MERT PROGRAMMER'S MANUAL

Second Edition (MERT Release 0) Program Generic PG-1C600 Issue 1

H. Lycklama, D. L. Bayer (Authors) Edited by Department 8234

October 1977

This manual is for use within the Bell System only.

@ Bell Laboratories, Murray Hill, New Jersey, 07974

AUTHORS' PREFACE to MERT Second Edition

We are grateful to the first users of the MERT system for their initial support and encouragement. These include: S. L. Arnold, W. A. Burnette, L. L. Hamilton, J. E. Laur, J. J. Molinelli, R. W. Peterson, M. A. Pilla and T. F. Tabloski. They made suggestions for additions and improvements, many of which were incorporated in MERT, and make MERT what it is today. We are particularly appreciative of the members of Department 8234, G. W. R. Luderer, E. A. Loikits and T. M. Raleigh who have done much to aid in the documentation and preparation of an exportable MERT system.

H. Lycklama D. L. Bayer

This manual was photocomposed in the Murray Hill Computation Center. The text of the manual was prepared using the UNIX* ed text editor and troff formatting program, as well as a Stare graphic hardcopy device for assistance in the proof correction process.

* UNIX is a Trademark of the Bell System.

First Printing

Reproduction, assembly and distribution: Technical Documentation Department Bell Laboratories, Whippany, New Jersey

TABLE OF CONTENTS MERT PROGRAMMER'S MANUAL

AUTHORS' PREFACE to Second Edition page 2 - Section 11
TABLE OF CONTENTS page 3 - Section 11
PERMUTED INDEX TO MERT PROGRAMMER'S MANUAL page 7 - Section 11
INTRODUCTION TO MERT PROGRAMMER'S MANUAL page 19 - Section 11
A SUPERVISOR PROCESS CALLS Intro-a addseg add a segment to the process address space adduser increment user count on a process alockseg alock
rti return from trap
segname get segment name
sendcpmsg send a capability message sendmsg send a message
sendmsg send a message

sendport setdspac setime setmap setprior setty sizeseg sleep sndmsgfrom spacaloc sswap sunswap sunswap sysproc toutset ulockid ulockseg unblkseg wakeup writeseg	set user-supervisor d-space bits set user-supervisor d-space bits set time set access, mode and starting segmentation register set priority of process set state of tty driver process set a bit pattern to sleep on send a message from a process allocate space for segment; add it to the proc. virtual addr. space remove non-swap status from a segment make a segment non-swap system process set time-out decrement lock count of segment decrement lock count of segment unblock a named segment wakeup all processes sleeping on a pattern wakeup all processes sleeping on a pattern force a segment to be written back			
unay wall for every	Hibrary Comment of the Comment of th			
B. KERNEL PROCE Intro-b alocmsg atchintr dequeuem dqtype dtchintr freemsg getarg getime iolock iomap ioqueuem messink psignal psleep ptimer putarg pwakeup	INTRO. TO KERNEL EMT CALLS allocate message buffer attach a process to an interrupt dequeue a message dequeue a particular message type detach a process from an interrupt free up message buffer get argument from SUP address space get system time lock segment for I/O map segid/offset to virtual address send message to I/O device driver return a message send events to processes on a control channel put process to sleep on bit pattern set time-out value for process put argument into SUP address space wake up processes sleeping on bit pattern			
queuem	queue message on input queue			
queuemn riteback segname sendevent setime timleft uniolock	queue message with no acknowledgement expected set altered bit on a segment get name of segment send event to a process set system time get time-out value for process unlock segment for I/O	1 1 3 3		
C. INTERPROCESS MESSAGE FORMATS Section 14				
Intro-c Intro-fm chdir chmod	INTRO. TO INTERPROCESS MESSAGE FORMATS INTRO. TO FILE MANAGER MESSAGES	S		
chown close .	change owner of file	2		

