Operators are followed by a cursor motion, and affect all text that would have been moved over. For example, since **w** moves over a word, **dw** deletes the word that would be moved over. Double the operator, e.g. **dd** to affect whole lines.

d	delete
c	change
y	yank lines to buffer
<	left shift
>	right shift
!	filter through command
=	indent for LISP

Miscellaneous Operations

C	change rest of line (**c$**)
D	delete rest of line (**d$**)
s	substitute chars (**cl**)
S	substitute lines (**cc**)
J	join lines
x	delete characters (**dl**)
X	... before cursor (**dh**)
Y	yank lines (**yy**)

Yank and Put

Put inserts the text most recently deleted or yanked. However, if a buffer is named, the text in that buffer is put instead.

p	put back text after cursor
P	put before cursor
″*x***p**	put from buffer *x*
″*x***y**	yank to buffer *x*
″*x***d**	delete into buffer *x*

Undo, Redo, Retrieve

u	undo last change
U	restore current line
.	repeat last change
″*d***p**	retrieve *d*'th last delete

AUTHOR

Vi and *ex* were developed by The University of California, Berkeley California, Computer Science Division, Department of Electrical Engineering and Computer Science.

SEE ALSO

ex (1).

Vi tutorial in *HP-UX Concepts and Tutorials.*

WARNINGS AND BUGS

Software tabs using ^**T** work only immediately after the *autoindent.*

Left and right shifts on intelligent terminals do not make use of insert and delete character operations in the terminal.

NAME

vis, inv - make unprintable characters in a file visible or invisible

SYNOPSIS

vis [**-n**] [**-s**] [**-u**] [**-w**] file ...

HP–UX COMPATIBILITY

Level: HP–UX/STANDARD

Origin: HP

Native Language Support:
8-bit and 16-bit data, customs, messages.

DESCRIPTION

Vis reads characters from each *file* in sequence and writes them to the standard output, converting those which are not printable into a visible form. *Inv* performs the inverse function, reading printable characters from each *file* and writing them, returned if appropriate to non–printable form, to standard out.

Non–printable characters are represented using C–like escape conventions:

****	backslash
\b	backspace
\e	escape
\f	form–feed
\n	new–line
\r	carriage return
\s	space
\t	horizontal tab
\v	vertical tab
*****n*	the 8–bit character whose ASCII code is the 3–digit octal number *n*.
\x*n*	the 8–bit character whose ASCII code is the 2–digit hexidecimal number *n*.

Space, horizontal tab, and new line may be treated as printable (and therefore passed unscathed to the output) or non–printable dependent on the options selected. Backslash, although printable, is expanded by *vis*, to a pair of backslashes so that when passed back through *inv*, it can be mapped back to a single backslash.

If no input file is given, or if the argument **-** is encountered, *vis* and *inv* read from the standard input file.

The options are:

-n causes new–line, space, and horizontal tab to be treated as non–printable characters. Thus *vis* expands them visibly as **\n**, **\s**, and **\t**, rather passing them directly to the output. *Inv* discards these character, expecting only the printable expansions. New–line characters are inserted by *vis* every 16 characters so that the output will be in form acceptable for most editors.

-s makes *vis* and *inv* silent about non–existent files, identical input and output, and write errors. Normally, no input file may be the same as the output file unless it is a special file.

-t treats horizontal tab and space as non–printable characters, in the same manner in which **-n** options treats them.

-u causes output to be unbuffered (character–by–character); normally, output is buffered.

-x causes *vis* output to be in hexidecimal form rather than the default octal form. Either form is accepted to *inv* as input.

EXAMPLE

If you encounter a file whose contents are unknown–perhaps binary or text–you can take a quick peek into it without jeopordizing your terminal:

```
vis -n file | head
```

This will safely show the first 160 bytes of the file.

SEE ALSO

cat(1), echo(1), od(1).

WARNING

Command formats such as

```
vis file1 file2 >file1
```

will cause the original data in file1 to be lost.

NAME

wait - await completion of process

SYNOPSIS

wait

HP–UX COMPATIBILITY

Level: HP–UX/NUCLEUS

Origin: System III

DESCRIPTION

Wait until all processes started with **&** have completed, and report on abnormal terminations.

Because the *wait*(2) system call must be executed in the parent process, the shell itself executes *wait*, without creating a new process.

SEE ALSO

sh(1), wait(2)

BUGS

Not all the processes of a 3– or more–stage pipeline are children of the shell, and thus cannot be waited for.

NAME

wc - word, line, and character count

SYNOPSIS

wc [**-lwc**] [names]

HP–UX COMPATIBILITY

Level: HP–UX/STANDARD

Origin: System V

Native Language Support:
8-bit data, customs, messages.

DESCRIPTION

Wc counts lines, words, and characters in the named files, or in the standard input if no *names* appear. It also keeps a total count for all named files. A word is a maximal string of characters delimited by spaces, tabs, or new–lines.

The options **l**, **w**, and **c** may be used in any combination to specify that a subset of lines, words, and characters are to be reported. The default is **-lwc**.

When *names* are specified on the command line, they will be printed along with the counts.

BUGS

Wc counts the number of new–lines to determine the line count. If an ASCII text file has a final line that is not terminated with a new–line character, the count will be off by one.

If there are very many characters, words, and/or lines in an input file, the output may be hard to read. This is because *wc* reserves a fixed column width for each count.

NAME

what - identify files for SCCS information

SYNOPSIS

what files

HP–UX COMPATIBILITY

Level: HP–UX/STANDARD

Origin: System III

DESCRIPTION

What searches the given files for all occurrences of the pattern that *get*(1) substitutes for %Z% (this is @(#) at this printing) and prints out what follows until the first ", >, new–line, \, or null character. For example, if the C program in file **f.c** contains

```
char ident[] = "@(#)identification information";
```

and **f.c** is compiled to yield **f.o** and **a.out**, then the command

```
what f.c f.o a.out
```

will print

```
f.c:
        identification information
f.o:
        identification information
a.out:
        identification information
```

What is intended to be used in conjunction with the SCCS command *get*(1), which automatically inserts identifying information, but it can also be used where the information is inserted manually.

SEE ALSO

get(1), help(1).

SCCS User's Guide in *HP–UX Concepts and Tutorials.*

DIAGNOSTICS

Use *help*(1) for explanations.

BUGS

It's possible that an unintended occurrence of the pattern @(#) could be found just by chance, but this causes no harm in nearly all cases.

NAME

whereis - locate source, binary, and/or manual for program

SYNOPSIS

whereis [**-sbm**] [**-u**] [**-SBM** dir ... **-f**] name ...

HP-UX COMPATIBILITY

Level: HP-UX/STANDARD

Origin: UCB

DESCRIPTION

Whereis locates source/binary and manuals sections for specified files. The supplied names are first stripped of leading pathname components and any (single) trailing extension of the form ".ext", e.g. ".c". Prefixes of "s." resulting from use of SCCS are also dealt with. *Whereis* then attempts to locate the desired program in a list of standard places. If any of the **-b, -s** or **-m** flags are given then *whereis* searches only for binaries, sources or manual sections respectively (or any two thereof). The **-u** flag may be used to search for unusual entries. A file is said to be unusual if it does not have one entry of each requested type. Thus "whereis –m –u *" asks for those files in the current directory which have no documentation.

Finally, the **-B -M** and **-S** flags may be used to change or otherwise limit the places where *whereis* searches. The **-f** file flag is used to terminate the last such directory list and signal the start of file names.

EXAMPLE

The following finds all the files in /usr/bin which are not documented in /usr/man/man1 with source in /usr/src/cmd:

```
cd /usr/bin
whereis -u -M /usr/man/man1 -S /usr/src/cmd -f *
```

FILES

/usr/src/*
/bin, /etc, /lib, /usr/{bin, games, lib}
/usr/man/*
/usr/local/{man/*, bin, games, include, lib}
/usr/contrib/{man/*, bin, games, include, lib}

BUGS

Since the program uses *chdir*(2) to run faster, pathnames given with the **-M -S** and **-B** must be full; i.e. they must begin with a "/".

NAME

who - who is on the system

SYNOPSIS

who [**-uTHlpdbrtasq**] [file]

who am i

who am I

HP-UX COMPATIBILITY

Level: HP-UX/STANDARD

Origin: System V

DESCRIPTION

Who can list the user's name, terminal line, login time, elapsed time since activity occurred on the line, and the process-ID of the command interpreter (shell) for each current HP-UX system user. It examines the **/etc/utmp** file to obtain its information. If *file* is given, that file is examined. Usually, *file* will be **/etc/wtmp**, which contains a history of all the logins since the file was last created.

Who with the **am i** or **am I** option identifies the invoking user.

Except for the default **-s** option, the general format for output entries is:

name [state] line time activity pid [comment] [exit]

With options, *who* can list logins, logoffs, reboots, and changes to the system clock, as well as other processes spawned by the *init* process. These options are:

-u This option lists only those users who are currently logged in. The *name* is the user's login name. The *line* is the name of the line as found in the directory **/dev**. The *time* is the time that the user logged in. The *activity* is the number of hours and minutes since activity last occurred on that particular line. A dot (**.**) indicates that the terminal has seen activity in the last minute and is therefore "current". If more than twenty-four hours have elapsed or the line has not been used since boot time, the entry is marked old**.** This field is useful when trying to determine whether a person is working at the terminal or not. The *pid* is the process-ID of the user's shell. The *comment* is the comment field associated with this line as found in **/etc/inittab** (see *inittab*(4)). This can contain information about where the terminal is located, the telephone number of the dataset, type of terminal if hard-wired, etc.

-T This option is the same as the **-u** option, except that the *state* of the terminal line is printed. The *state* describes whether someone else can write to that terminal. A **+** appears if the terminal is writable by anyone; a **-** appears if it is not. **Root** can write to all lines having a **+** or a **-** in the *state* field. If a bad line is encountered, a **?** is printed.

-l This option lists only those lines on which the system is waiting for someone to login. The *name* field is **LOGIN** in such cases. Other fields are the same as for user entries except that the *state* field does not exist.

-H This option will print column headings above the regular output.

-q This is a quick *who,* displaying only the names and the number of users currently logged on. When this option is used, all other options are ignored.

-p This option lists any other process which is currently active and has been previously spawned by *init*. The *name* field is the name of the program executed by *init* as found in **/etc/inittab**. The *state*, *line*, and *activity* fields have no meaning. The *comment* field shows the *id* field of the line from **/etc/inittab** that spawned this process. See *inittab*(4).

-d This option displays all processes that have expired and not been respawned by *init*. The *exit* field appears for dead processes and contains the termination and exit values (as returned by *wait*(2)), of the dead process. This can be useful in determining why a process terminated.

-b This option indicates the time and date of the last reboot.

-r This option indicates the current *run–level* of the *init* process. The last three fields contain the current state of *init*, the number of times that that state has been previously entered, and the previous state. These fields are updated each time *init* changes to a different run state.

-t This option indicates the last change to the system clock (via the *date*(1) command) by **root**. See *su*(1).

-a This option processes **/etc/utmp** or the named *file* with all options turned on.

-s This option is the default and lists only the *name*, *line*, and *time* fields.

FILES

/etc/utmp
/etc/wtmp
/etc/inittab

SEE ALSO

date(1), login(1), mesg(1), su(1), init(1M), wait(2), inittab(5), utmp(5).

NAME

whoami - print effective current user id

SYNOPSIS

whoami

HP–UX COMPATIBILITY

Level: HP–UX/STANDARD

Origin: UCB

DESCRIPTION

Whoami prints who you are. It works even if you are su'd, while 'who am i' does not since it uses /etc/utmp.

FILES

/etc/passwd Name data base

SEE ALSO

who (1)

NAME

write - interactively write (talk) to another user

SYNOPSIS

write user [line]

HP–UX COMPATIBILITY

Level: HP–UX/STANDARD

Origin: System V

Native Language Support:
8-bit data, customs, messages.

Remarks: Not supported on the Integral Personal Computer.

DESCRIPTION

Write copies lines from your terminal to that of another user. When first called, it sends the message:

Message from *yourname* **(tty**??**)** [*date*]...

to the person you want to talk to. When it has successfully completed the connection, it also sends two bells to your own terminal to indicate that what you are typing is being sent.

The recipient of the message should *write* back at this point. Communication continues until an end of file is read from the terminal, an interrupt is sent, or the recipient has executed "mesg n". At that point *write* writes **EOT** on the other terminal and exits.

If you want to write to a user who is logged in more than once, the *line* argument may be used to indicate which line or terminal to send to (e.g., **tty00**); otherwise, the first writable instance of the user found in **/etc/utmp** is assumed and the following message posted:

user is logged on more than one place.
You are connected to *"terminal"*.
Other locations are:
terminal

Permission to write may be denied or granted by use of the *mesg(1)* command. Writing to others is normally allowed by default. Certain commands, in particular *nroff*(1) and *pr*(1) disallow messages in order to prevent interference with their output. However, if the user has super–user permissions, messages can be forced onto a write–inhibited terminal.

If the character **!** is found at the beginning of a line, *write* calls the shell to execute the rest of the line as a command.

The following protocol is suggested for using *write*: when you first *write* to another user, wait for them to *write* back before starting to send. Each person should end a message with a distinctive signal (i.e., **(o)** for "over") so that the other person knows when to reply. The signal **(oo)** (for "over and out") is suggested when conversation is to be terminated.

FILES

/etc/utmp to find user
/bin/sh to execute **!**

SEE ALSO

mail(1), mesg(1), nroff(1), pr(1), sh(1), who(1).

DIAGNOSTICS

"*user is not logged on*" if the person you are trying to *write* to is not logged on.
"*Permission denied*" if the person you are trying to *write* to denies that permission (with *mesg*).
"*Warning: cannot respond, set mesg –y*" if your terminal is set to *mesg n* and the recipient cannot respond to you.

"*Can no longer write to user*" if the recipient has denied permission (*mesg n*) after you had started writing.

NAME

xargs - construct argument list(s) and execute command

SYNOPSIS

xargs [flags] [command [initial–arguments]]

HP–UX COMPATIBILITY

Level: HP–UX/STANDARD

Origin: System V

Remarks: Not supported on the Integral Personal Computer.

DESCRIPTION

Xargs combines the fixed *initial–arguments* with arguments read from standard input to execute the specified *command* one or more times. The number of arguments read for each *command* invocation and the manner in which they are combined are determined by the flags specified.

Command, which may be a shell file, is searched for, using one's **$PATH**. If *command* is omitted, **/bin/echo** is used.

Arguments read in from standard input are defined to be contiguous strings of characters delimited by one or more blanks, tabs, or new–lines; empty lines are always discarded. Blanks and tabs may be embedded as part of an argument if escaped or quoted. Characters enclosed in quotes (single or double) are taken literally, and the delimiting quotes are removed. Outside of quoted strings a backslash (\) will escape the next character.

