```
          /$$$$$$    /$$     /$$   /$$$$$$$   /$$$$$$  /$$$$$$$
         /$$__  $$ /$$$$   /$$$$  | $$__  $$ /$$__  $$| $$__  $$
        |__/  \ $$|_  $$  |_  $$  | $$  \ $$| $$  \__/| $$  \ $$
          /$$$$$$/  | $$    | $$  | $$$$$$$ |  $$$$$$ | $$  | $$
         /$$____/   | $$    | $$  | $$__  $$ \____  $$| $$  | $$
        | $$        | $$    | $$  | $$  \ $$ /$$  \ $$| $$  | $$
        | $$$$$$$$ /$$$$$$ /$$$$$$| $$$$$$$/|  $$$$$$/| $$$$$$$/
        |________/|______/|______/|_______/  \______/ |_______/

        ********************************************************
        ************** The 211BSD man page project *************
        ********************************************************
        **************** 3F - Fortran Library ******************
        ********************************************************
```

```
Inspired by:
================================================================================
  SimH      http://simh.trailing-edge.com/
  PiDP11    https://obsolescence.wixsite.com/obsolescence/pidp-11
  BSD 2.11  https://wfjm.github.io/home/211bsd/



Presented by the ShadowTron Blog
================================================================================
  https://www.youtube.com/c/shadowtronblog
  www.shadowtron.com
  shadowtronblog <at> gmail <dot> com



Other manuals in the series
================================================================================
    Manual 1  - Commands and Application Programs
    Manual 2  - System Calls
    Manual 3  - C Library Subroutines
==> Manual 3F - Fortran Library
    Manual 4  - Special Files
    Manual 5  - File Formats
    Manual 6  - Games
    Manual 7  - Miscellaneous
    Manual 8  - System Maintenance
```

Version 3.0

```
*************************************
**** Manual 3F - Fortran Library ****
*************************************
```

NAME

intro - introduction to FORTRAN library functions

DESCRIPTION

This section describes those functions that are in the Fortran run time library. The functions listed here provide an interface from f77 programs to the system in the same manner as the C library does for C programs. They are automatically loaded as needed by the Fortran compiler f77(1), except for the graphics interface routines. Those must be explicitly requested, see plot(3f).

The math intrinsics required by the 1977 Fortran standard are available, although not described here. In addition, the abs, sqrt, exp, log, sin, and cos intrinsics have been extended for double complex values. They may be referenced using the generic names listed above, or they may be referenced using their specific names that consist of the generic names preceded by either cd or z. For example, if zz is double complex, then sqrt(zz), zsqrt(zz), or cdsqrt(zz) compute the square root of zz. The dcmplx intrinsic forms a double complex value from two double precision variables or expressions, and the name of the specific function for the conjugate of a double complex value is dconjg.

Most of these functions are in libU77.a. Some are in libF77.a or libI77.a. A few intrinsic functions are described for the sake of completeness.

For efficiency, the SCCS ID strings are not normally included in the a.out file. To include them, simply declare

```
external f77lid
```

in any f77 module.

LIST OF FUNCTIONS

Name	Appears on Page	Description
abort	abort.3f	abnormal termination
access	access.3f	determine accessibility of a file
alarm	alarm.3f	execute a subroutine after a specified time
and	bit.3f	bitwise and
arc	plot.3f	f77 interface to plot(3x)
bessel	bessel.3f	bessel functions of two kinds for integer orders
box	plot.3f	f77 interface to plot(3x)
chdir	chdir.3f	change default directory
chmod	chmod.3f	change mode of a file
circle	plot.3f	f77 interface to plot(3x)
clospl	plot.3f	f77 interface to plot(3x)
cont	plot.3f	f77 interface to plot(3x)
ctime	time.3f	return system time
dffrac	flmin.3f	return extreme values

dflmax	flmin.3f	return extreme values
dflmin	flmin.3f	return extreme values
drand	rand.3f	return random values
drandm	random.3f	better random number generator
dtime	etime.3f	return elapsed execution time
erase	plot.3f	f77 interface to plot(3x)
etime	etime.3f	return elapsed execution time
exit	exit.3f	terminate process with status
falloc	malloc.3f	memory allocator
fdate	fdate.3f	return date and time in an ASCII string
ffrac	flmin.3f	return extreme values
fgetc	getc.3f	get a character from a logical unit
flmax	flmin.3f	return extreme values
flmin	flmin.3f	return extreme values
flush	flush.3f	flush output to a logical unit
fork	fork.3f	create a copy of this process
fpecnt	trpfpe.3f	trap and repair floating point faults
fputc	putc.3f	write a character to a fortran logical unit
free	malloc.3f	memory allocator
fseek	fseek.3f	reposition a file on a logical unit
fstat	stat.3f	get file status
ftell	fseek.3f	reposition a file on a logical unit
gerror	perror.3f	get system error messages
getarg	getarg.3f	return command line arguments
getc	getc.3f	get a character from a logical unit
getcwd	getcwd.3f	get pathname of current working directory
getenv	getenv.3f	get value of environment variables
getgid	getuid.3f	get user or group ID of the caller
getlog	getlog.3f	get user's login name
getpid	getpid.3f	get process id
getuid	getuid.3f	get user or group ID of the caller
gmtime	time.3f	return system time
hostnm	hostnm.3f	get name of current host
iargc	getarg.3f	return command line arguments
idate	idate.3f	return date or time in numerical form
ierrno	perror.3f	get system error messages
index	index.3f	tell about character objects
inmax	flmin.3f	return extreme values
ioinit	ioinit.3f	change f77 I/O initialization
irand	rand.3f	return random values
irandm	random.3f	better random number generator
isatty	ttynam.3f	find name of a terminal port
itime	idate.3f	return date or time in numerical form
kill	kill.3f	send a signal to a process
label	plot.3f	f77 interface to plot(3x)
len	index.3f	tell about character objects
line	plot.3f	f77 interface to plot(3x)
linemd	plot.3f	f77 interface to plot(3x)
link	link.3f	make a link to an existing file
lnblnk	index.3f	tell about character objects
loc	loc.3f	return the address of an object
long	long.3f	integer object conversion

