PERMUTED INDEX OF MERT AND UNIX PROGRAMMER'S MANUALS

Legend: (I-VIII) UNIX Sections; (a-g) MERT Sections

sync(VIII) update the super block
update(VIII) periodically update the super block
boot procedures(VIII) MERT startup
break, brk, sbrk(II) change core allocation
break in sh(I) exit from loop
break, brk, sbrk(II) change core allocation
setbuf in newio(III) set buffer size
alocmsg(b) allocate message buffer
getc, getw, fopen(III) buffered input
putc, putw, fcreat, fflush(III) buffered output
fflush in newio(III) flush buffer
freemsg(b) free up message buffer
mknod(VIII) build special file
cb(VI) C beautifier
cc(I) C compiler
lint(I) a C program verifier
hypot(III) calculate hypotenuse
dc(I) desk calculator
cal(VI) print calendar
call, lcall, vcall(II) create and execute a new process
indir(II) indirect system call
calloc in newio(III) allocate memory
Intro-b(b) INTRO. TO KERNEL EMT CALLS
Intro-f(f) INTRO. TO MERT UNIX SYSTEM CALLS
cal(VI) print calendar
fork(c) change count on open files and add capabilities
delcap(c) delete capability from process PCB
sendcpmsg(a) send a capability message
islower in newio(III) test for lower case
isupper in newio(III) test for upper case
in newio(III) translate to lower case...tolower
in newio(III) translate to upper case...toupper
ierror(III) catch Fortran errors
signal(II) catch or ignore signals
trap in sh(I) catch signals
cat(I) concatenate and print
cb(VI) C beautifier
cc(I) C compiler
chdir, cd(I) change working directory
floor, ceil(III) floor and ceiling functions
floor, ceil(III) floor and ceiling functions
cfree in newio(III) deallocate memory
jobchg(a) change control to next process
break, brk, sbrk(II) change core allocation
fork(c) change count on open files and add capabilities
passwd(I) change login password
chmod(c) change mode of file
chmod(II) change mode of file
chmod(I) change mode
chown(II) change owner and group of a file
chown(c) change owner of file
chown(VIII) change owner
chroot(I) change root directory for a command
chdir, cd(I) change working directory
chdir(c) change working directory
chdir(II) change working directory
pipe(II) create an interprocess channel
send events to processes on a control channel...psignal(b)
ungetc in newio(III) push character back
putchar(a) output characters to character device driver
gsi(VI) interpret extended character set on GSI terminal
ascii(VII) map of ASCII character set
fgetc in newio(III) get character
fputc in newio(III) put character
getc in newio(III) get character
getchar in newio(III) get character

fgrep(VI) search a file for lines containing keywords
cut(VI) cut out selected fields of each line of a file
cpio(I) copy file archives in and out
diff(I) differential file comparator
diff3(I) 3-way differential file comparison
dup(II) duplicate an open file descriptor
fileno in newio(III) get file descriptor
grep(I) search a file for a pattern
exec(c) open file for execution
egrep(VI) search a file for lines containing a pattern
fgrep(VI) search a file for lines containing keywords
mkpt(VIII) make prototype file for use by mkfs
pfile(g) process file format produced by ldp
ar(V) archive (library) file format
Intro-g(g) INTRO. TO MERT FILE FORMATS
Intro(II) INTROD. TO MERT FILE FORMATS
Intro(V) INTROD. TO FILE FORMATS
fmove(c) move file into a contiguous area
fmove(d) move file into contiguous area
fmove(f) move file into contiguous area
split(I) split a file into pieces
Intro-fm(c) INTRO. TO FILE MANAGER MESSAGES
setfil(III) specify Fortran file name
tell(II) get file offset
falloc(d) allocate contiguous file space
openi(c) open file specified by inode number
stat(c) get file status
stat(II) get file status
fsck(VIII) file system consistency check and interactive repair
check(VIII) file system consistency check
dcheck(VIII) file system directory consistency check
dump(VIII) incremental file system dump
init(c) initialize file system manager
restor(VIII) incremental file system restore
icheck(VIII) file system storage consistency check
mtab(VII) mounted file system table
Intro-d(d) INTRO. TO FILE SYSTEM UTILITY PROGRAMS
fs(g) format of MERT file system volume
fs(V) format of UNIX file system volume
mkfs(VIII) construct a file system
mount(c) mount file system
mount(II) mount file system
mount(VIII) mount file system
recon(d) reconfigure file system
umount(c) dismount file system
umount(II) dismount file system
umount(VIII) dismount file system
ftrunc(c) truncate file to given size
cut out selected fields of each line of a file...cut(VI)
fclose in newio(III) close file
fopen in newio(III) open file
fread in newio(III) read from file
freopen in newio(III) reopen file
fwrite in newio(III) write to file
file(I) determine format of file
system data structure definitions file...include(V)
basename(I) strip filename affixes
dirname(I) strip simple filename
fileno in newio(III) get file descriptor
fork(c) change count on open files and add capabilities
cpall(I) copy all files to a directory
mvall(I) move all files to a directory
col(VI) filter reverse line feeds
find(I) find files
hyphen(VI) find hyphenated words
wdleng in newio(III) find machine word size

