INTRODUCTION

This manual gives complete descriptions of all the publicly available features of UNIX. It provides neither a general overview (see "The UNIX Time-sharing System" for that) nor details of the implementation of the system (which remain to be disclosed).

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 Inevitably, this means that many sections will soon be out of date. (The rate of change of the system is so great that a dismayingly large number of early sections had to be modified while the rest were being written. The unbounded effort required to stay up-to-date is best indicated by the fact that several of the programs described were written specifically to aid in preparation of this manual!)

This manual is divided into seven sections:

I. Commands
II. System calls
III. Subroutines
IV. Special files
V. File formats
VI. User-maintained programs
VII. Miscellaneous

Commands are programs intended to be invoked directly by the user, in contradistinction to subroutines, which are intended to be called by the user's programs. Commands generally reside in directory /bin (for binary programs). This directory is searched automatically by the command line interpreter. Some programs classified as commands are located elsewhere; this fact is indicated in the appropriate sections.

System calls are entries into the UNIX supervisor. In assembly language, they are coded with the use of the opcode "sys", a synonym for the trap instruction.

The special files section discusses the characteristics of each system "file" which actually refers to an I/O device.

The file formats section documents the structure of particular kinds of files; for example, the form of the output of the loader and assembler is given. Excluded are files used only one command, for example the assembler's intermediate files.

User-maintained programs are not considered part of the UNIX system, and the principal reason for listing them is to indicate their existence without necessarily giving a complete
description. The author should be consulted for information.
The miscellaneous section gathers odds and ends.

Each section consists of a number of independent entries of a page or so each. The name of the entry is in the upper right corner of its pages, its preparation date in the upper left. Entries within each section are alphabetized. It was thought better to avoid page numbers, since it is hoped that the manual will be updated frequently.

All entries have a common format.

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:

Underlined 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 diagnostics that may be produced. This section tends to be as terse as the diagnostics themselves.

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

The **owner** section gives the name of the person or persons to be consulted in case of difficulty. The rule has been that the last one to modify something owns it, so the owner is not necessarily the author. The owner’s initials stand for:
These three-character names also happen to be UNIX user ID's, so messages may be transmitted by the mail command or, if the addressee is logged in, by write.

At the beginning of this document is a table of contents, organized by section and alphabetically within each section. There is also a permuted index derived from the table of contents. Within each index entry, the title of the writeup to which it refers is followed by the appropriate section number in parentheses. This fact is important because there is considerable name duplication among the sections, arising principally from commands which exist only to exercise a particular system call.

This manual was prepared using the UNIX text editor ed and the formatting program roff.
<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ar</td>
<td>archive (combine) files</td>
</tr>
<tr>
<td>as</td>
<td>assembler</td>
</tr>
<tr>
<td>b</td>
<td>compile B program</td>
</tr>
<tr>
<td>bas</td>
<td>BASIC dialect</td>
</tr>
<tr>
<td>bcd</td>
<td>convert ASCII to BCD</td>
</tr>
<tr>
<td>boot</td>
<td>reboot system</td>
</tr>
<tr>
<td>cat</td>
<td>concatenate (or print) files</td>
</tr>
<tr>
<td>chdir</td>
<td>change working directory</td>
</tr>
<tr>
<td>check</td>
<td>check consistency of file system</td>
</tr>
<tr>
<td>chmod</td>
<td>change access mode of files</td>
</tr>
<tr>
<td>chown</td>
<td>change owner of files</td>
</tr>
<tr>
<td>cmp</td>
<td>compare file contents</td>
</tr>
<tr>
<td>cp</td>
<td>copy file</td>
</tr>
<tr>
<td>date</td>
<td>get date and time of day</td>
</tr>
<tr>
<td>db</td>
<td>symbolic debugger</td>
</tr>
<tr>
<td>dbppt</td>
<td>write binary paper tape</td>
</tr>
<tr>
<td>dc</td>
<td>desk calculator</td>
</tr>
<tr>
<td>df</td>
<td>find free disk space</td>
</tr>
<tr>
<td>dsw</td>
<td>delete files interactively</td>
</tr>
<tr>
<td>dtf</td>
<td>format DECTape</td>
</tr>
<tr>
<td>du</td>
<td>find disk usage</td>
</tr>
<tr>
<td>ed</td>
<td>text editor</td>
</tr>
<tr>
<td>find</td>
<td>find file with given name</td>
</tr>
<tr>
<td>for</td>
<td>compile Fortran program</td>
</tr>
<tr>
<td>form</td>
<td>generate form letter</td>
</tr>
<tr>
<td>hup</td>
<td>hang up typewriter</td>
</tr>
<tr>
<td>lbppt</td>
<td>read binary paper tape</td>
</tr>
<tr>
<td>ld</td>
<td>link editor (loader)</td>
</tr>
<tr>
<td>ln</td>
<td>link to file</td>
</tr>
<tr>
<td>ls</td>
<td>list contents of directory</td>
</tr>
<tr>
<td>mail</td>
<td>send mail to another user</td>
</tr>
<tr>
<td>mesg</td>
<td>permit or deny messages</td>
</tr>
<tr>
<td>mkdir</td>
<td>create directory</td>
</tr>
<tr>
<td>mkfs</td>
<td>initialize file system</td>
</tr>
<tr>
<td>mount</td>
<td>mount detachable file system</td>
</tr>
<tr>
<td>mv</td>
<td>move or rename file</td>
</tr>
<tr>
<td>nm</td>
<td>print namelist</td>
</tr>
<tr>
<td>od</td>
<td>octal dump of file</td>
</tr>
<tr>
<td>pr</td>
<td>print file with headings</td>
</tr>
<tr>
<td>rew</td>
<td>rewind DECTape</td>
</tr>
<tr>
<td>rkd</td>
<td>dump disk to tape</td>
</tr>
<tr>
<td>rkf</td>
<td>format RK disk</td>
</tr>
<tr>
<td>rkl</td>
<td>load disk from tape</td>
</tr>
<tr>
<td>rm</td>
<td>remove (delete) file</td>
</tr>
<tr>
<td>rmdir</td>
<td>remove (delete) directory</td>
</tr>
<tr>
<td>roff</td>
<td>run off (format) text</td>
</tr>
<tr>
<td>sdate</td>
<td>adjust date and time</td>
</tr>
<tr>
<td>sh</td>
<td>command interpreter</td>
</tr>
<tr>
<td>stat</td>
<td>get file status</td>
</tr>
<tr>
<td>strip</td>
<td>remove symbols, relocation bits</td>
</tr>
<tr>
<td>su</td>
<td>become super-user</td>
</tr>
</tbody>
</table>
sum ......................... sum file
tap .......................... manipulate DECTape
tm .......................... get time information
tty .......................... find name of terminal
type ......................... print file on IBM 2741
umount ...................... dismount removable file system
un .......................... find undefined symbols
wc .......................... get (English) word count
who .......................... who is on the system
write ....................... write to another user

II. SYSTEM CALLS

break ....................... set program break
cemt ........................ catch EMT traps
chdir ....................... change working directory
chmod ........................ change mode of file
chown ....................... change owner of file
close ....................... close open file
creat ........................ create file
exec ........................ execute program file
exit ........................ terminate execution
fork ........................ create new process
fstat ........................ status of open file
getuid ...................... get user ID
gtty ........................ get typewriter mode
ilgins ...................... catch illegal instruction trap
intr ........................ catch or inhibit interrupts
link ........................ link to file
mkdir ........................ create directory
mount ....................... mount file system
open ........................ open file
quit ........................ catch or inhibit quits
read ........................ read file
rele ........................ release processor
seek ........................ move read or write pointer
setuid ...................... set user ID
smdate ...................... set date modified of file
stat ........................ get file status
stime ........................ set system time
stty ........................ set mode of typewriter
tell ........................ find read or write pointer
time ........................ get time of year
umount ...................... dismount file system
unlink ...................... remove (delete) file
wait ........................ wait for process
write ....................... write file

III. SUBROUTINES

atof ........................ convert ASCII to floating
atoi ........................ convert ASCII to integer
cftime ...................... convert time to ASCII
exp ........................ exponential function
fptrap..................floating-point simulator
ftoa........................convert floating to ASCII
get..................................get character
itoa................................convert integer to ASCII
log..................................logarithm base e
msg...............................print string on typewriter
ptime..................................print time
putc..................................write character or word
sin..................................sine, cosine
switch...............................transfer depending on value

IV. SPECIAL FILES

mem...........................core memory as file
ppt............................punched paper tape
rf0...............................RF disk file
rk0................................RK disk file
tap0,...,tap7...............DECtape file
tty.................................console typewriter
tty0,...,tty5..................remote typewriter

V. FILE FORMATS

a.out..........................assembler and loader output
archive.........................archive file
bppt................................binary paper tape format
core..............................core image file
directory........................directory format
file system.....................file system format
passwd..........................password file
uids................................map names to user ID’s
utmp.............................logged-in user information

VI. USER MAINTAINED PROGRAMS

basic..........................DEC supplied BASIC
bj..................................the game of black jack
cal..................................print calendar
chess.............................the game of chess
das..................................disassembler
dl1..................................load DEC binary paper tapes
dpt..................................read DEC ASCII paper tapes
moo..................................the game of MOO
sort..............................sort a file
ttt..................................the game of tic-tac-toe

VII. MISCELLANEOUS

as2..........................assembler’s pass 2
ascii.............................map of ASCII
ba.................................B assembler
bc.................................B compiler
bilib ...................... B interpreter library
bproc ...................... boot procedure
brt1,brt2 .................. B start and finish
f1,f2,f3,f4 ................ Fortran compiler passes
glob ...................... argument expander
init ...................... initializer process
kbd ...................... map of TTY 37 keyboard
liba ...................... standard assembly-language library
libb ...................... standard B library
libf ...................... standard Fortran library
login, logout .............. logging on and logging off the system
msh ...................... mini Shell
suftab .................... roff's suffix table
tabs ...................... set tab stops on typewriter
INDEX

chmod(I); change access mode of files
sdate(I); adjust date and time
mail(I); send mail to another user
write(I); write to another user
ar(I); a.out(V); assembler and loader output
archive(V); archive (combine) files
archive(V); archive file
archive(V); archive file
glob(VII); argument expander
ar(I); archive (combine) files
dpt(VI); read DEC ASCII paper tapes
tie(I); convert ASCII to BCD
atof(III); convert ASCII to floating
atoi(III); convert ASCII to integer
ascii(VII); map of ASCII
ctime(III); convert time to ASCII
convert floating to ASCII
itoa(III); convert integer to

a.out(V); assembler and loader output
as(I); assembler
ba(VII); B assembler
as2(VII); assembler's pass 2
liba(VII); standard assembly-language library
as2(VII); assembler's pass 2
atof(III); convert ASCII to floating
atoi(III); convert ASCII to integer

ba(VII); B compiler
bc(VII); B library
bilib(VII); standard B program
libb(VII); standard
b(I); compile B start and finish
bret1,bret2(VII); base e
log(III); logarithm
bas(I); BASIC dialect

basic(VI): DEC supplied BASIC
d(ba(VII); B assembler
bcd(I); convert ASCII to BCD
bc(VII); B compiler
su(I); become super-user
b(I); compile B program
bilib(VII); B interpreter library

bppt(V); binary paper tape format
dbppt(I); write binary paper tape
lbppt(I); read binary paper tape
dli(VI); load DEC binary paper tapes
remove symbols, relocation
bits...strip(I);
bj(VI); the game of black jack

- ix -
bj(VI): the game of black jack
bproc(VII): boot procedure
boot(I): reboot system
bppt(V): binary paper tape format
bproc(VII): boot procedure
break
break(I): set program break
brt1,brt2(VII): B start and finish calculator
cal(VI): print calendar
cal(VI): print calendar
cemt(I): catch EMT traps
cemt(I): catch EMT traps
ilgins(I): catch illegal instruction trap
intr(I): catch or inhibit interrupts
intr(I): catch or inhibit quits
quit(I): concatenate (or print) files
quit(I): concatenate (or print) files
chmod(I): change access mode of files
chmod(I): change mode of file
chown(I): change owner of file
chown(I): change owner of files
chdir(I): change working directory
chdir(I): change working directory
putc(I): write character or word
get(I): get character
chdir(I): change working directory
chdir(I): change working directory
check(I): check consistency of file system
check(I): check consistency of file system
chess(VI): the game of chess
chess(VI): the game of chess
chmod(I): change access mode of files
chmod(I): change mode of file
chown(I): change owner of files
chown(I): change owner of file
close(I): close open file
close(I): close open file
cmp(I): compare file contents
(combine) files
ar(I): archive
sh(I): command interpreter
cmp(I): compare file contents
b(I): compile B program
for(I): compile Fortran program
f1,f2,f3,f4(VII): Fortran compiler passes
bc(VII): B compiler
cat(I): concatenate (or print) files
cat(I): concatenate (or print) files
check(I): check consistency of file system
tty(IV): console typewriter
ls(I): list contents of directory
cmp(I): compare file contents
bcd(I): convert ASCII to BCD
atof(III): convert ASCII to floating
atof(III): convert ASCII to floating
atoi(III): convert ASCII to integer
ftoa(III): convert floating to ASCII
itoa(III): convert integer to ASCII
ctime(III): convert time to ASCII

-x-
cp(I): copy file
core(V): core image file
mem(IV): core memory as file
core(V): core image file
sin(III): sine,
cos(II): cosine
wc(I): get (English) word
count
mkdir(I): create directory
mkdir(II): create directory
creat(II): create file
fork(II): create new process
creat(II): create file
time(III): convert time to ASCII
das(VI): disassembler
date(I): get
date and time of day
date(I): get date and time of day
sdate(I): adjust
date modified of file
date(I): get date and time of day
smdate(II): set

db(I): symbolic debugger
dbppt(I): write binary paper tape
dc(I): desk calculator
debugger
basic(VI):
DEC ASCII paper tapes
dli(VI): load
DEC binary paper tapes
tap0,...,tap7(IV):
DEC supplied BASIC
dtf(I): format
DECTape
tap(I): manipulate
DECTape
rew(I): rewind
rmdir(I): remove
rmdir(II): remove
(delete) directory
rm(I): remove
(delete) file
dsw(I): delete files interactively
dsw(I):
dsp(I): delete files interactively
unlink(I): remove
don't messages
msg(I): permit or
depend on value
switch(III): transfer
dc(I):
desk calculator
dc(I):
detachable file system
df(I): find free disk space
mount(I): mount
dialog
detachable file system
df(I): find free disk space
directory
bas(I): BASIC
directory format
directory
changer(I): change working
directory
changer(II): change working
directory
ls(I): list contents of
directory
mkdir(I): create
directory
mkdir(II): create
directory
rmdir(I): remove (delete)
directory(II): change working
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
directory
rmdir(I): remove (delete)
umount(II): dismount file system
umount(I): dismount removable file system
dll(VI): load DEC binary paper tapes
dpt(VI): read DEC ASCII paper tapes
dsw(I): delete files interactively
dtf(I): format DECTape
du(I): find disk usage
dump disk to tape
dump of file
ed(I): text editor
ed(I): text editor
log(III): logarithm base
cemt(II): catch
wc(I): get
exec(II): execute program file
exec(II): execute program file
exit(II): terminate execution
exp(III): exponential function
exp(III): exponential function
cmp(I): compare
type(I): print
stat(I): get
stat(II): get
file system(V): file system format
check consistency of
mkfs(I): initialize
mount(I): mount detachable
mount(II): mount
umount(I): dismount removable
umount(II): dismount
find(I): find
pr(I): print
archive(V): archive
chmod(II): change mode of
chown(II): change owner of
close(II): close open
core(V): core image
cp(I): copy
creat(II): create
exec(II): execute program
fstat(II): status of open
link(II): link to
ln(I): link to
mem(IV): core memory as
mv(I): move or rename
od(I): octal dump of
open(II): open
passwd(V): password
read(II): read
rf0(IV): RF disk
rk0(IV): RK disk
rm(I): remove (delete)
dsw(I): delete files interactively
ar(I): archive (combine) files
concatenate (or print) files...cat(I):
change access mode of files...chmod(I):
chown(I): change owner of files
set date modified of file...smdate(II):
sort(VI): sort a file
sum(I): sum file
tap0,...,tap7(IV): DECTape file
unlink(II): remove (delete) file
du(I): find disk usage
find(I): find file with given name
df(I): find free disk space
tty(I): find name of terminal
tell(II): find read or write pointer
un(I): find undefined symbols
find(I): find file with given name

brt1,brt2(VII): B start and finish
ftoa(III): convert floating to ASCII
atof(III): convert ASCII to floating
fptrap(III): floating-point simulator
for(I): compile Fortran program
fork(II): create new process
form(I): generate form letter
dtf(I): format DECTape
rkf(I): format RK disk
roff(I): run off (format) text
bppt(V): binary paper tape format
directory(V): directory format
file system(V): file system format
form(I): generate form letter
f1,f2,f3,f4(VII): Fortran compiler passes
libf(VII): standard Fortran library
for(I): compile Fortran program
fptrap(III): floating-point simulator
df(I): find free disk space
rkI(II): load disk from tape
fstat(II): status of open file
ftoa(III): convert floating to ASCII function
exp(III): exponential

bj(VI): the game of black jack
chess(VI): the game of chess
moo(VI): the game of MOO
ttt(VI): the game of tic-tac-toe
form(I): generate form letter
get(III): get character
date(I): get date and time of day
wc(I): get (English) word count
stat(I): get file status
stat(II): get file status
tm(I): get time information
time(II): get time of year
gtty(II): get typewriter mode
getuid(II): get user ID
get(III): get character
find(I): find file with
    hup(I): hang up typewriter
pr(I): print file with
    getuid(I): get user ID
type(I): print file on
    getgrent(I): get user
getuid(I): set user
    uids(V): map names to user
    ilgins(I): catch illegal
    core(V): core
    tm(I): get time
    utmp(V): logged-in user
    intr(I): catch or
    quit(I): catch or
    init(VII): initialize file system
    init(VII): initializer process
    init(VII): initializer process
    init(VII): instruction trap
    illegal instruction trap
    image file
    information
    inhibit interrupts
    inhibit quits
    initialize file system
    initializer process
    intr(I): catch or inhibit
    interrupt

bj(VI): the game of black
kbd(VII): map of TTY 37

form(I): generate form
    library...