Each argument list is constructed starting with the *initial–arguments*, followed by some number of arguments read from standard input (Exception: see **-i** flag). Flags **-i**, **-l**, and **-n** determine how arguments are selected for each command invocation. When none of these flags are coded, the *initial–arguments* are followed by arguments read continuously from standard input until an internal buffer is full, and then *command* is executed with the accumulated args. This process is repeated until there are no more args. When there are flag conflicts (e.g., **-l** vs. **-n**), the last flag has precedence. *Flag* values are:

-l*number* *Command* is executed for each non–empty *number* lines of arguments from standard input. The last invocation of *command* will be with fewer lines of arguments if fewer than *number* remain. A line is considered to end with the first new–line *unless* the last character of the line is a blank or a tab; a trailing blank/tab signals continuation through the next non–empty line. If *number* is omitted, 1 is assumed. Option **-x** is forced.

-i*replstr* Insert mode: *command* is executed for each line from standard input, taking the entire line as a single arg, inserting it in *initial–arguments* for each occurrence of *replstr*. A maximum of 5 arguments in *initial–arguments* may each contain one or more instances of *replstr*. Blanks and tabs at the beginning of each line are thrown away. Constructed arguments may not grow larger than 255 characters, and option **-x** is also forced. **{}** is assumed for *replstr* if not specified.

-n*number* Execute *command* using as many standard input arguments as possible, up to *number* arguments maximum. Fewer arguments will be used if their total size is greater than *size* characters, and for the last invocation if there are fewer than *number* arguments remaining. If option **-x** is also coded, each *number* arguments must fit in the *size* limitation, else *xargs* ter–

minates execution.

-t Trace mode: The *command* and each constructed argument list are echoed to file descriptor 2 just prior to their execution.

-p Prompt mode: The user is asked whether to execute *command* each invocation. Trace mode (**-t**) is turned on to print the command instance to be executed, followed by a **?...** prompt. A reply of **y** (optionally followed by anything) will execute the command; anything else, including just a carriage return, skips that particular invocation of *command*.

-x Causes *xargs* to terminate if any argument list would be greater than *size* characters; **-x** is forced by the options **-i** and **-l**. When neither of the options **-i**, **-l**, or **-n** are coded, the total length of all arguments must be within the *size* limit.

-s*size* The maximum total size of each argument list is set to *size* characters; *size* must be a positive integer less than or equal to 470. If **-s** is not coded, 470 is taken as the default. Note that the character count for *size* includes one extra character for each argument and the count of characters in the command name.

-e*eofstr* *Eofstr* is taken as the logical end-of-file string. Underbar (_) is assumed for the logical **EOF** string if **-e** is not coded. The value **-e** with no *eofstr* coded turns off the logical **EOF** string capability (underbar is taken literally). *Xargs* reads standard input until either end-of-file or the logical **EOF** string is encountered.

Xargs will terminate if either it receives a return code of **-1** from, or if it cannot execute, *command*. When *command* is a shell program, it should explicitly *exit* (see *sh*(1)) with an appropriate value to avoid accidentally returning with **-1**.

EXAMPLES

The following will move all files from directory $1 to directory $2, and echo each move command just before doing it:

```
ls $1 | xargs -i -t mv $1/{} $2/{}
```

The following will combine the output of the parenthesized commands onto one line, which is then echoed to the end of file *log*:

```
(logname; date; echo $0 $*) | xargs >>log
```

The user is asked which files in the current directory are to be archived and archives them into *arch* (1.) one at a time, or (2.) many at a time.

```
1. ls | xargs -p -l ar r arch
2. ls | xargs -p -l | xargs ar r arch
```

The following will execute *diff*(1) with successive pairs of arguments originally typed as shell arguments:

```
echo $* | xargs -n2 diff
```

SEE ALSO

sh(1).

DIAGNOSTICS

Self-explanatory.

NAME

yacc - yet another compiler–compiler

SYNOPSIS

yacc [**-vdlt**] grammar

HP–UX COMPATIBILITY

Level: HP–UX/STANDARD

Origin: System V

DESCRIPTION

Yacc converts a context–free grammar into a set of tables for a simple automaton which executes an LR(1) parsing algorithm. The grammar may be ambiguous; specified precedence rules are used to break ambiguities.

The output file, **y.tab.c**, must be compiled by the C compiler to produce a program *yyparse*. This program must be loaded with the lexical analyzer program, *yylex*, as well as *main* and *yyerror*, an error handling routine. These routines must be supplied by the user; *lex*(1) is useful for creating lexical analyzers usable by *yacc*.

If the **-v** flag is given, the file **y.output** is prepared, which contains a description of the parsing tables and a report on conflicts generated by ambiguities in the grammar.

If the **-d** flag is used, the file **y.tab.h** is generated with the **#define** statements that associate the *yacc*–assigned "token codes" with the user–declared "token names". This allows source files other than **y.tab.c** to access the token codes.

If the **-l** flag is given, the code produced in **y.tab.c** will *not* contain any **#line** constructs. This should only be used after the grammar and the associated actions are fully debugged. This is useful when you intend to compile *y.tab.c* with the **–g** option for *cdb*, since #line constructs should not be used in file compiled with **–g**. (See *cc*(1), *cdb*(1)).

Runtime debugging code is always generated in **y.tab.c** under conditional compilation control. By default, this code is not included when **y.tab.c** is compiled. However, when *yacc*'s **-t** option is used, this debugging code will be compiled by default. Independent of whether the **-t** option was used, the runtime debugging code is under the control of **YYDEBUG**, a pre–processor symbol. If **YYDEBUG** has a non–zero value, then the debugging code is included. If its value is zero, then the code will not be included. The size and execution time of a program produced without the runtime debugging code will be smaller and slightly faster.

FILES

y.output	
y.tab.c	
y.tab.h	defines for token names
yacc.tmp,	
yacc.debug, yacc.acts	temporary files
/usr/lib/yaccpar	parser prototype for C programs

SEE ALSO

lex(1), malloc(3X).

LR Parsing by A. V. Aho and S. C. Johnson, Computing Surveys, June, 1974.

YACC - Yet Another Compiler Compiler in *HP–UX: Selected Articles.*

DIAGNOSTICS

The number of reduce–reduce and shift–reduce conflicts is reported on the standard error output; a more detailed report is found in the **y.output** file. Similarly, if some rules are not reachable from the start symbol, this is also reported.

BUGS

Because file names are fixed, at most one *yacc* process can be active in a given directory at a time.

PERMUTED INDEX

[..... test(1)
a64l ..... a64l(3C)
abort ..... abort(3C)
abs ..... abs(3C)
absolute value, floating point ..... floor(3M)
absolute value, integer ..... abs(3C)
access ..... access(2)
access long integer data in machine-independent manner ..... sputl(3X)
access modes, change memory segment ..... memchmd(2)
access terminfo database ..... tput(1)
access utmp file entry ..... getut(3C)
accessing discs, description of blocked/unblocked disc interface ..... disc(4)
accounting commands, miscellaneous ..... acct(1M)
accounting commands, overview ..... acct(1M)
accounting commands, process ..... acctcom(1)
accounting: connect-time ..... acctcon(1M)
accounting, convert binary wtmp records to ASCII ..... fwtmp(1M)
accounting, correct time/date stamps on wtmp records ..... fwtmp(1M)
accounting: daily ..... runacct(1M)
accounting file format ..... acct(5)
accounting files: merge or add total ..... acctmerg(1M)
accounting: generate disc usage data by user ID ..... diskusg(1M)
accounting: process accounting ..... acctprc(1M)
accounting, record login names and times ..... utmp(5)
accounting records command summary ..... acctcms(1M)
accounting: shell procedures ..... acctsh(1M)
acctcms ..... acctcms(1M)
acctcom ..... acctcom(1)
acctcon ..... acctcon(1M)
acctdisk ..... acct(1M)
acctdusg ..... acct(1M)
acctmerg ..... acctmerg(1M)
accton ..... acct(1M)
acctprc ..... acctprc(1M)
acctsh ..... acctsh(1M)
acctwtmp ..... acct(1M)
acos ..... trig(3M)
activity, terminate all current system activity ..... shutdown(1M)
adb ..... adb(1)
add a swap device for interleaved paging/signalling ..... swapon(2)
add backing store devices ..... vson(2)
add or change environment value ..... putenv(3C)
add or merge total accounting files ..... acctmerg(1M)
address space, allocate and free ..... memallc(2)
address space, lock/unlock for process ..... memlck(2)
addresses, get for program ..... end(3C)
adjust ..... adjust(1)
admin ..... admin(1)
advance ..... regexp(7)
advise OS about segment reference patterns ..... memadvise(2)
alarm ..... alarm(2)
alarm clock, set ..... alarm(2)
allocate a block of memory ..... malloc(3C)
allocate and free address space ..... memallc(2)
allocate backing store space to backing store device ..... vsadv(2)
allocate data segment space for process ..... brk(2)

a.out file format, description of ..... a.out(5)
append to an existing operating system ..... oscp(1M)
appointments, reminder service for ..... calendar(1)
ar ..... ar(1)
arc cosine function ..... trig(3M)
arc sine function ..... trig(3M)
arc tangent function ..... trig(3M)
archive, conversion to new format ..... arcv(1)
archive file format, description of ..... ar(5)
archive file format, description of cpio archive file format ..... cpio(5)
archive files on tape ..... tar(1)
archive library, find ordering relation for ..... lorder(1)
archive, table of contents format description ..... ranlib(5)
archives and libraries, create and maintain ..... ar(1)
archives, copy out to media ..... cpio(1)
archives, extract archive files from media ..... cpio(1)
arcv ..... arcv(1)
argument list handling facility, variable ..... varargs(7)
argv, get next option letter from ..... getopt(3C)
array, allocate memory space for ..... malloc(3C)
array, print formatted data into ..... printf(3S)
array, read and format data from ..... scanf(3S)
as ..... as(1)
asa ..... asa(1)
ASA carriage control characters, interpret ..... asa(1)
ascii ..... ascii(7)
ASCII, convert base 64 ASCII to long integer ..... a64l(3C)
ASCII, convert binary wtmp records to ..... fwtmp(1M)
ASCII, convert floating point value to ..... ecvt(3C)
ASCII, convert non-ASCII to ASCII ..... conv(3C)
ASCII, convert to numbers ..... atof(3C)
asctime ..... ctime(3C)
asin ..... trig(3M)
assembler for MC68000 ..... as(1)
assembler/linker executable output file, description of ..... a.out(5)
assembly language, translate ..... atrans(1)
assert ..... assert(3X)
assign buffering to an open file ..... setbuf(3S)
assistance, get for SCCS ..... help(1)
assure sufficient signal stack space ..... sigspace(2)
asynchronous terminal emulation ..... aterm(1)
at ..... at(1)
atan ..... trig(3M)
atan2 ..... trig(3M)
aterm ..... aterm(1)
atoa ..... atof(3C)
atof ..... atof(3C)
atoi ..... atof(3C)
atol ..... atof(3C)
atrans ..... atrans(1)
attributes, change program's internal ..... chatr(1)
automatically release blocked signals and wait for interrupt ..... sigpause(2)
awk ..... awk(1)
backing store devices, add/remove device from those available ..... vson(2)
backing store devices, allocate backing store space to ..... vsadv(2)

backing store usage, advise system about ..... vsadv(2)
backspaces and reverse line-feeds, interpret for nroff(1) ..... col(1)
backup ..... backup(1M)
backup Command Set 80 cartridge tape ..... tcio(1)
backup or archive file system ..... backup(1M)
banner ..... banner(1)
banners, make using large letters ..... banner(1)
base-64 ASCII, convert to long integer ..... a64l(3C)
basename ..... basename(1)
baud rate, settings for terminal ..... tty(4)
bcheckrc ..... brc(1M)
bdiff ..... bdiff(1)
Bell file system consistency check and interactive repair ..... biffsck(1)
Bell file system, construct ..... bifmkfs(1)
Bell file system debugger ..... biffsdb(1)
Bell Interchange Format file utilities ..... bif(5)
Berkeley compatibility for magnetic tape, description of ..... mt(4)
bessel functions ..... bessel(3M)
bfs ..... bfs(1)
BIF directory, list ..... bifls(1)
BIF directory, make ..... bifmkdir(1)
BIF file, change mode of ..... bifchmod(1)
BIF file copy ..... bifcp(1)
BIF files or directories, remove ..... bifrm(1)
bifchmod ..... bifchmod(1)
bifchown ..... bifchown(1)
bifcp ..... bifcp(1)
bifdf ..... bifdf(1)
biffind ..... biffind(1)
biffls ..... biffls(1)
biffsck ..... biffsck(1)
biffsdb ..... biffsdb(1)
bifmkdir ..... bifmkdir(1)
bifmkfs ..... bifmkfs(1)
bifrm ..... bifrm(1)
big file scanner ..... bfs(1)
binary search on a sorted table ..... bsearch(3C)
bit bucket, special file equivalent to ..... null(4)
block of memory, allocate ..... malloc(3C)
block of memory, change size of ..... malloc(3C)
block of memory, deallocate ..... malloc(3C)
block signals ..... sigblock(2)
block size, find for mounted file system ..... ustat(2)
block special file, create ..... mknod(2), mknod(1M)
blocked disc interface, description of ..... disc(4)
blocked signals, release and wait for interrupt ..... sigpause(2)
blocks, find number of free blocks for mounted file system ..... ustat(2)
blocks, report number of free disc blocks ..... df(1M)
boot area, allocate bytes for ..... sdfinit(1M)
boot area, copy OS from one or more SDF boot areas to another ..... oscp(1M)
boot area, set or get current settings for system parameters in ..... uconfig(1M)
brc ..... brc(1M)
break ..... sh(1)
break value, get maximum for process ..... ulimit(2)
break value, set or get ..... brk(2)

break-point debugging, enable for child process ..... ptrace(2)
brk ..... brk(2)
bsearch ..... bsearch(3C)
buffered file I/O package, description of ..... stdio(3S)
buffering, assign to open file ..... setbuf(3S)
buffers, flush those associated with an open file ..... fclose(3S)
byte offset of next I/O operation on file, set ..... fseek(3S)
byte swapping ..... swab(3C)
C compiler ..... cc(1)
C compiler, preprocessor for ..... cpp(1)
C flow graph, generate ..... cflow(1)
C preprocessor ..... cpp(1)
C program checker/verifier ..... lint(1)
C program, error message generator for ..... perror(3C)
C program formatter ..... cb(1)
cache buffers, specify size and number of ..... uconfig(1M)
calendar ..... calendar(1)
call another UNIX/HP-UX system ..... cu(1)
calloc ..... malloc(3C)
captoinfo ..... captoinfo(1M)
carriage control characters, interpret ASA ..... asa(1)
cartridge tape, Command Set 80 utility ..... tcio(1)
cartridge tape initialization ..... mediainit(1)
cartridge tape, perform input/output from/to ..... tcio(1)
cartridge tape, unpack/extract files from Command Set 80 ..... upm(1)
cat ..... cat(1)
cat, compress, uncompress files ..... compact(1)
catman ..... catman(1M)
catread ..... catread(3C)
cb ..... cb(1)
cc ..... cc(1)
ccat ..... compact(1)
cd ..... cd(1), sh(1)
cdc ..... cdc(1)
ceil ..... floor(3M)
certify file system consistency ..... fsck(1M)
certify SDF volume ..... sdfinit(1M)
cflow ..... cflow(1)
change bars, create file containing ..... diffmk(1)
change data segment space allocation ..... brk(2)
change delta commentary of SCCS delta ..... cdc(1)
change file mode ..... chmod(1), chmod(2)
change file owner or group ..... bifchown(1)
change file owner or group ..... chown(1), chown(2)
change group ID of user ..... newgrp(1), sh(1)
change login password ..... passwd(1)
change default login shell ..... chsh(1)
change memory segment access modes ..... memchmd(2)
change mode of a BIF file ..... bifchmod(1)
change or add value to environment ..... putenv(3C)
change or read real-time priority ..... rtprio(2)
change or set real-time priority ..... rtprio(1)
change program's internal attributes ..... chatr(1)
change root directory for a command ..... chroot(1M)
change root directory for duration of command ..... chroot(1), chroot(2)