lshift	bit.3f	left shift
lstat	stat.3f	get file status
ltime	time.3f	return system time
malloc	malloc.3f	memory allocator
move	plot.3f	f77 interface to plot(3x)
not	bit.3f	bitwise complement
openpl	plot.3f	f77 interface to plot(3x)
or	bit.3f	bitwise or
perror	perror.3f	get system error messages
point	plot.3f	f77 interface to plot(3x)
putc	putc.3f	write a character to a fortran logical unit
qsort	qsort.3f	quick sort
rand	rand.3f	return random values
random	random.3f	better random number generator
rename	rename.3f	rename a file
rindex	index.3f	tell about character objects
rshift	bit.3f	right shift
short	long.3f	integer object conversion
signal	signal.3f	change the action for a signal
sleep	sleep.3f	suspend execution for an interval
space	plot.3f	f77 interface to plot(3x)
stat	stat.3f	get file status
symlnk	symlnk.3f	make a symbolic link
system	system.3f	execute a UNIX command
tclose	topen.3f	f77 tape I/O
time	time.3f	return system time
topen	topen.3f	f77 tape I/O
traper	traper.3f	trap arithmetic errors
trapov	trapov.3f	trap and repair floating point overflow
tread	topen.3f	f77 tape I/O
trewin	topen.3f	f77 tape I/O
trpfpe	trpfpe.3f	trap and repair floating point faults
tskipf	topen.3f	f77 tape I/O
tstate	topen.3f	f77 tape I/O
ttynam	ttynam.3f	find name of a terminal port
twrite	topen.3f	f77 tape I/O
unlink	unlink.3f	remove a directory entry
wait	wait.3f	wait for a process to terminate
xor	bit.3f	bitwise exclusive or

NAME

abort - abnormal termination

SYNOPSIS

```
subroutine abort (string)
character*(*) string
```

DESCRIPTION

Abort cleans up the I/O buffers and then terminates execution. If string is given, it is written to logical unit 0 preceded by ``abort:''.

If the -g flag was specified during loading, then execution is terminated by calling abort (3) which aborts producing a core file in the current directory. If -g was not specified while loading, then *** Execution terminated is written on logical unit 0 and execution is terminated.

If the f77_dump_flag environment variable has been set to a value which begins with y, abort (3) is called whether or not -g was specified during loading. Similarly, if the value of f77_dump_flag begins with n, abort is not called.

FILES

/usr/lib/libF77.a

SEE ALSO

abort(3)

BUGS

String is ignored on the PDP11.

NAME

access - determine accessibility of a file

SYNOPSIS

```
integer function access (name, mode)
character*(*) name, mode
```

DESCRIPTION

Access checks the given file, name, for accessibility with respect to the caller according to mode. Mode may include in any order and in any combination one or more of:

r	test for read permission
w	test for write permission
x	test for execute permission
(blank)	test for existence

An error code is returned if either argument is illegal, or if the file cannot be accessed in all of the specified modes. 0 is returned if the specified access would be successful.

FILES

/usr/lib/libU77.a

SEE ALSO

access(2), perror(3F)

BUGS

Pathnames can be no longer than MAXPATHLEN as defined in <sys/param.h>.

NAME

alarm - execute a subroutine after a specified time

SYNOPSIS

```
integer function alarm (time, proc)
integer time
external proc
```

DESCRIPTION

This routine arranges for subroutine proc to be called after time seconds. If time is ``0'', the alarm is turned off and no routine will be called. The returned value will be the time remaining on the last alarm.

FILES

/usr/lib/libU77.a

SEE ALSO

alarm(3C), sleep(3F), signal(3F)

BUGS

Alarm and sleep interact. If sleep is called after alarm, the alarm process will never be called. SIGALRM will occur at the lesser of the remaining alarm time or the sleep time.

NAME

bessel functions - of two kinds for integer orders

SYNOPSIS

```
function besj0 (x)

function besj1 (x)

function besjn (n, x)

function besy0 (x)

function besy1 (x)

function besyn (n, x)

double precision function dbesj0 (x)
double precision x

double precision function dbesj1 (x)
double precision x

double precision function dbesjn (n, x)
double precision x

double precision function dbesy0 (x)
double precision x

double precision function dbesy1 (x)
double precision x

double precision function dbesyn (n, x)
double precision x
```

DESCRIPTION

These functions calculate Bessel functions of the first and second kinds for real arguments and integer orders.

DIAGNOSTICS

Negative arguments cause besy0, besy1, and besyn to return a huge negative value. The system error code will be set to EDOM (33).

FILES

/usr/lib/libF77.a

SEE ALSO

j0(3M), perror(3F)

NAME

bit - and, or, xor, not, rshift, lshift bitwise functions

SYNOPSIS

```
(intrinsic) function and (word1, word2)

(intrinsic) function or (word1, word2)

(intrinsic) function xor (word1, word2)

(intrinsic) function not (word)

(intrinsic) function rshift (word, nbits)

(intrinsic) function lshift (word, nbits)
```

DESCRIPTION

These bitwise functions are built into the compiler and return the data type of their argument(s). Their arguments must be integer or logical values.

The bitwise combinatorial functions return the bitwise ``and'' (and), ``or'' (or), or ``exclusive or'' (xor) of two operands. Not returns the bitwise complement of its operand.

Lshift, or rshift with a negative nbits, is a logical left shift with no end around carry. Rshift, or lshift with a negative nbits, is an arithmetic right shift with sign extension. No test is made for a reasonable value of nbits.