bilib(VII): B interpreter
    standard assembly-language
    libb(VII): standard B
    libf(VII): standard Fortran
ld(I): link editor (loader)
link(I): link to file
ln(I): link to file
link(I): link to file
ls(I): list contents of directory
ln(I): link to file

full(VI): load DEC binary paper tapes
rkl(I): load disk from tape

a.out(V): assembler and
    loader output
ld(I): link editor
    (loader)
log(III): logarithm base e
utmp(V): logged-in user information
logout(VII): logging on and
    logging off the system...login,
login, logout(VII): logging on and logging off the system
logging off the system... login,
the system... login,

mail(I): send

tap(I):
uids(V): map names to user ID’s
ascii(VII): map of ASCII
kbd(VII): map of TTY 37 keyboard
mem(IV): core memory as file
memory as file
mesg(I): permit or deny messages
msh(VII): mini Shell
mkdir(I): create directory
mkdir(II): create directory
mkfs(I): initialize file system
mode of file
mode of files
mode of typewriter
modified of file

chmod(II): change
chmod(I): change access
stty(II): set
getty(II): get typewriter
smdate(II): set date
moo(VI): the game of MOO

mount(I): mount detachable file system
mount(II): mount file system
mv(II): mount(II): mount detachable file system
mount(II): mount file system
move or rename file
move read or write pointer
msh(VII): mini Shell
mv(I): move or rename file
name of terminal

find(I): find file with given
nm(I): print
uids(V): map
fork(II): create
	
tty(I): find
find(I): find file
find file with given

nm(I): print
uids(V): map
fork(II): create

rcoff(I): run

close(II): close
fstat(II): status of
open(II): open file
open file
open file
open file
open file

rm(I): print namelist.
names to user ID’s
new process

od(I): octal dump of file
od(I): octal dump of file
off (format) text
open file
open file
open file
open file

ppp(I): punched
papertape
papertape
papertape
papertape

- xv -
dli(VI): load DEC binary     paper tapes
  dpt(VI): read DEC ASCII    paper tapes
  as2(VII): assembler's     pass 2
       Fortran compiler
  passwd(V):                passes...f1,f2,f3,f4(VII):  
  .  mesg(I):               passwd(V): password file
            permit or deny messages
seek(II): move read or write pointer
  tell(II): find read or write
  cal(VII):                 ppt(IV): punched paper tape
  type(I):                  pr(I): print file with headings
  pr(I): print calendar     print file on IBM 2741
  cat(I): concatenate (or   print file with headings
       nm(I): print namelist
       mesg(III): print string on typewriter
       ptime(III): print time
       bproc(VII): boot procedure
     fork(II): create new process
     init(VII): initializer process
     rele(II): release processor
     wait(II): wait for process
       break(II): set program break
exec(II): execute program
  b(I): compile B program
for(I): compile Fortran
  ptime(III): print time punched paper tape
quit(II): catch or inhibit quit(II): catch or inhibit quits
     quitpt(I): read binary paper tape
     dpt(VI): read DEC ASCII paper tapes
     read(II): read file
seek(II): move read or write pointer
  tell(II): find read or write pointer
boot(I): reboot system
  rele(II): release processor
strip(I): remove symbols, rele(II): release processor
       tty0,...,tty5(IV): relocation bits
       umount(I): dismount remote typewriter
       rmdir(I): remove (delete) directory
       rm(I): remove (delete) file
       unlink(I): remove (delete) file
strip(II): remove symbols, relocation bits
mv(I): move or rename file
rew(I): rewind DECtape
       rfo(IV): RF disk file
       rk0(IV): RK disk file
rkf(I): format RK disk
       rkd(I): dump disk to tape
rkf(I): format RK disk
rkl(I): load disk from tape
rk0(IV): RK disk file
rmdir(I): remove (delete) directory
rm(I): remove (delete) file
roff(I): run off (format) text

suftab(VII): roff’s suffix table
roff(I): run off (format) text
sdate(I): adjust date and time
seek(II): move read or write pointer

mail(I): send mail to another user
smdate(II): set date modified of file
stty(II): set mode of typewriter
break(II): set program break
stime(II): set system time
tabs(VII): set tab stops on typewriter
setuid(II): set user ID

msh(VII): mini Shell
sh(I): command interpreter

fptrap(III): floating-point simulator

sin(III): sine, cosine
sin(III): sine, cosine
smdate(II): set date modified of file
sort(VI): sort a file
sort(VI): sort a file

df(I): find free disk
liba(VII): standard assembly-language library
libb(VII): standard B library
libf(VII): standard Fortran library
brt1, brt2(VII): B start and finish
stat(I): get file status
stat(II): get file status
fstat(II): status of open file
stat(I): get file status
stat(II): get file status

tabs(VII): set tab stops on typewriter
msg(III): print
strip(I): remove symbols, relocation bits
stty(II): set mode of typewriter

suftab(VII): roff’s suffix table
suftab(VII): roff’s suffix table

sum(I): sum file
su(I): become super-user
sum(I): sum file

basic(VI): DEC supplied BASIC

db(I): symbolic debugger
strip(I): remove symbols, relocation bits
un(I): find undefined symbols
file system(V): file system format
stime(II): set system time
boot(I): reboot system
check consistency of file system...
check(I):
and logging off the
mkfs(I): initialize file
mount detachable file
mount(II): mount file
dismount removable file
umount(II): dismount file
   file
who(I): who is on the
tabs(IV): set
suftab(II): roff's suffix

bppt(V): binary paper
dbppt(I): write binary paper
lbppt(I): read binary paper
ppt(IV): punched paper
rkd(I): dump disk to
tape
rkl(I): load disk from
load DEC binary paper
dpt(IV): read DEC ASCII paper

tty(I): find name of
tty(II):
   ed(I):
roff(I): run off (format)
ttt(VI): the game of
tm(I): get
date(I): get date and
time(II): get
time(III): convert
ptime(III): print
sdate(I): adjust date and
stime(II): set system

switch(III):
catch illegal instruction
cemt(IV): catch EMT

kbd(VII): map of

gtty(II): get
tty(I): find name of terminal
tty(IV): console typewriter
tty0, ..., tty5(IV): remote typewriter
tty0, ..., tty5(IV): remote

un(I): find

system...login, logout(II): logging on system
system...mount(I): mount system
system...umount(I): umount system
system(V): file system format
system tab stops on typewriter
tabs(II): set tab stops on typewriter
tape format
tape
tape
tape
tape
tape
tape...dl(VI):
tapes
tape
tap(I): manipulate DECtape
tap0, ..., tap7(IV): DECtape file
tell(II): find read or write pointer
terminal
terminate execution
text editor
text
tic-tac-toe
time information
time of day
time of year
time to ASCII
time(II): get time of year
time
time
time
tm(I): get time information
transfer depending on value
trap...ilgins(II):
traps
ttt(VI): the game of tic-tac-toe
TTY 37 keyboard
tty(I): find name of terminal
tty(IV): console typewriter
tty0, ..., tty5(IV): remote typewriter

uids(IV): map names to user ID's
umount(I): dismount removable file system
umount(II): dismount file system,

undefined symbols
du(I): find disk
getuid(II): get
setuid(II): set
uids(V): map names to
utmp(V): logged-in
mail(I): send mail to another
write(I): write to another

transfer depending on
  wait(II):

  who(I):

find(I): find file
pr(I): print file
wc(I): get (English)
putc(III): write character or
  chdir(I): change
  chdir(II): change
  dbpct(I):
  putc(III):
  seek(II): move read or
tell(II): find read or
  write(I):

time(II): get time of
as2(VII): assembler's pass
type(I): print file on IBM
kbd(VII): map of TTY

un(I): find undefined symbols
unlink(II): remove (delete) file
usage
user ID
user ID's
user information
user
user
utmp(V): logged-in user information
value...switch(III):
wait for process
wait(II): wait for process
wc(I): get (English) word count
who is on the system
who(I): who is on the system
with given name
with headings
word count
word
working directory
working directory
write binary paper tape
write pointer
write pointer
write to another user
write(I): write to another user

year
  2
  2741
  37 keyboard
NAME
ar -- archive

SYNOPSIS
ar key afile name, ...

DESCRIPTION
ar maintains groups of files combined into a single archive file. Its main use is to create and update library files as used by the loader. It can be used, though, for any similar purpose.

key is one character from the set drtux, optionally concatenated with v. afile is the archive file. The names are constituent files in the archive file. The meanings of the key characters are:

d means delete the named files from the archive file.

r means replace the named files in the archive file. If the archive file does not exist, r will create it. If the named files are not in the archive file, they are appended.

l prints a table of contents of the archive file. If no names are given, all files in the archive are tabled. If names are given, only those files are tabled.

u is similar to r except that only those files that have been modified are replaced. If no names are given, all files in the archive that have been modified will be replaced by the modified version.

x will extract the named files. If no names are given, all files in the archive are extracted. In neither case does x alter the archive file.

v means verbose. Under the verbose option, ar gives a file-by-file description of the making of a new archive file from the old archive and the constituent files. The following abbreviations are used:

c copy
a append
d delete
r replace
x extract

FILES /tmp/vtma, vtmb ... temporary

SEE ALSO ld

DIAGNOSTICS "Bad usage", "afile -- not in archive format", "cannot open temp file", "name -- cannot open"
"name -- phase error", "name -- cannot create", "no archive file", "cannot create archive file", "name -- not found".

BUGS
Option 1 (table with more information) should be implemented.

There should be a way to specify the placement of a new file in an archive. Currently, it is placed at the end.

OWNER
ken, dmr
NAME as -- assembler

SYNOPSIS as name, ... 

DESCRIPTION as assembles the concatenation of name, ..., as is based on the DEC-provided assembler PAL-11R [references], although it was coded locally. Therefore, only the differences will be recorded.

Character changes are:

    for       use
    @         *
    #         $
    ;         /

In as, the character ";" is a logical new line; several operations may appear on one line if separated by ";". Several new expression operators have been provided:

\> right shift (logical)
\< left shift
* multiplication
\/ division
% remainder (no longer means "register")
! one's complement
[ ] parentheses for grouping
result has value of left, type of right

For example location 0 (relocatable) can be written "0^"; another way to denote register 2 is "2^r0".

All of the preceding operators are binary; if a left operand is missing, it is taken to be 0. The "!" operator adds its left operand to the one's complement of its right operand.

There is a conditional assembly operation code different from that of PAL-11R (whose conditionals are not provided):

    .if expression
    ... 
    .endif

If the expression evaluates to non-zero, the section of code between the ".if" and the ".endif" is assembled; otherwise it is ignored. ".if"s may be nested.

Temporary labels like those introduced by Knuth [reference] may be employed. A temporary label is defined as follows:
n:

where \( n \) is a digit 0 ... 9. Symbols of the form "nf" refer to the first label \( n: \) following the use of the symbol; those of the form "nb" refer to the last \( n: \). The same \( n \) may be used many times. Labels of this form are less taxing both on the imagination of the programmer and on the symbol table space of the assembler.

The PAL-11R opcodes ".eot" and ".end" are redundant and are omitted.

The symbols

\[
\begin{align*}
\text{r0} & \quad \ldots \quad \text{r5} \\
\text{sp} & \\
\text{pc} & \\
\text{ac} & \\
\text{mq} & \\
\text{div} & \\
\text{mul} & \\
\text{lsh} & \\
\text{ash} & \\
\text{nor} & \\
\text{csw} & \
\end{align*}
\]

are predefined with appropriate values. The symbol "csw" refers to the console switches. ".." is the relocation constant and is added to each relocatable symbol; normally it is 40000(8); it may be changed to assemble a section of code at a location different from that in which it will be executed.

It is illegal to assign a value to "." less than its current value.

The new opcode "sys" is used to specify system calls. Names for system calls are predefined. See the section on system calls for their names.

Strings of characters may be assembled in a way more convenient than PAL-11's "ascii" operation (which is, therefore, omitted). Strings are included between the string quotes "<" and ">":

<here is a string>

Escape sequences exist to enter non graphic and other difficult characters. These sequences are also effective in single and double character constants introduced by single ('') and double ("") quotes respectively.
The binary output of the assembler is placed on the file "a.out" in the current directory. a.out also contains the symbol table from the assembly and relocation bits. The output of the assembler is executable immediately if the assembly was error-free and if there were no unresolved external references. The link editor ld may be used to combine several assembly outputs and resolve global symbols.

The multiple location counter feature of PAL11R is not supported.

The assembler does not produce a listing of the source program. This is not a serious drawback; the debugger db discussed below is sufficiently powerful to render a printed octal translation of the source unnecessary.

FILES
/etc/as2 pass 2 of the assembler
a.tmp1 temporary
a.tmp2 temporary
a.tmp3 temporary
a.out object

SEE ALSO
ld, nm, sh, un, db, a.out (format of output)

DIAGNOSTICS
When an input file cannot be read, its name followed by a question mark is typed and assembly ceases.

When syntactic or semantic errors occur, a single-character diagnostic is typed out together with the line number and the file name in which it occurred. Errors in pass 1 cause cancellation of pass 2. The possible errors are:

) parentheses error
] parentheses error
* Indirection ("*") used illegally
A error in Address
B Branch instruction has too remote an address
E error in Expression
F error in local ("E" or "b") type symbol
G Garbage (unknown) character
M Multiply defined symbol as label
O Odd-- word quantity assembled at odd
address

P  Phase error—"." different in pass 2 from pass 1 value
R  Relocation error
U  Undefined symbol
X  Syntax error

BUGS  Symbol table overflow is not checked.

OWNER  dmr
NAME  B -- language

SYNOPSIS  sh rc /usr/b/rc name

DESCRIPTION  B is a language suitable for system programming. It is described in a separate publication B reference manual.

The canned shell sequence in /usr/b/rc will compile the program name.b into the executable file a.out. It involves running the B compiler, the B assembler, the assembler and the link editor. The process leaves the files name.i and name.s in the current directory.

FILES  name.b, name.i, name.s.

SEE ALSO  /etc/bc, /etc/ba, /etc/brt1, /etc/brt2, /etc/bilib, /etc/libb.a, B reference manual.

DIAGNOSTICS  see B reference manual

BUGS  There should be a B command.

OWNER  ken, dmr
NAME bas -- basic

SYNOPSIS bas [file]

DESCRIPTION bas is a dialect of basic. If a file argument is provided, the file is used for input before the console is read.

bas accepts lines of the form:

    statement
    integer statement

Integer numbered statements (known as internal statements) are stored for later execution. They are stored in sorted ascending order. Non-numbered statements are immediately executed. The result of an immediate expression statement (that does not have '=' as its highest operator) is printed.

Statements have the following syntax: (expr is short for expression)

expr
    The expression is executed for its side effects (assignment or function call) or for printing as described above.

done
    Return to system level.

draw expr expr expr
draw is used to draw on a 611-type storage scope through a TSP-1 plotter interface. The coordinates of the scope face are zero to one in both the x and y directions. (Zero, zero being the lower left corner.) The expressions are evaluated and designated X, Y, and Z. A line is drawn from the previous X, Y to the new X, Y. If Z is non-zero, the line is visible, otherwise the line is invisible.

for name = expr expr statement
for name = expr expr
...
next
The for statement repetatively executes a statement (first form) or a group of statements (second form) under control of a named variable. The variable takes on the value of the first expression, then is incremented by one on each loop, not to exceed the value of the second expression.
**goto expr**
The expression is evaluated, truncated to an integer and execution goes to the corresponding integer numbered statement. If executed from immediate mode, the internal statements are compiled first.

**if expr statement**
The statement is executed if the expression evaluates to non-zero.

**list [expr [expr]]**
*list* is used to print out the stored internal statements. If no arguments are given, all internal statements are printed. If one argument is given, only that internal statement is listed. If two arguments are given, all internal statements inclusively between the arguments are printed.

**print expr**
The expression is evaluated and printed.

**return expr**
The expression is evaluated and the result is passed back as the value of a function call.

**run**
The internal statements are compiled. The symbol table is re-initialized. The random number generator is re-set. Control is passed to the lowest numbered internal statement.

**Expressions have the following syntax:**

**name**
A name is used to specify a variable. Names are composed of a letter ('a' - 'z') followed by letters and digits. The first four characters of a name are significant.

**number**
A number is used to represent a constant value. A number is composed of digits, at most one decimal point ('.') and possibly a scale factor of the form e digits or e digits.

{ expr }
Parentheses are used to alter normal order of evaluation.

**expr op expr**
Common functions of two arguments are
abbreviated by the two arguments separated by an operator denoting the function. A complete list of operators is given below.

```
expr ( [expr [expr ...]] )
```

Functions of an arbitrary number of arguments can be called by an expression followed by the arguments in parentheses separated by commas. The expression evaluates to the line number of the entry of the function in the internally stored statements. This causes the internal statements to be compiled. If the expression evaluates negative, an builtin function is called. The list of builtin functions appears below.

```
name ( [expr [expr ...]] )
```

Arrays are not yet implemented.

The following is the list of operators:

```
=  
  = is the assignment operator. The left operand must be a name or an array element. The result is the right operand. Assignment binds right to left, all other operators bind left to right.

& |  
  & (logical and) has result zero if either of its arguments are zero. It has result one if both its arguments are non-zero.  | (logical or) has result zero if both of its arguments are zero. It has result one if either of its arguments are non-zero.

< <= > >= == <>  
  The relational operators (< less than, <= less than or equal, > greater than, >= greater than or equal, == equal to, <> not equal to) return one if their arguments are in the specified relation. They return zero otherwise. Relational operators at the same level extend as follows: a>b>c is the same as a>b&b>c.
```

```
+ -  
  Add and subtract.

* /  
  Multiply and divide.
```

```
^  
  Exponentiation.
```
The following is a list of builtin functions:

arg
Arg(i) is the value of the i
th actual
parameter on the current level of function
call.

exp
Exp(x) is the exponential function of x.

log
Log(x) is the logarithm base e of x.

sin
Sin(x) is the sine of x (radians).

cos
Cos(x) is the cosine of x (radians).

atn
Atan(x) is the arctangent of x. (Not implemented.)

rnd
Rnd() is a uniformly distributed random
number between zero and one.

expr
Expr() is the only form of program input.
A line is read from the input and evaluated
as an expression. The resultant value is
returned.

int
Int(x) returns x truncated to an integer.

FILES
/tmp/btma, btmb ... temporary

SEE ALSO
--

DIAGNOSTICS
Syntax errors cause the incorrect line to be
typed with an underscore where the parse failed.
All other diagnostics are self explanatory.

BUGS
Arrays [] are not yet implemented. In general,
program sizes, recursion, etc are not checked,
and cause trouble.

OWNER
ken
<table>
<thead>
<tr>
<th>NAME</th>
<th>bcd — binary coded decimal conversion</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYNOPSIS</td>
<td>bcd [ string ]</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>bcd will convert a string into GECOS card code. If no argument string is provided, bcd will read a line and convert it.</td>
</tr>
<tr>
<td>FILES</td>
<td>--</td>
</tr>
<tr>
<td>SEE ALSO</td>
<td>--</td>
</tr>
<tr>
<td>DIAGNOSTICS</td>
<td>--</td>
</tr>
<tr>
<td>BUGS</td>
<td>--</td>
</tr>
<tr>
<td>OWNER</td>
<td>dmr</td>
</tr>
</tbody>
</table>
NAME

boot -- reboot system

SYNOPSIS

/etc/boot

DESCRIPTION

boot logically a command, and is kept in /etc only to lessen the probability of its being in-
voked by accident or from curiosity. It reboots the system by jumping to the read-only memory,
which contains a disk boot program.

FILES

--

SEE ALSO

boot procedure

DIAGNOSTICS

--

BUGS

Should obviously not be executable by the general user. Also, it should reboot in a more direct
manner. The mechanism invoked by jumping to the ROM loader is sensitive to the contents of the
console switches, which makes the whole procedure even more dangerous.

Rather than jumping to the ROM, boot should simulate the ROM action with 173700 in the switches.
In this manner, It may be used when the switches are not set, and even in installation without a
ROM.

OWNER

ken
NAME

SYNOPSIS

DESCRIPTION

FILES

SEE ALSO

DIAGNOSTICS

BUGS

OWNER

cat -- concatenate and print

cat file1 ...

cat reads each file in sequence and writes it on the standard output stream. Thus:

    cat file

is about the easiest way to print a file. Also:

    cat file1 file2 >file3

is about the easiest way to concatenate files.

If no input file is given cat reads from the standard input file.

--

pr, cp

none; if a file cannot be found it is ignored.

--

ken, dmr
NAME        chdir -- change working directory

SYNOPSIS    chdir directory

DESCRIPTION directory becomes the new working directory.

Because a new process is created to execute each command, chdir would be ineffective if it were written as a normal command. It is therefore recognized and executed by the Shell.

FILES        --

SEE ALSO     sh

DIAGNOSTICS ?

BUGS         --

OWNER        ken, dmr
NAME
check -- file system consistency check

SYNOPSIS
check [ filesystem [ blockno1 ... ] ]

DESCRIPTION
check will examine a file system, build a bit map of used blocks, and compare this bit map against
the bit map maintained on the file system. If
the file system is not specified, a check of both
/dev/rf0 and /dev/rk0 is performed. Output in-
cludes the number of files on the file system,
the number of these that are 'large', the number
of used blocks, and the number of free blocks.

FILES
/dev/rf0, /dev/rk0

SEE ALSO
find

DIAGNOSTICS
Diagnostics are produced for blocks missing,
duplicated, and bad block addresses. Diagnostics
are also produced for block numbers passed as
parameters. In each case, the block number,
_inode, and block class (_ = inode, _ indirect,
_free) is printed.

BUGS
The checking process is two pass in nature. If
checking is done on an active file system, ex-
traneous diagnostics may occur.

The swap space on the RF file system is not ac-
counted for and will therefore show up as 'miss-
ing'.

OWNER
ken, dmr
NAME      chmod -- change mode

SYNOPSIS  chmod octal file1 ...

DESCRIPTION The octal mode replaces the mode of each of the files. The mode is constructed from the OR of the following modes:

  01 write for non-owner
  02 read for non-owner
  04 write for owner
  10 read for owner
  20 executable
  40 set-UID

Only the owner of a file may change its mode.

FILES      --

SEE ALSO   stat, ls

DIAGNOSTICS ?

BUGS       --

OWNER      ken, dmr
NAME

chown -- change owner

SYNOPSIS

chown owner file, ...

DESCRIPTION

owner becomes the new owner of the files. The owner may be either a decimal UID or a name found in /etc/u ids.

Only the owner of a file is allowed to change the owner. It is illegal to change the owner of a file with the set-user-ID mode.

FILES

/etc/uids

SEE ALSO

stat

DIAGNOSTICS

?

BUGS

--

OWNER

ken, dmr
NAME     cmp -- compare two files
SYNOPSIS  cmp file1 file2
DESCRIPTION The two files are compared for identical contents. Discrepancies are noted by giving the offset and the differing words.
FILES     --
SEE ALSO   --
DIAGNOSTICS Messages are given for inability to open either argument, premature EOF on either argument, and incorrect usage.
BUGS      If the two files differ in length by one byte, the extra byte does not enter into the comparison.
OWNER     dmr
NAME cp -- copy
SYNOPSIS cp file_11 file_12 file_21 file_22 ...
DESCRIPTION Files are taken in pairs; the first is opened for reading, the second created mode 17. Then the first is copied into the second.
FILES --
SEE ALSO cat, pr
DIAGNOSTICS Error returns are checked at every system call, and appropriate diagnostics are produced.
BUGS The second file should be created in the mode of the first.

A directory convention as used in mv should be adopted to cp.
OWNER ken, dmr
NAME date -- print the date

SYNOPSIS date

DESCRIPTION The current date is printed to the second.

FILES --

SEE ALSO sdate

DIAGNOSTICS --

BUGS --

OWNER dmr
NAME

db -- debug

SYNOPSIS

db [ core [ namelist ] ]

DESCRIPTION

Unlike many debugging packages (including DEC's ODT, on which db is loosely based) db is not loaded as part of the core image which it is used to examine; instead it examines files. Typically, the file will be either a core image produced after a fault or the binary output of the assembler. Core is the file being debugged; if omitted core is assumed. namelist is a file containing a symbol table. If it is omitted, a.out is the default. If no appropriate name list file can be found, db can still be used but some of its symbolic facilities become unavailable.

The format for most db requests is an address followed by a one character command.

Addresses are expressions built up as follows:

1. A name has the value assigned to it when the input file was assembled. It may be relocatable or not depending on the use of the name during the assembly.