change SCCS file parameters ..... admin(1)
change size of previously-allocated block of memory ..... malloc(3C)
change system state ..... init(1M)
change to another user ..... su(1)
change to different operating system or version ..... chsys(1M)
change working directory ..... cd(1), sh(1), chdir(2)
character classification ..... ctype(3C)
character conversion, lower-case to upper-case ..... conv(3C)
character conversion, non-ASCII to ASCII ..... conv(3C)
character conversion, upper-case to lower-case ..... conv(3C)
character count ..... wc(1)
character, description of special characters in terminal interface ..... tty(4)
character, push back into input stream ..... ungetc(3S)
character, read from buffered open file ..... getc(3S)
character, search for in string ..... string(3C)
character sets, NLS ..... ascii(7), kana8(7), roman8(7)
character size, settings for terminal ..... tty(4)
character special file, create ..... mknod(2), mknod(1M)
character, write on buffered open file or standard output ..... putc(3S)
characters, count number contained in file ..... wc(1)
characters, process characters from regular expression ..... regexp(7)
characters, translate into other characters ..... tr(1)
chatr ..... chatr(1)
chdir ..... chdir(2)
check C program ..... lint(1)
check file for accessibility ..... access(2)
check file system consistency ..... fsck(1M)
check integrity of OS in SDF boot area(s) ..... osck(1M)
check internal revision numbers of HP-UX files ..... revck(1M)
check password and group files ..... pwck(1M)
checklist, list of file systems to be checked by fsck(1M) ..... checklist(5)
chgrp ..... chown(1)
child process, enable break-point debugging of ..... ptrace(2)
child process, time execution of ..... times(2)
child process, wait for termination of ..... sh(1)
chmod ..... chmod(1), chmod(2)
chown ..... chown(1), chown(2)
chroot ..... chroot(1), chroot(2)
chroot ..... chroot(1M)
chsh ..... chsh(1)
chsys ..... chsys(1M)
classify characters for NLS ..... nl_ctype(3C)
clean up uucp spool directory ..... uuclean(1M)
clear ..... clear(1)
clear error indicator on open file ..... ferror(3S)
clear i-node by zeroing it out ..... clri(1M)
clear terminal screen ..... clear(1)
clear x.25 switched virtual circuit ..... clrsvc(1M)
clearerr ..... ferror(3S)
clock ..... clock(3C)
clock, set/print time and date ..... date(1)
close ..... close(2)
close a file descriptor ..... close(2)
close group file ..... getgrent(3C)
close or flush a stream ..... fclose(3S)

close password file ..... getpwent(3C)
close pipe between process and command ..... popen(3S)
close-on-exec flag, get/set ..... fcntl(2)
clri ..... clri(1M)
clrsvc ..... clrsvc(1M)
cmp ..... cmp(1)
code portability between HP-UX implementations, typedefs for ..... model(5)
code segments, specify maximum number of ..... uconfig(1M)
col ..... col(1)
collating sequence tables, NLS character set ..... col_seq_8(5), col_seq_16(5)
collation, non-ASCII string, used by NLS ..... nl_string(3C)
colon (:) command ..... sh(1)
combine object files into program ..... ld(1)
comm ..... comm(1)
command, create/close pipe between process and command ..... popen(3S)
command, execute from program ..... system(3S)
command, execute on another system ..... uux(1)
command, execute uucp commands on local system ..... uuxqt(1M)
command, execute with different root directory ..... chroot(1), chroot(2)
command interpreter, standard ..... sh(1)
command line options, parse ..... getopt(1)
command, report error information for ..... err(1)
command, run at lower or higher priority ..... nice(1), nice(2)
command, run immune to hangups, logouts, and quits ..... nohup(1)
Command Set 80 Cartridge Tape Utility ..... tcio(1)
command, set environment for ..... env(1)
command substitution ..... sh(1)
command summary: per-process accounting records ..... acctcms(1M)
command, time the execution of ..... time(1)
commands, execute at specified date(s) and time(s) ..... at(1), cron(1M)
commands, install in file system ..... install(1M)
commands, process accounting ..... acctcom(1)
common lines, find after comparing two files ..... comm(1)
common logarithm ..... exp(3M)
communication, establish interactive communication with another UNIX/HP-UX system ..... cu(1)
compact ..... compact(1)
compare two directories ..... dircmp(1)
compare two files ..... bdiff(1), cmp(1), diff(1)
compare two strings ..... string(3C)
compare two versions of SCCS file ..... sccsdiff(1)
compile ..... regexp(7)
compiled term file format ..... term(5)
compiler, C ..... cc(1)
compiler development ..... yacc(1)
compiler, FORTRAN 77 ..... fc(1), f77(1)
compiler, Pascal ..... pc(1)
compiler: terminfo ..... tic(1M)
compiler-compiler ..... yacc(1)
complementary error function and error function ..... erf(3M)
compress and uncompress files, and cat them ..... compact(1)
compress and uncompress files, and cat them ..... compact(1)
concatenate, copy, and/or print files ..... cat(1)
concatenate lines in one or more files ..... paste(1)
concatenate two strings ..... string(3C)
conditional expressions, evaluate and test ..... sh(1), test(1)

config ..... config(1M)
configure an HP-UX system ..... config(1M)
configure LP spooler system ..... mklp(1M)
connect to remote terminal ..... dial(3C)
connect-time accounting ..... acctcon(1M)
constants and functions, math ..... math(7)
construct a Bell file system ..... bifmkfs(1)
construct file system on special file ..... mkfs(1M)
construct new file system ..... newfs(1M)
contents of directory, list ..... ls(1)
context-free grammar, create ..... yacc(1)
continue ..... sh(1)
control characters, interpret ASA carriage ..... asa(1)
control device ..... ioctl(2), stty(2)
control-flow constructs, shell programming language ..... sh(1)
conventional terminal names ..... term(7)
convert archives to new format ..... arcv(1)
convert between 3-byte integers and long integers ..... l3tol(3C)
convert between long and base-64 ASCII ..... a64l(3C)
convert binary wtmp records into ASCII ..... fwtmp(1M)
convert date and time to ASCII ..... ctime(3C)
convert floating point value to ASCII string ..... ecvt(3C)
convert, reblock, translate, and copy a (tape) file ..... dd(1)
convert string to double-precision integer ..... strtod(3C)
convert string to integer ..... strtol(3C)
convert tape file ..... dd(1)
convert termcap description to terminfo description ..... captoinfo(1M)
copy an open file descriptor ..... dup(2), fcntl(2)
copy, concatenate, and/or print files ..... cat(1)
copy files between two systems ..... uucp(1), uuto(1)
copy files out to media ..... cpio(1)
copy files while simultaneously editing them ..... sed(1)
copy line from standard input to standard output ..... line(1)
copy, link, or move files ..... cp(1)
copy operating system from one or more SDF boot areas to another ..... oscp(1M)
copy string ..... string(3C)
copy tape file ..... dd(1)
copy to or from BIF files ..... bifcp(1)
copy to or from LIF files ..... lifcp(1)
core image, examine and/or modify for child process ..... ptrace(2)
core image file, description of ..... core(5)
cos ..... trig(3M)
cosh ..... sinh(3M)
cosine function ..... trig(3M)
cosine, hyperbolic ..... sinh(3M)
cp ..... cp(1)
cpio ..... cpio(1)
cpio archive format, description of ..... cpio(5)
cpio archives, unpack/extract from 5.25″ flexible discs ..... upm(1)
cpio archives, unpack/extract from Command Set 80 cartridge tape ..... upm(1)
cpp ..... cpp(1)
cpset ..... cpset(1M)
CPU type ..... machid(1)
creat ..... creat(2)
create a directory ..... mkdir(1)

create a directory file ..... mkdir(2)
create a name for a temporary file ..... tmpnam(3S)
create a new process ..... fork(2)
create a special file entry ..... mknod(5)
create an interprocess channel ..... pipe(2)
create and open temporary file ..... tmpfile(3S)
create cat files for the manual ..... catman(1M)
create delta (change) for SCCS file ..... delta(1)
create device files ..... mkdev(1M)
create directory, block/character special, fifo, or ordinary file ..... mknod(2), mknod(1M)
create encryption key ..... makekey(1M)
create libraries, archives ..... ar(1)
create link to file ..... link(1M), link(2)
create message catalog file for modification ..... findmsg(1)
create mnttab table ..... setmnt(1M)
create new file, overwrite existing file ..... creat(2)
create new operating system from ordinary files ..... oscp(1M)
create or change parameters of SCCS files ..... admin(1)
create unique file name ..... mktemp(3C)
creation mask, get/set for file ..... sh(1), umask(1), umask(2)
cron ..... cron(1M)
crontab ..... crontab(1)
CRT, facilitate viewing of continuous text on ..... more(1)
CRT, information about graphics devices with ..... graphics(4)
CRT screen handling and optimization routines ..... curses(3X)
crypt ..... crypt(3C)
C-source error messages into a file ..... mkstr(1)
ct ..... ct(4)
ctermid ..... ctermid(3S)
ctime ..... ctime(3C)
cu ..... cu(1)
current directory, print name of ..... pwd(1)
current events, print ..... news(1)
current user id ..... whoami(1)
current user in utmp file, find ..... ttyslot(3C)
current working directory, change ..... cd(1), sh(1), chdir(2)
current working directory pathname ..... getcwd(3C)
current working directory, print name of ..... pwd(1)
curses ..... curses(3X)
cursor handling and optimization routines ..... curses(3X)
cuserid ..... cuserid(3S)
cut ..... cut(1)
cut out selected fields of each line of a file ..... cut(1)
daily accounting ..... runacct(1M)
data access, long integer, machine independent ..... sputl(3X)
data base, relational data base operator ..... join(1)
Data Encryption Standard ..... crypt(3C)
data segment, change space allocation for ..... brk(2)
data segments, specify maximum number of ..... uconfig(1M)
data types, include file defining data types for system code ..... types(7)
database access ..... query(1)
datacomm, accept/reject files received through uucp or uuto ..... uuto(1)
datacomm, copy files between two systems ..... uucp(1), uuto(1)
datacomm, execute command on another system ..... uux(1)
datacomm, list of known system names ..... uucp(1)

datacomm, log of uucp and uux transactions ..... uucp(1)
date ..... date(1)
date and time, convert to ASCII string ..... ctime(3C)
date and time, get more precisely ..... ftime(2)
date, get/set ..... gettimeofday(2)
date, set ..... stime(2)
date, set and/or print ..... date(1)
dates, reminder service for important ..... calendar(1)
daylight ..... ctime(3C)
daylight saving time, time corrected for ..... ctime(3C)
dd ..... dd(1)
deallocate a block of memory ..... malloc(3C)
debug damaged file system ..... fsdb(1M)
debugger ..... adb(1)
debugging, enable break-point debugging for child process ..... ptrace(2)
decompiler: terminfo ..... untic(1M)
delays, settings and controls for terminal output ..... tty(4)
delta ..... delta(1)
delta, add to SCCS file ..... delta(1)
delta, change commentary of SCCS ..... cdc(1)
delta, inform user of any deltas being created for specific SCCS file ..... sact(1)
delta, remove from SCCS file ..... rmdel(1)
demand loadable, set for program ..... chatr(1)
deroff ..... deroff(1)
DES password encryption ..... crypt(3C)
description of environment ..... environ(7)
description of /etc/passwd, pwd.h files ..... passwd(5)
description of group file ..... group(5)
description of magic.h and magic numbers ..... magic(5)
description of OS management commands ..... osmgr(1M)
descriptor, close file ..... close(2)
descriptor, copy/duplicate file ..... dup(2), fcntl(2)
descriptor, get value of file ..... ferror(3S)
device, description of hpib interface to ..... hpib(4)
device driver, select virtual device driver ..... uconfig(1M)
device drivers, list ..... lsdev(1)
device file, create block/character ..... mknod(2), mknod(1M)
device files, create ..... mkdev(1M)
device files, perform functions on ..... ioctl(2), stty(2)
device names, pack/unpack for mknod(2) ..... mknod(5)
device I/O library ..... gpio_*(3I), hpib_*(3I), io_*(3I)
devices, backing store ..... vson(2)
devices, information about those with graphics crt's ..... graphics(4)
devnm ..... devnm(1M)
df ..... df(1M)
diagnostics, add to program ..... assert(3X)
dial ..... dial(3C)
dial out to a remote terminal ..... dial(3C)
dialup security control ..... dialups(5)
diff ..... diff(1)
differences between files, mark ..... diffmk(1)
differential file comparison, 3-way ..... diff3(1)
diffh ..... diff(1)
diffmk ..... diffmk(1)
digitizer, description of hpib interface to ..... hpib(4)

dircmp ..... dircmp(1)
directory, change root for duration of command ..... chroot(1), chroot(2)
directory, change working ..... cd(1), sh(1), chdir(2)
directory clean-up for uucp spool directory ..... uuclean(1M)
directory, compare two ..... dircmp(1)
directory, create ..... mkdir(1), mknod(2)
directory, description of internal SDF format of ..... dir(5)
directory, extract from path name ..... basename(1)
directory, list contents of ..... ls(1)
directory, list contents of LIF ..... lifls(1)
directory, move ..... mvdir(1M)
directory, print name of current working ..... pwd(1)
directory, remove ..... rm(1)
directory, remove ..... rmdir(2)
dirname ..... basename(1)
disc blocks, report number of free ..... df(1M)
disc description file ..... disktab(5)
disc drivers, information about blocked/unblocked interface ..... disc(4)
disc initialization ..... mediainit(1)
disc storage, preallocate ..... prealloc(1)
disc usage accounting by user ID ..... diskusg(1M)
disc usage, summarize ..... du(1)
disc, write current contents of memory to ..... sync(2), sync(1)
diskusg ..... diskusg(1M)
display buffering, specify number of pages of ..... uconfig(1M)
documentation, on-line ..... man(1)
documents, print using mm macros ..... mm(1)
dot (.) command ..... sh(1)
drand48 ..... drand48(3C)
driver, information about blocked/unblocked disc interface ..... disc(4)
drivers, list device ..... lsdev(1)
du ..... du(1)
dump, octal or hexadecimal ..... od(1)
dumpmsg ..... dumpmsg(1)
dup ..... dup(2)
dup2 ..... dup2(2)
duplicate an open file descriptor ..... dup(2), fcntl(2)
duplicate open file descriptor ..... dup2(2)
e ..... ex(1)
echo ..... echo(1)
echo (print) arguments after shell interpretation ..... echo(1)
ecvt ..... ecvt(3C)
ed ..... ed(1)
edata ..... end(3C)
edit ..... ex(1)
editing activity, print for SCCS file ..... sact(1)
editor, stream text ..... sed(1)
editor, text ..... ed(1), ex(1)
editor, visual text ..... vi(1)
effective current user id ..... whoami(1)
effective user/group ID's, get for process ..... getuid(2)
egrep ..... grep(1)
EMS ..... ems(2)
EMS, description of ..... ems(2)
emulation of asynchronous terminal ..... aterm(1)