	stidil saldan drive zanovne appur e beni
	creat creat a new file
	delcap delete capability from process PCB
	exec open file for execution
	falloc allocate contiguous space for a file
	fmove move file into a contiguous area
	fork change count on open files and add capabilities
	fsize get size of file
	fstat get status of open file
	ftrunc truncate file to given size
	init initialize file system manager
	link
	mdate modify date of file
	mknod make a directory or a special file
	mount mount file system
	nmcode get name code for segment
	open open file
	openi open file specified by inode number
	read read from file
	stat get file status
	sync update super-block
	umount dismount file system
	unlink remove directory entry
	write write to file
	I/o-messages INTRO. TO I/O PROCESS-MESSAGES
	Process-mgr INTRO. TO PROCESS-MANAGER MESSAGES
	(P-Mgr)P_CREAT create a process from a file
	(P-Mgr)MSTERM terminate a process and dump core
	(P-Mgr)pinit initialize the process manager
	(P-Mgr)pwait . message at termination of process-mgr-created process
	(Mem-Mgr)lock to memory manager: process lock a segment
	(Mem-Mgr) term to memory manager: terminate a process
	(System Scheduler) term to system scheduler: terminate a process
	lormal of MER I lie system vol
D. FIL.	E SYSTEM UTILITIES Section 15
	Intro-d INTRO. TO FILE SYSTEM UTILITY PROGRAMS
	falloc allocate contiguous file space
	fmove move file into contiguous area
	ignat
	icat concatenate i-node
	iclr
E. MEI	iclr
E. MEF	iclr
Е. МЕГ	iclr
E. MEF	iclr clear i-node icon do consistency check of i-nodes idmp dump i-node isnp snap i-node contents recdmn recon reconfiguration daemon recon reconfigure file system RT-UNIX PROGRAMS Section 16 Intro-e INTRO. TO MERT UTILITY PROGRAMS acp asynchronous copy errproc error logger
E. MEI	iclr clear i-node icon do consistency check of i-nodes idmp dump i-node isnp snap i-node contents recdmn recon reconfiguration daemon recon reconfigure file system RT-UNIX PROGRAMS Section 16 Intro-e INTRO. TO MERT UTILITY PROGRAMS acp asynchronous copy errproc error logger kdb kernel debugger
E. MEF	iclr clear i-node icon do consistency check of i-nodes idmp dump i-node isnp snap i-node contents recdmn recon reconfiguration daemon recon reconfigure file system RT-UNIX PROGRAMS Section 16 Intro-e INTRO. TO MERT UTILITY PROGRAMS acp asynchronous copy errproc error logger
E. MEF	iclr clear i-node icon do consistency check of i-nodes idmp dump i-node isnp snap i-node contents recdmn recon reconfiguration daemon recon reconfigure file system RT-UNIX PROGRAMS Section 16 Intro-e INTRO. TO MERT UTILITY PROGRAMS acp asynchronous copy errproc error logger kdb kernel debugger kdmp dump system state into core file
E. MEF	iclr clear i-node icon do consistency check of i-nodes idmp dump i-node isnp snap i-node contents recdmn reconfiguration daemon recon reconfigure file system RT-UNIX PROGRAMS Section 16 Intro-e INTRO. TO MERT UTILITY PROGRAMS acp asynchronous copy errproc asynchronous copy errproc error logger kdb kernel debugger kdmp dump system state into core file kpkill terminate a kernel process (superuser)
E. MER	iclr clear i-node icon do consistency check of i-nodes idmp dump i-node isnp snap i-node contents recdmn recon reconfiguration daemon recon reconfigure file system RT-UNIX PROGRAMS Section 16 Intro-e INTRO. TO MERT UTILITY PROGRAMS acp asynchronous copy errproc error logger kdb kernel debugger kdmp dump system state into core file kpkill terminate a kernel process (superuser) ktime give detailed kernel time of a command
E. MEF	iclr clear i-node icon do consistency check of i-nodes idmp dump i-node isnp snap i-node contents recdmn reconfiguration daemon recon reconfigure file system RT-UNIX PROGRAMS Intro-e INTRO. TO MERT UTILITY PROGRAMS acp asynchronous copy errproc error logger kdb shape dump system state into core file kpkill terminate a kernel process (superuser)

	pcp	load a user process with public libraries physical copy physical I/O terminate a process (superuser) run an environment (superuser) system generation program start up tty dump system state into core file
	tkill t	erminate all processes associated with a terminal extract user core image from process core dump
	Intro-f falloc fmove getseg lock msg msgport msgrecv msgsend pcreat plock qsleep qwait sendev setio statio sysproc	INTRO. TO MERT UNIX SYSTEM CALLS allocate space for contiguous file move file into contiguous area get user segment semaphores (USG Version) send and receive messages (USG Version) send message to a process connected to a port receive message send message to a process creat new process lock process in memory stop execution for small interval check for child process termination send event(s) set I/O mode of file get status of asynchronous I/O system ports
	waitev	wait for an event
G. MEF	RT FILE FORMATS Intro-g	Section 18 INTRO. TO MERT FILE FORMATS format of MERT file system volume kernel process translation file process file format produced by ldp

PERMUTED INDEX OF MERT PROGRAMMER'S MANUAL

mreceive(a) get a message of type -1 (acknowledgement)

queuemn(b) queue message with no acknowledgement expected mreceive(a) get a message of type -1 (acknowledgement)

setmap(a) set access, mode and starting segmentation register

acp(e) asynchronous copy

openseg(a) add a segment id to the process segment table addseg(a) add a segment to the process address space

fork(c) change count on open files and add capabilities

spacaloc(a) allocate space for segment; add it to the proc. virtual addr. space for segment; add it to the proc. virtual addr. space...spacaloc(a) allocate space addseg(a) add a segment to the process address space

drop a segment from a process virtual address space...dropseg(a) getarg(b) get argument from SUP address space putarg(b) put argument into SUP address space

remove a segment from a process virtual address space...rmovseg(a) iomap(b) map segid/offset to virtual address

addseg(a) add a segment to the process address space adduser(a) increment user count on a process

falloc(d) allocate contiguous file space

falloc(c) allocate contiguous space for a file alocmsg(b) allocate message buffer

falloc(f) allocate space for contiguous file

virtual addr. space...spacaloc(a) allocate space for segment; add it to the proc.