2. An octal number is an absolute quantity with the appropriate value.

3. An octal number immediately followed by "r" is a relocatable quantity with the appropriate value.

4. The symbol "." indicates the current pointer of db. The current pointer is set by many db requests.

5. Expressions separated by "+" or " " (blank) are expressions with value equal to the sum of the components. At most one of the components may be relocatable.

6. Expressions separated by "-" form an expression with value equal to the difference to the components. If the right component is relocatable, the left component must be relocatable.

7. Expressions are evaluated left to right.

Names for registers are built in:

r0 ... r5
sp
pc
ac
mq

These may be examined. Their values are deduced from the contents of the stack in a core image file. They are meaningless in a file that is not a core image.

If no address is given for a command, the current address (also specified by ".") is assumed. In general, "." points to the last word or byte printed by \texttt{db}.

There are \texttt{db} commands for examining locations interpreted as octal numbers, machine instructions, ASCII characters, and addresses. For numbers and characters, either bytes or words may be examined. The following commands are used to examine the specified file.

/ The addressed word is printed in octal.
\ The addressed byte is printed in octal.
" The addressed word is printed as two ASCII characters.
\ The addressed byte is printed as an ASCII character.
' The addressed word is multiplied by 2, then printed in octal (used with B programs, whose addresses are word addresses).
? The addressed word is interpreted as a machine instruction and a symbolic form of the instruction, including symbolic addresses, is printed. Usually, the result will appear exactly as it was written in the source program.
& The addressed word is interpreted as a symbolic address and is printed as the name of the symbol whose value is closest to the addressed word, possibly followed by a signed offset.

\texttt{<nl>} (i. e., the character "new line") This command advances the current location counter "." and prints the resulting location in the mode last specified by one of the above requests.

\ This character decrements "." and prints the resulting location in the mode last selected one of the above requests. It is
a converse to \( \langle n \rangle \).

It is illegal for the word-oriented commands to have odd addresses. The incrementing and decrementing of \( \langle n \rangle \) done by the \( \langle n \rangle \) and \( \langle n \rangle \) requests is by one or two depending on whether the last command was word or byte oriented.

The address portion of any of the above commands may be followed by a comma and then by an expression. In this case that number of sequential words or bytes specified by the expression is printed. \( "\) is advanced so that it points at the last thing printed.

There are two commands to interpret the value of expressions.

\[ = \]
When preceded by an expression, the value of the expression is typed in octal. When not preceded by an expression, the value of \( "\) is indicated. This command does not change the value of \( "\).

\[ : \]
An attempt is made to print the given expression as a symbolic address. If the expression is relocatable, that symbol is found whose value is nearest that of the expression, and the symbol is typed, followed by a sign and the appropriate offset. If the value of the expression is absolute, a symbol with exactly the indicated value is sought and printed if found; if no matching symbol is discovered, the octal value of the expression is given.

The following command may be used to patch the file being debugged.

\[ ! \]
This command must be preceded by an expression. The value of the expression is stored at the location addressed by the current value of \( "\). The opcodes do not appear in the symbol table, so the user must assemble them by hand.

The following command is used after a fault has caused a core image file to be produced.

\[ $ \]
causes the contents of the general registers and several other registers to be printed both in octal and symbolic format. The values are as they were at the time of the fault.
The only way to exit from db is to generate an end of file on the typewriter (EOT character).

FILES
---

SEE ALSO
as; core for format of core image.

DIAGNOSTICS
"File not found" if the first argument cannot be read; otherwise "?".

BUGS
Really, db should know about relocation bits, floating point operations, and PDP11/45 instructions.

OWNER
dmr
NAME     dbppt -- dump binary paper tape
SYNOPSIS  dbppt name [ output ]
DESCRIPTION dbppt produces binary paper tape in UNIX standard format, which includes checksums and a zero-suppression feature. File name is dumped; if the output argument is not given, output goes to /dev/ppt.
FILES     /dev/ppt
SEE ALSO lbppt to reload the tapes. bppt for binary paper tape format.
DIAGNOSTICS ?
BUGS      --
OWNER     ken
NAME
dc -- desk calculator

SYNOPSIS
dc

DESCRIPTION
dc is an arbitrary precision integer arithmetic package. The overall structure of dc is a stacking (reverse Polish) calculator. The following constructions are recognized by the calculator:

number
The value of the number is pushed on the stack. If the number starts with a zero, it is taken to be octal, otherwise it is decimal.

\[ \pm = \times \div \%
\]  
The top two values on the stack are added (\(+\)), subtracted (\(-\)), multiplied (\(\times\)), divided (\(\div\)), or remaindered (\(\%\)). The two entries are popped off of the stack, the result is pushed on the stack in their place.

sx
The top of the stack is popped and stored into a register named x, where x may be any character.

lx
The value in register x is pushed on the stack. The register x is not altered.

d
The top value on the stack is pushed on the stack. Thus the top value is duplicated.

p
The top value on the stack is printed in decimal. The top value remains unchanged.

f
All values on the stack are popped off and printed in decimal.

r
All values on the stack are popped.

q
exit.

h
print brief synopsis of commands to dc.

An example to calculate the monthly, weekly and
hourly rates for a $10,000/year salary.

10000
100* (now in cents)
dsa (non-destructive store)
12/ (pennies per month)
la52/ (pennies per week)
d10* (deci-pennies per week)
375/ (pennies per hour)
f (print all results)

(3) 512
(2) 19230
(1) 83333

FILES --

SEE ALSO --

DIAGNOSTICS ? (x) for unrecognized character x.

BUGS % doesn’t work correctly.

OWNER ken
NAME     df -- disk free
SYNOPSIS df [ filesystem ]
DESCRIPTION df prints out the number of free blocks available on a file system. If the file system is unspecified, the free space on /dev/rf0 and /dev/rk0 is printed.
FILES     /dev/rf0, /dev/rk0
SEE ALSO  check
DIAGNOSTICS --
BUGS      --
OWNER     ken, dmr
NAME  dsw  --  delete interactively

SYNOPSIS  dsw  [ directory ]

DESCRIPTION  For each file in the given directory ("." if not specified) dsw types its name. If 'y' is typed, the file is deleted; if 'x', dsw exits; if anything else, the file is not removed.

FILES  --

SEE ALSO  rm

DIAGNOSTICS  "?"

BUGS  The name "dsw" is a carryover from the ancient past. Its etymology is amusing but the name is nonetheless ill-advised.

OWNER  dmr, ken
NAME
dtf -- DECTape format

SYNOPSIS
/etc/df

DESCRIPTION
dtf will write timing tracks, mark tracks and
block numbers on a virgin DECTape. The format is
DEC standard of 578 blocks of 256 words each.
The end zones are a little longer than standard
DEC.

Before use, the tape to be formatted should be
mounted on drive 0. The 'wall' and 'wtm'
switches should be enabled. After the tape is
formatted, the switches should be disabled to
prevent damage to subsequent tapes due to a con-
troller logic error.

FILES

SEE ALSO
sdate

DIAGNOSTICS
"?" is typed for any error detected.

BUGS
This program does physical I/O on drive 0. The
processor priority is set very high due to very
stringent real time requirements. This means
that all time sharing activities are suspended
during the formatting (about 1.5 minutes) The
real time clock will also be slow.

OWNER
ken
NAME     du -- summarize disk usage

SYNOPSIS  du  [ -s ] [ -a ] [ name ... ]

DESCRIPTION du gives the number of blocks contained in all files and (recursively) directories within each specified directory or file name. If name is missing, . is used.

The optional argument -s causes only the grand total to be given. The optional argument -a causes an entry to be generated for each file. Absence of either causes an entry to be generated for each directory only.

A file which has two links to it is only counted once.

FILES     /

SEE ALSO  --

DIAGNOSTICS  --

BUGS      Files at the top level (not under -a option) are not listed.

Removable file systems do not work correctly since i-numbers may be repeated while the corresponding files are distinct. Du should maintain an i-number list per root directory encountered.

OWNER     dmr
NAME ed -- editor

SYNOPSIS ed [ name ]

DESCRIPTION ed is the standard text editor. ed is based on QED [reference] but is fully if succinctly described here. Differences between ed and QED are also noted to simplify the transition to the less powerful editor.

If the optional argument is given, ed simulates an e command on the named file; that is to say, the file is read into ed's buffer so that it can be edited.

ed operates on a copy of any file it is editing; changes made in the copy have no effect on the file until an explicit write (w) command is given. The copy of the text being edited resides in a temporary file called the buffer. There is only one buffer.

Commands to ed have a simple and regular structure: zero or more addresses followed by a single character command, possibly followed by parameters to the command. These addresses specify one or more lines in the buffer. Every command which requires addresses has default addresses, so that the addresses can often be omitted.

In general only one command may appear on a line. Certain commands allow the input of text. This text is placed in the appropriate place in the buffer. While ed is accepting text, it is said to be in input mode. In this mode, no commands are recognized; all input is merely collected. Input mode is left by typing a period (.) alone at the beginning of a line.

ed supports a limited form of regular expression notation. A regular expression is an expression which specifies a set of strings of characters. A member of this set of strings is said to be matched by the regular expression. The regular expressions allowed by ed are constructed as follows:

1. An ordinary character (not one of those discussed below) is a regular expression and matches that character.

2. A circumflex (^) at the beginning of a regular expression matches the null character at the beginning of a line.
3. A currency symbol ($) at the end of a regular expression matches the null character at the end of a line.

4. A period (.) matches any character but a new-line character.

5. A regular expression followed by an asterisk (*) matches any number of adjacent occurrences (including zero) of the regular expression it follows.

6. A string of characters enclosed in square brackets ([]) matches any character in the string but no others. If, however, the first character of the string is a circumflex (^) the regular expression matches any character but new-line and the characters in the string.

7. The concatenation of regular expressions is a regular expression which matches the concatenation of the strings matched by the components of the regular expression.

8. The null regular expression standing alone is equivalent to the last regular expression encountered.

Regular expressions are used in addresses to specify lines and in one command (s, see below) to specify a portion of a line which is to be replaced.

If it is desired to use one of the regular expression metacharacters as an ordinary character, that character may be preceded by "\". This also applies to the character bounding the regular expression (often "/") and to "\\" itself.

Addresses are constructed as follows. To understand addressing in ed it is necessary to know that at any time there is a current line. Generally speaking, the current line is the last line affected by a command; however, the exact effect on the current line by each command is discussed under the description of the command.

1. The character "." addresses the current line.

2. The character "$" addresses the last line of the buffer.

3. A decimal number n addresses the n-th line of the buffer.
4. A regular expression enclosed in slashes "/" addresses the first line found by searching toward the end of the buffer and stopping at the first line containing a string matching the regular expression. If necessary the search wraps around to the beginning of the buffer.

5. A regular expression enclosed in queries "?" addresses the first line found by searching toward the beginning of the buffer and stopping at the first line found containing a string matching the regular expression. If necessary the search wraps around to the end of the buffer.

6. An address followed by a plus sign "+" or a minus sign "-" followed by a decimal number specifies that address plus (resp. minus) the indicated number of lines. The plus sign may be omitted.

Commands may require zero, one, or two addresses. Commands which require no addresses regard the presence of an address as an error. Commands which require the presence of one address all assume a default address (often ".") but if given more than one address ignore any extras and use the last given. Commands which require two addresses have defaults in the case of zero or one address but use the last two if more than two are given.

Addresses are separated from each other typically by a comma (,). They may also be separated by a semicolon (;). In this case the current line "." is set to the the previous address before the next address is interpreted. This feature is used to control the starting line for forward and backward searches ("/", "?").

In the following list of ed commands, the default addresses are shown in parentheses. The parentheses are not part of the address, but are used to show that the given addresses are the default.

As mentioned, it is generally illegal for more than one command to appear on a line. However, any command may be suffixed by "p" (for "print"). In that case, the current line is printed after the command is complete.

In any two-address command, it is illegal for the
first address to lie after the second address.

(a)
<text>
The append command reads the given text and appends it after the addressed line. "" is left on the last line input, if there were any, otherwise at the addressed line. Address "0" is legal for this command; text is placed at the beginning of the buffer. (NOTE: the default address differs from that of QED.)

(c)
<text>
The change command deletes the addressed lines, then accepts input text which replaces these lines. "" is left at the last line input; if there were none, it is left at the first line not changed.

(d)
The delete command deletes the addressed lines from the buffer. "" is left at the first line not deleted.

e filename
The edit command causes the entire contents of the buffer to be deleted, and then the named file to be read in. "" is set to the last line of the buffer. The number of characters read is typed.

(g/regular expression/command
In the global command, the first step is to mark every line which matches the given regular expression. Then for every such line, the given command is executed with "" set to that line. The repeated command cannot be a, g, i, or c.

(i)
<text>
This command inserts the given text before the addressed line. "" is left at the last line input; if there were none, at the addressed line. This command differs from the a command only in the placement of the text.

(l)
The list command prints the addressed lines in an unambiguous way. Non-printing
characters are over-struck as follows:

<table>
<thead>
<tr>
<th>char</th>
<th>prints</th>
</tr>
</thead>
<tbody>
<tr>
<td>bs</td>
<td>\</td>
</tr>
<tr>
<td>tab</td>
<td>\r</td>
</tr>
<tr>
<td>ret</td>
<td>\t</td>
</tr>
<tr>
<td>SI</td>
<td>$</td>
</tr>
<tr>
<td>SO</td>
<td>@</td>
</tr>
</tbody>
</table>

All characters preceded by a prefix (ESC) character are printed over-struck with
without the prefix. Long lines are folded with the sequence \newline.

(...p

The print command prints the addressed
lines. "." is left at the last line print-
ed.

@

The quit command causes ed to exit. No
automatic write of a file is done.

($)r filename

The read command reads in the given file
after the addressed line. If no file name
is given, the file last mentioned in e, r,
or w commands is read. Address "0" is
legal for r and causes the file to be read
at the beginning of the buffer. If the
read is successful, the number of charac-
ters read is typed, "." is left at the
last line of the file.

(...s/regular expression/replacement/

The substitute command searches each ad-
dressed line for an occurrence of the speci-
fied regular expression. On each line in
which a match is found, the first (and only
first, compare QED) matched string is re-
placed by the replacement specified. It is
an error for the substitution to fail on
all addressed lines. Any character other
than space or new-line may be used instead
of "/" to delimit the regular expression
and the replacement. "." is left at the
last line substituted.

The ampersand "&" appearing in the replace-
ment is replaced by the regular expression
that was matched. The special meaning of
"&" in this context may be suppressed by
preceding it by "\".

(1,)$w filename

The write command writes the addressed
lines onto the given file. If no file name
is given, the file last named in e, r, or w
commands is written. "." is unchanged. If the command is successful, the number of characters written is typed.

($) =
The line number of the addressed line is typed. "." is unchanged by this command.

1 UNIX command
The remainder of the line after the "!" is sent to UNIX to be interpreted as a com-
mand. "." is unchanged.

<newline>
A blank line alone is equivalent to "+1p"; it is useful for stepping through text.

Ed can edit at most 1500 lines and the maximum size of a line is 256 characters. The differ-
ences between ed and QED are:

1. There is no "/f" character; input mode is left by typing "." alone on a line.

2. There is only one buffer and hence no "/b" stream directive.

3. The commands are limited to:
   
   a c d e g i l p q r s w = !
   
   where e is new.

4. The only special characters in regular expressions are:
   
   * ^ $ [ .
   
   which have the usual meanings. However, 
   "^" and "$" are only effective if they are
   the first or last character respectively of
   the regular expression. Otherwise suppres-
sion of special meaning is done by preced-
ing the character by "\", which is not oth-
erversely special.

5. In the substitute command, only the left-
   most occurrence of the matched regular
   expression is substituted.

7. The a command has a different default ad-
dress.

FILES
/tmp/etma, etmb, ... temporary
/etc/msh is used to implement the "!" command.
SEE ALSO

DIAGNOSTICS

"?" for any error

BUGS

ed is used as the shell for the editing system. It has the editing system UID built in and if invoked under this UID will give slightly different responses. This is a little kludgy.

OWNER

ken
NAME    find -- find file with given name
SYNOPSIS find name or number ...
DESCRIPTION find searches the entire file system hierarchy and gives the path names of all files with the specified names or (decimal) i-numbers.
FILES    --
SEE ALSO  --
DIAGNOSTICS --
BUGS      --
OWNER     dmr
NAME for -- fortran
SYNOPSIS for file
DESCRIPTION for is a nearly complete fortran compiler. file is the name of a fortran source program to be compiled. The following is a list of differences between for and ANSI standard fortran:

1. arbitrary combination of types are allowed in expressions. Not all combinations are expected to be supported in runtime. All of the normal conversions involving integer, real and double precision are allowed.

FILES
f.tmp1, 2 3 temporary
/etc/f1, 2 3 4 passes
/etc/xx runtime

SEE ALSO --

DIAGNOSTICS Diagnostics are given by number. If the source code is available, it is printed with an underline at the current character pointer. A listing of error numbers is available.

BUGS The following is a list of those features not yet implemented:

functions
arithmetic statement functions
data statements
complex constants
hollerith constants
continuation cards

OWNER dmr, ken
NAME

form -- form letter generator

SYNOPSIS

.form proto arg, ...

DESCRIPTION

form generates a form letter from a prototype letter, an associative memory, arguments and in a special case, the current date.

If form is invoked with the argument x, the following files come into play:

x.f prototype input
x.r form letter output
x.am associative memory
form.am associative memory if x.am not found.

Basically, form is a copy process from the file x.f to the file x.r. If an element of the form \n (where n is a digit from 1 to 9) is encountered, the nth argument is inserted in its place, and that argument is then rescanned. If \0 is encountered, the current date is inserted. If the desired argument has not been given, a message of the form "\n: " is typed. The response typed in then is used for that argument.

If an element of the form [name] is encountered, the name is looked up in the associative memory. If it is found, the contents of the memory under this name replaces the original element (again rescanned.) If the name is not found, a message of the form "name: " is typed. The response typed in is used for that element. If the associative memory is writable, the response is entered in the memory under the name. Thus the next search for that name will succeed without interaction.

In both of the above cases, the response is typed in by entering arbitrary text terminated by two new lines. Only the first of the two new lines is passed with the text. The process is instantly terminated if an end of file is encountered anywhere except in the associative memory.

FILES

x.f input file
x.r output file
x.am associative memory
form.am associative memory

SEE ALSO

type

DIAGNOSTICS

"setup error" when the appropriate files cannot be located or created.

BUGS

"setup" is misspelled.
11/3/71

OWNER: rhm, ken
NAME  hup  --  hang up typewriter

SYNOPSIS  hup

DESCRIPTION  hup hangs up the phone on the typewriter which uses it.

FILES  --

SEE ALSO  --

DIAGNOSTICS  --

BUGS  should not be used; sometimes causes the typewriter channel to be lost.

OWNER  dmr, ken
NAME
lbppt -- load binary paper tapes

SYNOPSIS
lbppt output [ input ]

DESCRIPTION
lbppt loads a paper tape in standard UNIX binary paper tape format. It is used to bring files to a UNIX installation. Currently there is a GECOS program to prepare a GECOS file in binary paper tape format.

If the input file is specified, the character stream from that input is expected to be in UNIX binary paper tape format. If it is not present, /dev/ppt is assumed. The input stream is interpreted, checksummed, and copied to the output file.

FILES
/dev/ppt

SEE ALSO
dbppt, bppt format

DIAGNOSTICS
"checksum"; "usage: "; "read error".

BUGS
--

OWNER
ken
NAME

ld -- link editor

SYNOPSIS

ld [ -usaol ] name

DESCRIPTION

ld combines several object programs into one; resolves external references; and searches libraries. In the simplest case the names of several object programs are given, and ld combines them, producing an object module which can be either executed or become the input for a further ld run.

The argument routines are concatenated in the order specified. The entry point of the output is the beginning of the first routine.

If any argument is a library, it is searched, and only those routines defining an unresolved external reference are loaded. If any routine loaded from a library refers to an undefined symbol which does not become defined by the end of the library, the library is searched again. Thus the order of libraries primarily affects the efficiency of loading, not what routines get loaded.

ld understands several flag arguments which are written preceded by a "-":

-s "squash" the output, that is, remove the symbol table and relocation bits to save space (but impair the usefulness of the debugger). This information can also be removed by strip.

-u take the following argument as a symbol and enter it as undefined in the symbol table. This is useful for loading wholly from a library, since initially the symbol table is empty and an unresolved reference is needed to force the loading of the first routine.

-o set the origin of the load to the octal number which is given as the next argument. This option affects only the definition of relocatable external symbols. See DMR before using.

-l This option is an abbreviation for a library name. "-l" alone stands for "/etc/liba.a", which is the standard system library for assembly language programs. 
"-lx" stands for "/etc/libx.a" where x is any character. There are libraries for Fortran (x='f') and B (x='b').
-a means "absolute" (load at origin absolute 0) but it doesn't work.

The output of ld is left on a.out. This file is executable only if no errors occurred during the load.

FILES

/etc/libx.a, for various x;
/etc/ltma, ltmb, ... (temporary)
a.out (output file)

SEE ALSO

as, strip, ar (maintains libraries)

DIAGNOSTICS

"can't create temp file" -- unwritable directory or someone else is using ld in the same directory.

"can't open temp file" -- maybe someone has deleted it out from under you.

"file not found" -- bad argument

"bad format" -- bad argument

"relocation error" -- bad argument (relocation bits corrupted)

"bad relocation" -- user error: a relocatable reference to an external symbol that turns out to be absolute.