enable swapping and paging ..... swapon(1M)
encrypt passwords ..... crypt(3C)
encryption key, generate ..... makekey(1M)
end ..... end(3C)
endgrent ..... getgrent(3C)
endpwent ..... getpwent(3C)
env ..... env(1)
environment, description of parameters and usage ..... sh(1), environ(7)
environment, install parameters in ..... sh(1)
environment, print current ..... env(1)
environment, set for duration of one command ..... env(1)
environment, set up at login time ..... profile(5)
environment variable, get value of ..... getenv(3C)
EOF (end-of-file) character, description of ..... tty(4)
EOF, indicate receipt of when reading file ..... ferror(3S)
EOL (end-of-line) character, description of ..... tty(4)
eqn, tbl, nroff, troff constructs, remove from text ..... deroff(1)
erase character, description of ..... tty(4)
erf ..... erf(3M)
erfc ..... erf(3M)
err ..... err(1)
errfile ..... errfile(5)
errinfo ..... errinfo(2)
errinfo, report value for last command failure ..... err(1)
errno ..... errno(2)
errno, report value for last command failure ..... err(1)
ERROR ..... regexp(7)
error function and complementary error function ..... erf(3M)
error handling, mathematical ..... matherr(3M)
error indicator ..... errinfo(2)
error indicator for system calls ..... errno(2)
error indicator, reset status of ..... ferror(3S)
error indicator while reading file ..... ferror(3S)
error information on last command failure ..... err(1)
error logging file for system ..... errfile(5)
error message generator from C programs ..... perror(3C)
etext ..... end(3C)
eval ..... sh(1)
evaluate arguments as an expression ..... expr(1)
ex ..... ex(1)
examine text, facilitate on soft-copy terminals ..... more(1)
exec ..... sh(1), exec(2)
execl ..... exec(2)
execle ..... exec(2)
execlp ..... exec(2)
executable file, extract symbol table (name list) entries from ..... nlist(3C)
executable file, get size of ..... size(1)
executable linker/assembler output file, description of ..... a.out(5)
execute a file in current process ..... exec(2)
execute command at lower or higher priority ..... nice(1), nice(2)
execute command immune to hangups, logouts, and quits ..... nohup(1)
execute command on another system ..... uux(1)
execute command using different root directory ..... chroot(1)
execute commands at specified date(s) and time(s) ..... at(1), cron(1M)
execute commands from file ..... sh(1)

execute new program in existing process ..... sh(1), exec(2)
execute process with real-time priority ..... rtprio(1)
execute HALGOL programs ..... opx25(1M)
execute uucp commands on local system ..... uuxqt(1M)
execute work requests on remote system ..... uucico(1M), uux(1)
execution profile, create for program ..... profil(2), monitor(3C)
execution, suspend process execution for time interval ..... sleep(1), sleep(3C)
execv ..... exec(2)
execve ..... exec(2)
execvp ..... exec(2)
_exit ..... exit(2)
exit ..... sh(1), exit(2)
exit from enclosing for or while loop ..... sh(1)
exp ..... exp(3M)
expand ..... expand(1)
expand tabs to spaces, and vice versa ..... expand(1)
exponent, raise 2 to a power ..... frexp(3C)
exponential function ..... exp(3M)
export ..... sh(1)
expr ..... expr(1)
expreserve ..... ex(1)
expression, evaluate arguments as ..... expr(1)
exrecover ..... ex(1)
Extended Memory System description ..... ems(2)
external symbols, examine execution profile for ..... prof(1)
extract entries from symbol table (name list) of executable file ..... nlist(3C)
extract error messages from C source into a file ..... mkstr(1)
extract files from 5.25″ flexible discs ..... upm(1)
extract files from Command Set 80 cartridge tape archives ..... upm(1)
extract files from media ..... cpio(1)
extract portions of path names ..... basename(1)
f77 ..... fc(1)
f77 ..... see fc(1)
fabs ..... floor(3M)
false ..... true(1)
fc ..... fc(1)
fclose ..... fclose(3S)
fcntl ..... fcntl(2)
fcntl(2), description of requests and arguments for ..... fcntl(7)
fcntl.h, description of ..... fcntl(7)
fcvt ..... ecvt(3C)
fdopen ..... fopen(3S)
feof ..... ferror(3S)
ferror ..... ferror(3S)
fflush ..... fclose(3S)
fgetc ..... getc(3S)
fgets ..... gets(3S)
fgrep ..... grep(1)
fifo special file, create ..... mknod(2), mknod(1M)
file, assign another file name to already open file ..... fopen(3S)
file, assign buffering to open ..... setbuf(3S)
file attributes file, description of ..... fs(5)
file, buffered read from ..... fread(3S)
file, buffered write to ..... fread(3S)
file, change group ID of ..... chown(1), chown(2)

file, change mode of ..... chmod(1), chmod(2)
file, change owner ..... chown(1), chown(2)
file, change permission bits ..... chmod(1), chmod(2)
file, check revision number for ..... revck(1M)
file, close a buffered open file ..... fclose(3S)
file comparison, three-way differential ..... diff3(1)
file control ..... fcntl(2)
file control constants, file containing definitions of ..... fcntl(7)
file, copy LIF in or out ..... lifcp(1)
file, copy to tape while performing certain conversions ..... dd(1)
file, count words, lines, and characters contained therein ..... wc(1)
file, create and open temporary ..... tmpfile(3S)
file, create device/special ..... mkdev(1M)
file, create or overwrite ordinary ..... creat(2)
file, create or remove link to/from ..... link(1M), link(2), unlink(2)
file, create ordinary ..... mknod(2)
file creation mask, set ..... sh(1), umask(1), umask(2)
file, description of buffered I/O ..... stdio(3S)
file, description of password file, /etc/passwd ..... passwd(5)
file, description of SCCS file format ..... sccsfile(5)
file descriptor, assign stream to ..... fopen(3S)
file descriptor, close ..... close(2)
file descriptor, copy/duplicate ..... dup(2), fcntl(2)
file descriptor, create file pointer using ..... fopen(3S)
file descriptor, determine if associated with terminal ..... ttyname(3C)
file descriptor, get value of ..... ferror(3S)
file, determine accessibility of ..... access(2)
file, error logging file for system ..... errfile(5)
file, find and/or remove duplicate lines in ..... uniq(1)
file, find spelling errors in ..... spell(1)
file format, per-process accounting ..... acct(5)
file, generate name for temporary ..... tmpnam(3S)
file, get information about ..... stat(2)
file, get/set status flags for ..... fcntl(2)
file, indicate the occurrence of an error while reading ..... ferror(3S)
file, indicate when EOF is encountered when reading from ..... ferror(3S)
file, locate in file system ..... find(1)
file, move to new position in ..... lseek(2)
file name, create file name vs. i-node list ..... ncheck(1M)
file name, create unique ..... mktemp(3C)
file name, extract from path name ..... basename(1)
file name, find special file for mounted file system on which file lies ..... devnm(1M)
file name, generate for temporary file ..... tmpnam(3S)
file name, generate for terminal ..... ctermid(3S)
file, open for reading or writing ..... open(2)
file, open with assigned buffering ..... fopen(3S)
file owner or group, change ..... bifchown(1)
file pointer, create using file descriptor ..... fopen(3S)
file pointer, move read/write (seek) ..... lseek(2)
file pointer, obtain for file ..... fopen(3S)
file pointer, re-assign to another file ..... fopen(3S)
file, print last part of ..... tail(1)
file, put line length specifications in text files ..... fspec(5)
file, put margin specifications in text files ..... fspec(5)
file, put tab specifications in text files ..... fspec(5)

file, read and execute commands from ..... sh(1)
file, read and format data from ..... scanf(3S)
file, read character from ..... getc(3S)
file, read from ..... read(2)
file, read string from ..... gets(3S)
file, read word from ..... getc(3S)
file, remove ..... rm(1)
file, remove a LIF ..... lifrm(1)
file, remove extra new-line characters from ..... rmnl(1)
file, remove selected fields from each line in ..... cut(1)
file, remove selected table column entries from ..... cut(1)
file, rename LIF ..... lifrename(1)
file, rewind before next I/O operation ..... fseek(3S)
file scanner, big ..... bfs(1)
file, search contents of for specified string(s) ..... grep(1)
file, set/clear set-user-ID, set-group-ID, sticky bits ..... chmod(1), chmod(2)
file size limit, get for process ..... ulimit(2)
file, sort contents of ..... sort(1)
file, split into pieces ..... split(1)
file system, backup file system on cpio archive ..... backup(1M)
file system (Bell) consistency check and interactive repair ..... biffsck(1)
file system (Bell) debugger ..... biffsdb(1)
file system consistency check and interactive repair ..... fsck(1M)
file system, construct on special file ..... mkfs(1M)
file system debugger ..... fsdb(1M)
file system descriptor file entry ..... getfsent(3X)
file system, find special file associated with ..... devnm(1M)
file system hierarchy ..... hier(7)
file system, install commands in ..... install(1M)
file system, list of those to be checked by fsck(1M) ..... checklist(5)
file system, mount or unmount ..... mount(1M), mount(2), umount(2)
file system name, get for mounted ..... ustat(2)
file system pack name, get for mounted ..... ustat(2)
file system shutdown status ..... fsclean(1M)
file system, table of mounted file systems ..... mnttab(5)
file, system's "bit bucket" special file ..... null(4)
file transfer: XMODEM protocol ..... umodem(1M)
file transfers: KERMIT-protocol ..... kermit(1M)
file tree walk ..... ftw(3C)
file, update access/modification/change times of ..... touch(1), utime(2)
file utilities, Bell Interchange Format ..... bif(5)
file, write character onto ..... putc(3S)
file, write formatted data onto ..... printf(3S)
file, write LIF volume header on ..... lifinit(1)
file, write string onto ..... puts(3S)
file, write to ..... write(2)
file, write word onto ..... putc(3S)
file-creation mode mask, get/set ..... umask(1), umask(2)
fileno ..... ferror(3S)
files, archive on tape ..... tar(1)
files, check password and group files ..... pwck(1M)
files, compare two ..... bdiff(1), cmp(1), diff(1)
files, compare two and create change bars ..... diffmk(1)
files, compare two and find lines common to both ..... comm(1)
files, compare two and find lines unique to each ..... comm(1)

files, concatenate two or more ..... cat(1)
files, copy ..... cat(1)
files, copy and simultaneously edit ..... sed(1)
files, copy between two systems ..... uucp(1), uuto(1)
files, copy out to media ..... cpio(1)
files, description of /etc/profile and $HOME/.profile ..... profile(5)
files, extract from media ..... cpio(1)
files, format and print ..... pr(1)
files, merge lines in one or more ..... paste(1)
files, move, link, or copy ..... cp(1)
files, print ..... cat(1)
files, unpack/extract from 5.25″ flexible discs ..... upm(1)
files, unpack/extract from Command Set 80 cartridge tape archives ..... upm(1)
filter reverse line-feeds and backspaces ..... col(1)
find ..... find(1)
find current user slot in utmp file ..... ttyslot(3C)
find duplicate lines in file ..... uniq(1)
find files ..... find(1)
find files in a BIF system ..... biffind(1)
find name of a terminal ..... ttyname(3C)
find strings for inclusion in message catalog ..... findstr(1)
findmsg ..... findmsg(1)
findstr ..... findstr(1)
fix manual pages for faster viewing with man(1) ..... fixman(1)
fixman ..... fisman(1)
flag, get/set close-on-exec ..... fcntl(2)
flags, mapping pwb/V6 UNIX terminal flags into current HP-UX ..... tty(4)
flags, set shell ..... sh(1)
flexible discs, unpack/extract files from ..... upm(1)
floating point number, split into integer and fractional parts ..... frexp(3C)
floating point to ASCII conversion ..... ecvt(3C)
floor ..... floor(3M)
flow graph, C, generate ..... cflow(1)
flush buffers associated with an open file ..... fclose(3S)
fmod ..... floor(3M)
fold long lines for finite-width output device ..... fold(1)
fopen ..... fopen(3S)
for loop, exit from enclosing ..... sh(1)
for loop, resume the next iteration of ..... sh(1)
fork ..... fork(2)
format and print files ..... pr(1)
format C program ..... cb(1)
format, compiled term file ..... term(5)
format data into string ..... printf(3S)
format data on buffered open file ..... printf(3S)
format data on standard output ..... printf(3S)
format, nlist structure ..... nlist(5)
format of an i-node, description of ..... inode(5)
format of a.out file, description of ..... a.out(5)
format of core image file, description of ..... core(5)
format of cpio archive, description of ..... cpio(5)
format of library/archive file, description of ..... ar(5)
format of SCCS file, description of ..... sccsfile(5)
format, privileged values ..... privgrp(5)
format SDF volume ..... sdfinit(1M)

format specifications, put in text file ..... fspec(5)
format tables for nroff or troff ..... tbl(1)
format text ..... nroff(1)
formatted output from varargs argument list ..... vprintf(3S)
formatted output with numbered arguments ..... printmsg(3C)
formatter, text, simple ..... adjust(1)
formatting text with the man macros ..... man(7)
formatting text with the mm macros ..... mm(7)
FORTRAN 77 compiler ..... fc(1), f77(1)
fprintf ..... printf(3S)
fputc ..... putc(3S)
fputs ..... puts(3S)
fread ..... fread(3S)
free ..... malloc(3C)
free blocks, find for mounted file system ..... ustat(2)
free disc blocks, report number of ..... bifdf(1)
free disc blocks, report number of ..... df(1M)
free i-nodes, find for mounted file system ..... ustat(2)
free memory space ..... memallc(2)
freopen ..... fopen(3S)
frexp ..... frexp(3C)
fscanf ..... scanf(3S)
fsck ..... fsck(1M)
fsck ..... fsck(1M)
fsck(1M), list of file systems to be checked by ..... checklist(5)
fsclean ..... fsclean(1M)
fsdb ..... fsdb(1M)
fseek ..... fseek(3S)
fstat ..... stat(2)
fstat(2)/stat(2), description of structure returned by these calls ..... stat(7)
ftell ..... fseek(3S)
ftime ..... ftime(2)
ftw ..... ftw(3C)
functions and constants, math ..... math(7)
fwrite ..... fread(3S)
fwtmp ..... fwtmp(1M)
gamma ..... gamma(3M)
gcvt ..... ecvt(3C)
gencat ..... gencat(1)
general terminal interface ..... termio(4)
generate a formatted message-catalog file ..... gencat(1)
generate C flow graph ..... cflow(1)
generate encryption key ..... makekey(1M)
generate uniformly-distributed pseudo-random numbers ..... drand48(3C)
get ..... get(1)
get date and time more precisely ..... ftime(2)
get entries from symbol table (name list) of executable file ..... nlist(3C)
get file system descriptor file entry ..... getfsent(3X)
get group access list ..... getgroups(2)
get message from a catalog ..... getmsg(3C)
get message queue ..... msgget(2)
get name of current host ..... gethostname(2)
get password file entry ..... getpwent(3C)
get pathname of current working directory ..... getcwd(3C)
get real/effective user, real/effective group IDs ..... getuid(2)