alockseg(a) lock a segment in memory and set write back

alocmsg(b) allocate message buffer alocseg(a) create a segment

riteback (b) set altered bit on a segment

fmove(c) move file into a contiguous area fmove(d) move file into contiguous area

fmove(f) move file into contiguous area

getarg(b) get argument from SUP address space putarg(b) put argument into SUP address space

tkill(e) terminate all processes associated with a terminal acp(e) asynchronous copy

statio(f) get status of asynchronous I/O

atchintr(b) attach a process to an interrupt atchintr(b) attach a process to an interrupt

attach(a) attach process to interrupt vector attach(a) attach process to interrupt vector

lock a segment in memory and set write back...alockseg(a) writeseg(a) force a segment to be written back

writeseg(a) force a segment to be written back mgetlim(a) get a message of type between given limits

riteback (b) set altered

bit on a segment

sleep(a) set a bit pattern to sleep on bit pattern

psleep(b) put process to sleep on pwakeup(b) wake up processes sleeping on bit pattern setdspac(a) set user-supervisor d-space bits

alocmsg(b) allocate message buffer

freemsg(b) free up message | buffer Intro-b(b) INTRO. TO KERNEL EMT CALLS

Intro-f(f) INTRO. TO MERT UNIX SYSTEM CALLS fork(c) change count on open files and add capabilities

delcap(c) delete capability from process PCB

sendcpmsg(a) send a capability message jobchg(a) change control to next process

fork(c) change count on open files and add capabilities

chmod(c) change mode of file chown(c) change owner of file

chdir(c) change working directory send events to processes on a control channel...psignal(b)

putchar(a) output characters to character device driver getchar(a) get characters from kernel process putchar(a) output characters to character device driver chdir(c) change working directory qwait(f) check for child process termination icon(d) do consistency check of i-nodes qwait(f) check for child process termination chmod(c) change mode of file chown(c) change owner of file clrevent(a) clear event flag(s) iclr(d) clear i-node close(c) close a file close(c) close a file clrevent(a) clear event flag(s) nmcode(c) get name code for segment ktime(e) give detailed kernel time of a command icat(d) concatenate i-node cwait(a) conditional wait for event msgport(f) send message to a process connected to a port icon(d) do consistency check of i-nodes getcsw(a) get console switch register setting isnp(d) snap i-node contents fmove(c) move file into a contiguous area fmove(d) move file into contiguous area fmove(f) move file into contiguous area falloc(d) allocate contiguous file space falloc(f) allocate space for contiguous file falloc(c) allocate contiguous space for a file psignal(b) send events to processes on a control channel jobchg(a) change control to next process copyseg(a) make a copy of a segment acp(e) asynchronous copy pcp(e) physical copy copyseg(a) make a copy of a segment extract user core image from process core dump...xusr(e) kdmp(e) dump system state into core file tdmp(e) dump system state into core file xusr(e) extract user core image from process core dump terminate a process and dump core...(P-Mgr)MSTERM(c) ulockid(a) decrement lock count of segment ulockseg(a) decrement lock count of segment adduser(a) increment user count on a process lockid(a) increment the lock count on a segment fork(c) change count on open files and add capabilities creat(c) creat a new file pcreat(f) creat new process creat(c) creat a new file (P-Mgr)P_CREAT(c) create a process from a file alocseg(a) create a segment cwait(a) conditional wait for event recdmn(d) reconfiguration daemon mdate(c) modify date of file kdb(e) kernel debugger growseg(a) increase or decrease the size of a segment ulockid(a) decrement lock count of segment ulockseg(a) decrement lock count of segment delcap(c) delete capability from process PCB delcap(c) delete capability from process PCB dequeuem (b) dequeue a message dequeue a particular message type dqtype(b) dequeuem (b) dequeue a message dtchintr(b) detach a process from an interrupt detach(a) detach process from interrupt vector detach(a) detach process from interrupt vector ktime(e) give detailed kernel time of a command

ioqueuem(b) send message to I/O device driver

putchar(a) output characters to character device driver unlink(c) remove directory entry

mknod(c) make a directory or a special file

chdir(c) change working directory

umount(c) dismount file system

dqtype(b) dequeue a particular message type

getty (a) get state of tty driver process setty(a) set state of tty driver process

ioqueuem(b) send message to I/O device driver

output characters to character device driver...putchar(a)

dropseg(a) drop a segment from a process virtual address space

address space... dropseg(a) drop a segment from a process virtual setdspac(a) set user-supervisor d-space bits

dtchintr(b) detach a process from an interrupt

(P-Mgr)MSTERM(c) terminate a process and dump core

idmp(d) dump i-node

kdmp(e) dump system state into core file tdmp(e) dump system state into core file

extract user core image from process core dump...xusr(e)