bc(I) arbitrary precision interactive language
file system consistency check and interactive repair...fsck(VIII)
dsw(I) delete interactively
typewriters...tty(IV) interface to low speed asynchronous devices including
dc(IV) DC-11 communications interface
dn(IV) DN-11 ACU interface
dp(IV) DP-11 201 data-phone interface
dr(IV) DR-11 general device interface
ht(IV) RH-11/TU-16 magtape interface
kl(IV) KL-11 or DL-11 asynchronous interface
tm(IV) TM-11/TU-10 magtape interface
spline(VI) interpolate smooth curve
gsi(VI) interpret extended character set on GSI terminal
fptrap(III) floating point interpreter
sno(VI) Snobol interpreter
pipe(II) create an interprocess channel
Intro-c(c) INTRO. TO INTERPROCESS MESSAGE FORMATS
return(I) terminate profile or interrupt processing routine
attach(a) attach process to interrupt vector
detach(a) detach process from interrupt vector
atchintr(b) attach a process to an interrupt
dtchintr(b) detach a process from an interrupt
qsleep(f) stop execution for small interval
sleep(I) suspend execution for an interval
sleep(II) stop execution for interval
Intro-fm(c) INTRO. TO FILE MANAGER MESSAGES
Intro-d(d) INTRO. TO FILE SYSTEM UTILITY PROGRAMS
Intro-c(c) INTRO. TO INTERPROCESS MESSAGE FORMATS
I/o-messages(c) INTRO. TO I/O PROCESS-MESSAGES
Intro-b(b) INTRO. TO KERNEL EMT CALLS
Intro-g(g) INTRO. TO MERT FILE FORMATS
Intro-f(f) INTRO. TO MERT UNIX SYSTEM CALLS
Intro-e(e) INTRO. TO MERT UTILITY PROGRAMS
Process-mgr(c) INTRO. TO PROCESS-MANAGER MESSAGES
Intro-a(a) INTRO. TO SUPERVISOR EMT TRAPS
Intro(IV) INTROD. TO DRIVERS
Intro(V) INTROD. TO FILE FORMATS
Intro(II) INTROD. TO MERT FILE FORMATS
Intro(III) INTROD. TO SUBROUTINES
Intro(VIII) INTROD. TO SYSTEM PROGRAMS
intss in newio(III) test for tss or batch
ncheck(VIII) generate names from i-numbers
ioqueuem(b) send message to I/O device driver
ioqueuem(a) send an I/O message
setio(f) set I/O mode of file
I/o-messages(c) INTRO. TO I/O PROCESS-MESSAGES
newio(III) a new IO subroutine package
iolock(b) lock segment for I/O
iolock(b) lock segment for I/O
iomap(b) map segid/offset to virtual address
I/o-messages(c) INTRO. TO I/O PROCESS-MESSAGES
pio(c) physical I/O
ioqueuem(a) send an I/O message
ioqueuem(b) send message to I/O device driver
statio(f) get status of asynchronous I/O
abort(III) generate an IOT fault
uniolock(b) unlock segment for I/O
isalpha in newio(III) test for alphabetic
isdigit in newio(III) test for numeric
islower in newio(III) test for lower case
isnp(d) snap i-node contents
isspace in newio(III) test for space
isupper in newio(III) test for upper case
spacaloc(a) allocate space for segment; add it to the proc. virtual addr. space
continue in sh(I) next iteration in loop
itol(III) integer to long integer conversion