get set of semaphores ..... semget(2)
get shared memory segment ..... shmget(2)
get special attributes for group ..... getprivgrp(1)
get x.25 line ..... getx25(1M)
getc ..... getc(3S)
GETC ..... regexp(7)
getchar ..... getc(3S)
getcwd ..... getcwd(3C)
getegid ..... getuid(2)
getenv ..... getenv(3C)
geteuid ..... getuid(2)
getfsent ..... getfsent(3X)
getgid ..... getuid(2)
getgrent ..... getgrent(3C)
getgrgid ..... getgrent(3C)
getgrnam ..... getgrent(3C)
getgroups ..... getgroups(2)
gethostname ..... gethostname(2)
getitimer ..... getitimer(2)
getlogin ..... getlogin(3C)
getmsg ..... getmsg(3C)
getmsg, insert calls using findstring output ..... insertmsg(1)
getopt ..... getopt(1)
getopt ..... getopt(3C)
getpass ..... getpass(3C)
getpgrp ..... getpid(2)
getpid ..... getpid(2)
getppid ..... getpid(2)
getprivgrp ..... getprivgrp(1), getprivgrp(2), setprivgrp(1M), privgrp(5)
getpw ..... getpw(3C)
getpwent ..... getpwent(3C)
getpwnam ..... getpwent(3C)
getpwuid ..... getpwent(3C)
gets ..... gets(3S)
get/set date and time ..... gettimeofday(2)
get/set special attributes for group ..... getprivgrp(2)
get/set value of interval timer ..... getitimer(2)
gettimeofday ..... gettimeofday(2)
getty ..... getty(1M)
getuid ..... getuid(2)
getut ..... getut(3C)
getw ..... getc(3S)
getx25 ..... getx25(1M)
gmtime ..... ctime(3C)
goto, non-local ..... setjmp(3C)
grammar, create context-free ..... yacc(1)
graphics devices, information for those with crt's ..... graphics(4)
grep ..... grep(1)
group ..... group(5)
group access list, set ..... setgroups(2)
group, change ID of user ..... newgrp(1)
group file, close ..... getgrent(3C)
group file, description of /etc/group ..... group(5)
group file, read one line from ..... getgrent(3C)
group file, rewind ..... getgrent(3C)

group file, search for matching group ID ..... getgrent(3C)
group file, search for matching group name ..... getgrent(3C)
group ID, change for file ..... chown(1), chown(2)
group ID, change for user ..... newgrp(1), sh(1)
group ID, get for process ..... getpid(2)
group ID, print ..... id(1)
group ID, search group file for matching ..... getgrent(3C)
group ID, set ..... setuid(2)
group ID, set for process ..... setpgrp(2)
group memberships, show ..... groups(1)
group name, search group file for matching ..... getgrent(3C)
group/password file checkers ..... pwck(1M)
groups ..... groups(1)
group special attributes, get ..... getprivgrp(1)
group special attributes, set ..... setprivgrp(1M)
grpck ..... pwck(1M)
grp.h ..... group(5)
gsignal ..... ssignal(3C)
gtty ..... stty(2)
handling facility, variable argument list ..... varargs(7)
hangups, run command immune to ..... nohup(1)
hardware name, get ..... uname(1), uname(2)
hardware trap numbers, list of ..... trapno(2)
hash search tables ..... hsearch(3C)
header, write LIF volume on file ..... lifinit(1)
heap size, change for program ..... chatr(1)
help ..... help(1)
help, get for SCCS routines ..... help(1)
hexadecimal, octal dump ..... od(1)
hier ..... hier(7)
hierarchy, file system ..... hier(7)
host name, get ..... gethostname(2)
host name, set ..... sethostname(2)
host system, set/print name of current ..... hostname(1)
hostname ..... hostname(1)
hpib interface, description of ..... hpib(4)
hpnls ..... hpnls(7)
HP-UX implementations, conditional compilation depending on ..... model(5)
HP-UX implementations, definition of constants which identify ..... model(5)
HP-UX machine identification ..... model(5)
HP-UX revision information, get ..... revision(1)
HP-UX version name, get ..... uname(1), uname(2)
hsearch ..... hsearch(3C)
hyperbolic functions ..... sinh(3M)
hypot ..... hypot(3M)
hypotenuse, function for calculating ..... hypot(3M)
id ..... id(1)
ID's, set user and group ..... setuid(2)
init ..... init(1M)
INIT ..... regexp(7)
init(1M), control information for ..... inittab(5)
initgroups ..... initgroups(3C)
initialization of system state and processes ..... init(1M)
initialize group access list ..... initgroups(3C)
initialize hard disc, flexible disc, or cartridge tape media ..... mediainit(1)

initialize SDF volume ..... sdfinit(1M)
initialize terminal type and mode on login ..... tset(1)
inittab ..... inittab(5)
i-node, clear i-node by zeroing it out ..... clri(1M)
i-node, description of i-node format ..... inode(5)
i-node, enable access to i-node for file system repair ..... fsdb(1M)
i-nodes, create file name vs. i-node list ..... ncheck(1M)
i-nodes, find number of free i-nodes in mounted file system ..... ustat(2)
input and format data from buffered open file ..... scanf(3S)
input and format data from standard input ..... scanf(3S)
input and format data from string ..... scanf(3S)
input commands to shell ..... sh(1)
input control, description of input control for terminal ..... tty(4)
input/output between process and command ..... popen(3S)
input/output, description of buffered file ..... stdio(3S)
input/output operation, get current byte offset of ..... fseek(3S)
input/output operation, reposition next ..... fseek(3S)
input/output, output character/word to open file or standard output ..... putc(3S)
input/output, push character back into input stream ..... ungetc(3S)
input/output redirection ..... sh(1)
input/output, write string to open file or standard output ..... puts(3S)
insert calls to getmsg using findstring output ..... insertmsg(1)
install ..... install(1M)
install commands into file system ..... install(1M)
install object files in binary directories ..... cpset(1M)
integer, get largest integer smaller than x ..... floor(3M)
integer, get smallest integer larger than x ..... floor(3M)
integers, convert between 3-byte and long ..... l3tol(3C)
integer trap control ..... intrapoff(3M)
integrity check of operating system in SDF boot area(s) ..... osck(1M)
interactive IMAGE database access ..... query(1)
interactively write (talk) to another user ..... write(1)
interface, description of hpib ..... hpib(4)
interface to blocked/unblocked disc, description of ..... disc(4)
interface to terminal I/O, description of ..... tty(4)
interleave factor, establish for SDF volume ..... sdfinit(1M)
interprocess communication, create ..... pipe(2)
inter-process communication facilities status ..... ipcs(1)
inter-process communication routines ..... stdipc(3C)
interrupt character, description of ..... tty(4)
intrapoff ..... intrapoff(3M)
I/O between process and command ..... popen(3S)
I/O, description of buffered file ..... stdio(3S)
I/O operation, get current byte offset of ..... fseek(3S)
I/O operation, reposition next ..... fseek(3S)
I/O, output character/word to open file or standard output ..... putc(3S)
I/O, push character back into input stream ..... ungetc(3S)
I/O redirection ..... sh(1)
I/O: GPIO routines (device I/O library) ..... gpio_*(3I)
I/O: HP-IB routines (device I/O library) ..... hpib_*(3I)
I/O: I/O routines (device I/O library) ..... io_*(3I)
I/O, write string to open file or standard output ..... puts(3S)
ioctl ..... ioctl(2)
ioctl(2) system calls, description of ..... tty(4)
iomap ..... iomap(4)

ipcrm ..... ipcrm(1)
ipcs ..... ipcs(1)
isalnum ..... ctype(3C)
isalpha ..... ctype(3C)
isascii ..... ctype(3C)
isatty ..... ttyname(3C)
iscntrl ..... ctype(3C)
isdigit ..... ctype(3C)
isgraph ..... ctype(3C)
islower ..... ctype(3C)
isprint ..... ctype(3C)
ispunct ..... ctype(3C)
isspace ..... ctype(3C)
issue identification file ..... issue(5)
isupper ..... ctype(3C)
isxdigit ..... ctype(3C)
j0 ..... bessel(3M)
j1 ..... bessel(3M)
jn ..... bessel(3M)
join ..... join(1)
join, perform join of two data base relations ..... join(1)
kana8 ..... kana8(7)
kermit ..... kermit(1M)
key, generate encryption ..... makekey(1M)
kill ..... kill(1)
kill character, description of ..... tty(4)
killall ..... killall(1M)
l ..... ls(1)
l3tol ..... l3tol(3C)
l64a ..... a64l(3C)
langid ..... langid(7)
langinfo ..... langinfo(3C)
language identification variable ..... langid(7)
language information ..... langinfo(3C)
last-accessed time, update for file ..... touch(1), utime(2)
last-changed time, update for file ..... touch(1)
last-modified time, update for file ..... touch(1), utime(2)
ld ..... ld(1)
ldexp ..... frexp(3C)
leave ..... leave(1)
length of string, get ..... string(3C)
lex ..... lex(1)
lexical analysis of text, generate programs for ..... lex(1)
libraries and archives, create and maintain ..... ar(1)
library file format, description of ..... ar(5)
library file format, description of cpio archive format ..... cpio(5)
library, find ordering relation for object ..... lorder(1)
library, table of contents format description ..... ranlib(5)
LIF directory, list contents of ..... lifls(1)
LIF file, remove ..... lifrm(1)
LIF file, rename ..... lifrename(1)
LIF files, copy in or out ..... lifcp(1)
LIF volume header, write on file ..... lifinit(1)
lifcp ..... lifcp(1)
lifinit ..... lifinit(1)

lifls ..... lifls(1)
lifrename ..... lifrename(1)
lifrm ..... lifrm(1)
line ..... line(1)
line, copy from standard input to standard output ..... line(1)
line count ..... wc(1)
line length, put line length specifications in text files ..... fspec(5)
linear search and update ..... lsearch(3C)
lines, count number contained in file ..... wc(1)
lines, find common lines in two files ..... comm(1)
lines, find unique lines in two files ..... comm(1)
lines, merge in one or more files ..... paste(1)
link ..... link(1M), link(2)
link, copy, or move files ..... cp(1)
link, create to or remove from file ..... link(1M), link(2), unlink(2)
link editor ..... ld(1)
link information utility, object files ..... linkinfo(1)
linker ..... ld(1)
linker/assembler executable output file, description of ..... a.out(5)
linkinfo ..... linkinfo(1)
lint ..... lint(1)
list active processes in system ..... ps(1)
list contents of BIF directories ..... bifls(1)
list contents of directories ..... ls(1)
list contents of LIF directory ..... lifls(1)
list current users on system ..... who(1)
list device drivers ..... lsdev(1)
list file names with associated i-nodes ..... ncheck(1M)
list spooled uucp transactions grouped by transaction ..... uuls(1)
list users and their current processes ..... whodo(1M)
ll ..... ls(1)
ln ..... cp(1)
localtime ..... ctime(3C)
locate files in file system ..... find(1)
locate source, binary, and/or manual for program ..... whereis(1)
lock ..... lock(1)
lock process, text, or data in memory ..... plock(2)
lock terminal ..... lock(1)
lockf ..... lockf(2)
lock/unlock process address space or segment ..... memlck(2)
log ..... exp(3M)
log gamma function ..... gamma(3M)
log results of work requests on remote system ..... uucico(1M)
log10 ..... exp(3M)
logarithm, common ..... exp(3M)
logarithm, natural ..... exp(3M)
logging file for system errors ..... errfile(5)
logging in on HP-UX ..... login(1)
logical block, set number of bytes per logical block ..... sdfinit(1M)
Logical Interchange Format description ..... lif(5)
login ..... login(1)
login, establish baud rate and communication with terminal during ..... getty(1M)
login name, get ..... logname(1), getlogin(3C)
login name, get ASCII string representing ..... cuserid(3S)
login name, print ..... id(1)

login name, record for each user (accounting) ..... utmp(5)
login shell, change default ..... chsh(1)
login time, record for each user (accounting) ..... utmp(5)
logname ..... logname(1)
logouts, run command immune to ..... nohup(1)
long integer, convert to base-64 ASCII ..... a64l(3C)
long integer data access, machine independent ..... sputl(3X)
long integers, convert to/from 3-byte integers ..... l3tol(3C)
longjmp ..... setjmp(3C)
lorder ..... lorder(1)
lower-case to upper-case character conversion ..... conv(3C)
ls ..... ls(1)
lsdev ..... lsdev(1)
lsearch ..... lsearch(3C)
lseek ..... lseek(2)
lsf ..... ls(1)
lsr ..... ls(1)
lsx ..... ls(1)
ltol3 ..... l3tol(3C)
m4 ..... m4(1)
machid ..... machid(1)
machine ID, get ..... uname(1), uname(2)
machine processor type ..... machid(1)
machine-dependent values ..... values(7)
macro processor ..... m4(1)
macros for formatting entries in the HP-UX Reference manual ..... man(7)
macros for formatting text ..... mm(7)
magic numbers, description of ..... magic(5)
magic.h, description of ..... magic(5)
magnetic tape, description of raw interface and controls ..... mt(4)
magnetic tape, manipulate and/or position ..... mt(1)
mail ..... mail(1)
mail, read or send to other users ..... mail(1)
maintain libraries, archives ..... ar(1)
maintain, update, recompile programs ..... make(1)
make ..... make(1)
make a BIF directory ..... bifmkdir(1)
make file system on special file ..... mkfs(1M)
make posters in large letters ..... banner(1)
make unprintable characters in a file visible or invisible ..... vis(1)
makekey ..... makekey(1M)
malloc ..... malloc(3C)
man ..... man(1)
man macros, description of ..... man(7)
manage binary search trees ..... tsearch(3C)
manage hash search tables ..... hsearch(3C)
manipulate wtmp records ..... fwtmp(1M)
mantissa, get from floating point value ..... frexp(3C)
manual, create preformatted manual pages for on-line ..... catman(1M)
manual, on-line ..... man(1)
manual page (on-line), locate for program ..... whereis(1)
map characters into other characters during copy to standard output ..... tr(1)
mapping, physical address ..... iomap(4)
margins, put margin specifications in text files ..... fspec(5)
mark Command Set 80 cartridge tape ..... tcio(1)