Intro-b(b) INTRO. TO KERNEL EMT CALLS Intro-a(a) INTRO. TO SUPERVISOR EMT TRAPS

enevent(a) enable event flag(s)

enevent(a) enable event flag(s)

unlink(c) remove directory entry

run(e) run an environment (superuser)

errproc(e) error logger

errproc(e) error logger

clrevent(a) clear event flag(s) enevent(a) enable event flag(s) event(a) send event to a process

sendevent(b) send event to a process

event(a) send event to a process

cwait(a) conditional wait for event

psignal(b) send events to processes on a control channel

sendev(f) send event(s) waitev(f) wait for an event

exec(c) open file for execution

execute(a) execute new process

execute (a) execute new process

qsleep(f) stop execution for small interval

exec(c) open file for execution

queue message with no acknowledgement expected...queuemn(b)

xusr(e) extract user core image from process core dump

falloc(c) allocate contiguous space for a file falloc(d) allocate contiguous file space falloc(f) allocate space for contiguous file

exec(c) open file for execution

pfile (g) process file format produced by ldp

Intro-g(g) INTRO. TO MERT FILE FORMATS

fmove(c) move file into a contiguous area fmove(d) move file into contiguous area fmove(f) move file into contiguous area

Intro-fm(c) INTRO. TO FILE MANAGER MESSAGES

falloc(d) allocate contiguous file space

openi(c) open file specified by inode number

stat(c) get file status

init(c) initialize file system manager

Intro-d(d) INTRO. TO FILE SYSTEM UTILITY PROGRAMS

fs(g) format of MERT file system volume

mount(c) mount file system

recon(d) reconfigure file system umount(c) dismount file system

ftrunc(c) truncate file to given size fork(c) change count on open files and add capabilities

clrevent(a) clear event flag(s) enevent(a) enable event flag(s)

```
fmove(c) move file into a contiguous area
                                                fmove(d) move file into contiguous area
                                                fmove(f) move file into contiguous area
                                     writeseg(a) force a segment to be written back
                                                fork(c) change count on open files and add capabilities
                                         fs(g) format of MERT file system volume
                              pfile(g) process file format produced by ldp
Intro-c(c) INTRO. TO INTERPROCESS MESSAGE FORMATS
               Intro-g(g) INTRO. TO MERT FILE FORMATS
                                     freemsg(b) free up message buffer
                                                freemsg(b) free up message buffer
                                                freeseg(a) remove a segment ID from proc-sgm-table
                                                fs(g) format of MERT file system volume
                                                fsize(c) get size of file
                                                fstat(c) get status of open file
                                                ftrunc(c) truncate file to given size
                          sgen(e) system generation program
                          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
                                     getmsg(a) get a message
                                     receive(a) get a message
                                     getarg(b) get argument from SUP address space
                                     getchar(a) get characters from kernel process
                                      getcsw(a) get console switch register setting
                                         stat(c) get file status
                                     nmcode(c) get name code for segment
                                    segname(b) get name of segment
                                    segname(a) get segment name
                                        fsize(c) get size of file
                                      sizeseg(a) get size of segment
                                       getty(a) get state of tty driver process
                                        statio(f) get status of asynchronous I/O
                                        fstat(c) get status of open file
                                      getime(b) get system time
                                      getime(a) get time
                                     timleft(b) get time-out value for process
                                     getseg(f) get user segment
                                              getarg(b) get argument from SUP address space
                                                getchar(a) get characters from kernel process
                                        getcsw(a) get console switch register setting
                                        getime(a) get time
                                              getime(b) get system time
                                           getmsg(a) get a message
                                       getseg(f) get user segment
                                            getty (a) get state of tty driver process
                                         gettype(a) get a message of given type
                                    ktime(e) give detailed kernel time of a command
         mgetlim(a) get a message of type between given limits
                         ftrunc(c) truncate file to given size
                      gettype(a) get a message of given type
                     mgettype(a) get a message of given type
                     msgtype (a) get a message of given type
                                                 growseg (a) increase or decrease the size of a segment
                                                 icat(d) concatenate i-node
                                                 iclr(d) clear i-node
                                                 icon(d) do consistency check of i-nodes
                     freeseg(a) remove a segment ID from proc-sgm-table
                        openseg(a) add a segment id to the process segment table
                                                 idmp(d) dump i-node
                         xusr(e) extract user core image from process core dump
                                     growseg(a) increase or decrease the size of a segment
                                      lockid(a) increment the lock count on a segment
```

adduser(a) increment user count on a process

inhibit(a) run process at priority one init(c) initialize file system manager init(c) initialize file system manager (P-Mgr)pinit(c) initialize the process manager isnp(d) snap i-node contents openi(c) open file specified by inode number icat(d) concatenate i-node iclr(d) clear i-node idmp(d) dump i-node icon(d) do consistency check of i-nodes queuem (b) queue message on input queue Intro-c(c) INTRO. TO INTERPROCESS MESSAGE FORMATS 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 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 ioqueuem(b) send message to I/O device driver ioqueuem (a) send an 1/O message setio(f) set I/O mode of file I/o-messages(c) INTRO. TO I/O PROCESS-MESSAGES 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(e) physical ioqueuem (a) send an I/O message ioqueuem(b) send message to I/O device driver statio(f) get status of asynchronous I/O uniolock (b) unlock segment for 1/0 isnp(d) snap i-node contents spacaloc(a) allocate space for segment; add it to the proc. virtual addr. space jobchg(a) change control to next process kdb(e) kernel debugger kdmp(e) dump system state into core file kernel debugger kdb(e) Intro-b(b) INTRO. TO KERNEL EMT CALLS kpkill(e) terminate a kernel process (superuser) kprc(g) kernel process translation file kernel process getchar(a) get characters from ktime(e) give detailed kernel time of a command kpkill(e) terminate a kernel process (superuser) kprc(g) kernel process translation file ktime(e) give detailed kernel time of a command ldp(e) load a process pfile (g) process file format produced by Idu(e) load a user process with public libraries Idu(e) load a user process with public libraries get a message of type between given limits...mgetlim(a) link(c) link to a file link(c) link to a file ldp(e) load a process (Mem-Mgr)load(c) to memory manager: load a process ldu(e) load a user process with public libraries alockseg(a) lock a segment in memory and set write back