"multiply defined" -- same symbol defined twice in same load

"un" -- stands for "undefined symbol"

"symbol not found" -- loader bug

BUGS

Option "-a" doesn't work at all; option "-o" doesn't work right.

OWNER

dmr
NAME
ln -- make a link

SYNOPSIS
ln name₁ [ name₂ ]

DESCRIPTION
ln creates a link to an existing file name₁. If name₂ is given, the link has that name; otherwise it is placed in the current directory and its name is the last component of name₁.

It is forbidden to link to a directory or to link across file systems.

FILES
--

SEE ALSO
rm, to `unlink`

DIAGNOSTICS
"?"

BUGS
There is nothing particularly wrong with ln, but links don't work right with respect to the backup system: one copy is backed up for each link, and (more serious) in case of a file system reload both copies are restored and the information that a link was involved is lost.

OWNER
ken, dmr
NAME
ls -- list contents of directory

SYNOPSIS
ls [ -ltda ] name1 ...

DESCRIPTION
ls lists the contents of one or more directories under control of several options:

l list in long format, giving i-number, mode, owner, size in bytes, and time of last modification for each file. (see stat for format of the mode)

t sort by time modified (latest first) instead of by name, as is normal

a list all entries; usually those beginning with "." are suppressed

s give size in blocks for each entry

d if argument is a directory, list only its name, not its contents (mostly used with "-l" to get status on directory)

If no argument is given, "." is listed. If an argument is not a directory, its name is given.

FILES
/etc/uids to get user ID's for ls -l

SEE ALSO
stat

DIAGNOSTICS
"name nonexistent"; "name unreadable"; "name unstatable."

BUGS
In ls -l, when a user cannot be found in /etc/uids, the user number printed instead of a name is incorrect. It is correct in stat.

OWNER
dmr, ken
NAME  mail -- send mail to another user

SYNOPSIS  mail  [ letter person ... ]

DESCRIPTION  mail without an argument searches for a file called mailbox, prints it if present, and asks if it should be saved. If the answer is "y", the mail is renamed mail, otherwise it is deleted. The answer to the above question may be supplied in the letter argument.

When followed by the names of a letter and one or more people, the letter is appended to each person's mailbox. Each letter is preceded by the sender's name and a postmark.

A person is either the name of an entry in the directory /usr, in which case the mail is sent to /usr/person/mailbox, or the path name of a directory, in which case mailbox in that directory is used.

When a user logs in he is informed of the presence of mail.

FILES  /etc/uids to map the sender's numerical user ID to name; mail and mailbox in various directories.

SEE ALSO  init

DIAGNOSTICS  "Who are you?" if the user cannot be identified for some reason (a bug). "Cannot send to user" if mailbox cannot be opened.

BUGS  --

OWNER  ken
NAME        mesg -- permit or deny messages
SYNOPSIS    mesg [ n ][ y ]
DESCRIPTION mesg n forbids messages via write by revoking non-user write permission on the user's typewriter. mesg y reinstates permission. mesg with no argument reverses the current permission. In all cases the previous state is reported.
FILES       /dev/ttyn
SEE ALSO    write
DIAGNOSTICS "?" if the standard input file is not a typewriter
BUGS        --
OWNER       dmr, ken
NAME       mkdir -- make a directory
SYNOPSIS   mkdir  dirname
DESCRIPTION mkdir creates directory dirname.
The standard entries "." and ".." are made automatically.
FILES      --
SEE ALSO   rmdir to remove directories
DIAGNOSTICS "?"
BUGS       No permissions are checked. The system’s user ID, not that of the creator of the directory, becomes the owner of the directory.
OWNER      ken, dmr
NAME
mkfs -- make file system

SYNOPSIS
/etc/mkfs t
/etc/mkfs r

DESCRIPTION
mkfs initializes either a DECtape (argument "t") or an RK03 disk pack (argument "r") so that it contains an empty file system. mkfs or its equivalent must be used before a tape or pack can be mounted as a file system.

In both cases the super-block, i-list, and free list are initialized, and a root directory containing entries for .. and .. are created. For RK03's the number of available blocks is 4872, for tapes 578.

This program is kept in /etc to avoid inadvertent use and consequent destruction of information.

FILES
/dev/tap0, /dev/rk0

SEE ALSO
--

DIAGNOSTICS
"Arg count", "Unknown argument", "Open error".

BUGS
--

OWNER
ken, dmr
NAME
mount -- mount file system

SYNOPSIS
mount special dir

DESCRIPTION
mount announces to the system that a removable file system has been mounted on the device corresponding to special file special. Directory dir (which must exist already) becomes the name of the root of the newly mounted file system.

FILES
--

SEE ALSO
umount

DIAGNOSTICS
"?", if the special file is already in use, cannot be read, or if dir does not exist.

BUGS
Should be usable only by the super-user.

OWNER
ken, dmr
NAME
mv -- move or rename a file

SYNOPSIS
mv name_1 name_2 ... ...

DESCRIPTION
mv changes the name of name_1 by linking to it
under the name name_2 and then unlinking name_1.
Several pairs of arguments may be given. If the
new name is a directory, the file is moved to
that directory under its old name. Directories
may only be moved within the same parent directory (just renamed).

FILES
--

SEE ALSO
--

DIAGNOSTICS
"?a"-- incorrect argument count
"?d"-- attempt to move a directory
"?s"-- moving file to itself
"?l"-- link error; old file doesn't exist or
can't write new directory
"?u"-- can't unlink old name

BUGS
If mv succeeds in removing the target file, but
then in unable to link back to the old file, the
result is ?l and the removal of the target file.
This is common with demountable file systems and
should be circumvented. Also in such cases, mv
should copy if it can.

OWNER
ken, dmr
NAME
nm -- get name list

SYNOPSIS
nm [ name ]

DESCRIPTION
nm prints the symbol table from the output file of an assembler or loader run. Only relocatable, global, and undefined symbols— not absolute— are given. Each defined symbol is preceded by its value; each undefined symbol by blanks. Global symbols have their first character underlined. The output is sorted alphabetically.

If no file is given, the symbols in a.out are listed.

FILES
a.out

SEE ALSO
as, ld

DIAGNOSTICS
"?"

BUGS
--

OWNER
dmr, ken
NAME od -- octal dump

SYNOPSIS od name [ origin ]

DESCRIPTION od dumps a file in octal, eight words per line with the origin of the line on the left. If an octal origin is given it is truncated to 0 mod 16 and dumping starts from there, otherwise from 0. Printing continues until halted by sending an interrupt signal.

FILES --

SEE ALSO db

DIAGNOSTICS "?"

BUGS Dumping does not cease at the end of the file; instead the file appears to be padded with garbage to a length of 511 mod 512 bytes.

OWNER ken, dmr
NAME
pr  --  print file

SYNOPSIS
pr  [  -lcm  ] name, ...  

DESCRIPTION
pr  produces a printed listing of one or more files. The output is separated into pages headed by the name of the file, a date, and the page number.

The optional flag  -l  causes each page to contain 78 lines instead of the standard 66 to accommodate legal size paper.

The optional flags  -c  (current date) and  -m  (modified date) specify which date will head all subsequent files.  -m  is default.

FILES
/dev/ttyn to suspend messages.

SEE ALSO
cat, cp, mesg

DIAGNOSTICS
--  (files not found are ignored)

BUGS
none

OWNER
ken, dmr
NAME rew -- rewind tape
SYNOPSIS rew [ digit ]
DESCRIPTION rew rewinds DECtape drives. The digit is the logical tape number, and should range from 0 to 7. A missing digit indicates drive 0.
FILES /dev/tap0, ..., /dev/tap7
SEE ALSO --
DIAGNOSTICS "?" if there is no tape mounted on the indicated drive or if the file cannot be opened.
BUGS --
OWNER ken, dmr
NAME  rkd  --  dump RK disk to tape

SYNOPSIS  /etc/rkd

DESCRIPTION  rkd copies an RK03/RK05 disk pack onto nine DECTapes.

Physical I/O is done and interrupts are disabled, so time-sharing is suspended
during operation of the command.

The sequence of tape drives is: 0, 1, 0, 1, ....

rkd exits if 0 appears in the console switches.

FILES  --

SEE ALSO  rkl

DIAGNOSTICS  none; errors are retried forever

BUGS  --

OWNER  ken
NAME                  rkf -- format RK03 disk pack

SYNOPSIS              rkf

DESCRIPTION           rkf formats a virgin disk pack. Because it dest-
                     roys all information on that pack, and because 
it is not interlocked against file system activi-
ty on the pack, the rkf program is not maintained 
in executable form. Instead the source must be 
located and assembled.

FILES                 none (uses physical I/O on drive 0).

SEE ALSO              --

DIAGNOSTICS          "error" is printed and a core image is produced 
                      if a write error occurs. A copy of the RK status 
                      register is in register 5.

BUGS                  As mentioned, rkf is not interlocked against sys-
                      tem I/O; if I/O is already occurring, it will be 
                      badly disrupted. In any event, all information 
on the pack is destroyed.

OWNER                 ken, dmr
<table>
<thead>
<tr>
<th>NAME</th>
<th>rkl -- reload RK disk from tape</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYNOPSIS</td>
<td><code>/etc/rkl</code></td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>rkl loads an RK05/RK05 disk pack from nine DECTapes.</td>
</tr>
<tr>
<td></td>
<td>The program uses physical I/O with interrupts disabled; therefore time-sharing is suspended.</td>
</tr>
<tr>
<td></td>
<td>Only the super-user may invoke this command.</td>
</tr>
<tr>
<td></td>
<td>The sequence of drives is: 0, 1, 0, 1, .... rkl will cease if 0 appears in the console switches.</td>
</tr>
<tr>
<td>FILES</td>
<td><code>--</code></td>
</tr>
<tr>
<td>SEE ALSO</td>
<td>rkd</td>
</tr>
<tr>
<td>DIAGNOSTICS</td>
<td>none; errors are retried forever</td>
</tr>
<tr>
<td>BUGS</td>
<td><code>--</code></td>
</tr>
<tr>
<td>OWNER</td>
<td>ken</td>
</tr>
</tbody>
</table>
NAME    rm    -- remove (unlink) files

SYNOPSIS  rm name, ...

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.

Directories cannot be removed by rm; cf. rmdir.

FILES    none

SEE ALSO    rmdir, for removing directories.

DIAGNOSTICS  If the file cannot be removed or does not exist,
the name of the file followed by a question mark
is typed.

BUGS    rm probably should ask whether a read-only file
is really to be removed.

OWNER    ken, dmr
NAME
rmdir -- remove directory

SYNOPSIS
rmdir dir1 ...

DESCRIPTION
rmdir removes (deletes) directories. The directory must be empty (except for the standard entries "." and "..", which rmdir itself removes). Write permission is required in the directory in which the directory appears.

FILES
none

SEE ALSO
--

DIAGNOSTICS
"dir?" is printed if directory dir cannot be found, is not a directory, or is not removable.

"dir -- directory not empty" is printed if dir has entries other than "." or "..

BUGS
--

OWNER
ken, dmr
NAME  roff -- format text

SYNOPSIS  roff [ +number ] [ -number ] name, ...

DESCRIPTION  roff formats text according to control lines embedded in the text. The optional argument "+number" causes printing to begin at the first page with the appropriate number; "-number" causes printing to cease at the first page with a higher number.

roff is fully described in a separate publication [reference].

FILES  /etc/sufstab contains a list of suffixes used to guide hyphenation. /tmp/rtma, rtmb, ... temporary. /dev/tty to suspend messages.

SEE ALSO  [reference], mesg

DIAGNOSTICS  none -- files not found are ignored

BUGS  roff does not check for various kinds of buffer overflow. If a fault occurs, check the input in the region where the error occurred.

OWNER  jfo, dmr, ken
NAME

sdate -- set date and time

SYNOPSIS

sdate mmddhhmm

DESCRIPTION

sdate adjusts the system's idea of the date and time. mm is the month number; dd is the day number in the month; hh is the hour number (24-hour system); mm is the minute number. For example,

sdate 10080045

sets the date to Oct. 8, 12:45 AM.

FILES

none

SEE ALSO

date

DIAGNOSTICS

"?" if the date is syntactically incorrect.

BUGS

none

OWNER

ken, dmr
NAME

sh -- shell (command interpreter)

SYNOPSIS

sh [ name [ arg₁ ... [ argₙ ] ] ]

DESCRIPTION

sh is the standard command interpreter. It is
the program which reads and arranges the execu-
tion of the command lines typed by most users.
It may itself be called as a command to interpret
files of command lines. Before discussing the
arguments to the shell used as a command, the
structure of command lines themselves will be
given.

Command lines are sequences of commands separated
by command delimiters. Each command is a se-
quence of non-blank command arguments separated
by blanks. The first argument specifies the name
of a command to be executed. Except for certain
types of special arguments discussed below, the
arguments other than the command name are simply
passed to the invoked command.

If the first argument represents the path name of
an executable file, it is invoked; otherwise the
string "%/bin/" is prepended to the argument. (In
this way the standard commands, which reside in
%/bin", are found.) If this search too fails a
diagnostic is printed.

The remaining non-special arguments are simply
passed to the command without further interpreta-
tion by the shell.

There are three command delimiters: the new
line, ";", and "&". The semicolon ";" specifies
sequential execution of the commands so
separated; that is,

coma; comb

causes the execution first of command coma, then
of comb. The ampersand "&" causes simultaneous
execution:

coma & comb

causes coma to be called, followed immediately by
comb without waiting for coma to finish. Thus
coma and comb execute simultaneously. As a spe-
cial case,

coma &

causes coma to be executed and the shell immedi-
ately to request another command without waiting
for coma.
Two characters cause the immediately following string to be interpreted as a special argument to the shell itself, not passed to the command. An argument of the form "<arg" causes the file arg to be used as the standard input file of the given command; an argument of the form ">arg" causes file "arg" to be used as the standard output file for the given command.

If any argument contains either of the characters "?" or "*", it is treated specially as follows. The current directory is searched for files which match the given argument. The character "*" in an argument matches any string of characters in a file name (including the null string); "?" matches any single character in a file name. Other argument characters match only the same character in the file name. For example, "*" matches all file names; "?" matches all one-character file names; "ab*.s" matches all file names beginning with "ab" and ending with "s".

If the argument with "*" or "?" also contains a "/", a slightly different procedure is used: instead of the current directory, the directory used is the one obtained by taking the argument up to the last "/" before a "*" or "?". The matching process matches the remainder of the argument after this "/" against the files in the derived directory. For example; "/usr/dmr/a*.s" matches all files in directory "/usr/dmr" which begin with "a" and end with "s".

In any event, a list of names is obtained which match the argument. This list is sorted into alphabetical order, and the resulting sequence of arguments replaces the single argument containing the "*" or "?". The same process is carried out for each argument with a "*" or "?" (the resulting lists are not merged) and finally the command is called with the resulting list of arguments.

For example: directory /usr/dmr contains the files a1.s, a2.s, ..., a9.s. From any directory, the command

    as /usr/dmr/a?.s

calls as with arguments /usr/dmr/a1.s, /usr/dmr/a2.s, ..., /usr/dmr/a9.s in that order.

The character "\" causes the immediately following character to lose any special meaning it may have to the shell; in this way "<", ">", and other characters meaningful to the shell may be passed as part of arguments. A special case of
this feature allows the continuation of commands onto more than one line: a new-line preceded by \\ is translated into a blank.

Sequences of characters enclosed in double (" ) or single ( ' ) quotes are also taken literally.

When the shell is invoked as a command, it has additional string processing capabilities. Recall that the form in which the shell is invoked is

    sh [ name [ arg1 ... [ arg9 ] ] ]

The name is the name of a file which will be read and interpreted. If not given, this subinstance of the shell will continue to read the standard input file.

In the file, character sequences of the form "$n", where n is a digit 0, ..., 9, are replaced by the nth argument to the invocation of the shell (argn). "$0" is replaced by name.

An end-of-file in the shell’s input causes it to exit. A side effect of this fact means that the way to log out from UNIX is to type an end of file.

FILES
/etc/glob

SEE ALSO [reference], which gives the theory of operation of the shell.

DIAGNOSTICS
"?", in case of any difficulty. The most common problem is inability to find the given command. Others: input file ("<") cannot be found; no more processes can be created (this will alleviate itself with the passage of time). Note that no diagnostic is given for inability to create an output (">") file; the standard output file has already been closed when the condition is discovered and there is no place to write the diagnostic.

If a "*" or "?" is used, the glob routine is invoked; it types "No command" if it cannot find the given command, and "No match" if there were no files which matched an argument with "?" or "*".

BUGS
Better diagnostics should be provided. If a "*" or "?" is used, the command must be in /bin. (Not, for example, in the user’s directory.) This is actually a glob bug.
11/3/71

OWNER

DMR, Ken
NAME
stat -- get file status

SYNOPSIS
stat name1 ...

DESCRIPTION
stat gives several kinds of information about one
or more files:

  i-number
  access mode
  number of links
  owner
  size in bytes
  date and time of last modification
  name (useful when several files are named)

All information is self-explanatory except the
mode. The mode is a six-character string whose
characters mean the following:

  1 s: file is small (smaller than 4096 bytes)
     l: file is large

  2 d: file is a directory
     x: file is executable
     u: set user ID on execution
     -: none of the above

  3 r: owner can read
     -: owner cannot read

  4 w: owner can write
     -: owner cannot write

  5 r: non-owner can read
     -: non-owner cannot read

  6 w: non-owner can write
     -: non-owner cannot write

The owner is almost always given in symbolic
form; however if he cannot be found in
"/etc/uids" a number is given.

If the number of arguments to stat is not exactly
1 a header is generated identifying the fields of
the status information.

FILES
/etc/uids

SEE ALSO
ls with the "-l" option gives the same information as stat.

DIAGNOSTICS
"name?" for any error.

BUGS
none
NAME  strip -- remove symbols and relocation bits

SYNOPSIS  strip name1 ... 

DESCRIPTION  strip removes the symbol table and relocation bits ordinarily attached to the output of the assembler and loader. This is useful to save space after a program has been debugged.

The effect of strip is the same as use of the -s option of ld.

FILES  /tmp/stma, stmb ... temporary file

SEE ALSO  ld, as

DIAGNOSTICS  Diagnostics are given for: non-existent argument; inability to create temporary file; improper format (not an object file); inability to re-read temporary file.

BUGS  --

OWNER  dmr
su -- become privileged user

DESCRIPTION
su allows one to become the super-user, who has all sorts of marvelous powers. In order for su to do its magic, the user must pass as an argument a password. If the password is correct, su will execute the shell with the UID set to that of the super-user. To restore normal UID privileges, type an end-of-file to the super-user shell.

SEE ALSO
shell

DIAGNOSTICS
"Sorry" if password is wrong

BUGS
--

OWNER
dmr, ken
NAME  sum -- sum file

SYNOPSIS  sum name

DESCRIPTION  sum sums the contents of a file. In practice, it is most often used to verify that all of a DECTape can be read without error.

FILES  none

SEE ALSO  --

DIAGNOSTICS  "?" if the file cannot be read at all or if an error is discovered during the read.

BUGS  none

OWNER  ken
NAME

tap -- manipulate DECTape

SYNOPSIS

tap [ key ] [ name ... ]

DESCRIPTION

tap saves and restores selected portions of the
file system hierarchy on DECTape. Its actions
are controlled by the key argument. The key is a
string of characters containing at most one func-
tion letter and possibly one or more function
modifiers. Other arguments to the command are
file or directory names specifying which files
are to be dumped, restored, or tabled.

The function portion of the key is specified by
one of the following letters:

r The indicated files and directories, to-
gether with all subdirectories, are dumped
onto the tape. If files with the same
names already exist, they are replaced
(hence the "r"). "Same" is determined by
string comparison, so "/abc" can never be
the same as "/usr/dmr/abc" even if
"/usr/dmr" is the current directory. If no
file argument is given, "/" is the default.

u updates the tape. u is the same as r, but
a file is replaced only if its modification
date is later than the date stored on the
tape; that is to say, if it has changed
since it was dumped. u is the default com-
mand if none is given.

d deletes the named files and directories
from the tape. At least one file argument
must be given.

x extracts the named files from the tape to
the file system. The owner, mode, and
date-modified are restored to what they
were when the file was dumped. If no file
argument is given, the entire contents of
the tape are extracted.

t lists the names of all files stored on the
tape which are the same as or are hierarch-
ically below the file arguments. If no
file argument is given, the entire contents
of the tape are tabled.

l is the same as t except that an expanded
listing is produced giving all the avail-
able information about the listed files.

The following characters may be used in addition
to the letter which selects the function desired.
0, ..., 7 This modifier selects the drive on which the tape is mounted. "0" is the default.

v Normally tap does its work silently. The v (verbose) option causes it to type the name of each file it treats preceded by a letter to indicate what is happening.

    r file is being replaced
    a file is being added (not there before)
    x file is being extracted
    d file is being deleted

The v option can be used with r, u, d, and x only.

c means a fresh dump is being created; the tape directory will be zeroed before begin- ning. Usable only with r and u.

f causes new entries copied on tape to be 'fake' in that only the entries, not the data associated with the entries are updat ed. Such fake entries cannot be extracted. Usable only with r and u.

w causes tap to pause before treating each file, type the indicative letter and the file name (as with v) await the user's response. Response 'y" means "yes", so the file is treated. Null response means "no", and the file does not take part in whatever is being done. Response 'x' means "exit"; the tap command terminates immediately. In the x function, files previously asked about have been extracted already. With r, u, and d no change has been made to the tape.

m make (create) directories during an x if necessary.

i ignore tape errors. It is suggested that this option be used with caution to read damaged tapes.