bj(VI) the game of black jack
jobchg(a) change control to next process
join(VI) relational data base operator
kdb(e) kernel debugger
kdmp(e) dump system state into core file
kdb(e) kernel debugger
Intro-b(b) INTRO. TO KERNEL EMT CALLS
kpkill(e) terminate a kernel process (superuser)
kprc(g) kernel process translation file
getchar(a) get characters from kernel process
ktime(e) give detailed kernel time of a command
search a file for lines containing keywords...fgrep(VI)
kill(I) terminate a process
kill(II) send signal to a process
kl(IV) KL-11 or DL-11 asynchronous interface
kl(IV) KL-11 or DL-11 asynchronous interface
mem, kmem, null(IV) core memory
kpkill(e) terminate a kernel process (superuser)
kprc(g) kernel process translation file
ktime(e) give detailed kernel time of a command
bc(I) arbitrary precision interactive language
sh(I) shell command programming language
end, etext, edata(III) last locations in program
tail(I) deliver the last part of a file
call, lcall, vcall(II) create and execute a new process
ld(I) link editor
ldp(e) load a process
pfile(g) process file format produced by ldp
ldu(e) load a user process with public libraries
strlen in newio(III) obtain string length
lex(VI) generate programs for simple lexical tasks
lex(VI) generate programs for simple lexical tasks
ldu(e) load a user process with public libraries
ar(V) archive (library) file format
ar(I) archive and library maintainer
get a message of type between given limits...mgetlim(a)
read(I) read one line at a time
col(VI) filter reverse line feeds
cut(VI) cut out selected fields of each line of a file
lpd(VIII) line printer daemon
lpr(I) line printer spooler
lp(IV) line printer
getpw in newio(III) get password line
comm(I) print lines common to two files
egrep(VI) search a file for lines containing a pattern
fgrep(VI) search a file for lines containing keywords
uniq(I) report repeated lines in a file
rev(VI) reverse lines of a file
paste(VI) merge the same lines of all files
a.out(V) assembler and link editor output
ld(I) link editor
link(c) link to a file
link(II) link to a file
dmc(IV) network link with DDCMP protocol
link(c) link to a file
link(II) link to a file
ln(I) make a link
lint(I) a C program verifier
ls(I) list contents of directory
cref(I) make cross reference listing
nlist(III) get entries from name list
nm(I) print name list
ln(I) make a link
lnxx(III) return name of current terminal
ldp(e) load a process
(Mem-Mgr)load(c) to memory manager: load a process

mkpt(VIII) make prototype file for use by mkfs
make(I) make a program
(Mem-Mgr)load(c) to memory manager: load a process
Intro-fm(c) INTRO. TO FILE MANAGER MESSAGES
(Mem-Mgr)lock(c) to memory manager: process lock a segment
(Mem-Mgr)term(c) to memory manager: terminate a process
init(c) initialize file system manager
(P-Mgr)pinit(c) initialize the process manager
man(I) print on-line documentation
tp(I) manipulate DECtape and magtape
mtm(I) magnetic tape manipulation
man(V) manual page format
tmac(VI) ms macros for formatting manuscripts
man(V) manual page format
ascii(VII) map of ASCII character set
iomap(b) map segid/offset to virtual address
neqn(I) typeset mathematics on terminal
eqn(I) typeset mathematics
mdate(c) modify date of file
mem, kmem, null(IV) core memory
(Mem-Mgr)load(c) to memory manager: load a process
segment... (Mem-Mgr)lock(c) to memory manager: process lock a
(Mem-Mgr)term(c) to memory manager: terminate a process
alockseg(a) lock a segment in memory and set write back
agen(VI) generate associative memory drivers
(Mem-Mgr)load(c) to memory manager: load a process
(Mem-Mgr)lock(c) to memory manager: process lock a segment
(Mem-Mgr)term(c) to memory manager: terminate a process
calloc in newio(III) allocate memory
cfree in newio(III) deallocate memory
lockseg(a) lock a segment in memory
mem, kmem, null(IV) core memory
plock(f) lock process in memory
sort(I) sort or merge files
paste(VI) merge the same lines of all files
mesg(I) permit or deny messages
mesg(III) write message on typewriter
(P-Mgr)pwait(c) message at termination of process-mgr-created process
alocmsg(b) allocate message buffer
freemsg(b) free up message buffer
Intro-c(c) INTRO. TO INTERPROCESS MESSAGE FORMATS
sndmsgfrom(a) send a message from a process
gettype(a) get a message of given type
mgettype(a) get a message of given type
msgtype(a) get a message of given type
mreceive(a) get a message of type -1 (acknowledgement)
mgetlim(a) get a message of type between given limits
queuem(b) queue message on input queue
mesg(III) write message on typewriter
sendport(a) send message through port
msgport(f) send message to a process connected to a port
msgsend(f) send message to a process
ioqueuem(b) send message to I/O device driver
dqtype(b) dequeue a particular message type
queuemn(b) queue message with no acknowledgement expected
dequeuem(b) dequeue a message
getmsg(a) get a message
ioqueuem(a) send an I/O message
messink(b) return a message
msgrecv(f) receive message
receive(a) get a message
msg(f) send and receive messages (USG Version)
sendcpmsg(a) send a capability message
sendmsg(a) send a message
Intro-fm(c) INTRO. TO FILE MANAGER MESSAGES
mesg(I) permit or deny messages