lockseg(a)

lock a segment in memory

```
(Mem-Mgr)lock(c) to memory manager: process lock a segment
                         ulockid(a) decrement lock count of segment
                        ulockseg(a) decrement lock count of segment
                        lockid(a) increment the lock count on a segment
                                     plock(f) lock process in memory
                                     iolock(b) lock segment for I/O
                                               lock(f) semaphores (USG Version)
                                               lockid(a) increment the lock count on a segment
                                               lockseg(a) lock a segment in memory
                               errproc(e) error
                                               logger
                                   copyseg(a)
                                               make a copy of a segment
                                    mknod(c) make a directory or a special file
                                   punswap(a) make a process non-swap
                                   sunswap(a) make a segment non-swap
                 (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
                                               manager: terminate a process
                 (Mem-Mgr)term(c) to memory
                     init(c) initialize file system
                                               manager
                                               manager
            (P-Mgr)pinit(c) initialize the process
                                               map segid/offset to virtual address
                                     iomap(b)
                                                mdate(c) modify date of file
                                                (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
                   alockseg(a) lock a segment in
                                                memory and set write back
                                                memory manager: load a process
                          (Mem-Mgr)load(c) to
                                                memory manager: process lock a segment
                          (Mem-Mgr)lock(c) to
                                                memory manager: terminate a process
                         (Mem-Mgr)term(c) to
                    lockseg(a) lock a segment in
                                                memory
                        plock (f) lock process in
                                                memory
                                                message at termination of process-mgr-created process
                               (P-Mgr)pwait(c)
                                                message buffer
                            alocmsg(b) allocate
                            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
                               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
Process-mgr(c) INTRO. TO PROCESS-MANAGER MESSAGES
                                                 messink(b) return a message
                                                 mgetlim(a) get a message of type between given limits
                                                 mgettype(a) get a message of given type
                                                 mknod(c) make a directory or a special file
                            setmap(a) set access, mode and starting segmentation register
                               chmod(c) change mode of file
```

setio(f) set I/O mode of file

mdate(c) modify date of file mount(c) mount file system mount file system
mount(c) mount file system fmove(c) move file into a contiguous area fmove(d) move file into contiguous area fmove(f) move file into contiguous area mreceive(a) get a message of type -1 (acknowledgement) msg(f) send and receive messages (USG Version) msgport(f) send message to a process connected to a port msgrecv(f) receive message msgsend(f) send message to a process msgtype(a) get a message of given type nmcode(c) get name code for segment segname(b) get name of segment unblkseg(a) unblock a named segment segname(a) get segment name jobchg(a) change control to next process nmcode(c) get name code for segment queuemn(b) queue message with no acknowledgement expected pswap(a) remove non-swap status from a process sswap(a) remove non-swap status from a segment punswap(a) make a process non-swap sunswap(a) make a segment non-swap openi(c) open file specified by inode number inhibit(a) run process at priority one sleep(a) set a bit pattern to sleep on exec(c) open file for execution openi(c) open file specified by inode number fstat(c) get status of open file open(c) open file fork(c) change count on open files and add capabilities open(c) open file openi(c) open file specified by inode number openseg(a) add a segment id to the process segment table putchar(a) output characters to character device driver chown(c) change owner of file dqtype(b) dequeue a particular message type sleep(a) set a bit pattern to sleep on psleep(b) put process to sleep on bit pattern wake up processes sleeping on bit pattern...pwakeup(b) wakeup all processes sleeping on a pattern...wakeup(a) delcap(c) delete capability from process PCB pcp(e) physical copy pcreat(f) creat new process permit(a) run process at priority zero pfile(g) process file format produced by ldp pcp(e) physical copy pio(e) physical I/O pio(e) physical I/O pkill(e) terminate a process (superuser) plock(f) lock process in memory (P-Mgr)MSTERM(c) terminate a process and dump core (P-Mgr)P_CREAT(c) create a process from a file (P-Mgr)pinit(c) initialize the process manager process-mgr-created process... (P-Mgr)pwait(c) message at termination of send message to a process connected to a port...msgport(f) sendport(a) send message through port sysproc(f) system ports setprior(a) set priority of process inhibit(a) run process at priority one permit(a) run process at priority zero allocate space for segment; add it to the proc. virtual addr. space...spacaloc(a) tkill(e) terminate all processes associated with a terminal psignal(b) send events to processes on a control channel wakeup(a) wakeup all processes sleeping on a pattern