FILES /dev/tap0 ... /dev/tap7

SEE ALSO rk

DIAGNOSTICS RK open error
RK read error
RK write error
Directory checksum
Directory overflow
RK overflow
Phase error (a file has changed after it was selected for dumping but before it was dumped)

BUGS
All references to "RK" should read "tape." The m option does not work correctly in all cases. The i option is not yet implemented.

OWNER
ken
NAME       tm  -- provide time information

SYNOPSIS   tm [ command arg1 .... ]

DESCRIPTION tm is used to provide timing information. When used without an argument, output like the following is given:

    tim    77:43:20    29.2
    ovh    13:59:42    1.2
    dsk    12:06:30    4.1
    idl    352:31:37   23.7
    usr    3:32:15     0.1
    der    5, 171      0, 0

The first column of numbers gives totals in the named categories since the last time the system was cold-booted; the second column gives the changes since the last time tm was invoked. The tim row is total real time (hours:minutes:seconds); unlike the other times, its origin is the creation date of tm's temporary file. ovh is time spent executing in the system; dsk is time spent waiting for both kinds of disk I/O; idl is idle time; usr is user execution time; der is RF disk error count (left number) and RK disk error count (right number).

tm can be invoked with arguments which are assumed to constitute a command to be timed. In this case the output is as follows:

    tim    2.2
    ovh    0.3
    dsk    1.8
    idl    0.0
    usr    0.0

The given times represent the number of seconds spent in each category during execution of the command.

FILES      /tmp/ttmp, /dev/rfo (for absolute times) contains the information used to calculate the differential times.

SEE ALSO   format of file system (which tells where the times come from)

DIAGNOSTICS   "?" if the command cannot be executed; "can't creat temp file" if trouble with ttmp; "can't read super-block" if times cannot be read from system.

BUGS       (1) when invoked with a command argument, everything going on at the moment is counted, not just the command itself. (2) Two users doing tm...
simultaneously interfere with each other's use of the temporary file.

OWNER

ken, dmr
NAME  tty  -- get tty name

SYNOPSIS  tty

DESCRIPTION  tty gives the name of the user’s typewriter in the form "ttyn" for n a digit. The actual path name is then "/dev/ttyn".

FILES  --

SEE ALSO  --

DIAGNOSTICS  "not a tty" if the standard input file is not a typewriter.

BUGS  --

OWNER  dmr, ken
NAME type -- type on 2741

SYNOPSIS type name, ...

DESCRIPTION type produces output on an IBM 2741 terminal with a Correspondence type ball.

type uses typewriter tty5, which, because of the lack of access ports, is also used as a standard communication channel. Therefore, who should be used to verify the absence of a user on tty5.

The method is as follows: type the type command. It will wait until tty5 is dialled up. When the phone answers, depress the interrupt button after paper has been loaded, and the first file will be typed. type spaces out to the end of a sheet of paper and waits until the interrupt button is depressed before beginning each new file.

FILES /dev/tty5

SEE ALSO who

DIAGNOSTICS --

BUGS Obviously some scheme is needed to prevent interference between normal users and type. The best thing would be to support 2741's as a standard terminal.

OWNER dmr
NAME     umount -- dismount file system

SYNOPSIS   umount special

DESCRIPTION  umount announces to the system that the removable file system previously mounted on special file special is to be removed.

Only the super-user may issue this command.

FILES     --

SEE ALSO   mount

DIAGNOSTICS  ?

BUGS       This command should be restricted to the super-user.

OWNER     ken, dmr
NAME    un -- undefined symbols

SYNOPSIS un [ name ]

DESCRIPTION un prints a list of undefined symbols from an assembly or loader run. If the file argument is not specified, a.out is the default. Names are listed alphabetically except that non-global symbols come first. Undefined global symbols (un-resolved external references) have their first character underlined.

FILES a.out

SEE ALSO as, ld

DIAGNOSTICS "?" if the file cannot be found.

BUGS --

OWNER dmr, ken
NAME  wc -- get (English) word count

SYNOPSIS  wc name, ...

DESCRIPTION  wc provides a count of the words, text lines, and roff control lines for each argument file.

A text line is a sequence of characters not beginning with "." and ended by a new-line. A roff control line is a line beginning with ".". A word is a sequence of characters bounded by the beginning of a line, by the end of a line, or by a blank or a tab.

FILES  --

SEE ALSO  roff

DIAGNOSTICS  none; arguments not found are ignored.

BUGS  --

OWNER  jfo
NAME who -- who is on the system
SYNOPSIS who
DESCRIPTION who lists the name, typewriter channel, and login time for each current UNIX user.
FILES /tmp/utmp contains the necessary information; it is maintained by init.
SEE ALSO /etc/init
DIAGNOSTICS --
BUGS --
OWNER dmr, ken
NAME
write -- write to another user

SYNOPSIS
write user

DESCRIPTION
write copies lines from your typewriter to that of another user. When first called, write sends the message

message from yourname...

The recipient of the message should write back at this point. Communication continues until an end of file is read from the typewriter or an interrupt is sent. At that point write writes "EOT" on the other terminal.

Permission to write may be denied or granted by use of the mesg command. At the outset writing is allowed. Certain commands, in particular roff and pr, disallow messages in order to prevent messy output.

If the character "!" is found at the beginning of a line, write calls the mini-shell msh 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 him to write back before starting to send. Each party should end each message with a distinctive signal ("(o)" for "over" is conventional) that the other may reply. "(oo)" (for "over and out") is suggested when conversation is about to be terminated.

FILES
/tmp/utmp is used to discover the target user's typewriter channel and the sending user's name.

msh is used to execute commands.

SEE ALSO
mesg

DIAGNOSTICS
"user not logged in"; "permission denied".

BUGS
--

OWNER
dmr, ken
NAME       break -- set program break
SYNOPSIS   sys break; addr / break = 17.
DESCRIPTION break sets the system's idea of the highest location used by the program to addr. Locations greater than addr and below the stack pointer are not swapped and are thus liable to unexpected modification.

If the argument is 0 or higher than the stack pointer the entire 4K word user core area is swapped.

When a program begins execution via exec the break is set at the highest location defined by the program and data storage areas. Ordinarily, therefore, only programs with growing data areas need to use break.

FILES       --
SEE ALSO    exec
DIAGNOSTICS none; strange addresses cause the break to be set to include all of core.
BUGS        --
OWNER       ken, dmr
NAME  cemt  --  catch emt traps  
SYNOPSIS  sys cemt; arg / cemt = 29.; not in assembler  
DESCRIPTION  This call allows one to catch traps resulting from the emt instruction. Arg is a location within the program; emt traps are sent to that location. The normal effect of emt traps may be restored by giving an arg equal to 0. 

Prior to the use of this call, the result of an emt instruction is a simulated rts instruction. The operand field is interpreted as a register, and an rts instruction is simulated for that register (after verifying that various registers have appropriate values). This feature is useful for debugging, since the most dangerous program bugs usually involve an rts with bad data on the stack or in a register.  
FILES  --  
SEE ALSO  --  
DIAGNOSTICS  --  
BUGS  --  
OWNER  ken, dmr
<table>
<thead>
<tr>
<th>NAME</th>
<th>chdir -- change working directory</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYNOPSIS</td>
<td>sys chdir; dirname / chdir = 12.</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>dirname is address of the pathname of a directory, terminated by a 0 byte. chdir causes this directory to become the current working directory.</td>
</tr>
<tr>
<td>FILES</td>
<td>--</td>
</tr>
<tr>
<td>SEE ALSO</td>
<td>--</td>
</tr>
<tr>
<td>DIAGNOSTICS</td>
<td>The error bit (c-bit) is set if the given name is not that of a directory.</td>
</tr>
<tr>
<td>BUGS</td>
<td>--</td>
</tr>
<tr>
<td>OWNER</td>
<td>ken, dmr</td>
</tr>
</tbody>
</table>
NAME   chmod  --  change mode of file
SYNOPSIS  sys chmod; name; mode       / chmod = 15.
DESCRIPTION  The file whose name is given as the null-terminated string pointed to by name has its mode changed to mode. Modes are constructed by oring together some combination of the following:

  01 write, non-owner  
  02 read, non-owner   
  04 write, owner     
 10 read, owner       
 20 executable       
 40 set user ID on execution

Only the owner of a file (or the super-user) may change the mode.

FILES  --
SEE ALSO  --
DIAGNOSTICS  Error bit (c-bit) set if name cannot be found or if current user is neither the owner of the file nor the super-user.
BUGS  --
OWNER  ken, dmr
NAME        chown -- change owner of file

SYNOPSIS    sys chown; name; owner / chown = 16.

DESCRIPTION The file whose name is given by the null-terminated string pointed to by name has its owner changed to owner. Only the present owner of a file (or the super-user) may donate the file to another user. Also, one may not change the owner of a file with the set-user-ID bit on, otherwise one could create Trojan Horses able to misuse other's files.

FILES        --

SEE ALSO     /etc/uids has the mapping between user names and user numbers.

DIAGNOSTICS The error bit (c-bit) is set on illegal owner changes.

BUGS         --

OWNER        ken, dmr
NAME

  close -- close a file

SYNOPSIS

  (file descriptor in r0)
  sys   close   / close = 6.

DESCRIPTION

  Given a file descriptor such as returned from an
  open or creat call, close closes the associated
  file. A close of all files is automatic on exit,
  but since processes are limited to 10 simultaneously
  open files, close is necessary to programs
  which deal with many files.

FILES

  --

SEE ALSO

  creat, open

DIAGNOSTICS

  The error bit (c-bit) is set for an unknown file
descriptor.

BUGS

  --

OWNER

  ken, dmr
creaT -- create a new file

SYNOPSIS
sys creaT; name; mode / creaT = 8.
{file descriptor in r0}

DESCRIPTION
creaT creates a new file or prepares to rewrite
an existing file called name; name is the address
of a null-terminated string. If the file did not
exist, it is given mode mode; if it did exist,
it's mode and owner remain unchanged but it is
truncated to 0 length.

The file is also opened for writing, and its file
descriptor is returned in r0.

The mode given is arbitrary; it need not allow
writing. This feature is used by programs which
deal with temporary files of fixed names. The
creation is done with a mode that forbids writ-
ing. Then if a second instance of the program
attempts a creaT, an error is returned and the
program knows that the name is unusable for the
moment.

If the last link to an open file is removed, the
file is not destroyed until the file is closed.

FILES
--

SEE ALSO
write, close

DIAGNOSTICS
The error bit (c-bit) may be set if: a needed
directory is not readable; the file does not
exist and the directory in which it is to be
created is not writable; the file does exist and
is writable; the file is a directory.

BUGS
--

OWNER
ken, dmr
NAME
exec -- execute a file

SYNOPSIS
sys exec; name; args / exec = 11.

name: <...\0>
...
args: arg1; arg2; ...; 0
arg1: <...\0>
...

DESCRIPTION
exec overlays the calling process with the named file, then transfers to the beginning of the core image of the file. The first argument to exec is a pointer to the name of the file to be executed. The second is the address of a list of pointers to arguments to be passed to the file. Conventionally, the first argument is the name of the file. Each pointer addresses a string terminated by a null byte.

There can be no return from the file; the calling core image is lost.

The program break is set from the executed file; see the format of a.out.

Once the called file starts execution, the arguments are passed as follows. The stack pointer points to the number of arguments. Just above this number is a list of pointers to the argument strings.

sp-> nargs
arg1
...
argn

arg1: <arg1\0>
...
argn: <argn\0>

The arguments are placed as high as possible in core: just below 60000(8).

Files remain open across exec calls. However, the illegal instruction, emt, quit, and interrupt trap specifications are reset to the standard values. (See ilqins, cent, quit, intr.)

Each user has a real user ID and an effective (The real ID identifies the person using the system; the effective ID determines his access privileges.) exec changes the effective user ID to the owner of the executed file if the file has the "set-user-ID" mode. The real user ID is not affected.
FILES

SEE ALSO fork

DIAGNOSTICS If the file cannot be read or if it is not executable, a return from exec constitutes the diagnostic. The error bit (c-bit) is set.

BUGS

OWNER ken, dmr
NAME       exit   -- terminate process
SYNOPSIS   sys exit    / exit = 1
DESCRIPTION exit is the normal means of terminating a process. All files are closed and the parent process is notified if it is executing a wait.

This call can never return.

FILES      --
SEE ALSO    sys wait
DIAGNOSTICS -
BUGS        --
OWNER       ken, dmr
NAME          fork         -- spawn new process

SYNOPSIS     sys   fork   / fork = 2.
              (new process return)
              (old process return)

DESCRIPTION  fork is the only way new processes are created. The new process's core image is a copy of that of the caller of fork; the only distinction is the return location and the fact that r0 in the old process contains the process ID of the new process. This process ID is used by wait.

FILES         --

SEE ALSO      sys wait, sys exec

DIAGNOSTICS  The error bit (c-bit) is set in the old process if a new process could not be created because of lack of swap space.

BUGS          See wait for a subtle bug in process destruction.

OWNER         ken, dmr
NAME      fstat -- get status of open file

SYNOPSIS  (file descriptor in r0)
           sys    fstat; buf     / fstat = 28.

DESCRIPTION This call is identical to stat, except that it
operates on open files instead of files given by
name. It is most often used to get the status of
the standard input and output files, whose names
are unknown.

FILES      --

SEE ALSO   sys stat

DIAGNOSTICS The error bit (c-bit) is set if the file descrip-
tor is unknown.

BUGS       --

OWNER      ken, dmr
NAME getuid -- get user identification

SYNOPSIS sys getuid / getuid = 24.
(user ID in r0)

DESCRIPTION getuid returns the real user ID of the current
process. The real user ID identifies the person
who is logged in, in contradistinction to the
effective user ID, which determines his access
permission at each moment. It is thus useful to
programs which operate using the "set user ID"
mode, to find out who invoked them.

FILES /etc/uids can be used to map the user ID number
into a name.

SEE ALSO setuid

DIAGNOSTICS --

BUGS --

OWNER ken, dmr
NAME       gttty -- get typewriter status

SYNOPSIS   (file descriptor in r0)
            sys gttty; arg / gttty = 32.; not in assembler
            ...
            arg: .=.+6

DESCRIPTION gttty stores in the three words addressed by arg
            the status of the typewriter whose file descrip-
            tor is given in r0. The format is the same as
            that passed by stty.

FILES      --

SEE ALSO   stty

DIAGNOSTICS Error bit (c-bit) is set if the file descriptor
            does not refer to a typewriter.

BUGS       --

OWNER      ken, dmr
NAME    ilgins -- catch illegal instruction trap
SYNOPSIS sys ilgins; arg / ilgins = 33.; not in assembler
DESCRIPTION ilgins allows a program to catch illegal instruction traps. If arg is zero, the normal instruction trap handling is done: the process is terminated and a core image is produced. If arg is a location within the program, control is passed to arg when the trap occurs.

This call is used to implement the floating point simulator, which catches and interprets 11/45 floating point instructions.

FILES    --
SEE ALSO fptrap, the floating point package
DIAGNOSTICS --
BUGS      --
OWNER     ken, dmr
NAME       intr  --  set interrupt handling
SYNOPSIS   sys    intr; arg    / intr = 27.
DESCRIPTION When arg is 0, interrupts (ASCII DELETE) are ignored. When arg is 1, interrupts cause their normal result, that is, force an exit. When arg is a location within the program, control is transferred to that location when an interrupt occurs.

After an interrupt is caught, it is possible to resume execution by means of an rti instruction; however, great care must be exercised, since all I/O is terminated abruptly upon an interrupt. In particular, reads of the typewriter tend to return with 0 characters read, thus simulating an end of file.

FILES       --
SEE ALSO    quit
DIAGNOSTICS --
BUGS        It should be easier to resume after an interrupt, but I don't know how to make it work.
OWNER       ken, dmr
NAME  link -- link to a file
SYNOPSIS  sys link; name$_1$; name$_2$  / link = 9.
DESCRIPTION  A link to name$_1$ is created; the link has name name$_2$. Either name may be an arbitrary path name.
FILES  --
SEE ALSO  unlink
DIAGNOSTICS  The error bit (c-bit) is set when name$_1$ cannot be found; when name$_2$ already exists; when the directory of name$_2$ cannot be written; when an attempt is made to link to a directory by a user other than the super-user.
BUGS  --
OWNER  ken, dmr
NAME
mkdir -- make a directory

SYNOPSIS
sys mkdir; name; mode / mkdir = 14.

DESCRIPTION
mkdir creates an empty directory whose name is
the null-terminated string pointed to by name.
The mode of the directory is mode. The special
entries "." and ".." are not present.

mkdir can only be invoked by the super-user.

FILES
--

SEE ALSO
mkdir command

DIAGNOSTICS
Error bit (c-bit) is set if the directory already
exists or if the user is not the super-user.

BUGS
--

OWNER
ken, dmr
NAME

mount -- mount file system

SYNOPSIS

sys mount; special; name / mount = 21.; not in assembler

DESCRIPTION

mount announces to the system that a removable file system has been mounted on special file special; from now on, references to file name will refer to the root file on the newly mounted file system. Special and name are pointers to null-terminated strings containing the appropriate path names.

Name must exist already. If it had useful contents, they are inaccessible while the file system is mounted.

Almost always, name should be a directory so that an entire file system, not just one file, may exist on the removable device.

FILES

--

SEE ALSO

umount

DIAGNOSTICS

Error bit (c-bit) set if special is inaccessible or dir does not exist.

BUGS

At most one removable device can be mounted at a time. The use of this call should be restricted to the super-user.

OWNER

ken, dmr
NAME open -- open for reading or writing

SYNOPSIS sys open; name; mode / open = 5.
{descriptor in r0}

DESCRIPTION open opens the file name for reading (if mode is 0) or writing (if mode is non-zero). name is the address of a string of ASCII characters representing a path name, terminated by a null character.

The file descriptor should be saved for subsequent calls to read (or write) and close.

In both the read and write case the file pointer is set to the beginning of the file.

If the last link to an open file is removed, the file is not destroyed until it is closed.

FILES --

SEE ALSO creat, read, write, close

DIAGNOSTICS The error bit (c-bit) is set if the file does not exist, if one of the necessary directories does not exist or is unreadable, or if the file is not readable.

BUGS --

OWNER ken, dmr
NAME        quit -- turn off quit signal

SYNOPSIS    sys quit; flag / quit = 26.

DESCRIPTION When flag is 0, this call disables quit signals from the typewriter (ASCII FS). When flag is 1, quits are re-enabled, and cause execution to cease and a core image to be produced. When flag is an address in the program, a quit causes control to be sent to that address.

Quits should be turned off only with due consideration.

FILES       --

SEE ALSO    sys intr turns off interrupts

DIAGNOSTICS --

BUGS        --

OWNER       ken, dmr
NAME
read -- read from file

SYNOPSIS
(sys descriptor in r0)
sys read; buffer; nchars / read = 3.
(nread in r0)

DESCRIPTION
A file descriptor is a word returned from a successful open call.

Buffer is the location of nchars contiguous bytes into which the input will be placed. It is not
guaranteed that all nchars bytes will be read, however; for example if the file refers to a
typewriter at most one line will be returned. In any event the number of characters read is re-
turned in r0.

If r0 returns with value 0, then end-of-file has been reached.

FILES
--

SEE ALSO
open

DIAGNOSTICS
As mentioned, r0 is 0 on return when the end of
the file has been reached. If the read was
otherwise unsuccessful the error bit (c-bit) is
set. Many conditions, all rare, can generate an
error: physical I/O errors, bad buffer address,
preposterous nchars, file descriptor not that of
an input file.

BUGS
--

OWNER
ken, dmr
NAME rele -- release processor

SYNOPSIS sys rele / rele = 0; not in assembler

DESCRIPTION This call causes the process to be swapped out immediately if another process wants to run. Its main reason for being is internal to the system, namely to implement timer-runout swaps. However, it can be used beneficially by programs which wish to loop for some reason without consuming more processor time than necessary.

FILES --

SEE ALSO --

DIAGNOSTICS --

BUGS --

OWNER ken, dmr
NAME

seek -- move read/write pointer

SYNOPSIS

(file descriptor in r0)
sys seek; offset; ptrname / seek = 19.

DESCRIPTION

The file descriptor refers to a file open for reading or writing. The read (or write) pointer for the file is set as follows:

if ptrname is 0, the pointer is set to offset.
if ptrname is 1, the pointer is set to its current location plus offset.
if ptrname is 2, the pointer is set to the size of the file plus offset.

FILES

--

SEE ALSO

tell

DIAGNOSTICS

The error bit (c-bit) is set for an undefined file descriptor.

BUGS

A file can conceptually be as large as 2**20 bytes. Clearly only 2**16 bytes can be addressed by seek. The problem is most acute on the tape files and RK and RF. Something is going to be done about this.

OWNER

ken, dmr
NAME    setuid -- set process ID

SYNOPSIS (process ID in r0)
         sys    setuid    / setuid = 23.

DESCRIPTION The user ID of the current process is set to the argument in r0. Both the effective and the real user ID are set. This call is only permitted to the super-user.

FILES   --

SEE ALSO getuid

DIAGNOSTICS Error bit (c-bit) is set if the current user ID is not that of the super-user.