These functions may be used to create a variety of general routines, as in the following statement function definitions:

```
integer bitset, bitclr, getbit, word, bitnum

bitset( word, bitnum ) = or(word,lshift(1,bitnum))
bitclr( word, bitnum ) = and(word,not(lshift(1,bitnum)))
getbit( word, bitnum ) = and(rshift(word,bitnum),1)
```

FILES

These functions are generated in-line by the f77 compiler.

NAME

chdir - change default directory

SYNOPSIS

```
integer function chdir (dirname)
character*(*) dirname
```

DESCRIPTION

The default directory for creating and locating files will be changed to dirname. Zero is returned if successful; an error code otherwise.

FILES

/usr/lib/libU77.a

SEE ALSO

chdir(2), cd(1), perror(3F)

BUGS

Pathnames can be no longer than MAXPATHLEN as defined in <sys/param.h>.

Use of this function may cause inquire by unit to fail.

NAME
chmod - change mode of a file

SYNOPSIS
```
integer function chmod (name, mode)
character*(*) name, mode
```

DESCRIPTION
This function changes the filesystem mode of file name. Mode can be any specification recognized by chmod(1). Name must be a single pathname.

The normal returned value is 0. Any other value will be a system error number.

FILES
```
/usr/lib/libU77.a
/bin/chmod          exec'ed to change the mode.
```

SEE ALSO
chmod(1)

BUGS
Pathnames can be no longer than MAXPATHLEN as defined in <sys/param.h>.

NAME

etime, dtime - return elapsed execution time

SYNOPSIS

```
function etime (tarray)
real tarray(2)

function dtime (tarray)
real tarray(2)
```

DESCRIPTION

These two routines return elapsed runtime in seconds for the calling process. Dtime returns the elapsed time since the last call to dtime, or the start of execution on the first call.

The argument array returns user time in the first element and system time in the second element. The function value is the sum of user and system time.

The resolution of all timing is 1/HZ sec. where HZ is currently 60.

FILES

/usr/lib/libU77.a

SEE ALSO

times(2)

NAME
     exit - terminate process with status

SYNOPSIS
```
     subroutine exit (status)
     integer status
```

DESCRIPTION
     Exit flushes and closes all the process's files, and notifies the parent process if it is executing a wait. The low-order 8 bits of status are available to the parent process. (Therefore status should be in the range 0 - 255)

     This call will never return.

     The C function exit may cause cleanup actions before the final `sys exit'.

FILES
     /usr/lib/libF77.a

SEE ALSO
     exit(2), fork(2), fork(3F), wait(2), wait(3F)

NAME

fdate - return date and time in an ASCII string

SYNOPSIS

```
subroutine fdate (string)
character*(*) string

character*(*) function fdate()
```

DESCRIPTION

Fdate returns the current date and time as a 24 character string in the format described under ctime(3). Neither `newline' nor NULL will be included.

Fdate can be called either as a function or as a subroutine. If called as a function, the calling routine must define its type and length. For example:

```
   character*24    fdate
   external   fdate

   write(*,*) fdate()
```

FILES

/usr/lib/libU77.a

SEE ALSO

ctime(3), time(3F), itime(3F), idate(3F), ltime(3F)

NAME

flmin, flmax, ffrac, dflmin, dflmax, dffrac, inmax - return extreme values

SYNOPSIS

```
function flmin()

function flmax()

function ffrac()

double precision function dflmin()

double precision function dflmax()

double precision function dffrac()

function inmax()
```

DESCRIPTION

Functions flmin and flmax return the minimum and maximum positive floating point values respectively. Functions dflmin and dflmax return the minimum and maximum positive double precision floating point values. Function inmax returns the maximum positive integer value.

The functions ffrac and dffrac return the fractional accuracy of single and double precision floating point numbers respectively. This is the difference between 1.0 and the smallest real number greater than 1.0.

These functions can be used by programs that must scale algorithms to the numerical range of the processor.

FILES

/usr/lib/libF77.a

NAME
flush - flush output to a logical unit

SYNOPSIS
```
subroutine flush (lunit)
```

DESCRIPTION
Flush causes the contents of the buffer for logical unit lunit to be flushed to the associated file. This is most useful for logical units 0 and 6 when they are both associated with the control terminal.

FILES
/usr/lib/libI77.a

SEE ALSO
fclose(3S)

NAME
fork - create a copy of this process

SYNOPSIS
integer function fork()

DESCRIPTION
Fork creates a copy of the calling process. The only distinction between the 2 processes is that the value returned to one of them (referred to as the `parent' process) will be the process id of the copy. The copy is usually referred to as the `child' process. The value returned to the `child' process will be zero.

All logical units open for writing are flushed before the fork to avoid duplication of the contents of I/O buffers in the external file(s).

If the returned value is negative, it indicates an error and will be the negation of the system error code. See perror(3F).

A corresponding exec routine has not been provided because there is no satisfactory way to retain open logical units across the exec. However, the usual function of fork/exec can be performed using system(3F).

FILES
/usr/lib/libU77.a

SEE ALSO
fork(2), wait(3F), kill(3F), system(3F), perror(3F)

NAME

fseek, ftell - reposition a file on a logical unit

SYNOPSIS

```
integer function fseek (lunit, offset, from)
integer offset, from

integer function ftell (lunit)
```

DESCRIPTION

lunit must refer to an open logical unit.   offset is an offset in bytes relative to the position specified by from. Valid values for from are:

```
   0 meaning `beginning of the file'
   1 meaning `the current position'
   2 meaning `the end of the file'
```

The value returned by fseek will be 0 if successful, a system error code otherwise.   (See perror(3F))

Ftell returns the current position of the file associated with the specified logical unit. The value is an offset, in bytes, from the beginning of the file. If the value returned is negative, it indicates an error and will be the negation of the system error code. (See perror(3F))

FILES

/usr/lib/libU77.a

SEE ALSO

fseek(3S), perror(3F)

NAME

getarg, iargc - return command line arguments

SYNOPSIS

```
subroutine getarg (k, arg)
character*(*) arg

function iargc ()
```

DESCRIPTION

A call to getarg will return the kth command line argument in character string arg. The 0th argument is the command name.