pwakeup(b) wake up processes sleeping on bit pattern

Process-mgr(c) INTRO. TO PROCESS-MANAGER MESSAGES to memory manager: terminate a process...(Mem-Mgr)term(c) I/o-messages(c) INTRO. TO I/O PROCESS-MESSAGES (P-Mgr)pwait(c) message at termination of process-mgr-created process at termination of process-mgr-created process...(P-Mgr)pwait(c) message to system scheduler: terminate a process... (System Scheduler) term (c) freeseg(a) remove a segment ID from proc-sgm-table pfile(g) process file format produced by ldp Intro-d(d) INTRO. TO FILE SYSTEM UTILITY PROGRAMS Intro-e(e) INTRO. TO MERT UTILITY PROGRAMS psignal(b) send events to processes on a control channel psleep(b) put process to sleep on bit pattern pstart(a) start process pswap(a) remove non-swap status from a process ptimer(b) set time-out value for process ldu(e) load a user process with public libraries punswap(a) make a process non-swap putarg(b) put argument into SUP address space psleep(b) put process to sleep on bit pattern putarg(b) put argument into SUP address space putchar(a) output characters to character device driver pwakeup(b) wake up processes sleeping on bit pattern qsleep(f) stop execution for small interval queuem(b) queue message on input queue queuemn(b) queue message with no acknowledgement expected queuem (b) queue message on input queue expected... queuemn(b) queue message with no acknowledgement queuem (b) queue message on input queue qwait(f) check for child process termination read(c) read from file read(c) read from file recdmn(d) reconfiguration daemon msgrecv(f) receive message msg(f) send and receive messages (USG Version) receive(a) get a message recon(d) reconfigure file system recdmn(d) reconfiguration daemon recon(d) reconfigure file system getcsw(a) get console switch register setting set access, mode and starting segmentation register...setmap(a) rmovseg(a) remove a segment from a process virtual address space freeseg(a) remove a segment ID from proc-sgm-table unlink(c) remove directory entry pswap(a) remove non-swap status from a process sswap(a) remove non-swap status from a segment messink(b) return a message rti(a) return from trap riteback (b) set altered bit on a segment address space... rmovseg(a) remove a segment from a process virtual rti(a) return from trap run(e) run an environment (superuser) inhibit(a) run process at priority one permit(a) run process at priority zero run(e) run an environment (superuser) (System Scheduler)term(c) to system scheduler: terminate a process process...(System Scheduler) term(c) to system scheduler: terminate a iomap(b) map segid/offset to virtual address spacaloc(a) allocate space for segment; add it to the proc. virtual addr. space

iolock(b) lock
uniolock(b) unlock
dropseg(a) drop a
rmovseg(a) remove a
freeseg(a) remove a
openseg(a) add a
alockseg(a) lock a
lockseg(a) lock a
segment for I/O
segment from a process virtual address space
segment ID from proc-sgm-table
segment id to the process segment table
segment in memory and set write back
segment in memory

segname(a) get segment name sunswap(a) make a segment non-swap openseg(a) add a segment id to the process segment table writeseg(a) force a segment to be written back addseg(a) add a segment to the process address space setmap(a) set access, mode and starting segmentation register increase or decrease the size of a segment...growseg(a) to memory manager: process lock a segment...(Mem-Mgr)lock(c) segname(a) get segment name segname(b) get name of segment lock(f) semaphores (USG Version) sendcpmsg(a) send a capability message sndmsgfrom(a) send a message from a process sendmsg(a) send a message ioqueuem(a) send an I/O message msg(f) send and receive messages (USG Version) event(a) send event to a process sendevent(b) send event to a process psignal(b) send events to processes on a control channel sendev(f) send event(s) 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 sendcpmsg(a) send a capability message sendevent(b) send event to a process sendev(f) send event(s) sendmsg(a) send a message sendport(a) send message through port sleep(a) set a bit pattern to sleep on setmap(a) set access, mode and starting segmentation register riteback(b) set altered bit on a segment setio(f) set I/O mode of file setprior(a) set priority of process setty(a) set state of tty driver process setime(b) set system time ptimer(b) set time-out value for process toutset(a) set time-out setime(a) set time setdspac(a) set user-supervisor d-space bits alockseg(a) lock a segment in memory and set write back setdspac(a) set user-supervisor d-space bits setime(a) set time setime(b) set system time setio(f) set I/O mode of file register... setmap(a) set access, mode and starting segmentation setprior(a) set priority of process getcsw(a) get console switch register setting setty(a) set state of tty driver process sgen(e) system generation program growseg(a) increase or decrease the size of a segment fsize(c) get size of file sizeseg(a) get size of segment ftrunc(c) truncate file to given sizeseg(a) get size of segment psleep(b) put process to sleep on bit pattern sleep(a) set a bit pattern to sleep on sleep(a) set a bit pattern to sleep on wakeup(a) wakeup all processes sleeping on a pattern pwakeup(b) wake up processes sleeping on bit pattern qsleep(f) stop execution for small interval isnp(d) snap i-node contents sndmsgfrom(a) send a message from a process

spacaloc(a) allocate space for segment; add it to

the proc. virtual addr. space...