BUGS    --

OWNER   ken, dmr
NAME       smdate -- set modified date on file

SYNOPSIS   (time to AC-MQ)
            sys  smdate; file / smdate = 30.; not in assembler

DESCRIPTION File is the address of a null-terminated string
giving the name of a file. The modified time of
the file is set to the time given in the AC-MQ
registers.

This call is allowed only to the super-user.

FILES      --

SEE ALSO   --

DIAGNOSTICS Error bit is set if the user is not the super-
user or if the file cannot be found.

BUGS       --

OWNER      ken, dmr
NAME        stat  --  get file status

SYNOPSIS    sys stat; name; buf / stat = 18.

DESCRIPTION name points to a null-terminated string naming a file; buf is the address of a 34(10) byte buffer into which information is placed concerning the file. It is unnecessary to have any permissions at all with respect to the file, but all directories leading to the file must be readable.

After stat, buf has the following format:

buf, +1     i-number
+2,+3       flags (see below)
+4          number of links
+5          user ID of owner
+6,+7       size in bytes
+8,+9       first indirect block or contents block
...
+22,+23     eighth indirect block or contents block
+24,+25,+26,+27 creation time
+28,+29,+30,+31 modification time
+32,+33     unused

The flags are as follows:

100000 used (always on)
040000 directory
020000 file has been modified (always on)
010000 large file
000040 set user ID
000020 executable
000010 read, owner
000004 write, owner
000002 read, non-owner
000001 write, non-owner

FILES

SEE ALSO    fstat

DIAGNOSTICS Error bit (c-bit) is set if the file cannot be found.

BUGS        The format is going to change someday.

OWNER       ken, dmr
NAME        stime -- set time

SYNOPSIS    (time in AC-MQ)
            sys    stime / stime = 25.; not in assembler

DESCRIPTION stime sets the system's idea of the time and
date. Only the super-user may use this call.

FILES       --

SEE ALSO    sys time

DIAGNOSTICS Error bit (c-bit) set if user is not the super-
user.

BUGS        --

OWNER       ken, dmr
NAME       stty -- set mode of typewriter

SYNOPSIS  (file descriptor in r0)
    sys    stty; arg / stty = 31.; not in assembler
    ...
    arg:  dcrsr; dcpsr; mode

DESCRIPTION stty sets mode bits for a typewriter whose file
descriptor is passed in r0. First, the system
delays until the typewriter is quiescent. Then,
the argument dcrsr is placed into the typewri-
ter's reader control and status register, and
dcpsr is placed in the printer control and status
register. The DC-11 manual must be consulted for
the format of these words. For the purpose of
this call, the most important role of these argu-
ments is to adjust to the speed of the typewriter.

The mode arguments contains several bits which
determine the system's treatment of the
typewriter:

   200  even (M37 tty) parity allowed
   100  odd  (non-M37 tty) allowed
   040  raw mode: wake up on all characters
   020  map CR into LF; echo LF or CR as CR-LF
   010  don't echo (half duplex)
   004  map upper case to lower case on input (M33 TTY)

Characters with the wrong parity, as determined
by bits 200 and 100, are ignored.

In raw mode, every character is passed back im-
mediately to the program. No erase or kill pro-
cessing is done; the end-of-file character (EOT),
the interrupt character (DELETE) and the quit
character (FS) are not treated specially.

Mode 020 causes input carriage returns to be
turned into new-lines; input of either CR or LF
causes CR-LF both to be echoed (used for GE Ter-
miNet 300's).

FILES    --

SEE ALSO  gtty

DIAGNOSTICS The error bit (c-bit) is set if the file descrip-
tor does not refer to a typewriter.

BUGS     This call should be used with care. It is all
too easy to turn off your typewriter.

OWNER    ken, dmr
NAME  tell -- get file pointer

SYNOPSIS (file descriptor in r0)
sys tell; offset; ptrname / tell = 20.
(value returned in r0)

DESCRIPTION The file descriptor refers to an open file. The value returned in r0 is one of:

if ptrname is 0, the value returned is offset;
if ptrname is 1, the value is the current pointer plus offset;
if ptrname is 2, the value returned is the number of bytes in the file plus offset.

FILES --

SEE ALSO seek

DIAGNOSTICS The error bit (c-bit) is set if the file descriptor is unknown.

BUGS Tell doesn't work. Complain if you need it.

OWNER ken, dmr
NAME       time  --  get time of year

SYNOPSIS   sys     time   / time = 13.
            (time AC-MQ)

DESCRIPTION time returns the time since 00:00:00, Jan. 1, 1971, measured in sixtieths of a second. The high order word is in the AC register and the low order is in the MQ.

FILES      --

SEE ALSO   --

DIAGNOSTICS --

BUGS       The chronological-minded user will note that 2**32 sixtieths of a second is only about 2.5 years.

OWNER      ken, dmr
NAME

umount  -- dismount file system

SYNOPSIS

sys  umount;  special  /  umount = 22.; not in assembler

DESCRIPTION

umount announces to the system that special file special is no longer to contain a removable file system. The file associated with the special file reverts to its ordinary interpretation (see mount).

The user must take care that all activity on the file system has ceased.

FILES

--

SEE ALSO

mount

DIAGNOSTICS

Error bit (c-bit) set if no file system was mounted on the special file.

BUGS

Use of this call should be restricted to the super-user.

OWNER

ken, dmr
NAME  unlink -- remove directory entry

SYNOPSIS  sys    unlink; name    / unlink = 10.

DESCRIPTION  Name points to a null-terminated string. Unlink removes the entry for the file pointed to by name from its directory. If this entry was the last link to the file, the contents of the file are freed and the file is destroyed. If, however, the file was open in any process, the actual destruction is delayed until it is closed, even though the directory entry has disappeared.

FILES

SEE ALSO  link

DIAGNOSTICS  The error bit (c-bit) is set to indicate that the file does not exist or that its directory cannot be written. Write permission is not required on the file itself. It is also illegal to unlink a directory (except for the super-user).

BUGS  Probably write permission should be required to remove the last link to a file, but this gets in other problems (namely, one can donate an undeletable file to someone else).

If the system crashes while a file is waiting to be deleted because it is open, the space is lost.

OWNER  ken, dmr
NAME
wait -- wait for process to die

SYNOPSIS
sys wait / wait = 7.
(process ID in r0)

DESCRIPTION
wait causes its caller to delay until one of its child processes terminates. If any child has already died, return is immediate; if there are no children, return is immediate with the error bit set. In the case of several children several waits are needed to learn of all the deaths.

FILES
--

SEE ALSO
fork

diagonistics
error bit (c-bit) on if no children not previously waited for.

BUGS
A child which dies but is never waited for is not really gone in that it still consumes disk swap and system table space. This can make it impossible to create new processes. The bug can be noticed when several "&" separators are given to the shell not followed by an command without an ampersand. Ordinarily things clean themselves up when an ordinary command is typed, but it is possible to get into a situation in which no commands are accepted, so no waits are done; the system is then hung.

The fix, probably, is to have a new kind of fork which creates a process for which no wait is necessary (or possible); also to limit the number of active or inactive descendants allowed to a process.

OWNER
ken, dmr
NAME  write -- write on file

SYNOPSIS  (file descriptor in r0)
sys write; buffer; nchars / write = 4.
(number written in r0)

DESCRIPTION  A file descriptor is a word returned from a successful open or creat call.

buffer is the address of nchars contiguous bytes which are written on the output file. The number of characters actually written is returned in r0. It should be regarded as an error if this is not the same as requested.

For disk and tape files, writes which are multiples of 512 characters long and begin on a 512-byte boundary are more efficient than any others.

FILES  --

SEE ALSO  sys creat, sys open

DIAGNOSTICS  The error bit (c-bit) is set on an error: bad descriptor, buffer address, or count. physical I/O errors;

BUGS  --

OWNER  ken, dmr
NAME    atof -- ascii to floating

SYNOPSIS jsr  r5,atof; subr

DESCRIPTION atof will convert an ascii stream to a floating number returned in fr0. The subroutine subr is called on r5 for each character of the ascii stream. subr should return the character in r0. The first character not used in the conversion is left in r0. The floating point simulation should be active in either floating or double mode, but in single precision integer mode.

FILES    kept in /etc/liba.a

SEE ALSO fpstrap

DIAGNOSTICS --

BUGS    The subroutine subr should not disturb any registers.

OWNER    ken
NAME           atoi -- ascii to integer

SYNOPSIS       jsr r5,atoi; subr

DESCRIPTION    atoi will convert an ascii stream to a binary number returned in mq. The subroutine subr is called on r5 for each character of the ascii stream. subr should return the character in r0. The first character not used in the conversion is left in r0.

FILES          kept in /etc/liba.a

SEE ALSO       --

DIAGNOSTICS    --

BUGS           The subroutine subr should not disturb any registers.

OWNER          ken
NAME

ctime -- convert date and time to ASCII

SYNOPSIS

(move time to AC-MQ)
mov $buffer,r0
jsr pc,ctime

DESCRIPTION

The buffer is 15 characters long. The time has the format

Oct 9 17:32:24

The input time is in the AC and MQ registers in the form returned by sys time.

FILES

kept in /etc/liba.a

SEE ALSO

ptime, to print time; sys time

DIAGNOSTICS

--

BUGS

The time is not taken modulo 1 year. (Jan 1 comes out Dec 32.) Also, the clock period is only a couple of years.

dmr
NAME exp -- exponential function

SYNOPSIS jsr r5,exp

DESCRIPTION The exponential of fr0 is returned in fr0. The floating point simulation should be active in either floating or double mode, but in single precision integer mode.

FILES kept in /etc/liba.a

SEE ALSO fptrap

DIAGNOSTICS --

BUGS Large arguments will cause an overflow fault from the floating point simulator.

OWNER ken
**NAME**  fptrap -- floating point simulator

**SYNOPSIS**  sys 33.; fptrap

**DESCRIPTION**  fptrap is a program designed to pick up illegal instruction in order to simulate a sub-set of the 11/45 floating point hardware.

**FILES**  kept in /etc/liba.a

**SEE ALSO**  as, PDP-11/45 manual

**DIAGNOSTICS**  none, hardware gives no diagnostics.

**BUGS**  The simulation, if unsuccessful for any reason gives an IOT fault from inside the simulator. This should be handled better.

**OWNER**  ken, dm
NAME       ftoa -- floating to ascii conversion
SYNOPSIS   jsr     r5,ftoa; subr
DESCRIPTION ftoa will convert the floating point number in fr0 into ascii in the form [-]d,ddddddddd[e-]dd*. The floating point simulator should be active in either floating or double mode, but in single integer mode. For each character generated by ftoa, the subroutine subr is called on register r5 with the character in r0.
FILES      kept in /etc/liba.a
SEE ALSO   fptrap
DIAGNOSTICS --
BUGS       The subroutine subr should not disturb any registers.
OWNER      ken
NAME
getw, getc, fopen -- buffered input

SYNOPSIS
mov $filename, r0
jsr r5, fopen; iobuf

jsr r5, getc; iobuf
(character in r0)

jsr r5, getw; iobuf
(word in r0)

DESCRIPTION
These routines are used to provide a buffered input facility. iobuf is the address of a
134(10) byte buffer area whose contents are maintained by these routines. Its format is:

ioptr: .=.+2       / file descriptor
 .=.+2       / characters left in buffer
 .=.+2       / ptr to next character
 .=.+128.    / the buffer

fopen should be called initially to open the file. On return, the error bit (c-bit) is set if
the open failed. If fopen is never called, get will read from the standard input file.

getc returns the next byte from the file in r0. The error bit is set on end of file or a read
error.

getw returns the next word in r0. getc and getw may be used alternately; there are no odd/even
problems.

iobuf must be provided by the user; it must be on a word boundary.

FILES
kept in /etc/liba.a

SEE ALSO
sys open, sys read; putc, putw, fcreat

DIAGNOSTICS
c-bit set on EOF or error

BUGS
for greater speed, the buffer should be 512 bytes long. Unfortunately, this will cause several
existing programs to stop working.

OWNER
dmr
NAME       itoa -- integer to ascii conversion

SYNOPSIS   jsr r5,itoa; subr

DESCRIPTION itoa will convert the number in r0 into ascii decimal possibly preceded by a - sign. For each character generated by itoa, the subroutine subr is called on register r5 with the character in r0.

FILES      kept in /etc/liba,a

SEE ALSO   --

DIAGNOSTICS --

BUGS       The subroutine subr should not disturb any registers.

OWNER      ken
NAME log -- logarithm base e

SYNOPSIS jsr r5,log

DESCRIPTION The logarithm base e of fr0 is returned in fr0. The floating point simulation should be active in either floating or double mode, but in single precision integer mode.

FILES kept in /etc/liba.a

SEE ALSO fptrap

DIAGNOSTICS The error bit (c-bit) is set if the input argument is less than or equal to zero.

BUGS --

OWNER ken
NAME

mesg -- write message on typewriter

SYNOPSIS

jsr r5,mesg; (Now is the time\0); .even

DESCRIPTION

mesg writes the string immediately following its call onto the standard output file. The string is terminated by a 0 byte.

FILES

kept in /etc/liba.a, standard output file

SEE ALSO

--

DIAGNOSTICS

--

BUGS

--

OWNER

ken, dmr
NAME

ptime -- print date and time

SYNOPSIS

(move time to ac-mq)
mov file, r0
jsr pc, ptime

DESCRIPTION

ptime prints the date and time in the form

Oct 9 17:20:33

on the file whose file descriptor is in r0. The string is 15 characters long. The time to be printed is placed in the AC and MQ registers in the form returned by sys time.

FILES

kept in /etc/liba.a

SEE ALSO

sys time, ctime (used to do the conversion)

DIAGNOSTICS

--

BUGS

see ctime

OWNER

dmr, ken
NAME
putc, putw, fcreat, flush -- buffered output

SYNOPSIS
mov $filename,r0
jsr r5, fcreat; iobuf

(get byte in r0)
jsr r5, putc; iobuf

(get word in r0)
jsr r5, putw; iobuf

jsr r5, flush; iobuf

DESCRIPTION
fcreat creates the given file (mode 17) and sets up the buffer iobuf (size 134(10) bytes); putc and putw write a byte or word respectively onto the file; flush forces the contents of the buffer to be written, but does not close the file. The format of the buffer is:

iobuf: .=,+2             / file descriptor
      .=,+2             / characters unused in buffer
      .=,+2             / ptr to next free character
      .=,+128           / buffer

fcreat sets the error bit (c-bit) if the file creation failed; none of the other routines return error information.

Before terminating, a program should call flush to force cut the last of the output.

The user must supply iobuf, which should begin on a word boundary.

FILES
kept in ./etc/liba.a

SEE ALSO
syscreat; syswrite; getc, getw, fopen

DIAGNOSTICS
error bit possible on fcreat call

BUGS
buffers should be changed to 512 bytes.

OWNER
dmr
NAME     sin, cos -- sine cosine
SYNOPSIS jsr    r5, sin (cos)
DESCRIPTION The sine (cosine) of fr0 (radians) is returned in
    fr0. The floating point simulation should be
    active in either floating or double mode, but in
    single precision integer mode. All floating
    registers are used.
FILES    kept in /etc/liba.a
SEE ALSO fptrap
DIAGNOSTICS --
BUGS     Size of the argument should be checked to make
    sure the result is meaningful.
OWNER    ken, dmr
NAME

switch  --  switch on value

SYNOPSIS

(switch value in r0)
jsr    r5,switch; swtab
(not-found return)

... swtab: val1; lab1;
...     valn; labn
...;  0

DESCRIPTION

switch compares the value of r0 against each of the val_i; if a match is found, control is transferred to the corresponding lab_i (after popping the stack once). If no match has been found by the time a null lab_i occurs, switch returns.

FILES

kept in /etc/liba.a

SEE ALSO

--

DIAGNOSTICS

--

BUGS

--

OWNER

ken, dmr
NAME      mem -- core memory

SYNOPSIS  --

DESCRIPTION mem maps the core memory of the computer into a file. It may be used, for example, to examine, and even to patch the system using the debugger.

Mem is a byte-oriented file; its bytes are numbered 0 to 65,535.

FILES     --

SEE ALSO  --

DIAGNOSTICS --

BUGS      If a location not corresponding to implemented memory is read or written, the system will incur a bus-error trap and, in panic, will reboot itself.

OWNER     ken, dmr
NAME       ppt -- punched paper tape
SYNOPSIS   --
DESCRIPTION ppt refers to the paper tape reader or punch, depending on whether it is read or written.

When ppt is opened for writing, a 100-character leader is punched. Thereafter each byte written is punched on the tape. No editing of the characters is performed. When the file is closed, a 100-character trailer is punched.

When ppt is opened for reading, the process waits until tape is placed in the reader and the reader is on-line. Then requests to read cause the characters read to be passed back to the program, again without any editing. This means that several null characters will usually appear at the beginning of the file; they correspond to the tape leader. Likewise several nulls are likely to appear at the end. End-of-file is generated when the tape runs out.

Seek calls for this file are meaningless and are effectively ignored (however, the read/write pointers are maintained and an arbitrary sequence of reads or writes intermixed with seeks will give apparently correct results when checked with tell).

FILES      --
SEE ALSO   lppt, dbppt, bppt format
DIAGNOSTICS --
BUGS       Previously, there were separate special files for ASCII tape (which caused null characters to be suppressed) and binary tape (which used a blocked format with checksums). These notions were conceptually quite attractive, but they were discarded to save space in the system.

OWNER      ken, dmr
NAME       rf0 -- RF11-RS11 fixed-head disk file

SYNOPSIS    --

DESCRIPTION This file refers to the entire RF disk. It may be either read or written, although writing is inherently very dangerous, since a file system resides there.

The disk contains 1024 256-word blocks, numbered 0 to 1023. Like the other block-structured devices (tape, RK disk) this file is addressed in blocks, not bytes. This has two consequences: seek calls refer to block numbers, not byte numbers; and sequential reading or writing always advance the read or write pointer by at least one block. Thus successive reads of 10 characters from this file actually read the first 10 characters from successive blocks.

FILES        --

SEE ALSO     /dev/tap0, /dev/rk0

DIAGNOSTICS --

BUGS         The fact that this device is addressed in terms of blocks, not bytes, is extremely unfortunate. It is due entirely to the fact that read and write pointers (and consequently the arguments to seek and tell) are single-precision numbers. This really has to be changed but unfortunately the repercussions are serious.

OWNER        ken, dmr
NAME       rk0 -- RK03 (or RK05) disk
SYNOPSIS   --
DESCRIPTION rk0 refers to the entire RK03 disk as a single sequentially-addressed file. Its 256-word blocks are numbered 0 to 4871. Like the RF disk and the tape files, its addressing is block-oriented. Consult the /dev/rf0 section.
FILES      --
SEE ALSO   /dev/rf0, /dev/tap0
DIAGNOSTICS --
BUGS       See /dev/rf0
OWNER      ken, dmr
NAME

tap0 ... tap7

SYNOPSIS

--

DESCRIPTION

These files refer to DECTape drives 0 to 7. Since the logical drive number can be manually set, all eight files exist even though at present there are only two physical drives.

The 256-word blocks on a standard DECTape are numbered 0 to 577. However, the system makes no assumption about this number; a block can be read or written if it exists on the tape and not otherwise. An error is returned if a transaction is attempted for a block which does not exist.

Like the RK and RF special files, addressing on the tape files is block-oriented. See the RF0 section.

FILES

--

SEE ALSO

/dev/ra0, /dev/ra0

DIAGNOSTICS

--

BUGS

see /dev/ra0

OWNER

ken, dmr
NAME tty -- console typewriter

SYNOPSIS --

DESCRIPTION tty (as distinct from tty0, ..., tty5) refers to the console typewriter hard-wired to the PDP-11. Most of the time it is turned off and so has little general use.

Generally, the disciplines involved in dealing with tty are similar to those for tty0 ... and the appropriate section should be consulted. The following differences are salient:

The system calls stty and gttty do not apply to this device. It cannot be placed in raw mode; on input, upper case letters are always mapped into lower case letters; a carriage return is echoed when a line-feed is typed.

The quit character is not FS (as with tty0...) but is generated by the key labelled "alt mode."

By appropriate console switch settings, it is possible to cause UNIX to come up as a single-user system with I/O on this device.

FILES --

SEE ALSO /dev/tty0...; init

DIAGNOSTICS --

BUGS --

OWNER ken, dmr
NAME

tty0 ... tty5 -- communications interfaces

SYNOPSIS

--

DESCRIPTION

These files refer to DC11 asynchronous communications interfaces. At the moment there are six of them, but the number is subject to change. Names for up to four others will be constructed by an obvious algorithm.

When one of these files is opened, it causes the process to wait until a connection is established. (In practice, however, user's programs seldom open these files; they are opened by init and become a user's standard input and output file.) The very first typewriter file open in a process becomes the control typewriter for that process. The control typewriter plays a special role in the handling quit or interrupt signals, as discussed below. The control typewriter is inherited by a child process during a fork.

A terminal associated with one of these files ordinarily operates in full-duplex mode. Characters may be typed at any time, even while output is occurring, and are only lost when the system's character input buffers become completely choked, which is very rare.

When first opened, the interface expects the terminal to use 15 odd-parity, 10-bit ASCII characters per second and to have the new-line function. Finally, the system calculates delays after sending the code for certain functions (e.g., new-line, tab) on the assumption that the terminal is a Teletype model 37. All this is merely a long way of saying that the system expects to be used by a TTY 37. However, most of these assumptions can be changed by a special system call: in particular, the expected parity can be changed; the speed, character size, and stop bits can be changed (speeds available are 134.5, 150, 300, 1200 baud; see the DC11 manual); the new-line function can be simulated by a combination of the carriage-return and line-feed functions; carriage return can be translated into new-line on input; upper case letters can be mapped into lower case letters; echoing can be turned off so the terminal operates in half duplex. See the system call stty. (Also see init for the way 300-baud terminals are detected.)