mark SDF operating system file as loadable/non-loadable ..... osmark(1M)
mark/unmark volume as HP-UX root volume ..... rootmark(1M)
mask, get/set file-creation ..... sh(1), umask(1), umask(2)
master device information table ..... master(5)
math ..... math(7)
math functions and constants ..... math(7)
mathematical error handling ..... matherr(3M)
matherr ..... matherr(3M)
MC68000 assembler ..... as(1)
mediainit ..... mediainit(1)
memadvise ..... memadvise(2)
memallc ..... memallc(2)
memberships, show group ..... groups(1)
memchmd ..... memchmd(2)
memfree ..... memallc(2)
memlck ..... memlck(2)
memory ..... memory(3C)
memory, allocate a block of ..... malloc(3C)
memory, allocate for array ..... malloc(3C)
memory, change size of previously-allocated block ..... malloc(3C)
memory, deallocate block of ..... malloc(3C)
memory management, inform operating system about segment reference patterns ..... memadvise(2)
memory management, modify segment length ..... memvary(2)
memory operations ..... memory(3C)
memory segment access modes, change ..... memchmd(2)
memory space, allocate and free ..... memallc(2)
memory, write to disc ..... sync(2), sync(1)
memulck ..... memlck(2)
memvary ..... memvary(2)
merge contents of several files ..... sort(1)
merge lines in one or more files ..... paste(1)
merge or add total accounting files ..... acctmerg(1M)
mesg ..... mesg(1)
message catalogs: MPE/RTE ..... catread(3C)
message control operations ..... msgctl(2)
message operations ..... msgop(2)
messages, permit/deny to your terminal ..... mesg(1)
messages, read or send to other users ..... mail(1)
messages, send to all users ..... wall(1M)
messages, send to another user interactively ..... write(1)
mkdev ..... mkdev(1M)
mkdir ..... mkdir(1)
mkdir ..... mkdir(2)
mkfs ..... mkfs(1M)
mklp ..... mklp(1M)
mknod ..... mknod(2), mknod(1M)
mknod.h, description of ..... mknod(5)
mkstr ..... mkstr(1)
mktemp ..... mktemp(3C)
mm ..... mm(1)
mm macros, description of ..... mm(7)
mm macros, print documents formatted with ..... mm(1)
mnttab table, create ..... setmnt(1M)
mnttab.h, description of ..... mnttab(5)
mod function, floating point ..... floor(3M)

mode, change for file ..... chmod(1), chmod(2)
model, Native Language Support ..... hpnls(7)
model.h, description of ..... model(5)
modem ..... modem(4)
modem control special file ..... modem(4)
modf ..... frexp(3C)
modify parameters of SCCS files ..... admin(1)
modify segment length ..... memvary(2)
monitor ..... monitor(3C)
monitor uucp network ..... uusub(1M)
more ..... more(1)
mount ..... mount(1M), mount(2)
mount or unmount file system ..... mount(1M), mount(2), umount(2)
mounted devices, create table of ..... setmnt(1M)
mounted devices, table of those mounted by mount(1M) ..... mnttab(5)
mounted file system, find special file associated with ..... devnm(1M)
mounted file system statistics ..... ustat(2)
move a directory ..... mvdir(1M)
move, link, or copy files ..... cp(1)
move read/write file pointer; seek ..... lseek(2)
move to new working directory ..... cd(1), sh(1), chdir(2)
msgctl ..... msgctl(2)
msgget ..... msgget(2)
msgop ..... msgop(2)
mt ..... mt(1)
multiple line-feeds, remove from output ..... ssp(1)
mv ..... cp(1)
mvdir ..... mvdir(1M)
name, get login ..... logname(1), getlogin(3C)
name list (symbol table), extract entries from executable file's name list ..... nlist(3C)
name list (symbol table), print from object file ..... nm(1)
Native Language Support model ..... hpnls(7)
natural logarithm ..... exp(3M)
ncheck ..... ncheck(1M)
network, monitor uucp activity on ..... uusub(1M)
network special file, create ..... mknod(2), mknod(1M)
new file system ..... newfs(1M)
newfs ..... newfs(1M)
newfs ..... newfs(1M)
newgrp ..... newgrp(1), sh(1)
new-line character, description of ..... tty(4)
new-line characters, remove extras from file ..... rmnl(1)
news ..... news(1)
news, print current events ..... news(1)
nice ..... nice(1), nice(2)
nlist ..... nlist(3C)
nlist structure format ..... nlist(5)
NLS character classification ..... nl_ctype(3C)
NLS character set collating sequence tables ..... col_seq_8, col_seq_16
NLS character sets ..... ascii(7), kana8(7), roman8(7)
NLS model ..... hpnls(7)
NLS native language information ..... langinfo(3C)
NLS non-ASCII string collation ..... nl_string(3C)
NLS translate characters ..... nl_conv(3C)
nl_string ..... nl_string(3C)

nm ..... nm(1)
nodename, get ..... revision(1), uname(1), uname(2)
nodename, set/print name of current ..... hostname(1)
nohup ..... nohup(1)
non-ASCII string collation used by NLS ..... nl_string(3C)
nroff ..... nroff(1)
nroff, format tables for ..... tbl(1)
nroff, interpret output from nroff for printing ..... col(1)
nroff, troff, tbl, eqn constructs, remove from text ..... deroff(1)
numbered-argument print output formatting ..... printmsg(3C)
object code, locate for program ..... whereis(1)
object file, debugger for ..... adb(1)
object file, extract symbol table (name list) entries from ..... nlist(3C)
object file, get size of ..... size(1)
object file link information utility ..... linkinfo(1)
object file, print symbol table (name list) of ..... nm(1)
object file, remove symbol table and relocation bits from ..... strip(1)
object files, combine into program ..... ld(1)
object library, find ordering relation for ..... lorder(1)
octal, hexadecimal dump ..... od(1)
od ..... od(1)
on-line manual command ..... man(1)
on-line manual, create preformatted manual pages for ..... catman(1M)
open ..... open(2)
open a file and assign buffering to it ..... fopen(3S)
open file, assign buffering to ..... setbuf(3S)
open file descriptor, duplicate ..... dup2(2)
open file for reading or writing ..... open(2)
operating system, append to an existing operating system ..... oscp(1M)
operating system, change to different OS or different version of same OS ..... chsys(1M))
operating system, check integrity of OS in SDF boot area(s) ..... osck(1M)
operating system, copy from one or more SDF boot areas to another ..... oscp(1M)
operating system, create new operating system from ordinary files ..... oscp(1M)
operating system management package description ..... osmgr(1M)
operating system, mark as loadable or non-loadable ..... osmark(1M)
operating system, shut down OS with optional re-boot ..... stopsys(1M)
operating system, split into one or more ordinary files ..... oscp(1M)
optarg ..... getopt(3C)
opterr ..... getopt(3C)
optimization routines: CRT screen and cursor control ..... curses(3X)
optind ..... getopt(3C)
option letter, get from argv ..... getopt(3C)
options, parse command line ..... getopt(1)
options, set for terminal ..... stty(1)
options, set shell ..... sh(1)
opx25 ..... opx25(1M)
ordering relation, find for object library or archive file ..... lorder(1)
ordinary file, create ..... mknod(2)
ordinary file, create or overwrite ..... creat(2)
OS, append to an existing operating system ..... oscp(1M)
OS, change to different OS or different version of same OS ..... chsys(1M)
OS, check integrity of operating system in SDF boot area(s) ..... osck(1M)
OS, copy from one or more SDF boot areas to another ..... oscp(1M)
OS, create new operating system from ordinary files ..... oscp(1M)
OS management package description ..... osmgr(1M)

OS, mark as loadable or non-loadable ..... osmark(1M)
OS, shut down operating system with optional re-boot ..... stopsys(1M)
OS, split operating system into one or more ordinary files ..... oscp(1M)
osck ..... osck(1M)
oscp ..... oscp(1M)
osmark ..... osmark(1M)
osmgr ..... osmgr(1M)
output character or word to open file or standard output ..... putc(3S)
output, description of formatted/unformatted output to printer ..... lp(4)
output, description of system handling of terminal output ..... tty(4)
output, print formatted data into string ..... printf(3S)
output, print formatted data on buffered open file ..... printf(3S)
output, print formatted data on standard output ..... printf(3S)
output string to open file or standard output ..... puts(3S)
overlay program onto existing process and execute ..... sh(1), exec(2)
overview of accounting commands ..... acct(1M)
owner, change for file ..... chown(1), chown(2)
page ..... more(1)
page size, set for paged data ..... uconfig(1M)
paged data, set for program ..... chatr(1)
paging and swapping enable ..... swapon(1M)
PAM ..... pam(1)
parameter substitution ..... sh(1)
parameters, environment ..... sh(1), environ(7)
parameters, install in environment ..... sh(1)
parameters, mark as readonly ..... sh(1)
parameters, perform left-shift on positional ..... sh(1)
parameters, set for terminal ..... stty(1)
parameters, set for terminal on login ..... tset(1)
parent process ID, get for process ..... getpid(2)
parity, settings for terminal ..... tty(4)
parse command line options ..... getopt(1)
Pascal compiler ..... pc(1)
passwd ..... passwd(1)
password, change login ..... passwd(1)
password encryption ..... crypt(3C)
password file, close ..... getpwent(3C)
password file, description of ..... passwd(5)
password file, get line containing matching user ID ..... getpw(3C)
password file, output line similar to those contained in ..... putpwent(3C)
password file, read one line from ..... getpwent(3C)
password file, rewind ..... getpwent(3C)
password file, search for matching user ID ..... getpwent(3C)
password file, search for matching user name ..... getpwent(3C)
password, read from /dev/tty or standard input ..... getpass(3C)
password/group file checkers ..... pwck(1M)
paste ..... paste(1)
path name, get for terminal ..... ttyname(3C)
path name, isolate directory name from ..... basename(1)
path name, isolate file name from ..... basename(1)
pattern, find and process within text ..... awk(1)
pattern, search contents of file for ..... grep(1)
pause ..... pause(2)
pause, suspend process for interval ..... sleep(3C)
pc ..... pc(1)

pclose ..... popen(3S)
PEEKC ..... regexp(7)
periodic, automatic sync ..... syncer(1)
permission bits, change for file ..... chmod(1), chmod(2)
per-process accounting file format ..... acct(5)
perror ..... perror(3C)
personal applications manager, a command shell ..... pam(1)
physical address mapping ..... iomap(4)
pipe ..... pipe(2)
pipe, create/close between process and command ..... popen(3S)
pipe, get intermediate data from ..... tee(1)
pipeline, create ..... pipe(2)
pipeline, get intermediate data from ..... tee(1)
place error messages from C source into a file ..... mkstr(1)
plock ..... plock(2)
plotter, description of hpib interface to ..... hpib(4)
popen ..... popen(3S)
port, database listing terminal type connected to each ..... ttytype(5)
portable code between HP-UX implementations, typedefs for ..... model(5)
position magnetic tape ..... mt(1)
positional parameters, perform left-shift on ..... sh(1)
posters, make using large letters ..... banner(1)
pow ..... exp(3M)
power function ..... exp(3M)
powerfail ..... brc(1M)
pr ..... pr(1)
prealloc ..... prealloc(1)
preallocate disc storage ..... prealloc(1)
preprocessor for C compiler ..... cpp(1)
print and format files ..... pr(1)
print and summarize an SCCS file ..... prs(1)
print arguments after shell interpretation ..... echo(1)
print, copy, and/or concatenate files ..... cat(1)
print current SCCS file editing activity ..... sact(1)
print documents formatted with mm macros ..... mm(1)
print effective current user id ..... whoami(1)
print formatted data on standard output, open file, or string ..... printf(3S)
print formatted output from varargs argument list ..... vprintf(3S)
print formatted output with numbered arguments ..... printmsg(3C)
print last part of file ..... tail(1)
print list of users and their current processes ..... whodo(1M)
print name list (symbol table) of object file ..... nm(1)
print name of current working directory ..... pwd(1)
print news items ..... news(1)
print time and date ..... date(1)
print user, group IDs and names ..... id(1)
printer, description of formatted/unformatted output ..... lp(4)
printer, description of hpib interface to ..... hpib(4)
printer options, set ..... slp(1)
printf ..... printf(3S)
printmsg ..... printmsg(3C)
priority, run command at lower or higher ..... nice(1), nice(2)
privileged values format ..... privgrp(5)
procedures: shell procedures for accounting ..... acctsh(1M)
process accounting ..... acctprc(1M)

process accounting commands ..... acctcom(1)
process and system state initialization ..... init(1M)
process, change data segment space allocation for ..... brk(2)
process, change root directory of ..... chroot(1), chroot(2)
process, create a new ..... fork(2)
process, create/close pipe between process and command ..... popen(3S)
process, enable break-point debugging of child process ..... ptrace(2)
process, format of core image of terminated process ..... core(5)
process, get ID, group ID, and parent process ID of ..... getpid(2)
process, get real/effective user and real/effective group ID's for ..... getuid(2)
process, get/set file size limit for ..... ulimit(2)
process group ID, set ..... setpgrp(2)
process, lock/unlock address space or segment ..... memlck(2)
process number, get ..... getpid(2)
process, overlay new program onto existing ..... sh(1), exec(2)
process, print accumulated user and system time elapsed for ..... sh(1)
process, send SIGIOT to ..... abort(3C)
process, send signal to ..... kill(1), kill(2), abort(3C)
process, set group ID for ..... setpgrp(2)
process status, report ..... ps(1)
process, suspend execution for interval of time ..... sleep(1), sleep(3C)
process, suspend until signal ..... pause(2)
process, terminate ..... kill(1), sh(1), exit(2), kill(2), abort(3C)
process, time execution of ..... times(2)
process, wait for completion of ..... sh(1), wait(1), wait(2)
processes, list active ..... ps(1)
processes, send signal to all user processes ..... killall(1M)
processes, specify maximum number of processes per user ..... uconfig(1M)
processes, terminate all user processes ..... shutdown(1M)
processor type ..... machid(1)
prof ..... prof(1)
profil ..... profil(2)
profile, create for program during execution ..... profil(2), monitor(3C)
profile data, display ..... prof(1)
profile files, description of /etc/profile and $HOME/.profile ..... profile(5)
program, add diagnostics to ..... assert(3X)
program, change internal attributes of ..... chatr(1)
program, check/verify C ..... lint(1)
program, create execution profile for ..... profil(2), monitor(3C)
program, create from object files ..... ld(1)
program, debugger for ..... adb(1)
program, execute command from ..... system(3S)
program, force action associated with signal to be taken ..... ssignal(3C)
program, format C ..... cb(1)
program, generate for lexical analysis of text ..... lex(1)
program, get particular addresses associated with ..... end(3C)
program, get size of ..... size(1)
program, locate source, binary, and/or on-line manual page for ..... whereis(1)
program, maintain, update, and recompile ..... make(1)
program, overlay onto existing process and execute ..... sh(1), exec(2)
program, run immune to hangups, logouts, and quits ..... nohup(1)
program, set up signal handling for ..... signal(2), ssignal(3C)
program verification ..... assert(3X)
provide semaphores and record locking on files ..... lockf(2)
provide truth value about your processor type ..... machid(1)