Iargc returns the index of the last command line argument.

FILES

/usr/lib/libU77.a

SEE ALSO

getenv(3F), execve(2)

NAME

getc, fgetc - get a character from a logical unit

SYNOPSIS

```
integer function getc (char)
character char

integer function fgetc (lunit, char)
character char
```

DESCRIPTION

These routines return the next character from a file associated with a fortran logical unit, bypassing normal fortran I/O. Getc reads from logical unit 5, normally connected to the control terminal input.

The value of each function is a system status code. Zero indicates no error occurred on the read; -1 indicates end of file was detected. A positive value will be either a UNIX system error code or an f77 I/O error code. See perror(3F).

FILES

/usr/lib/libU77.a

SEE ALSO

getc(3S), intro(2), perror(3F)

NAME

getcwd - get pathname of current working directory

SYNOPSIS

```
integer function getcwd (dirname)
character*(*) dirname
```

DESCRIPTION

The pathname of the default directory for creating and locating files will be returned in dirname. The value of the function will be zero if successful; an error code otherwise.

FILES

/usr/lib/libU77.a

SEE ALSO

chdir(3F), perror(3F)

BUGS

Pathnames can be no longer than MAXPATHLEN as defined in <sys/param.h>.

NAME
getenv - get value of environment variables

SYNOPSIS
```
subroutine getenv (ename, evalue)
character*(*) ename, evalue
```

DESCRIPTION
Getenv searches the environment list (see environ(7)) for a string of the form ename=value and returns value in evalue if such a string is present, otherwise fills evalue with blanks.

FILES
/usr/lib/libU77.a

SEE ALSO
environ(7), execve(2)

NAME

getlog - get user's login name

SYNOPSIS

```
subroutine getlog (name)
character*(*) name

character*(*) function getlog()
```

DESCRIPTION

Getlog will return the user's login name or all blanks if the process is running detached from a terminal.

FILES

/usr/lib/libU77.a

SEE ALSO

getlogin(3)

NAME

getpid - get process id

SYNOPSIS

```
integer function getpid()
```

DESCRIPTION

Getpid returns the process ID number of the current process.

FILES

/usr/lib/libU77.a

SEE ALSO

getpid(2)

NAME
getuid, getgid - get user or group ID of the caller

SYNOPSIS
```
integer function getuid()

integer function getgid()
```

DESCRIPTION
These functions return the real user or group ID of the user of the process.

FILES
/usr/lib/libU77.a

SEE ALSO
getuid(2)

NAME
     hostnm - get name of current host

SYNOPSIS
```
     integer function hostnm (name)
     character*(*) name
```

DESCRIPTION
     This function puts the name of the current host into character string name. The return value should be 0; any other value indicates an error.

FILES
     /usr/lib/libU77.a

SEE ALSO
     gethostname(2)

NAME
     idate, itime - return date or time in numerical form

SYNOPSIS
```
     subroutine idate (iarray)
     integer iarray(3)

     subroutine itime (iarray)
     integer iarray(3)
```

DESCRIPTION
     Idate returns the current date in iarray. The order is: day, mon, year. Month will be in the range 1-12. Year will be > 1969.

     Itime returns the current time in iarray. The order is: hour, minute, second.

FILES
     /usr/lib/libU77.a

SEE ALSO
     ctime(3F), fdate(3F)

NAME
     index, rindex, lnblnk, len - tell about character objects

SYNOPSIS
```
     (intrinsic) function index (string, substr)
     character*(*) string, substr

     integer function rindex (string, substr)
     character*(*) string, substr

     function lnblnk (string)
     character*(*) string

     (intrinsic) function len (string)
     character*(*) string
```

DESCRIPTION
     Index (rindex) returns the index of the first (last) occurrence of the substring substr in string, or zero if it does not occur. Index is an f77 intrinsic function; rindex is a library routine.

     Lnblnk returns the index of the last non-blank character in string. This is useful since all f77 character objects are fixed length, blank padded. Intrinsic function len returns the size of the character object argument.

FILES
     /usr/lib/libF77.a

NAME

ioinit - change f77 I/O initialization

SYNOPSIS

```
logical function ioinit (cctl, bzro, apnd, prefix, vrbose)
logical cctl, bzro, apnd, vrbose
character*(*) prefix
```

DESCRIPTION

This routine will initialize several global parameters in the f77 I/O system, and attach externally defined files to logical units at run time. The effect of the flag arguments applies to logical units opened after ioinit is called. The exception is the preassigned units, 5 and 6, to which cctl and bzro will apply at any time. Ioinit is written in Fortran-77.

By default, carriage control is not recognized on any logical unit. If cctl is .true. then carriage control will be recognized on formatted output to all logical units except unit 0, the diagnostic channel. Otherwise the default will be restored.

By default, trailing and embedded blanks in input data fields are ignored. If bzro is .true. then such blanks will be treated as zeros. Otherwise the default will be restored.

By default, all files opened for sequential access are positioned at their beginning. It is sometimes necessary or convenient to open at the END-OF-FILE so that a write will append to the existing data. If apnd is .true. then files opened subsequently on any logical unit will be positioned at their end upon opening. A value of .false. will restore the default behavior.

Ioinit may be used to associate file names with Fortran logical unit numbers through environment variables (see "Introduction to the f77 I/O Library" for a more general way of doing this). If the argument prefix is a non-blank string, then names of the form prefixNN will be sought in the program environment. The value associated with each such name found will be used to open logical unit NN for formatted sequential access. For example, if f77 program myprogram included the call

```
   call ioinit (.true., .false., .false., 'FORT', .false.)
```

then when the following sequence

```
    % setenv FORT01 mydata
    % setenv FORT12 myresults
```

```
          % myprogram
```

would result in logical unit 1 opened to file mydata and logical unit 12 opened to file myresults.   Both files would be positioned at their beginning. Any formatted output would have column 1 removed and interpreted as carriage control. Embedded and trailing blanks would be ignored on input.