Normally, a typewriter operates in units of lines. This means that a program attempting to read will be suspended until an entire line has been typed. Also, no matter how many characters
are requested in the read call, at most one line will be returned. It is not however necessary to read a whole line at once; any number of characters may be requested in a read, even one, without losing information.

The EOT character may be used to generate an end of file from a typewriter. When an EOT is received, all the characters waiting to be read are immediately passed to the program, without waiting for a new-line. Thus if there are no characters waiting, which is to say the EOT occurred at the beginning of a line, zero characters will be passed back, and this is the standard end-of-file signal.

When the carrier signal from the dataset drops (usually because the user has hung up his terminal) any read returns with an end-of-file indication. Thus programs which read a typewriter and are sensitive to end-of-file on their inputs (which all programs which should be) will terminate appropriately when hung up on.

Two characters have a special meaning when typed. The ASCII DEL character (sometimes called "rub-out") is the interrupt signal. When this character is received from a given typewriter, a search is made for all processes which have this typewriter as their control typewriter, and which have not informed the system that they wish to ignore interrupts. If there is more than one such process, one of these is selected, for practical purposes at random. Then either the process is forced to exit or a trap is simulated to an agreed-upon location in the process. See sys intr for more information.

The ASCII character FS is the quit signal. Its treatment is identical to the interrupt signal except that unless the receiving process has made other arrangements it will not only be terminated but a core image file will be written. (See sys quit for more information.)

During input, erase and kill processing is normally done. The character "#" erases the last character typed, except that it will not erase beyond the beginning of a line or an EOF. The character "@" kills the entire line up to the point where it was typed, but not beyond an EOF. Both these characters operate on a keystroke basis independently of any backspacing or tabbing that may have been done. Either "@" or "#" may be entered literally by preceding it by " "; the erase or kill character remains, but the
disappears.

It is also possible (again by `sys stty`) to put the typewriter into raw mode. In this mode, the program reading is wakened on each character, and when a program reads, it waits only until at least one character has been typed. In raw mode, no erase or kill processing is done; and the EOT, quit and interrupt characters are not treated specially.

Output is prosaic compared to input. It should be noted, however, that when one or more characters are written, they are actually transmitted to the terminal as soon as previously-written characters have finished typing. When a program produces characters too rapidly to be typed, as is very common, it may be suspended for a time.

Odd parity is always generated on output, except that the characters EOT and NAK have the wrong parity. Thus the 37 TTY will not hang up (EOT) or lock its keyboard (NAK) if a program accidentally prints these characters.

FILES

SEE ALSO tty

DIAGNOSTICS

BUGS

As has been suggested, UNIX has a heavy predisposition towards 37 Teletype terminals. However, it is quite possible to use 300-baud terminals such as the GE TermiNet 300. (See `init` for the procedure.) The main difficulty in practice is 37-oriented delay calculations.

Terminals such as the IBM 2741 would theoretically be very desirable but there are many difficulties related to its inadequate and non-ASCII character sets (the 2741 has two, count 'em) and the inherently half-duplex nature of the terminal. It is possible to produce output on a 2741; `cf type`.

OWNER

ken, dmr
NAME

a.out -- assembler and link editor output

SYNOPSIS

--

DESCRIPTION

a.out is the output file of the assembler as and the link editor ld. In both cases, a.out is executable provided there were no errors and no unresolved external references.

This file has four sections: a header, the program text, a symbol table, and relocation bits. The last two may be empty if the program was loaded with the "-s" option of ld or if the symbols and relocation have been removed by strip.

The header always contains 6 words:

1  a "br .+14" instruction (205(8))
2  The size of the program text
3  The size of the symbol table
4  The size of the relocation bits area
5  The size of a data area
6  A zero word (unused at present)

The sizes of the program, symbol table, and relocation area are in bytes but are always even. The branch instruction serves both to identify the file and to jump to the text entry point. The program text size includes the 6-word header.

The data area is used when the file is executed; the exec system call sets the program break to the sum of the text size and this data size. The data area is generated by the assembler when the location counter "." lies beyond the last assembled data, for example when the program ends with one or more constructions of the form ".=+.n"; it is preserved by the loader for the last program in a load. (Routines other than the last have the appropriate number of 0 words inserted, since there is no other provision for zero-suppression in an a.out file.)

The symbol table consists of 6-word entries. The first four contain the ASCII name of the symbol, null-padded. (In fact, the assembler generates symbols of at most 7 bytes.) The next word is a flag indicating the type of symbol. The following values are possible:

00 undefined symbol
01 absolute symbol
02 register symbol
03 relocatable symbol
40 undefined global symbol
41 absolute global symbol
43 relocatable global symbol

An undefined global corresponds to a GMAP "symref" and an absolute or relocatable global to a "symdef" or absolute or relocatable value respectively. Values other than those given above may occur if the user has defined some of his own instructions.

The last word of a symbol table entry contains the value of the symbol. Its contents are not specified if the symbol is undefined.

If a.out contains no unresolved global references, header and text portions are exactly as they will appear in core when the file is executed. If the value of a word in the text portion involves a reference to an undefined global, the word is replaced by the offset in the symbol table of the appropriate symbol. (That is, possible offsets are 0, 12(10), 24(10), ....) Such a word will have appropriate relocation bits.

The relocation bits portion uses a variable-length encoding. There is a string of bits for each word in the text portion. The scheme has at least two bits for each word, plus possibly two more to extend the codes available; in either case the bits may be followed by a 16-bit string to represent an offset to an external symbol. The bits are packed together without regard to word boundaries. The last word is filled out with 0's on the right.

The possible relocation bit configurations are:

00  
word is absolute

01  
word is relocatable

10  
word is a relative reference to an undefined global symbol with no offset. Currently, the word contains the offset in the symbol table of the symbol. When the symbol becomes defined, say with value x, this location will contain x-.2, where "." is the location of the word.

1100xxxxxxxxxxxxx  
word is a relative reference to an external symbol with an offset. It is the same as the previous relocation type, except that the 16-bit offset is added in when the symbol
becomes defined.

**1101**

word is a reference to an undefined external symbol with no offset. At present the word contains the symbol table offset of the symbol. When the symbol becomes defined, the word will contain the value of the symbol.

**1110xxxxxxxxxxxxxxx**

word is a reference to an undefined external symbol with an offset. At present, the word contains the symbol table offset of the symbol. When the symbol becomes defined, the word will contain the value of the symbol plus the given 16-bit offset.

FILES
---

SEE ALSO
as ld, strip, nm, un

DIAGNOSTICS
---

BUGS
Soon, there will be a new type of symbol: the data area symbol. In the text, it will appear as an ordinary external reference. However, it need not be defined; this will be done by the loader. Watch this space for more details.

OWNER
dmr
NAME
archive (library) file format

SYNOPSIS
--

DESCRIPTION
The archive command `ar` is used to combine several files into one. Its use has three benefits: when files are combined, the file space consumed by the breakage at the end of each file (256 bytes on the average) is saved; directories are smaller and less confusing; archive files of object programs may be searched as libraries by the loader `ld`.

A file produced by `ar` has a "magic number" at the start, followed by the constituent files, each preceded by a file header. The magic number is -147(10), or 177555(8) (it was chosen to be unlikely to occur anywhere else). The header of each file is 16 bytes long:

0-7
  file name, null padded on the right

8-11
  Modification time of the file

12
  User ID of file owner

13
  file mode

14-15
  file size

If the file is an odd number of bytes long, it is padded with a null byte, but the size in the header is correct.

Notice there is no provision for empty areas in an archive file.

FILES
--

SEE ALSO
`ar`, `ld`

DIAGNOSTICS
--

BUGS
--

OWNER
ken, dmr
NAME

binary punched paper tape format

SYNOPSIS

--

DESCRIPTION

Binary paper tape is used to pass and store arbitrary information on paper tape. The format chosen has the following features: a) no format of the data is assumed. b) check summing c) zero suppression

The format is as follows:

Between records, NULL characters are ignored. The beginning of the tape is considered between records, thus the leader is ignored.

The first non-null character specifies the type and size of the record. If the character is positive (1 to 177), the record is a data record consisting of that many characters. All but the last of these characters are data, the last being a checksum. The checksum is calculated such that the sum of the entire record is zero mod 256.

If the first character is negative (200-376) the record is a zero suppression record. It is identical to minus that number of zeros of data. One character of checksum follows this negative character. It is the positive of the negative character.

The special case of a record looking like a single zero character suppressed (377;1) causes no data transfer, but is an end-of-file indication.

FILES

--

SEE ALSO

lbppt, dbppt

DIAGNOSTICS

--

BUGS

--

OWNER

ken, dmr
NAME
format of core image

SYNOPSIS
--

DESCRIPTION
Three conditions cause UNIX to write out the core image of an executing program: the program generates an unexpected trap (by a bus error or illegal instruction); the user sends a "quit" signal (which has not been turned off by the program); a trap is simulated by the floating point simulator. The core image is called "core" and is written in the current working directory (provided it can be; normal access controls apply). It is exactly 8192+64 bytes long. The first 8192 represent the actual contents of memory at the time of the fault; the last 64 are the contents of the system's per-user data area for this process. Only the first word of this area will be described.

When any trap which is not an I/O interrupt occurs, all the useful registers are stored on the stack. After all the registers have been stored, the contents of sp are placed in the first cell of the user area; this cell is called u.sp. Therefore, within the core image proper, there is an area which contains the following registers in the following order (increasing addresses):

(u.sp)->sc
  mq
  ac
  r5
  r4
  r3
  r2
  r1
  r0
  pc (at time of fault)
  processor status (at time of fault)

The last two are stored by the hardware. It follows that the contents of sp at the time of the fault were (u.sp) plus 22(10).

The t-bit (trap bit) in the stored status will be on when a quit caused the generation of the core image, since this bit is used in the implementation of quits.

FILES
--

SEE ALSO
--

DIAGNOSTICS
--
11/3/71

BUGS

OWNER  ken, dmr
NAME

format of directories

SYNOPSIS

--

DESCRIPTION

A directory behaves exactly like an ordinary file, save that no user may write into a directory. The fact that a file is a directory is indicated by a bit in the flag word of its i-node entry.

Directory entries are 10 bytes long. The first word is the i-node of the file represented by the entry, if non-zero; if zero, the entry is empty.

Bytes 2-9 represent the (8-character) file name, null padded on the right. These bytes are not necessarily cleared for empty slots.

By convention, the first two entries in each directory are for "." and "..". The first is an entry for the directory itself. The second is for the parent directory. The meaning of ".." is modified for the root directory of the master file system and for the root directories of removable file systems. In the first case, there is no parent, and in the second, the system does not permit off-device references without a mount system call. Therefore in both cases ".." has the same meaning as ".."

FILES

--

SEE ALSO

file system format

DIAGNOSTICS

--

BUGS

--

OWNER

ken, dmr
format of file system

Every file system storage volume (e.g. RF disk, RK disk, DECTape reel) has a common format for certain vital information.

Every such volume is divided into a certain number of 256 word (512 byte) blocks. Blocks 0 and 1 are collectively known as the super-block for the device; they define its extent and contain an i-node map and a free-storage map. The first word contains the number of bytes in the free-storage map; it is always even. It is followed by the map. There is one bit for each block on the device; the bit is "1" if the block is free. Thus if the number of free-map bytes is n, the blocks on the device are numbered 0 through 8n-1. The free-map count is followed by the free map itself. The bit for block k of the device is in byte k/8 of the map; it is offset k(mod 8) bits from the right. Notice that bits exist for the superblock and the i-list, even though they are never allocated or freed.

After the free map is a word containing the byte count for the i-node map. It too is always even. I-numbers below 41(10) are reserved for special files, and are never allocated; the first bit in the i-node free map refers to i-number 41. Therefore the byte number in the i-node map for i-node i is (i-41)/8. It is offset (i-41)(mod 8) bits from the right; unlike the free map, a "0" bit indicates an available i-node.

I-numbers begin at 1, and the storage for i-nodes begins at block 2. Also, i-nodes are 32 bytes long, so 16 of them fit into a block. Therefore, i-node i is located in block (i+31)/16 of the file system, and begins 32*(i+31)(mod 16) bytes from its start.

There is always one file system which is always mounted; in standard UNIX it resides on the RF disk. This device is also used for swapping. The swap areas are at the high addresses on the device. It would be convenient if these addresses did not appear in the free list, but in fact this is not so. Therefore a certain number of blocks at the top of the device appear in the free map, are not marked free, yet do not appear within any file. These are the blocks that show up "missing" in a check of the RF disk.

Again on the primary file system device, there
are several pieces of information following that previously discussed. They contain basically the information typed by the `um` command; namely, the times spent since a cold boot in various categories, and a count of I/O errors. In particular, there are two words with the calendar time (measured since 00:00 Jan 1, 1971); two words with the time spent executing in the system; two words with the time spent waiting for I/O on the RF and RK disks; two words with the time spent executing in a user’s core; one byte with the count of errors on the RF disk; and one byte with the count of errors on the RK disk. All the times are measured in sixtieths of a second.

I-node 41(10) is reserved for the root directory of the file system. No i-numbers other than this one and those from 1 to 40 (which represent special files) have a built-in meaning. Each i-node represents one file. The format of an i-node is as follows, where the left column represents the offset from the beginning of the i-node:

<table>
<thead>
<tr>
<th>Offset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-1</td>
<td>flags (see below)</td>
</tr>
<tr>
<td>2</td>
<td>number of links</td>
</tr>
<tr>
<td>3</td>
<td>user ID of owner</td>
</tr>
<tr>
<td>4-5</td>
<td>size in bytes</td>
</tr>
<tr>
<td>6-7</td>
<td>first indirect block or contents block</td>
</tr>
<tr>
<td>20-21</td>
<td>eighth indirect block or contents block</td>
</tr>
<tr>
<td>22-25</td>
<td>creation time</td>
</tr>
<tr>
<td>26-29</td>
<td>modification time</td>
</tr>
<tr>
<td>30-31</td>
<td>unused</td>
</tr>
</tbody>
</table>

The flags are as follows:

<table>
<thead>
<tr>
<th>Flag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000000</td>
<td>i-node is allocated</td>
</tr>
<tr>
<td>0400000</td>
<td>directory</td>
</tr>
<tr>
<td>0200000</td>
<td>file has been modified (always on)</td>
</tr>
<tr>
<td>0100000</td>
<td>large file</td>
</tr>
<tr>
<td>000040</td>
<td>set user ID on execution</td>
</tr>
<tr>
<td>000020</td>
<td>executable</td>
</tr>
<tr>
<td>000010</td>
<td>read, owner</td>
</tr>
<tr>
<td>000004</td>
<td>write, owner</td>
</tr>
<tr>
<td>000002</td>
<td>read, non-owner</td>
</tr>
<tr>
<td>000001</td>
<td>write, non-owner</td>
</tr>
</tbody>
</table>

The allocated bit (flag 100000) is believed even if the i-node map says the i-node is free; thus corruption of the map may cause i-nodes to become unallocatable, but will not cause active nodes to be reused.

Byte number $n$ of a file is accessed as follows: $n$ is divided by 512 to find its logical block number (say $b$) in the file. If the file is small
(flag 010000 is 0), then \( b \) must be less than \( 8 \), and the physical block number corresponding to \( b \) is the \( b \)th entry in the address portion of the i-node.

If the file is large, \( b \) is divided by 256 to yield a number which must be less than 8 (or the file is too large for UNIX to handle). The corresponding slot in the i-node address portion gives the physical block number of an indirect block. The residue mod 256 of \( b \) is multiplied by two (to give a byte offset in the indirect block) and the word found there is the physical address of the block corresponding to \( b \).

If block \( b \) in a file exists, it is not necessary that all blocks less than \( b \) exist. A zero block number either in the address words of the i-node or in an indirect block indicates that the corresponding block has never been allocated. Such a missing block reads as if it contained all zero words.

**FILES**

**SEE ALSO** format of directories

**DIAGNOSTICS**

**BUGS**

Two blocks are not enough to handle the i- and free-storage maps for an R02 disk pack, which contains around 10 million words.

**OWNER**

\[
\frac{N}{512} = \frac{n}{siz} \leq b \leq 5 \times 256 \frac{8 \times 2^{31}}{2^{18}} = 2^{26} \text{ bytes}
\]

\[
\frac{n}{siz} = \text{block actually limited by offset in kbytes} \leq 2^{16} \text{ bytes}
\]
NAME passwd -- password file

SYNOPSIS --

DESCRIPTION passwd contains for each user the following information:

  name (login name)
  password
  numerical user ID
  default working directory
  program to use as Shell

This is an ASCII file. Each field within each user's entry is separated from the next by a colon. Each user is separated from the next by a new-line. If the password field is null, no password is demanded; if the Shell field is null, the Shell itself is used.

This file, naturally, is inaccessible to anyone but the super-user.

This file resides in directory /etc.

FILES --

SEE ALSO /etc/init

DIAGNOSTICS --

BUGS --

OWNER super-user
NAME /etc/uids -- map user names to user IDs

SYNOPSIS --

DESCRIPTION
This file allows programs to map user names into user numbers and vice versa. Anyone can read it. It resides in directory /etc, and should be updated along with the password file when a user is added or deleted.

The format is an ASCII name, followed by a colon, followed by a decimal ASCII user ID number.

FILES --

SEE ALSO --

DIAGNOSTICS --

BUGS --

OWNER dmr, ken
NAME   /tmp/utmp -- user information

SYNOPSIS --

DESCRIPTION This file allows one to discover information about who is currently using UNIX. The file is binary; each entry is 16(10) bytes long. The first eight bytes contain a user's login name or are null if the table slot is unused. The low order byte of the next word contains the last character of a typewriter name (currently, '0' to '5' for /dev/tty0 to /dev/tty5). The next two words contain the user's login time. The last word is unused.

This file resides in directory /tmp.

FILES --

SEE ALSO /etc/init, which maintains the file.

DIAGNOSTICS --

BUGS --

OWNER ken, dmr
NAME basic -- DEC supplied BASIC

SYNOPSIS basic [file]

DESCRIPTION Basic is the standard BASIC V000 distributed as a stand alone program. The optional file argument is read before the console. See DEC-11-AJPB-D manual.

Since bas is smaller and faster, basic is not maintained on line.

FILES --

SEE ALSO bas

DIAGNOSTICS See manual

BUGS GOK

OWNER dmr
NAME
bj -- the game of black jack

SYNOPSIS
/usr/games/bj

DESCRIPTION
Black jack is a serious attempt at simulating the dealer in the game of black jack (or twenty-one) as might be found in Reno.

The following rules apply:

The bet is $2 every hand.

A player 'natural' (black jack) pays $3. A dealer natural loses $2. Both dealer and player naturals is a 'push' (no money exchange).

If the dealer has an ace up, the player is allowed to make an 'insurance' bet against the chance of a dealer natural. If this bet is not taken, play resumes as normal. If the bet is taken, it is a side bet where the player wins $2 if the dealer has a natural and loses $1 if the dealer does not.

If the player is dealt two cards of the same value, he is allowed to 'double'. He is allowed to play two hands, each with one of these cards. (The bet is doubled also; $2 on each hand.)

If a dealt hand has a total of ten or eleven, the player may 'double down'. He may double the bet ($2 to $4) and receive exactly one more card on that hand.

Under normal play, the player may 'hit' (draw a card) as long as his total is not over twenty-one. If the player 'busts' (goes over twenty-one), the dealer wins the bet.

When the player 'stands' (decides not to hit), the dealer hits until he attains a total of seventeen or more. If the dealer busts, the player wins the bet.

If both player and dealer stand, the one with the largest total wins. A tie is a push.

The machine deals and keeps score. The following questions will be asked at appropriate times. Each question is answered by y followed by a new line for 'yes', or just new line for 'no'.

? means 'do you want a hit?'

Insurance?
Double down?

Every time the deck is shuffled, the dealer so states and the 'action' (total bet) and 'standing' (total won or loss) is printed. To exit, hit the interrupt key (DEL) and the action and standing will be printed.

FILES
SEE ALSO
DIAGNOSTICS
BUGS
OWNER ken
NAME       cal -- print calendar
SYNOPSIS   /usr/ken/cal year
DESCRIPTION Cal will print a calendar for the given year. The year can be between 0 (really 1 BC) and 9999. For years when several calendars were in vogue in different countries, the calendar of England (and therefore her colonies) is printed.

P.S. try cal of 1752.

FILES      --
SEE ALSO    --
DIAGNOSTICS --
BUGS        --
OWNER       ken
NAME  
chess -- the game of chess

SYNOPSIS  
/usr/games/chess

DESCRIPTION
Chess is an attempt at computer chess. The program 'speaks' in algebraic chess notation. The initial board configuration in this notation is as follows:

8 R N B Q K B N R
7 P P P P P P P P
6 * * * * * * *
5 * * * * * * *
4 * * * * * * *
3 * * * * * * *
2 p p p p p p p p
1 r n b q k b n r
a b c d e f g h

A move is specified by the 'from' co-ordinate followed by the 'to' co-ordinate. Thus the white P-K4 move would be 'e2e4'. The black P-K4 would be 'e7e5'.