falloc(c) allocate contiguous space for a file

falloc(f) allocate space for contiguous file

```
space...spacaloc(a) allocate space for segment; add it to the proc. virtual addr.
    add a segment to the process address space...addseg(a)
a segment from a process virtual address space...dropseg(a) drop
         falloc(d) allocate contiguous file space
getarg(b) get argument from SUP address
                                          space
putarg(b) put argument into SUP address
                                          space
a segment from a process virtual address
                                          space...rmovseg(a) remove
 segment; add it to the proc. virtual addr. space...spacaloc(a) allocate space for
         mknod(c) make a directory or a special file
                      openi(c) open file
                                         specified by inode number
                                          sswap(a) remove non-swap status from a segment
                               pstart(a)
                                          start process
                              startty(e) start up tty
         setmap(a) set access, mode and starting segmentation register
                                          startty (e) start up tty
                                          stat(c) get file status
                  kdmp(e) dump system state into core file
                  tdmp(e) dump system state into core file
                            getty (a) get state of tty driver process
                             setty(a) set state of tty driver process
                                          statio(f) get status of asynchronous I/O
             pswap(a) remove non-swap status from a process
              sswap(a) remove non-swap status from a segment
                            statio(f) get status of asynchronous I/O
                             fstat(c) get status of open file
                          stat(c) get file status
                               qsleep(f) stop execution for small interval
                              stoptty(e) stop tty
                                          stoptty(e) stop tty
                                          sunswap(a) make a segment non-swap
            getarg(b) get argument from SUP address space
            putarg(b) put argument into SUP address space
                         sync(c) update super-block
      kpkill(e) terminate a kernel process
                                          (superuser)
             pkill(e) terminate a process
                                         (superuser)
             run(e) run an environment (superuser)
                  Intro-a(a) INTRO. TO SUPERVISOR EMT TRAPS
                   getcsw(a) get console switch register setting
                                          sync(c) update super-block
                                          sysproc(a) system process
                                          sysproc(f) system ports
     Intro-f(f) INTRO. TO MERT UNIX SYSTEM CALLS
                                sgen(e) system generation program
                     init(c) initialize file system manager
                              sysproc(f) system ports
                              sysproc(a) system process
           (System Scheduler) term(c) to system scheduler: terminate a process
                   terminate a process... (System Scheduler) term (c) to system scheduler:
                         kdmp(e) dump system state into core file
                         tdmp(e) dump system state into core file
                          getime(b) get system time
                           setime(b) set system time
            Intro-d(d) INTRO. TO FILE SYSTEM UTILITY PROGRAMS
              fs(g) format of MERT file system volume
 add a segment id to the process segment table...openseg(a)
                                          tdmp(e) dump system state into core file
 terminate all processes associated with a terminal...tkill(e)
                               kpkill(e) terminate a kernel process (superuser)
                   (P-Mgr)MSTERM(c) terminate a process and dump core
                                pkill(e) terminate a process (superuser)
(Mem-Mgr)term(c) to memory manager: terminate a process
  Scheduler) term(c) to system scheduler: terminate a process... (System
                                 tkill(e) terminate all processes associated with a terminal
             (P-Mgr)pwait(c) message at termination of process-mgr-created process
```

qwait(f) check for child process termination
 sendport(a) send message through port

ktime(e) give detailed kernel time of a command

PA-1C600-01 Section 11 Issue 1, October 1977 AT&TCo SPCS

```
getime(a) get time
                       getime(b) get system time
                               ptimer(b) set time-out value for process
                               timleft(b) get time-out value for process
                               toutset(a) set time-out
                               setime(a) set time
                        setime(b) set system time
                                              timleft(b) get time-out value for process
                                  terminal...
                                             tkill(e) terminate all processes associated with a
                                              toutset(a) set time-out
                      kprc(g) kernel process translation file
                           rti(a) return from trap
  Intro-a(a) INTRO. TO SUPERVISOR EMT TRAPS
                                   ftrunc(c) truncate file to given size
                         getty(a) get state of tty driver process
                         setty(a) set state of tty driver process
                           startty(e) start up tty
                             stoptty(e) stop tty
               mreceive(a) get a message of type -1 (acknowledgement)
                mgetlim(a) get a message of type between given limits
      dqtype(b) dequeue a particular message type
            gettype(a) get a message of given type
          mgettype(a) get a message of given type
          msgtype(a) get a message of given type
                                              ulockid(a) decrement lock count of segment
                                              ulockseg(a) decrement lock count of segment
                                              umount(c) dismount file system
                                              unblkseg(a) unblock a named segment
                                unblkseg(a)
                                             unblock a named segment
                                              uniolock(b) unlock segment for I/O
                                              unlink(c) remove directory entry
                                uniolock(b) unlock segment for I/O
                                    sync(c) update super-block
                             xusr(e) extract user core image from process core dump
                       adduser(a) increment user count on a process
                               ldu(e) load a user process with public libraries
                               getseg(f) get user segment
                             setdspac(a) set user-supervisor d-space bits
                         lock(f) semaphores (USG Version)
           msg(f) send and receive messages (USG Version)
      Intro-d(d) INTRO. TO FILE SYSTEM UTILITY PROGRAMS
               Intro-e(e) INTRO. TO MERT UTILITY PROGRAMS
                      ptimer(b) set time-out value for process
                      timleft(b) get time-out value for process
         attach(a) attach process to interrupt vector
      detach(a) detach process from interrupt vector
                   lock (f) semaphores (USG
                                             Version)
     msg(f) send and receive messages (USG
                                             Version)
        space for segment; add it to the proc.
                                             virtual addr. space...spacaloc(a) allocate
   dropseg(a) drop a segment from a process virtual address space
rmovseg(a) remove a segment from a process virtual address space
               iomap(b) map segid/offset to virtual address
           fs(g) format of MERT file system volume
                                   waitev(f) wait for an event
                         cwait(a) conditional wait for event
                                              waitev(f) wait for an event
                                pwakeup(b)
                                             wake up processes sleeping on bit pattern
                                 wakeup(a)
                                             wakeup all processes sleeping on a pattern
                                             wakeup(a) wakeup all processes sleeping on a pattern
                            chdir(c) change
                                             working directory
          lock a segment in memory and set write back...alockseg(a)
                                    write(c) write to file
                                             write(c) write to file
                                             writeseg(a) force a segment to be written back
```