If the argument vrbose is .true. then ioinit will report on its activity.

The effect of

```
   call ioinit (.true., .true., .false., '', .false.)
```

can be achieved without the actual call by including ``-lI66'' on the f77 command line. This gives carriage control on all logical units except 0, causes files to be opened at their beginning, and causes blanks to be interpreted as zero's.

The internal flags are stored in a labeled common block with the following definition:

```
   integer*2 ieof, ictl, ibzr
   common /ioiflg/ ieof, ictl, ibzr
```

FILES

```
     /usr/lib/libI77.a      f77 I/O library
     /usr/lib/libI66.a      sets older fortran I/O modes
```

SEE ALSO

getarg(3F), getenv(3F), ``Introduction to the f77 I/O Library''

BUGS

Prefix can be no longer than 30 characters. A pathname associated with an environment name can be no longer than 255 characters.

The ``+'' carriage control does not work.

NAME

kill - send a signal to a process

SYNOPSIS

```
function kill (pid, signum)
integer pid, signum
```

DESCRIPTION

Pid must be the process id of one of the user's processes. Signum must be a valid signal number (see sigvec(2)). The returned value will be 0 if successful; an error code otherwise.

FILES

/usr/lib/libU77.a

SEE ALSO

kill(2), sigvec(2), signal(3F), fork(3F), perror(3F)

NAME

link - make a link to an existing file

SYNOPSIS

```
function link (name1, name2)
character*(*) name1, name2

integer function symlnk (name1, name2)
character*(*) name1, name2
```

DESCRIPTION

Name1 must be the pathname of an existing file. Name2 is a pathname to be linked to file name1. Name2 must not already exist. The returned value will be 0 if successful; a system error code otherwise.

Symlnk creates a symbolic link to name1.

FILES

/usr/lib/libU77.a

SEE ALSO

link(2), symlink(2), perror(3F), unlink(3F)

BUGS

Pathnames can be no longer than MAXPATHLEN as defined in <sys/param.h>.

NAME
     long, short - integer object conversion

SYNOPSIS
```
     integer*4 function long (int2)
     integer*2 int2

     integer*2 function short (int4)
     integer*4 int4
```

DESCRIPTION
     These functions provide conversion between short and long integer objects. Long is useful when constants are used in calls to library routines and the code is to be compiled with ``-i2''. Short is useful in similar context when an otherwise long object must be passed as a short integer.

FILES
     /usr/lib/libF77.a

NAME

malloc, free, falloc - memory allocator

SYNOPSIS

```
subroutine malloc (size, addr)
integer size, addr

subroutine free (addr)
integer addr

subroutine falloc (nelem, elsize, clean, basevec, addr, offset)
integer nelem, elsize, clean, addr, offset
```

DESCRIPTION

Malloc, falloc and free provide a general-purpose memory allocation package. Malloc returns in addr the address of a block of at least size bytes beginning on an even-byte boundary.

Falloc allocates space for an array of nelem elements of size elsize and returns the address of the block in addr. It zeros the block if clean is 1. It returns in offset an index such that the storage may be addressed as basevec(offset+1) ... basevec(offset+nelem). Falloc gets extra bytes so that after address arithmetic, all the objects so addressed are within the block.

The argument to free is the address of a block previously allocated by malloc or falloc; this space is made available for further allocation, but its contents are left undisturbed. To free blocks allocated by falloc, use addr in calls to free, do not use basevec(offset+1).

Needless to say, grave disorder will result if the space assigned by mallocorfalloc is overrun or if some random number is handed to free.

DIAGNOSTICS

Malloc and falloc set addr to 0 if there is no available memory or if the arena has been detectably corrupted by storing outside the bounds of a block.

The following example shows how to obtain memory and use it within a subprogram:

```
    integer addr, work(1), offset
       ...
    call falloc ( n, 4, 0, work, addr, offset )
    do 10 i = 1, n
    work(offset+i) = ...
10    continue
```

The next example reads in dimension information, allocates space for two arrays and two vectors, and calls subroutine doit to do the computations:

```
    integer addr, dummy(1), offs
    read *, k, l, m
    indm1   = 1
    indm2   = indm1 + k*l
    indm3   = indm2 + l*m
    indsym  = indm3 + k*m
    lsym = n*(n+1)/2
    indv  = indsym + lsym
    indtot = indv + m
    call falloc ( indtot, 4, 0, dummy, addr, offs )
    call doit( dummy(indm1+offs), dummy(indm2+offs),
   .         dummy(indm3+offs), dummy(indsym+offs),
   .         dummy(indv +offs), m, n, lsym )
    end
    subroutine doit( arr1, arr2, arr3, vsym, vec, m, n, lsym )
    real arr1(k,l), arr2(l,m), arr3(k,m), vsym(lsym), v2(m)
       ...
```

FILES
/usr/lib/libU77.a

SEE ALSO
malloc(3)

NAME

perror, gerror, ierrno - get system error messages

SYNOPSIS

```
subroutine perror (string)
character*(*) string

subroutine gerror (string)
character*(*) string

character*(*) function gerror()

function ierrno()
```

DESCRIPTION

Perror will write a message to fortran logical unit 0 appropriate to the last detected system error. String will be written preceding the standard error message.

Gerror returns the system error message in character variable string. Gerror may be called either as a subroutine or as a function.

Ierrno will return the error number of the last detected system error. This number is updated only when an error actually occurs. Most routines and I/O statements that might generate such errors return an error code after the call; that value is a more reliable indicator of what caused the error condition.

FILES

/usr/lib/libU77.a

SEE ALSO

intro(2), perror(3)
D. L. Wasley, Introduction to the f77 I/O Library

BUGS

String in the call to perror can be no longer than 127 characters.

The length of the string returned by gerror is determined by the calling program.