prs ..... prs(1)
ps ..... ps(1)
pseudo-random number generator ..... drand48(3C)
pseudo-random numbers ..... drand48(3C)
pseudo-terminal driver ..... pty(4)
ptrace ..... ptrace(2)
pty ..... pty(4)
public UNIX-to-UNIX file copy ..... uuto(1)
push character back into input stream ..... ungetc(3S)
putc ..... putc(3S)
putchar ..... putc(3S)
putenv ..... putenv(3C)
putpwent ..... putpwent(3C)
puts ..... puts(3S)
putw ..... putc(3S)
pwck ..... pwck(1M)
pwd ..... pwd(1)
pwd.h ..... passwd(5)
Pythagorean theorem function ..... hypot(3M)
qsort ..... qsort(3C)
query ..... query(1)
quit character, description of ..... tty(4)
quits, run command immune to ..... nohup(1)
quoting, as used by the shell ..... sh(1)
rand ..... rand(3C)
random number generator ..... drand48(3C)
random number generator ..... rand(3C)
randomized library/archive, table of contents format description ..... ranlib(5)
ranlib.h, description of ..... ranlib(5)
raw interface to disc, description of ..... disc(4)
raw mode, description of raw mode interface to magnetic tape ..... mt(4)
raw mode, description of raw output to printer ..... lp(4)
rc ..... brc(1M)
read ..... sh(1), read(2)
read and format data from buffered open file ..... scanf(3S)
read and format data from standard input ..... scanf(3S)
read and format data from string ..... scanf(3S)
read character from buffered open file ..... getc(3S)
read error indicator on open file ..... ferror(3S)
read from a file using buffers ..... fread(3S)
read from file ..... read(2)
read from standard input ..... sh(1)
read operation, reposition next ..... fseek(3S)
read password from /dev/tty or standard input ..... getpass(3C)
read text in convenient chunks on soft-copy terminal ..... more(1)
read word from buffered open file ..... getc(3S)
read-ahead, set number of buffers allocated to ..... uconfig(1M)
readonly ..... sh(1)
read/write file pointer, move (seek) ..... lseek(2)
real group ID, get for process ..... getuid(2)
real user ID, get for process ..... getuid(2)
realloc ..... malloc(3C)
real-time priority, change or read ..... rtprio(2)
real-time priority, execute process with ..... rtprio(1)
reblock tape file ..... dd(1)

reboot ..... reboot(1M)
reboot ..... reboot(2)
re-boot operating system after shut-down ..... stopsys(1M)
reboot system ..... reboot(1M)
reboot the system ..... reboot(2)
record locking and semaphores on files ..... lockf(2)
record login names, login times, and tty names for user ..... utmp(5)
regexp.h, description of ..... regexp(7)
regular expression compile and match routines ..... regexp(7)
relational database operator ..... join(1)
release blocked signals and wait for interrupt ..... sigpause(2)
release Command Set 80 cartridge tape ..... tcio(1)
release number, get current ..... revision(1), uname(1), uname(2)
relocation bits, remove from object file ..... strip(1)
remind you when you have to leave ..... leave(1)
remind you when you have to leave ..... leave(1)
reminder service ..... calendar(1)
remote system, execute work requests on ..... uucico(1M), uux(1)
remove a directory file ..... rmdir(2)
remove a LIF file ..... lifrm(1)
remove backing store devices ..... vson(2)
remove BIF files or directories ..... bifrm(1)
remove delta from SCCS file ..... rmdel(1)
remove duplicate lines in file ..... uniq(1)
remove extra new-line characters from file ..... rmnl(1)
remove files or directories ..... rm(1)
remove link to file ..... link(1M), unlink(2)
remove message queue ..... ipcrm(1)
remove multiple line-feeds from output ..... ssp(1)
remove nroff/troff, tbl, and eqn constructs ..... deroff(1)
remove selected fields from each line of a file ..... cut(1)
remove selected table column entries from file ..... cut(1)
remove semaphore set ..... ipcrm(1)
remove shared memory id ..... ipcrm(1)
remove symbol table and relocation bits from object file ..... strip(1)
rename LIF files ..... lifrename(1)
repair file system inconsistencies ..... fsck(1M), fsdb(1M)
report inter-process communication facilities status ..... ipcs(1)
report number of free disc blocks ..... bifdf(1)
report CPU time used ..... clock(3C)
reserve a terminal ..... lock(1)
reset error indicator on open file ..... ferror(3S)
RETURN ..... regexp(7)
revck ..... revck(1M)
reverse line-feeds and backspaces, interpret for nroff(1) ..... col(1)
reverse previous *get*(1) of SCCS file ..... unget(1)
revision ..... revision(1)
revision information, get HP-UX ..... revision(1)
revision numbers, check for HP-UX files ..... revck(1M)
rewind ..... fseek(3S)
rewind a file ..... fseek(3S)
rewind group file ..... getgrent(3C)
rewind magnetic tape ..... mt(1)
rewind password file ..... getpwent(3C)
rm ..... rm(1)

rmail ..... mail(1)
rmdel ..... rmdel(1)
rmdir ..... rm(1)
rmdir ..... rmdir(2)
rmnl ..... rmnl(1)
roman8 ..... roman8(7)
root directory, change for duration of command ..... chroot(1), chroot(2)
root volume, mark/unmark volume as HP-UX root volume ..... rootmark(1M)
rootmark ..... rootmark(1M)
rtprio ..... rtprio(1)
run a command at low priority ..... nice(1), nice(2)
run a command immune to hangups, logouts, and quits ..... nohup(1)
run daily accounting ..... runacct(1M)
runacct ..... runacct(1M)
CPU time report ..... clock(3C)
CS/80 cartridge tape special file ..... ct(4)
GPIO routines (device I/O library) ..... gpio_*(3I)
HALGOL programs ..... opx25(1M)
HP-IB routines (device I/O library) ..... hpib_*(3I)
IMAGE database access ..... query(1)
I/O routines (device I/O library) ..... io_*(3I)
KERMIT-protocol file transfer program ..... kermit(1M)
LP spooler system, configure ..... mklp(1M)
MPE/RTE-style message catalog support ..... catread(3C)
MPE/RTE-style message catalog support ..... catread(3C)
UUCP system snapshot ..... uusnap(1)
XMODEM protocol file transfer program ..... umodem(1M)
XMODEM protocol file transfer program ..... umodem(1M)
sact ..... sact(1)
sbrk ..... brk(2)
scan text for pattern and process ..... awk(1)
scanf ..... scanf(3S)
SCCS, ask for help concerning ..... help(1)
SCCS file, change delta commentary of ..... cdc(1)
SCCS file, check for validity ..... val(1)
SCCS file, compare two versions of ..... sccsdiff(1)
SCCS file, create delta (change) for ..... delta(1)
SCCS file, description of SCCS file format ..... sccsfile(5)
SCCS file, get identification information from ..... what(1)
SCCS file, get version of ..... get(1)
SCCS file, print and summarize ..... prs(1)
SCCS file, print current editing activity for ..... sact(1)
SCCS file, print delta summary of ..... get(1)
SCCS file, remove delta from ..... rmdel(1)
SCCS file, reverse previous get(1) of ..... unget(1)
SCCS files, create or change parameters of ..... admin(1)
sccsdiff ..... sccsdiff(1)
schedule commands at specified date(s) and time(s) ..... at(1), cron(1M)
screen handling and optimization routines ..... curses(3X)
SDF boot area, copy OS from one or more SDF boot areas to another ..... oscp(1M)
SDF, description of ..... dir(5)
SDF, description of SDF volume ..... fs(5)
SDF volume, format, initialize, and certify ..... sdfinit(1M)
sdfinit ..... sdfinit(1M)
search an ASCII file for pattern ..... grep(1)

search tables, hash-coded ..... hsearch(3C)
security control, dialup ..... dialups(5)
sed ..... sed(1)
seek to new position in file ..... lseek(2)
segment length, modify ..... memvary(2)
segment, lock/unlock for process ..... memlck(2)
segment reference patterns, inform operating system about ..... memadvise(2)
select ..... select(2)
select/reject common lines of two files ..... comm(1)
semaphore control operations ..... semctl(2)
semaphore operations ..... semop(2)
semaphores and record locking on files ..... lockf(2)
semaphores, get ..... semget(2)
semctl ..... semctl(2)
semget ..... semget(2)
semop ..... semop(2)
send mail to users or read mail ..... mail(1)
send signal to all user processes ..... killall(1M)
set ..... sh(1)
set current signal mask ..... sigsetmask(2)
set group access list ..... setgroups(2)
set name of host cpu ..... sethostname(2)
set options for terminal port ..... stty(1)
set or change real-time priority ..... rtprio(1)
set or print name of current host system ..... hostname(1)
set printer options ..... slp(1)
set process's alarm clock ..... alarm(2)
set special attributes for group ..... setprivgrp(1M)
set system parameters ..... uconfig(1M)
set tabs on a terminal ..... tabs(1)
set the modes of a terminal ..... getty(1M)
set time and date ..... date(1), stime(2)
set user and group IDs ..... setuid(2)
setbuf ..... setbuf(3S)
setgid ..... setuid(2)
setgrent ..... getgrent(3C)
set-group-ID bit, set/clear for file ..... chmod(1), chmod(2)
setgroups ..... setgroups(2)
sethostname ..... sethostname(2)
setitimer ..... setitimer(2)
setjmp ..... setjmp(3C)
setkey ..... crypt(3C)
setmnt ..... setmnt(1M)
setpgrp ..... setpgrp(2)
setprivgrp ..... setprivgrp(1M)
setprivgrp ..... setprivgrp(1M), setprivgrp(2)
setpwent ..... getpwent(3C)
settimeofday ..... settimeofday(2)
setuid ..... setuid(2)
set-user-ID bit, set/clear for file ..... chmod(1), chmod(2)
sh ..... sh(1)
shareable, mark or unmark program code as ..... chatr(1)
shared memory control operations ..... shmctl(2)
shared memory operations ..... shmop(2)
shared memory segment, get ..... shmget(2)

shell ..... sh(1)
shell, change default login ..... chsh(1)
shell command, issue from program ..... system(3S)
shell, command, Personal Applications Manager ..... pam(1)
shell, input commands to ..... sh(1)
shell procedures for accounting ..... acctsh(1M)
shell programming language ..... sh(1)
shell scripts, system initialization ..... brc(1M)
shell, set/clear flags to ..... sh(1)
shift ..... sh(1)
shmctl ..... shmctl(2)
shmget ..... shmget(2)
shmop ..... shmop(2)
show group memberships ..... groups(1)
shut down operating system with optional re-boot ..... stopsys(1M)
shutdown ..... shutdown(1M)
shutdown status of specified file system ..... fsclean(1M)
sigblock ..... sigblock(2)
sign on ..... login(1)
signal ..... signal(2)
signal facilities, software ..... sigvector(2)
signal, force action associated with signal to be taken ..... ssignal(3C)
signal handling for program, set up ..... signal(2), ssignal(3C)
signal mask, set ..... sigsetmask(2)
signal, send SIGIOT to process ..... abort(3C)
signal, send to all user processes ..... killall(1M)
signal, send to process ..... kill(1), kill(2), abort(3C)
signal, set trap for ..... sh(1)
signal stack space ..... sigspace(2)
signal, suspend process until receipt of ..... pause(2)
signgam ..... gamma(3M)
signs, make using large letters ..... banner(1)
sigpause ..... sigpause(2)
sigsetmask ..... sigsetmask(2)
sigspace ..... sigspace(2)
sigvector ..... sigvector(2)
simple text formatter ..... adjust(1)
sin ..... trig(3M)
sine function ..... trig(3M)
sine, hyperbolic ..... sinh(3M)
sinh ..... sinh(3M)
size ..... size(1)
size of an object file ..... size(1)
sleep ..... sleep(1)
sleep ..... sleep(3C)
slp ..... slp(1)
snapshot of the UUCP system ..... uusnap(1)
software signal facilities ..... sigvector(2)
sort ..... sort(1)
sort algorithm ..... qsort(3C)
sort and/or merge files ..... sort(1)
sort, topological ..... tsort(1)
source code, locate for program ..... whereis(1)
spaces, convert to tabs, and vice versa ..... expand(1)
special characters in terminal interface, description of ..... tty(4)

special file, create block/character/network ..... mkdev(1M), mknod(2), mknod(1M)
special file, create fifo ..... mknod(2), mknod(1M)
special file, identify for file name on mounted file system ..... devnm(1M)
special file, modem control ..... modem(4)
special file, CS/80 cartridge tape ..... ct(4)
special file, system "bit bucket" ..... null(4)
special files, perform functions on ..... ioctl(2), stty(2)
special files, utilities used in creating special files ..... mknod(5)
spell ..... spell(1)
spellin ..... spell(1)
spelling errors, find ..... spell(1)
spellout ..... spell(1)
split ..... split(1)
split a file into pieces ..... split(1)
split operating system into one or more ordinary files ..... oscp(1M)
spool directory clean-up for uucp ..... uuclean(1M)
sprintf ..... printf(3S)
sputl ..... sputl(3X)
sqrt ..... exp(3M)
square root function ..... exp(3M)
srand ..... rand(3C)
sscanf ..... scanf(3S)
ssignal ..... ssignal(3C)
ssp ..... ssp(1)
stack size, specify size in bytes ..... uconfig(1M)
standard input, copy one line from to standard output ..... line(1)
standard input, read from ..... sh(1)
standard inter-process communication package ..... stdipc(3C)
start character, resume output, description of ..... tty(4)
stat ..... stat(2)
stat(2)/fstat(2), description of structure returned by these calls ..... stat(7)
state, defining system states for init(1M) ..... inittab(5)
state, initialization of system state and processes ..... init(1M)
stat.h, description of ..... stat(7)
status flags, get/set for file ..... fcntl(2)
status, get for file ..... stat(2)
status, inter-process communication facilities ..... ipcs(1)
stdio ..... stdio(3S)
stdipc ..... stdipc(3C)
step ..... regexp(7)
sticky bit, set/clear for file ..... chmod(1), chmod(2)
stime ..... stime(2)
stop character, suspend output, description of ..... tty(4)
stop operating system with optional re-boot ..... stopsys(1M)
stopsys ..... stopsys(1M)
strcat ..... string(3C)
strchr ..... string(3C)
strcmp ..... string(3C)
strcpy ..... string(3C)
strcspn ..... string(3C)
stream, close or flush ..... fclose(3S)
stream text editor ..... sed(1)
string collation, non-ASCII, used by NLS ..... nl_string(3C)
string, copy ..... string(3C)
string, get length of ..... string(3C)

string, print formatted data into ..... printf(3S)
string, read and format data from ..... scanf(3S)
string, read from buffered open file ..... gets(3S)
string, search contents of file for specified ..... grep(1)
string, search for particular character in ..... string(3C)
string to double-precision integer conversion ..... strtod(3C)
string, write to open file or standard output ..... puts(3S)
strings, compare two ..... string(3C)
strings, concatenate two ..... string(3C)
string-to-integer conversion ..... strtol(3C)
strip ..... strip(1)
strip multiple line-feeds from output ..... ssp(1)
strlen ..... string(3C)
strncat ..... string(3C)
strncmp ..... string(3C)
strncpy ..... string(3C)
strpbrk ..... string(3C)
strrchr ..... string(3C)
strspn ..... string(3C)
strtod ..... strtod(3C)
strtok ..... string(3C)
strtol ..... strtol(3C)
structure, definition of structure returned by stat(2) and fstat(2) ..... stat(7)
Structured Directory Format, description of ..... dir(5)
Structured Directory Format, description of SDF volume ..... fs(5)
Structured Directory Format volume, format, initialize, and certify ..... sdfinit(1M)
stty ..... stty(1)
stty ..... stty(2)
sttyv6 ..... sttyv6(4)
su ..... su(1)
summarize and print SCCS file ..... prs(1)
superblock, description of superblock in SDF volume ..... fs(5)
suspend process execution for interval of time ..... sleep(1), sleep(3C)
suspend process until signal ..... pause(2)
swab ..... swab(3C)
swap bytes ..... swab(3C)
swap device, add ..... swapon(2)
swap time, set for virtual segment ..... uconfig(1M)
swapon ..... swapon(1M)
swapon ..... swapon(2)
swapping and paging enable ..... swapon(1M)
symbol table, extract entries from executable file's symbol table (name list) ..... nlist(3C)
symbol table, print from object file ..... nm(1)
symbol table, remove from object file ..... strip(1)
symbols, examine execution profile for ..... prof(1)
sync ..... sync(2), sync(1), syncer(1)
syncer ..... syncer(1M)
sync, automatic periodic ..... syncer(1M)
synchronous I/O multiplexing ..... select(2)
sys_errlist ..... perror(3C)
sys_nerr ..... perror(3C)
system ..... system(3S)
system activity, terminate all current activity ..... shutdown(1M)
system calls, error indicator for ..... errno(2)
system configuration ..... config(1M)