writeseg(a) force a segment to be written back

PA-1C600-01 GRELICATION INSTRUCTION Section 11 Issue 1, October 1977 AT&TCo SPCS

xusr(e) extract user core image from process core dump

permit(a) run process at priority zero

INTRODUCTION TO MERT PROGRAMMER'S MANUAL

This manual provides a description of the internal features of the MERT operating system. It is meant to be used as a supplement to the UNIX PROGRAMMER'S MANUAL. A more general overview of the MERT operating system is provided by the two technical memoranda:

- H. Lycklama, D. L. Bayer, A Structured Operating System for a PDP-11/45, TM-75-1352-4, May 6, 1975.
- D. L. Bayer, H. Lycklama, *MERT, A Multi-Environment Real-Time Operating System*, TM-75-1352-7, July 18, 1975. Also published in: ACM Operating Systems Review, Volume 9, Number 5, November 1975, pp.33-42.

Within the area it surveys, this manual attempts to be as complete and timely as possible. A conscious decision was made to describe each program in exactly the state it was in at the time its manual section was prepared. In particular, the desire to describe something as it should be, not as it is, was resisted. Inevitably, this means that many sections will soon be out of date.

This manual is divided into seven sections:

- A. Supervisor Process Calls
- B. Kernel Process Calls
- C. Inter-Process Message Formats
- D. File System Utilities
 - E. MERT-UNIX Programs
 - F. MERT-UNIX System Calls
 - G. MERT File Formats (new) and a seed that a seed the seed to be a see

The PCB (Process Control Block) of a supervisor-user process is described in section A. A supervisor-user process has entries into the kernel by means of EMT traps. Each one of these is described in detail. The binary object code for each routine is kept in the library file "/lib/libe.a".

The kernel process header is described in section B. A kernel process has entries into the kernel by means of another set of EMT traps. Each of these is described in detail here. The binary object code for each routine is kept in the library file "/lib/libk.a".

Inter-process communication is achieved mainly by means of messages. The header of a message is described in section C along with the contents of the various message types which are recognized by the basic system processes. These processes include kernel I/O drivers, file manager, process manager, memory manager and system process scheduler.

Section D describes the utility programs which deal with the file system. For those which are not described, the reader is referred to the UNIX PROGRAMMER'S MANUAL.

Section E describes the various utility programs which are used to build special MERT files such as process images and the boot image. It also includes desriptions of programs which will run under MERT-UNIX making use of the new "system call's" added to the MERT version of the UNIX supervisor.

Section F describes all of the new "UNIX system call's" added to the MERT-UNIX supervisor. The binary object code for each routine is kept in the library file "/lib/libr.a".

Section G describes the format of various files, most notably the layout of a file system volume and the i-node structure.

Most sections begin with an introduction section. Each section consists of a number of independent entries of a page or so. Entries within each section are alphabetized behind the introduction section, except for Section C where these rules apply only to the first part dealing with file manager messages. The last part of Section C deals with various resource management processes; see the table of contents to understand the organization. The page numbers of each entry start at 1.

All entries are based on a common format, not all of whose subsections will always appear.

The name section repeats the entry name and gives a very short description of its purpose.

The *synopsis* summarizes the use of the program being described. A few conventions are used, particularly in the Commands section:

B o 1 d f a c e words are considered literals, and are typed just as they appear.

Square brackets ([]) around an argument indicate that the argument is optional. When an argument is given as "name", it always refers to a file name.

Ellipses "..." are used to show that the previous argument-prototype may be repeated.

A final convention is used by the commands themselves. An argument beginning with a minus sign "-" is often taken to mean some sort of flag argument even if it appears in a position where a file name could appear. Therefore, it is unwise to have files whose names begin with "-".

The *description* section discusses in detail the subject at hand.

The *files* section gives the names of files which are built into the program.

A see also section gives pointers to related information.

A diagnostics section discusses the diagnostic indications which may be produced. Messages which are intended to be self-explanatory are not listed.

The *bugs* section gives known bugs and sometimes deficiencies. Occasionally also the suggested fix is described.