NOTES

UNIX system error codes are described in intro(2). The f77 I/O error codes and their meanings are:

```
   100  ``error in format''
   101  ``illegal unit number''
   102  ``formatted i/o not allowed''
   103  ``unformatted i/o not allowed''
```

```
        104  ``direct i/o not allowed''
        105  ``sequential i/o not allowed''
        106  ``can't backspace file''
        107  ``off beginning of record''
        108  ``can't stat file''
        109  ``no * after repeat count''
        110  ``off end of record''
        111  ``truncation failed''
        112  ``incomprehensible list input''
        113  ``out of free space''
        114  ``unit not connected''
        115  ``invalid data for integer format term''
        116  ``invalid data for logical format term''
        117  ``'new' file exists''
        118  ``can't find 'old' file''
        119  ``opening too many files or unknown system error''
        120  ``requires seek ability''
        121  ``illegal argument''
        122  ``negative repeat count''
        123  ``illegal operation for unit''
        124  ``invalid data for d, e, f, or g format term''
```

NAME
    plot: openpl et al. - f77 library interface to plot (3X) libraries.

SYNOPSIS
```
subroutine openpl()

subroutine erase()

subroutine label(str)
character str*(*)

subroutine line(ix1, iy1, ix2, iy2)

subroutine box(ix1, iy1, ix2, iy2)
Draw a rectangle and leave the cursor at ( ix2,iy2).

subroutine circle(ix, iy, ir)

subroutine arc(ix, iy, ix0, iy0, ix1, iy1)

subroutine move(ix, iy)

subroutine cont(ix, iy)

subroutine point(ix, iy)

subroutine linemd(str)
character str*(*)

subroutine space(ix0, iy0, ix1, iy1)

subroutine clospl()
```

DESCRIPTION
    These are interface subroutines, in the library -lf77plot, allowing f77 users to call the plot(3X) graphics routines which generate graphic output in a relatively device-independent manner. The f77 subroutine names are the same as the C function names except that linemod and closepl have been shortened to linemd and clospl . See plot(5) and plot(3X) for a description of their effect.

    Only the first 255 character in string arguments to label and linemd are used.

    This library must be specified in the f77(1) command before the device specific graphics library; for example, to compile and load a FORTRAN program in prog.f to run on a Tektronix 4014 terminal:

```
f77 prog.f -lf77plot -l4014
```

See plot(3X) for a complete list of device specific plotting libraries.

SEE ALSO

plot(5), plot(1G), plot(3X), graph(1G)

NAME
     putc, fputc - write a character to a fortran logical unit

SYNOPSIS
```
     integer function putc (char)
     character char

     integer function fputc (lunit, char)
     character char
```

DESCRIPTION
     These funtions write a character to the file associated with a fortran logical unit bypassing normal fortran I/O. Putc writes to logical unit 6, normally connected to the control terminal output.

     The value of each function will be zero unless some error occurred; a system error code otherwise. See perror(3F).

FILES
     /usr/lib/libU77.a

SEE ALSO
     putc(3S), intro(2), perror(3F)

NAME

qsort - quick sort

SYNOPSIS

```
subroutine qsort (array, len, isize, compar)
external compar
integer*2 compar
```

DESCRIPTION

One dimensional array contains the elements to be sorted. len is the number of elements in the array. isize is the size of an element, typically -

```
   4 for integer and real
   8 for double precision or complex
   16 for double complex
   (length of character object) for character arrays
```

Compar is the name of a user supplied integer*2 function that will determine the sorting order. This function will be called with 2 arguments that will be elements of array. The function must return -

```
   negative if arg 1 is considered to precede arg 2
   zero if arg 1 is equivalent to arg 2
   positive if arg 1 is considered to follow arg 2
```

On return, the elements of array will be sorted.

FILES

/usr/lib/libU77.a

SEE ALSO

qsort(3)

NAME

rand, drand, irand - return random values

SYNOPSIS

```
function irand (iflag)

function rand (iflag)

double precision function drand (iflag)
```

DESCRIPTION

The newer random(3f) should be used in new applications; rand remains for compatibilty.

These functions use rand(3C) to generate sequences of random numbers. If iflag is '1', the generator is restarted and the first random value is returned. If iflag is otherwise non-zero, it is used as a new seed for the random number generator, and the first new random value is returned.

Irand returns positive integers in the range 0 through 2147483647. Rand and drand return values in the range 0. through 1.0 .

FILES

/usr/lib/libF77.a

SEE ALSO

random(3F), rand(3C)

BUGS

The algorithm returns a 15 bit quantity on the PDP11; a 31 bit quantity on the VAX. Irand on the PDP11 calls rand(3C) twice to form a 31 bit quantity, but bit 15 will always be 0.

NAME

random, drandm, irandm - better random number generator

SYNOPSIS

```
function irandm (iflag)

function random (iflag)

double precision function drandm (iflag)
```

DESCRIPTION

These functions use random(3) to generate sequences of random numbers, and should be used rather than the older functions described in man 3f rand. If iflag is non-zero, it is used as a new seed for the random number generator, and the first new random value is returned.

Irandm returns positive integers in the range 0 through 2147483647 (2**31-1). Random and drandm return values in the range 0. through 1.0 by dividing the integer random number from random(3) by 2147483647 .

FILES

/usr/lib/libF77.a

SEE ALSO

random(3)

NAME
     rename - rename a file

SYNOPSIS
```
     integer function rename (from, to)
     character*(*) from, to
```

DESCRIPTION
     From must be the pathname of an existing file. To will become the new pathname for the file. If to exists, then both from and to must be the same type of file, and must reside on the same filesystem. If to exists, it will be removed first.

     The returned value will be 0 if successful; a system error code otherwise.

FILES
     /usr/lib/libU77.a

SEE ALSO
     rename(2), perror(3F)

BUGS
     Pathnames can be no longer than MAXPATHLEN as defined in <sys/param.h>.