system error logging file ..... errfile(5)
System III compatibility for magnetic tape, description of ..... mt(4)
system initialization shell scripts ..... brc(1M)
system name, get ..... revision(1), uname(1), uname(2)
system names, list of those known to uucp ..... uucp(1)
system parameters, set or list ..... uconfig(1M)
system reboot ..... reboot(1M)
system reconfiguration ..... uconfig(1M)
system state, defining states for init(1M) ..... inittab(5)
system state, initialization of ..... init(1M)
table of contents format description for archives/libraries ..... ranlib(5)
table of devices mounted by mount(1M) ..... mnttab(5)
table of mounted devices, create ..... setmnt(1M)
table search, binary ..... bsearch(3C)
tables, format for nroff/troff ..... tbl(1)
tabs ..... tabs(1)
tabs, expand to spaces, and vice versa ..... expand(1)
tabs, put tab specifications in text files ..... fspec(5)
tabs, set on terminal ..... tabs(1)
tail ..... tail(1)
tan ..... trig(3M)
tangent function ..... trig(3M)
tangent, hyperbolic ..... sinh(3M)
tanh ..... sinh(3M)
tape, archive files on ..... tar(1)
tape, Command Set 80 cartridge utility ..... tcio(1)
tape density, how to set for magnetic tape ..... mt(4)
tape, description of magnetic tape raw interface and controls ..... mt(4)
tape file archiver ..... tar(1)
tape file, convert, reblock, translate and/or copy ..... dd(1)
tape initialization ..... mediainit(1)
tape, manipulate and/or position ..... mt(1)
tape, unpack/extract files from Command Set 80 cartridge ..... upm(1)
tar ..... tar(1)
tbl ..... tbl(1)
tbl, nroff, troff, eqn constructs, remove from text ..... deroff(1)
tcio ..... tcio(1)
tee ..... tee(1)
temporary file, create and open ..... tmpfile(3S)
temporary file, generate name for ..... tmpnam(3S)
termcap ..... termcap(3C), terminfo(5)
termcap description to terminfo description, convert ..... captoinfo(1M)
terminal capabilities, database for *vi* editor ..... terminfo(5)
terminal capabilities in terminfo(5), access ..... termcap(3C)
terminal commands, description of ioctl(2) system call commands ..... tty(4)
terminal, database listing terminal type for each port ..... ttytype(5)
terminal dependent initialization ..... tset(1)
terminal, description of general interface to ..... tty(4)
terminal driver, pseudo- ..... pty(4)
terminal emulation, asynchronous ..... aterm(1)
terminal, establish communication with terminal for login ..... getty(1M)
terminal, facilitate viewing of continuous text on ..... more(1)
terminal, find baud rate of terminal during login process ..... getty(1M)
terminal flags, mapping between pwb/V6 UNIX and current HP-UX ..... tty(4)
terminal, generate file name for ..... ctermid(3S)

terminal, get path name of ..... ttyname(3C)
terminal, get path name of user's ..... tty(1)
terminal input control, description of ..... tty(4)
terminal interface, general ..... termio(4)
terminal interface, version 6/PWD-compatibility ..... sttyv6(4)
terminal, permit/deny messages to ..... mesg(1)
terminal screen, clear ..... clear(1)
terminal screen handling and optimization routines ..... curses(3X)
terminal, set options for ..... stty(1)
terminal, set tabs on ..... tabs(1)
terminal, set type and mode on login ..... tset(1)
terminal, test file descriptor for association with ..... ttyname(3C)
terminals, list of recognized terminal names ..... term(7)
terminals, list of supported terminals in terminfo(5) ..... term(7)
terminate a process ..... kill(1), sh(1), exit(2), kill(2), abort(3C)
terminate all users' processes ..... shutdown(1M)
terminfo compiler ..... tic(1M)
terminfo database access ..... tput(1)
terminfo description from termcap description, convert ..... captoinfo(1M)
termio ..... termio(4)
test ..... sh(1), test(1)
test conditional expressions ..... sh(1), test(1)
text editor ..... ed(1), ex(1)
text editor, database of terminal capabilities for *vi* ..... terminfo(5)
text editor, stream ..... sed(1)
text editor (variant of ex for casual users) ..... edit(1)
text editor, visual ..... vi(1)
text, facilitate CRT viewing of continuous ..... more(1)
text file, put format specifications in ..... fspec(5)
text, find spelling errors in ..... spell(1)
text format specifications, put in text file ..... fspec(5)
text formatter ..... nroff(1)
text formatter, simple ..... adjust(1)
text formatting, description of man macros ..... man(7)
text formatting, description of mm macros ..... mm(7)
text formatting, remove nroff/troff/tbl/eqn constructs from text ..... deroff(1)
text, generate programs for lexical analysis of ..... lex(1)
text pattern scanning and processing language ..... awk(1)
text, print using mm macros ..... mm(1)
tgetent ..... termcap(3C)
tgetflag ..... termcap(3C)
tgetnum ..... termcap(3C)
tgetstr ..... termcap(3C)
tgoto ..... termcap(3C)
three-way differential file comparison ..... diff3(1)
tic ..... tic(1M)
time ..... time(1)
time ..... time(2)
time a command ..... time(1)
time and date, convert to ASCII string ..... ctime(3C)
time and date, get more precisely ..... ftime(2)
time, corrected for daylight saving time and time zone ..... ctime(3C)
time execution of a process and its child processes ..... times(2)
time, get seconds since 00:00:00 GMT, January 1, 1970 ..... time(2)
time, get/set ..... gettimeofday(2)

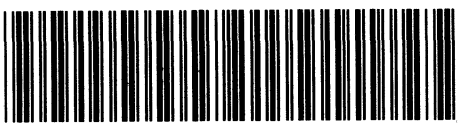
time, print elapsed user and system time for process ..... sh(1)
time, set and/or print ..... date(1), stime(2)
time to leave ..... leave(1)
time zone, time corrected for ..... ctime(3C)
time/date stamps, correct those on wtmp records ..... fwtmp(1M)
times ..... sh(1), times(2)
timezone ..... ctime(3C)
tmpfile ..... tmpfile(3S)
tmpnam ..... tmpnam(3S)
toascii ..... conv(3C)
_tolower ..... conv(3C)
tolower ..... conv(3C)
topological sort ..... tsort(1)
touch ..... touch(1)
_toupper ..... conv(3C)
toupper ..... conv(3C)
tput ..... tput(1)
tputs ..... termcap(3C)
tr ..... tr(1)
transfer files between two systems ..... uucp(1), uuto(1)
translate assembly language ..... atrans(1)
translate characters during copy from standard input to standard output ..... tr(1)
translate characters for NLS ..... nl_conv(3C)
translate tape file ..... dd(1)
trap ..... sh(1)
trap numbers for hardware ..... trapno(2)
trap, set for particular signal ..... sh(1), signal(2), ssignal(3C)
trapno ..... trapno(2)
trapno, report value for last command failure ..... err(1)
trigonometric functions ..... trig(3M)
troff, format tables for ..... tbl(1)
troff, nroff, tbl, eqn constructs, remove from text ..... deroff(1)
true ..... true(1)
truth value about your processor type ..... machid(1)
truth values ..... true(1)
tset ..... tset(1)
tsort ..... tsort(1)
tty ..... tty(1)
tty name, record for each user (accounting) ..... utmp(5)
tty port, database listing terminal type connected to each ..... ttytype(5)
ttyname ..... ttyname(3C)
ttyslot ..... ttyslot(3C)
tune a file system ..... tunefs(1M)
type declarations, data type definitions for system code ..... types(7)
typedefs for code portability between HP-UX implementations ..... model(5)
types.h, description of ..... types(7)
tzname ..... ctime(3C)
tzset ..... ctime(3C)
uconfig ..... uconfig(1M)
ul ..... ul(1)
ulimit ..... ulimit(2)
umask ..... sh(1), umask(1), umask(2)
umodem ..... umodem(1M)
umount ..... mount(1M), umount(2)
uname ..... uname(1), uname(2)

unblocked disc interface, description of ..... disc(4)
uncompact ..... compact(1)
uncompiler: terminfo ..... untic(1M)
underlining, translate underscores to terminal escape sequence ..... ul(1)
underscores, translate to terminal escape sequence for underlining ..... ul(1)
unexpand ..... expand(1)
unget ..... unget(1)
UNGETC ..... regexp(7)
ungetc ..... ungetc(3S)
uniformly-distributed pseudo-random number generator ..... drand48(3C)
uniq ..... uniq(1)
unique lines, find after comparing two files ..... comm(1)
UNIX/HP-UX system, establish communication with another ..... cu(1)
unlink ..... link(1M), unlink(2)
unlock/lock process address space or segment ..... memlck(2)
unmount or mount file system ..... mount(1M), mount(2), umount(2)
unpack cpio archives from HP media ..... upm(1)
unprintable characters in a file visible or invisible ..... vis(1)
untic ..... untic(1M)
update access/modification/change times of file ..... touch(1), utime(2)
update, maintain, recompile programs ..... make(1)
update super-block ..... sync(2), sync(1)
upm ..... upm(1)
upper-case to lower-case character conversion ..... conv(3C)
use findstring output to insert calls to getmsg ..... insertmsg(1)
user crontab file ..... crontab(1)
user environment, description of ..... environ(7)
user ID, get line from password file with matching ..... getpw(3C)
user ID, print ..... id(1)
user ID, search password file for matching ..... getpwent(3C)
user ID, set ..... setuid(2)
user name, print ..... id(1)
user name, search password file for matching ..... getpwent(3C)
user processes, terminate all ..... shutdown(1M)
user, switch to another ..... su(1)
users, print list of current ..... who(1)
users, print list of users and their current processes ..... whodo(1M)
ustat ..... ustat(2)
utilities, Bell Interchange Format file operations ..... bif(5)
utime ..... utime(2)
utmp accounting file, description of ..... utmp(5)
utmp file current user slot ..... ttyslot(3C)
utmp.h, description of ..... utmp(5)
uucico ..... uucico(1M)
uuclean ..... uuclean(1M)
uucp ..... uucp(1)
uucp command execution ..... uuxqt(1M)
uucp network, monitor activity ..... uusub(1M)
uucp spool directory clean-up ..... uuclean(1M)
uucp system names, list of ..... uucp(1)
uucp transactions grouped by transaction, list ..... uuls(1)
uucp/uux transactions, log of ..... uucp(1)
uulog ..... uucp(1)
uuls ..... uuls(1)
uuname ..... uucp(1)

uupick ..... uuto(1)
uusnap ..... uusnap(1)
uusub ..... uusub(1M)
uuto ..... uuto(1)
uux ..... uux(1)
uuxqt ..... uuxqt(1M)
val ..... val(1)
validate password and group files ..... pwck(1M)
validate SCCS file ..... val(1)
values ..... values(7)
values, machine-dependent ..... values(7)
varargs ..... varargs(7)
varargs argument list, print formatted output from ..... vprintf(3S)
variable argument list handling facility ..... varargs(7)
verify C program ..... lint(1)
verify Command Set 80 cartridge tape ..... tcio(1)
verify file system consistency ..... fsck(1M)
verify password and group files ..... pwck(1M)
version 6/PWD-compatibility terminal interface ..... sttyv6(4)
version name, get for HP-UX ..... uname(1), uname(2)
version number, get ..... revision(1)
versions, compare two SCCS file versions ..... sccsdiff(1)
vfork ..... fork(2)
vi ..... vi(1)
vi editor, database of terminal capabilities for ..... terminfo(5)
view ..... vi(1)
viewing text, facilitate on soft-copy terminals ..... more(1)
virtual memory page pool, specify maximum size of ..... uconfig(1M)
virtual memory usage, set or clear for program ..... chatr(1)
virtual memory working set ratio, set ..... uconfig(1M)
virtual segment, establish time segment remains memory resident ..... uconfig(1M)
vis ..... vis(1)
visual text editor ..... vi(1)
volume, description of SDF volume superblock ..... fs(5)
volume, format, initialize, and certify SDF volume ..... sdfinit(1M)
volume header, write LIF on file ..... lifinit(1)
volume, mark/unmark as HP-UX root volume ..... rootmark(1M)
vprintf ..... vprintf(3S)
vsadv ..... vsadv(2)
vsoff ..... vson(2)
vson ..... vson(2)
wait ..... sh(1), wait(1), wait(2)
wait for completion of process ..... sh(1), wait(1), wait(2)
walk a file tree ..... ftw(3C)
wall ..... wall(1M)
wc ..... wc(1)
wc ..... wc(1)
what ..... what(1)
whereis ..... whereis(1)
while loop, exit from enclosing ..... sh(1)
while loop, resume the next iteration of ..... sh(1)
who ..... who(1)
whoami ..... whoami(1)
whodo ..... whodo(1M)
word count ..... wc(1)

word, read from buffered open file ..... getc(3S)
word, write on buffered open file or standard output ..... putc(3S)
words, count number contained in file ..... wc(1)
working directory, change ..... cd(1), sh(1), chdir(2)
working directory, print name of ..... pwd(1)
write ..... write(1), write(2)
write character on buffered open file or standard output ..... putc(3S)
write current contents of memory to disc ..... sync(2), sync(1)
write interactively to another user ..... write(1)
write LIF volume header on file ..... lifinit(1)
write on a file ..... write(2)
write operation, reposition next ..... fseek(3S)
write password file entry ..... putpwent(3C)
write string to open file or standard output ..... puts(3S)
write to a file using buffers ..... fread(3S)
write to all users ..... wall(1M)
write word on buffered open file or standard output ..... putc(3S)
wtmp accounting file, description of ..... utmp(5)
wtmp records, convert from binary to ASCII ..... fwtmp(1M)
wtmp records, correct time/date stamps on ..... fwtmp(1M)
wtmpfix ..... fwtmp(1M)
x.25 line, get ..... getx25(1M)
xd ..... od(1)
y0 ..... bessel(3M)
y1 ..... bessel(3M)
yacc ..... yacc(1)
yn ..... bessel(3M)

HP Part Number
09000-90008
Printed in U.S.A. 6/86



09000-90657
For Internal Use Only