The following commands are recognized by the chess program:

move
Make the move if legal. The program does not keep track of who is to play. The move is made for whatever side is specified.

move x
Make the move regardless of legality. This is a good way to either set up a desired situation or to cheat. The initial move 'e2e8x' is a winner.

mw
The program will compute and make a move for the white pieces.

m
The program will compute and make a move for the black pieces.

lab
Set the level parameters to a and b, where a and b are numbers between 0 and 9. The initial settings are 2 and 8. The first parameter increases computation time rapidly while the second parameter only increases computation exponentially. Currently move times run from 20 seconds to 10 minutes. It was hoped that these numbers would be usefully related to the program's competence.
The board is printed.

The last move is un-made. This is another good way to cheat.

All the moves to date are printed.

The current game situation is saved on the file c.tmp.

The game situation on the file c.tmp is restored.

!command
    The unix command is executed by the mini-shell.

An interrupt (DEL) will pull the program out of its computation. If it is trying to make a move, the best move to date is made.

FILES
    c.tmp

SEE ALSO
    msh

DIAGNOSTICS
    ? if an illegal move is attempted, or if an unknown command is typed.

BUGS
    The current version does not recognize castling, promotion and en passant. A new version is in the mill.

OWNER
    ken
<table>
<thead>
<tr>
<th>NAME</th>
<th>das -- disassembler</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYNOPSIS</td>
<td>--</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>A PDP-11 disassembler exists. Contact the author for more information.</td>
</tr>
<tr>
<td>FILES</td>
<td>--</td>
</tr>
<tr>
<td>SEE ALSO</td>
<td>--</td>
</tr>
<tr>
<td>DIAGNOSTICS</td>
<td>--</td>
</tr>
<tr>
<td>BUGS</td>
<td>--</td>
</tr>
<tr>
<td>OWNER</td>
<td>ken</td>
</tr>
</tbody>
</table>
NAME  dli -- load DEC binary paper tapes
SYNOPSIS  dli output [input]
DESCRIPTION  dli will load a DEC binary paper tape into the output file. The binary format paper tape is read from the input file (/dev/ppt is default.)
FILES  /dev/ppt
SEE ALSO  --
DIAGNOSTICS  "checksum"
BUGS  --
OWNER  dmr
NAME     dpt -- read DEC ASCII paper tape
SYNOPSIS dpt output [input]
DESCRIPTION  dpt reads the input file (/dev/ppt default) assuming the format is a DEC generated ASCII paper tape of an assembly language program. The output is a UNIX ASCII assembly program.
FILES       /dev/ppt
SEE ALSO    --
DIAGNOSTICS --
BUGS        Almost always a hand pass is required to get a correct output.
OWNER       ken, dmr
| NAME            | moo -- a game          |
| SYNOPSIS       | /usr/games/moo         |
| DESCRIPTION    | moo is a guessing game imported from England. |
| FILES          | --                     |
| SEE ALSO       | --                     |
| DIAGNOSTICS    | --                     |
| BUGS           | --                     |
| OWNER          | ken                    |
NAME    sort -- sort a file
SYNOPSIS sort input output
DESCRIPTION sort will sort the input file and write the sorted file on the output file. Wide options are available on collating sequence and ignored characters.
FILES    --
SEE ALSO  --
DIAGNOSTICS --
BUGS     --
OWNER    dmr, ken
NAME       ttt -- tic-tac-toe
SYNOPSIS   /usr/games/ttt
DESCRIPTION ttt is the X's and O's game popular in 1st grade. This is a learning program that never makes the same mistake twice.
FILES      ttt.k -- old mistakes
SEE ALSO   --
DIAGNOSTICS --
BUGS       --
OWNER      ken
NAME       as2 -- assembler pass 2
SYNOPSIS   --
DESCRIPTION as2 is invoked by the assembler as to perform its second pass.
FILES      see as
SEE ALSO    as
DIAGNOSTICS see as
BUGS       --
OWNER      dmr
NAME ascii -- map of ASCII character set

SYNOPSIS cat /etc/ascii

DESCRIPTION ascii is a map of the ASCII character set, to be printed as needed. It contains:

| 000 nul | 001 soh | 002 stx | 003 etx | 004 eot | 005 enq | 006 ack | 007 bel |
| 010 bs  | 011 ht  | 012 nl  | 013 vt  | 014 np  | 015 cr  | 016 so  | 017 si  |
| 020 dle | 021 dc1 | 022 dc2 | 023 dc3 | 024 dc4 | 025 nak | 026 syn | 027 etb |
| 030 can | 031 em  | 032 sub | 033 esc | 034 fs  | 035 gs  | 036 rs  | 037 us  |
| 040 sp  | 041 !  | 042 "  | 043 #  | 044 $  | 045 %  | 046 &  | 047 '  |
| 050 (   | 051 )  | 052 *  | 053 +  | 054 ,  | 055 -  | 056 .  | 057 /  |
| 060 0   | 061 1  | 062 2  | 063 3  | 064 4  | 065 5  | 066 6  | 067 7  |
| 070 8   | 071 9  | 072 :  | 073 ;  | 074 <  | 075 =  | 076 >  | 077 ?  |
| 100 @   | 101 A  | 102 B  | 103 C  | 104 D  | 105 E  | 106 F  | 107 G  |
| 110 H   | 111 I  | 112 J  | 113 K  | 114 L  | 115 M  | 116 N  | 117 O  |
| 120 P   | 121 Q  | 122 R  | 123 S  | 124 T  | 125 U  | 126 V  | 127 W  |
| 130 X   | 131 Y  | 132 Z  | 133 \  | 134 /  | 135 ]  | 136 ^  | 137 _  |
| 140 '   | 141 a  | 142 b  | 143 c  | 144 d  | 145 e  | 146 f  | 147 g  |
| 150 h   | 151 i  | 152 j  | 153 k  | 154 l  | 155 m  | 156 n  | 157 o  |
| 160 p   | 161 q  | 162 r  | 163 s  | 164 t  | 165 u  | 166 v  | 167 w  |
| 170 x   | 171 y  | 172 z  | 173 {  | 174 | 175 }  | 176 ~  | 177 del |

FILES

SEE ALSO

DIAGNOSTICS

BUGS

OWNER jfo
NAME      ba  --  B assembler

SYNOPSIS  /etc/ba name

DESCRIPTION ba is invoked by the B command in order to turn the B intermediate code into assembly language.

FILES     name.i (input), name.s (output)

SEE ALSO  b command, /etc/bc

DIAGNOSTICS --

BUGS      At the moment, the b command is defunct, and ba is invoked via a command file.

OWNER     ken
NAME
bc -- B compiler

SYNOPSIS
/etc/bc name.b name.i

DESCRIPTION
bc is the B compiler proper; it turns B source into intermediate code. It is invoked from the b command.

FILES
name.b (input), name.i (intermediate output)

SEE ALSO
b (command), /etc/ba

DIAGNOSTICS
--

BUGS
The b command is defunct at the moment; bc is called from a command file.

OWNER
ken
NAME  biblib -- B interpreter library

SYNOPSIS --

DESCRIPTION biblib is the library of B runtime operators. It is searched during the loading of a B-compiled program.

Standard B subroutines are contained in /etc/libb.a.

FILES --

SEE ALSO b (command); ar, ld

DIAGNOSTICS --

BUGS The following assignment binary operators are missing: b102 (|=), b103 (=&), b104 (==), b105 (=!=), b106 (=<), b107 (=<), b110 (=>), b111 (=>), b112 (=>), b113 (<<), b120 (=/).

OWNER ken, dmr
NAME                  bos, maki, rom, vcboot, msys, et al

SYNOPSIS              --

DESCRIPTION           On the RF disk, the highest 16K words are reserved for stand-alone programs. These 16K words are allocated as follows:

bos                   (1K)
Warm UNIX             (6K)
Cold UNIX              (6K)
unassigned             (3K)

The UNIX read only memory (ROM) is home cut with 2 programs of 16 words each. The first (address 173700) reads bos from the RF disk into core location 54000 and transfers to 54000. The other ROM program (address 173740) reads a DECtape sitting in the end-zone on drive 0 into core location 0 and transfers to 0. This latter operation is compatible with part of DEC's standard ROM. The disassembled code for the UNIX ROM follows:

173700:   mov  $177472,r0  12700;177472
          mov  $3,-(r0)    12740;3
          mov  $140000,-(r0) 12740;140000
          mov  $54000,-(r0)  12740;54000
          mov  $-2000,-(r0)  12740;176000
          mov  $5,-(r0)     12740;5
          tstb (r0)         105710
          bge   .-2         2376
          jmp   *$54000     137;54000

173740:   mov  $177350,r0  12700;177350
          clr   -(r0)      5040
          mov  r0,-(r0)    10040
          mov  $3,-(r0)    12740;3
          tstb (r0)       105710
          bge   .-2        2376
          tst   *$177350   5737;177350
          bne   .          1377
          movb $5,(r0)    112710;5
          tstb (r0)       105710
          bge   .-2        2376
          clr   pc         5007

The program bos (Bootstrap Operating System) examines the console switches and executes one of several internal programs depending on the setting. If no setting is recognizable, bos loops waiting for a recognizable setting. The following settings are currently recognized:

173700 73700 Will read Warm UNIX from the RF into core location 0 and transfer to 400.
1 Will read Cold UNIX from the RF into core location 0 and transfer to 400.

2 Will read the unassigned 3K program into core location 0 and transfer to 400.

10 Will dump 12K words of memory from core location 0 onto DECTape drive 7.

0 Will load a standard UNIX binary paper tape into core location 0 and transfer to 0.

57500 Will load the standard DEC absolute and binary loaders and transfer to 57500.

Thus we come to the UNIX warm boot procedure: put 173700 into the switches, push load address and then push start. The alternate switch setting of 73700 that will load warm UNIX is used as a signal to bring up a single user system for special purposes. See /etc/init.

Cold boots can be accomplished with the Cold UNIX program, but they’re not. Thus the Cold UNIX slot on the RF may have any program desired. This slot is, however, used during a cold boot. Mount the UNIX INIT DECTape on drive 0 positioned in the end-zone. Put 173740 into the switches. Push load address. Put 1 into the switches. Push start. This reads a program called vcboot from the tape into core location 0 and transfers to it. vcboot then reads 16K words from the DECTape (blocks 1–32) and copies the data to the highest 16K words of the RF. Thus this initializes the read-only part of the RF. vcboot then reads in bos and executes it. bos then reads in Cold UNIX and executes that. Cold UNIX halts for a last chance before it completely initializes the RF file system. Push continue, and Cold UNIX will initialize the RF. It then sets into execution a user program that reads the DECTape for initialization files starting from block 33. When this is done, the program executes /etc/init which should have been on the tape.

The INIT tape is made by the program maki running under UNIX. maki writes vcboot on block 0 of /dev/tape. It then copies the RF 16K words (using /dev/rf0) onto blocks 1 thru 32. It has internally a list of files to be copied from block 33 on. This list follows:

/etc/init
/bin/chmod
/bin/chown
/bin/cp
/bin/ln
/bin/ls
/bin/mkdir
/bin/mv
/bin/rm
/bin/rmdir
/bin/sh
/bin/stat
/bin/tap

Thus this is the set of programs available after a cold boot. /etc/init and /bin/sh are mandatory. /bin/tap and /bin/mkdir are used to load up the file system. The rest of the programs are frosting. As soon as possible, an sdate should be done.

The last link in this incestuous daisy chain is the program msys.

msys char file

will copy the file file onto the RF read only slot specified by the character char. Char is taken from the following set:

b bos
u Warm UNIX
1 Cold UNIX
2 unassigned

Due to their rarity of use, maki and msys are maintained off line and must be reassembled before used.

FILES
/dev/rf0, /dev/tapn

SEE ALSO /etc/init, /bin/tap, /bin/sh, /bin/mkdir, bppt format

DIAGNOSTICS
--

BUGS The files /bin/mount, /bin/sdate, and /bin/date should be included in the initialization list of maki.

OWNER ken
NAME       brt1, brt2 -- B runtime routines

SYNOPSIS   --

DESCRIPTION The first of these routines must be loaded first in an executable B program; the second must be loaded last, after all other routines. They are not in /etc/bilib only because having them separate is the easiest way to assure the order of loading.

FILES      --

SEE ALSO   b command, bilib

DIAGNOSTICS --

BUGS       --

OWNER      ken
NAME       f1, f2, f3, f4 -- Fortran compiler
SYNOPSIS   --
DESCRIPTION These programs represent the four phases of a Fortran compilation:

    f1: specification statements
    f2: common and equivalence allocation
    f3: executable statements
    f4: cleanup

Each exec's the next; the first is called by the for command.

FILES      f.tmp1, f.tmp2, f.tmp3
SEE ALSO    for
DIAGNOSTICS --
BUGS        Besides the fact that there is a good deal of the Fortran language missing, there is no for command; Fortran is invoked via a command file.

OWNER       ken, dmr
NAME    glob -- global

SYNOPSIS --

DESCRIPTION glob is used to expand arguments to the shell containing "*" or "?". It is passed the argument list containing the metacharacters; glob expands the list and calls the command itself.

FILES --

SEE ALSO sh

DIAGNOSTICS "No match", "no command"

BUGS glob will only load a command from /bin. Also if any "*" or "?" argument fails to generate matches, "No match" is typed and the command is not executed.

OWNER dmr
NAME

init -- process initialization

SYNOPSIS

DESCRIPTION

init is invoked inside UNIX as the last step in the boot procedure. It first carries out several housekeeping duties: it must change the modes of the tape files and the RK disk file to 17, because if the system crashed while a tap or rk command was in progress, these files would be inaccessible; it also truncates the file /tmp/utmp, which contains a list of UNIX users, again as a recovery measure in case of a crash. Directory usr is assigned via sys mount as resident on the RK disk.

init then forks several times so as to create one process for each typewriter channel on which a user may log in. Each process changes the mode of its typewriter to 15 (read/write owner, write-only non-owner; this guards against random users stealing input) and the owner to the super-user. Then the typewriter is opened for reading and writing. Since these opens are for the first files open in the process, they receive the file descriptors 0 and 1, the standard input and output file descriptors. It is likely that no one is dialled in when the read open takes place; therefore the process waits until someone calls. At this point, init types its "login:" message and reads the response, which is looked up in the password file. The password file contains each user's name, password, numerical user ID, default working directory, and default shell. If the lookup is successful and the user can supply his password, the owner of the typewriter is changed to the appropriate user ID. An entry is made in /tmp/utmp for this user to maintain an up-to-date list of users. Then the user ID of the process is changed appropriately, the current directory is set, and the appropriate program to be used as the Shell is executed.

At some point the process will terminate, either because the login was successful but the user has now logged out, or because the login was unsuccessful. The parent routine of all the children of init has meanwhile been waiting for such an event. When return takes place from the sys wait, init simply forks again, and the child process again awaits a user.

There is a fine point involved in reading the login message. UNIX is presently set up to handle automatically two types of terminals: 150 baud, full duplex terminals with the line-feed
function (typically, the Model 37 Teletype terminal), and 300 baud, full duplex terminals with only the line-space function (typically the GE TerminiNet terminal). The latter type identifies itself by sending a line-break (long space) signal at login time. Therefore, if a null character is received during reading of the login line, the typewriter mode is set to accommodate this terminal and the "login:" message is typed again (because it was garbled the first time).

Init, upon first entry, checks the switches for 73700. If this combination is set, init will open /dev/tty as standard input and output and directly execute /bin/sh. In this manner, UNIX can be brought up with a minimum of hardware and software.

FILES
/tmp/utmp, /dev/tty0 ... /dev/ttyn

SEE ALSO
sh

DIAGNOSTICS
"No directory", "No shell". There are also some halts if basic I/O files cannot be found in /dev.

BUGS

OWNER
ken, dmr
NAME kbd -- keyboard map

SYNOPSIS cat /etc/kbd

DESCRIPTION kbd contains a map to the keyboard for model 37 Teletype terminals with the extended character set feature. If kbd is printed on such a terminal, the following will appear:

<[1234567890-_"\>qwertyuiop asdfghjkl;: zxcvbnm,./
<\1234567890--\3\5\7> y ;: ,./
<{!"#$%&'()=_,-}~!>QWERTYUIOP ASDFGHJKL+* ZXCVBNM,.?
< !"#$%&'()=_,- >ξΔΛΠΘΨΩΠ ΑΕΦΨΠΡΛ+* ΟΨΨΒΗΜ,.?

FILES --

SEE ALSO --

DIAGNOSTICS --

BUGS --

OWNER jfo
NAME liba.a -- assembly language library

SYNOPSIS --

DESCRIPTION This library is the standard location for assembly-language subroutines of general use. A section of this manual is devoted to its contents.

This library is searched when the link editor ld encounters the "-l" argument.

FILES --

SEE ALSO ld; library manual

DIAGNOSTICS --

BUGS --

OWNER dmr, ken
NAME        libb.a -- B library

SYNOPSIS    --

DESCRIPTION This library contains all B-callable subroutines of general utility. Its contents are detailed in the library section of the B manual. At present its contents are:

    char
    getchr
    putchr
    exit
    printf
    seek
    setuid
    stat
    time
    unlink
    wait
    lchar
    chdir
    chmod
    chown
    close
    creat
    execl
    execv
    fork
    fstat
    getuid
    intr
    link
    mkdir
    open
    read
    write
    ctime

FILES        --

SEE ALSO     b

DIAGNOSTICS --

BUGS         --

OWNER        ken, dmr
NAME  /etc/libf.a  -- Fortran library
SYNOPSIS  --
DESCRIPTION  This library contains all the Fortran runtime routines. Many are missing.
FILES  --
SEE ALSO  f1, f2, f3, f4
DIAGNOSTICS  --
BUGS  Will be renamed, and libf.a reserved for subroutines and functions.
OWNER  ken, dmr
NAMElogging in and logging out

SYNOPSIS--

DESCRIPTIONUNIX must be called from an appropriate terminal. The two general classes of terminals which UNIX supports are typified by the 37 Teletype on the one hand and the GE TermiNet 300 and Memorex 1240 on the other. The principal difference is the baud rate (150 vs. 300) and the treatment of the carriage return character. Most terminals operating at 150, 300, or 1200 baud using the ASCII character set either work (more or less) at the moment or can be used by special arrangement. In particular, special arrangement is necessary for terminals which do not generate lower-case ASCII characters.

It is also necessary to have a valid UNIX user ID and (if desired) password. These may be obtained, together with the telephone number, from the system administrators.

The same telephone number serves terminals operating at both the standard speeds. When a connection is established via a 150-baud terminal (e.g. TTY 37) UNIX types out "login:" you respond with your user name, and, if a mask is typed, with a password. If the login was successful, the "$" character is typed by the Shell to indicate login is complete and commands may be issued. A message of the day may be typed if there are any announcements. Also, if there is a file called "mailbox", you are notified that someone has sent you mail. (See the mail command.)

From a 300-baud terminal, the procedure is slightly different. Such terminals often have a full-duplex switch, which should be turned on (or conversely, half-duplex should be turned off). When a connection with UNIX is established, a few garbage characters are typed (these are the "login:" message at the wrong speed). You should depress the "break" key; this is a speed-independent signal to UNIX that a 300-baud terminal is in use. It will type "login:" (at the correct speed this time) and from then on the procedure is the same as described above.

Logging out is simple by comparison (in fact, sometimes too simple). Simply generate an end-of-file at Shell level by using the EOT character; the "login:" message will appear again to indicate that you may log in again.
It is also possible to log out simply by hanging up the terminal; this simulates an end-of-file on the typewriter.

FILES
--

SEE ALSO
init

DIAGNOSTICS
--

BUGS
Hanging up on programs which never read the typewriter or which ignore end-of-files is very dangerous; in the worst cases, the programs can only be halted by restarting the system.

OWNER
ken, dmr
NAME  msh -- mini-shell

SYNOPSIS  --

DESCRIPTION  msh is a heavily simplified version of the Shell. It reads one line from the standard input file, interprets it as a command, and calls the command.

The mini-shell supports few of the advanced features of the Shell; none of the following characters is special:

   > $ \ ; &

However, "*" and "?" are recognized and glob is called. The main use of msh is to provide a command-executing facility for various interactive sub-systems.

FILES  --

SEE ALSO  sh, glob

DIAGNOSTICS  "?"

BUGS  --

OWNER  ken, dmr
NAME   suftab  --  suffix table

SYNOPSIS  --

DESCRIPTION  suftab is a table of suffixes used to guide
hyphenation in roff. Its first 12 words are not
used (see a.out Format.) Its next 26 words point
to the beginning of the subtabels for each of the
26 initial letters of a suffix. The first entry
for each suffix is a count of the number of bytes
in the suffix. The second byte of each entry is
a flag indicating the type of suffix. The suffix
itself follows; the high bits of each letter
indicate where the hyphens come. The table for
each initial suffix letter ends with a zero count
byte.

FILES  --

SEE ALSO  roff

DIAGNOSTICS  --

BUGS  --

OWNER  jfo, dmr, ken
NAME
tabs -- tab stop set

SYNOPSIS

cat /etc/tabs

DESCRIPTION

When printed on a suitable terminal, this file will set tab stops at columns 8, 16, 24, 32, .... Suitable terminals include the Teletype model 37 and the GE TermiNet 300.

Since UNIX times delays assuming tabs set every 8, this has become a defacto 'standard.'

FILES

--

SEE ALSO

--

DIAGNOSTICS

--

BUGS

--

OWNER

ken