NAME
     signal - change the action for a signal

SYNOPSIS
```
     integer function signal(signum, proc, flag)
     integer signum, flag
     external proc
```

DESCRIPTION

When a process incurs a signal (see signal(3C)) the default action is usually to clean up and abort. The user may choose to write an alternative signal handling routine. A call to signal is the way this alternate action is specified to the system.

Signum is the signal number (see signal(3C)). If flag is negative, then proc must be the name of the user signal handling routine. If flag is zero or positive, then proc is ignored and the value of flag is passed to the system as the signal action definition. In particular, this is how previously saved signal actions can be restored. Two possible values for flag have specific meanings: 0 means "use the default action" (See NOTES below), 1 means "ignore this signal".

A positive returned value is the previous action definition. A value greater than 1 is the address of a routine that was to have been called on occurrence of the given signal. The returned value can be used in subsequent calls to signal in order to restore a previous action definition. A negative returned value is the negation of a system error code. (See perror(3F))

FILES
     /usr/lib/libU77.a

SEE ALSO
     signal(3C), kill(3F), kill(1)

NOTES

f77 arranges to trap certain signals when a process is started. The only way to restore the default f77 action is to save the returned value from the first call to signal.

If the user signal handler is called, it will be passed the signal number as an integer argument.

NAME
     sleep - suspend execution for an interval

SYNOPSIS
```
     subroutine sleep (itime)
```

DESCRIPTION
     Sleep causes the calling process to be suspended for itime seconds. The actual time can be up to 1 second less than itime due to granularity in system timekeeping.

FILES
     /usr/lib/libU77.a

SEE ALSO
     sleep(3)

NAME

stat, lstat, fstat - get file status

SYNOPSIS

```
integer function stat (name, statb)
character*(*) name
integer statb(12)

integer function lstat (name, statb)
character*(*) name
integer statb(12)

integer function fstat (lunit, statb)
integer statb(12)
```

DESCRIPTION

These routines return detailed information about a file. Stat and lstat return information about file name; fstat returns information about the file associated with fortran logical unit lunit. The order and meaning of the information returned in array statb is as described for the structure stat under stat(2). The ``spare'' values are not included.

The value of either function will be zero if successful; an error code otherwise.

FILES

/usr/lib/libU77.a

SEE ALSO

stat(2), access(3F), perror(3F), time(3F)

BUGS

Pathnames can be no longer than MAXPATHLEN as defined in <sys/param.h>.

NAME

system - execute a UNIX command

SYNOPSIS

```
integer function system (string)
character*(*) string
```

DESCRIPTION

System causes string to be given to your shell as input as if the string had been typed as a command. If environment variable SHELL is found, its value will be used as the command interpreter (shell); otherwise sh(1) is used.

The current process waits until the command terminates. The returned value will be the exit status of the shell. See wait(2) for an explanation of this value.

FILES

/usr/lib/libU77.a

SEE ALSO

exec(2), wait(2), system(3)

BUGS

String can not be longer than NCARGS-50 characters, as defined in <sys/param.h>.

NAME
     time, ctime, ltime, gmtime - return system time

SYNOPSIS
```
     integer function time()

     character*(*) function ctime (stime)
     integer stime

     subroutine ltime (stime, tarray)
     integer stime, tarray(9)

     subroutine gmtime (stime, tarray)
     integer stime, tarray(9)
```

DESCRIPTION
     Time returns the time since 00:00:00 GMT, Jan. 1, 1970, measured in seconds. This is the value of the UNIX system clock.

     Ctime converts a system time to a 24 character ASCII string. The format is described under ctime(3). No 'newline' or NULL will be included.

     Ltime and gmtime disect a UNIX time into month, day, etc., either for the local time zone or as GMT.   The order and meaning of each element returned in tarray is described under ctime(3).

FILES
     /usr/lib/libU77.a

SEE ALSO
     ctime(3), itime(3F), idate(3F), fdate(3F)

NAME

topen, tclose, tread, twrite, trewin, tskipf, tstate - f77 tape I/O

SYNOPSIS

```
integer function topen (tlu, devnam, label)
integer tlu
character*(*) devnam
logical label

integer function tclose (tlu)
integer tlu

integer function tread (tlu, buffer)
integer tlu
character*(*) buffer

integer function twrite (tlu, buffer)
integer tlu
character*(*) buffer

integer function trewin (tlu)
integer tlu

integer function tskipf (tlu, nfiles, nrecs)
integer tlu, nfiles, nrecs

integer function tstate (tlu, fileno, recno, errf, eoff,
eotf, tcsr)
integer tlu, fileno, recno, tcsr
logical errf, eoff, eotf
```

DESCRIPTION

These functions provide a simple interface between f77 and magnetic tape devices. A ``tape logical unit'', tlu, is ``topen''ed in much the same way as a normal f77 logical unit is ``open''ed. All other operations are performed via the tlu. The tlu has no relationship at all to any normal f77 logical unit.

Topen associates a device name with a tlu. Tlu must be in the range 0 to 3. The logical argument label should indicate whether the tape includes a tape label. This is used by trewin below. Topen does not move the tape. The normal returned value is 0. If the value of the function is negative, an error has occured. See perror(3F) for details.

Tclose closes the tape device channel and removes its association with tlu. The normal returned value is 0. A negative value indicates an error.

Tread reads the next physical record from tape to buffer. Buffer must be of type character. The size of buffer should be large enough to hold the largest physical record to be read. The actual number of bytes read will be returned as the value of the function. If the value is 0, the end-of-file has been detected. A negative value indicates an error.

Twrite writes a physical record to tape from buffer. The physical record length will be the size of buffer. Buffer must be of type character. The number of bytes written will be returned. A value of 0 or negative indicates an error.

Trewin rewinds the tape associated with tlu to the beginning of the first data file. If the tape is a labelled tape (see topen above) then the label is skipped over after rewinding. The normal returned value is 0. A negative value indicates an error.

Tskipf allows the user to skip over files and/or records. First, nfiles end-of-file marks are skipped. If the current file is at EOF, this counts as 1 file to skip. (Note: This is the way to reset the EOF status for a tlu.) Next, nrecs physical records are skipped over. The normal returned value is 0. A negative value indicates an error.

Finally, tstate allows the user to determine the logical state of the tape I/O channel and to see the tape drive control status register. The values of fileno and recno will be returned and indicate the current file and record number. The logical values errf, eoff, and eotf indicate an error has occurred, the current file is at EOF, or the tape has reached logical end-of-tape. End-of-tape (EOT) is indicated by an empty file, often referred to as a double EOF mark. It is not allowed to read past EOT although it is allowed to write. The value of tcsr will reflect the tape drive control status register. See ht(4) for details.

FILES

/usr/lib/libU77.a

SEE ALSO

ht(4), perror(3F), rewind(1)

NAME
    traper - trap arithmetic errors

SYNOPSIS
    integer function traper (mask)

DESCRIPTION
    NOTE: This routine applies only to the VAX. It is ignored on the PDP11.

    Integer overflow and floating point underflow are not normally trapped during execution. This routine enables these traps by setting status bits in the process status word. These bits are reset on entry to a subprogram, and the previous state is restored on return. Therefore, this routine must be called inside each subprogram in which these conditions should be trapped. If the condition occurs and trapping is enabled, signal SIGFPE is sent to the process. (See signal(3C))

    The argument has the following meaning:

value	meaning
0	do not trap either condition
1	trap integer overflow only
2	trap floating underflow only
3	trap both the above

    The previous value of these bits is returned.

FILES
    /usr/lib/libF77.a

SEE ALSO
    signal(3C), signal(3F)

NAME

trapov - trap and repair floating point overflow

SYNOPSIS

```
subroutine trapov (numesg, rtnval)
double precision rtnval
```

DESCRIPTION

NOTE: This routine applies only to the older VAX 11/780's. VAX computers made or upgraded since spring 1983 handle errors differently. See trpfpe(3F) for the newer error handler. This routine has always been ineffective on the VAX 11/750. It is a null routine on the PDP11.

This call sets up signal handlers to trap arithmetic exceptions and the use of illegal operands. Trapping arithmetic exceptions allows the user's program to proceed from instances of floating point overflow or divide by zero. The result of such operations will be an illegal floating point value. The subsequent use of the illegal operand will be trapped and the operand replaced by the specified value.

The first numesg occurrences of a floating point arithmetic error will cause a message to be written to the standard error file. If the resulting value is used, the value given for rtnval will replace the illegal operand generated by the arithmetic error. Rtnval must be a double precision value. For example, ``0d0'' or ``dflmax()''.

FILES

/usr/lib/libF77.a

SEE ALSO

trpfpe(3F), signal(3F), range(3F)

BUGS

Other arithmetic exceptions can be trapped but not repaired.

There is no way to distinguish between an integer value of 32768 and the illegal floating point form. Therefore such an integer value may get replaced while repairing the use of an illegal operand.

NAME

trpfpe, fpecnt - trap and repair floating point faults

SYNOPSIS

```
subroutine trpfpe (numesg, rtnval)
double precision rtnval

integer function fpecnt ()

common /fpeflt/ fperr
logical fperr
```

DESCRIPTION

NOTE: This routine applies only to Vax computers.   It is a null routine on the PDP11.

Trpfpe sets up a signal handler to trap arithmetic exceptions. If the exception is due to a floating point arithmetic fault, the result of the operation is replaced with the rtnval specified. Rtnval must be a double precision value. For example, ``0d0'' or ``dflmax()''.

The first numesg occurrences of a floating point arithmetic error will cause a message to be written to the standard error file. Any exception that can't be repaired will result in the default action, typically an abort with core image.

Fpecnt returns the number of faults since the last call to trpfpe.

The logical value in the common block labelled fpeflt will be set to .true. each time a fault occurs.

FILES

/usr/lib/libF77.a

SEE ALSO

signal(3F), range(3F)

BUGS

This routine works only for faults, not traps. This is primarily due to the Vax architecture.

If the operation involves changing the stack pointer, it can't be repaired. This seldom should be a problem with the f77 compiler, but such an operation might be produced by the optimizer.

The POLY and EMOD opcodes are not dealt with.

NAME
     ttynam, isatty - find name of a terminal port

SYNOPSIS
     character*(*) function ttynam (lunit)

     logical function isatty (lunit)

DESCRIPTION
     Ttynam returns a blank padded path name of the terminal device associated with logical unit lunit.

     Isatty returns .true. if lunit is associated with a terminal device, .false. otherwise.

FILES
     /dev/*
     /usr/lib/libU77.a

DIAGNOSTICS
     Ttynam returns an empty string (all blanks) if lunit is not associated with a terminal device in directory `/dev'.

NAME

unlink - remove a directory entry

SYNOPSIS

```
integer function unlink (name)
character*(*) name
```

DESCRIPTION

Unlink causes the directory entry specified by pathname name to be removed. If this was the last link to the file, the contents of the file are lost. The returned value will be zero if successful; a system error code otherwise.

FILES

/usr/lib/libU77.a

SEE ALSO

unlink(2), link(3F), filsys(5), perror(3F)

BUGS

Pathnames can be no longer than MAXPATHLEN as defined in <sys/param.h>.

NAME

wait - wait for a process to terminate

SYNOPSIS

```
integer function wait (status)
integer status
```

DESCRIPTION

Wait causes its caller to be suspended until a signal is received or one of its child processes terminates. If any child has terminated since the last wait, return is immediate; if there are no children, return is immediate with an error code.

If the returned value is positive, it is the process ID of the child and status is its termination status (see wait(2)). If the returned value is negative, it is the negation of a system error code.

FILES

/usr/lib/libU77.a

SEE ALSO

wait(2), signal(3F), kill(3F), perror(3F)