A Source Book from The Open Group

Quick Interface Reference to the Base Specifications, Issue 6

The Open Group

A Source Book from The Open Group

Quick Interface Reference to the Base Specifications, Issue 6

Published in the U.K. by The Open Group, April 2003.

Any comments relating to the material contained in this document may be submitted to:

The Open Group
Apex Plaza
Forbury Road
Reading
Berkshire, RG1 1AX
United Kingdom

or by Electronic Mail to:

OGSpecs@opengroup.org

Chapter 1

System Interfaces Reference

This chapter contains a brief reference for each system interface defined in XSH, Issue 6.

_longjmp, _setjmp

Non-local goto

XSI
```
#include <setjmp.h>

void _longjmp(jmp_buf env, int val);
int _setjmp(jmp_buf env);
```

_tolower

Transliterate uppercase characters to lowercase

XSI
```
#include <ctype.h>

int _tolower(int c);
```

_toupper

Transliterate lowercase characters to uppercase

XSI
```
#include <ctype.h>

int _toupper(int c);
```

a64l, l64a

Convert between a 32-bit integer and a radix-64 ASCII string

XSI
```
#include <stdlib.h>

long a64l(const char *s);
char *l64a(long value);
```

abort

Generate an abnormal process abort

```
#include <stdlib.h>

void abort(void);
```

abs

Return an integer absolute value

```
#include <stdlib.h>

int abs(int i);
```

accept

Accept a new connection on a socket

```
#include <sys/socket.h>

int accept(int socket, struct sockaddr *restrict address,
    socklen_t *restrict address_len);
```

access

Determine accessibility of a file

```
#include <unistd.h>

int access(const char *path, int amode);
```

acos, acosf, acosl

Arc cosine functions

```
#include <math.h>

double acos(double x);
float acosf(float x);
long double acosl(long double x);
```

acosh, acoshf, acoshl

Inverse hyperbolic cosine functions

```
#include <math.h>

double acosh(double x);
float acoshf(float x);
long double acoshl(long double x);
```

aio_cancel

Cancel an asynchronous I/O request (**REALTIME**)

AIO

```
#include <aio.h>

int aio_cancel(int fildes, struct aiocb *aiocbp);
```

aio_error

Retrieve errors status for an asynchronous I/O operation (**REALTIME**)

AIO

```
#include <aio.h>

int aio_error(const struct aiocb *aiocbp);
```

aio_fsync

Asynchronous file synchronization (**REALTIME**)

AIO
```
#include <aio.h>

int aio_fsync(int op, struct aiocb *aiocbp);
```

aio_read

Asynchronous read from a file (**REALTIME**)

AIO
```
#include <aio.h>

int aio_read(struct aiocb *aiocbp);
```

aio_return

Retrieve return status of an asynchronous I/O operation (**REALTIME**)

AIO
```
#include <aio.h>

ssize_t aio_return(struct aiocb *aiocbp);
```

aio_suspend

Wait for an asynchronous I/O request (**REALTIME**)

AIO
```
#include <aio.h>

int aio_suspend(const struct aiocb *const list[], int nent,
    const struct timespec *timeout);
```

aio_write

Asynchronous write to a file (**REALTIME**)

AIO
```
#include <aio.h>

int aio_write(struct aiocb *aiocbp);
```

alarm

Schedule an alarm signal

```
#include <unistd.h>

unsigned alarm(unsigned seconds);
```

asctime, asctime_r

Convert date and time to a string

```
#include <time.h>

char *asctime(const struct tm *timeptr);
TSF char *asctime_r(const struct tm *restrict tm, char *restrict buf);
```

asin, asinf, asinl

Arc sine function

```
#include <math.h>

double asin(double x);
float asinf(float x);
long double asinl(long double x);
```

asinh, asinfh, asinfl

Inverse hyperbolic sine functions

```
#include <math.h>

double asinh(double x);
float asinhf(float x);
long double asinhl(long double x);
```

assert

Insert program diagnostics

```
#include <assert.h>

void assert(scalar expression);
```

atan, atanf, atanl

Arc tangent function

```
#include <math.h>

double atan(double x);
float atanf(float x);
long double atanl(long double x);
```

atan2, atan2f, atan2l

Arc tangent functions

```
#include <math.h>

double atan2(double y, double x);
float atan2f(float y, float x);
long double atan2l(long double y, long double x);
```

atanh, atanhf, atanhl

Inverse hyperbolic tangent functions

```
#include <math.h>

double atanh(double x);
float atanhf(float x);
long double atanhl(long double x);
```

atexit

Register a function to run at process termination

```
#include <stdlib.h>

int atexit(void (*func)(void));
```

atof

Convert a string to double-precision number

```
#include <stdlib.h>

double atof(const char *str);
```

atoi

Convert a string to an integer

```
#include <stdlib.h>

int atoi(const char *str);
```

atol, atoll

Convert a string to a long integer

```
#include <stdlib.h>

long atol(const char *str);
long long atoll(const char *nptr);
```

basename

Return the last component of a pathname

XSI

```
#include <libgen.h>

char *basename(char *path);
```

bcmp

Memory operations (**LEGACY**)

XSI
```
#include <strings.h>

int bcmp(const void *s1, const void *s2, size_t n);
```

bcopy

Memory operations (**LEGACY**)

XSI
```
#include <strings.h>

void bcopy(const void *s1, void *s2, size_t n);
```

bind

Bind a name to a socket

```
#include <sys/socket.h>

int bind(int socket, const struct sockaddr *address,
    socklen_t address_len);
```

bsd_signal

Simplified signal facilities

OB XSI
```
#include <signal.h>

void (*bsd_signal(int sig, void (*func)(int)))(int);
```

bsearch

Binary search a sorted table

```
#include <stdlib.h>

void *bsearch(const void *key, const void *base, size_t nel,
    size_t width, int (*compar)(const void *, const void *));
```

btowc

Single byte to wide character conversion

```
#include <stdio.h>
#include <wchar.h>

wint_t btowc(int c);
```

bzero

Memory operations (**LEGACY**)

XSI
```
#include <strings.h>

void bzero(void *s, size_t n);
```

cabs, cabsf, cabsl

Return a complex absolute value

```
#include <complex.h>

double cabs(double complex z);
float cabsf(float complex z);
long double cabsl(long double complex z);
```

cacos, cacosf, cacosl

Complex arc cosine functions

```
#include <complex.h>

double complex cacos(double complex z);
float complex cacosf(float complex z);
long double complex cacosl(long double complex z);
```

cacosh, cacoshf, cacoshl

Complex arc hyperbolic cosine functions

```
#include <complex.h>

double complex cacosh(double complex z);
float complex cacoshf(float complex z);
long double complex cacoshl(long double complex z);
```

calloc

A memory allocator

```
#include <stdlib.h>

void *calloc(size_t nelem, size_t elsize);
```

carg, cargf, cargl

Complex argument functions

```
#include <complex.h>

double carg(double complex z);
float cargf(float complex z);
long double cargl(long double complex z);
```

casin, casinf, casinl

Complex arc sine functions

```
#include <complex.h>

double complex casin(double complex z);
float complex casinf(float complex z);
long double complex casinl(long double complex z);
```

casinh, casinhf, casinhl

Complex arc hyperbolic sine functions

```
#include <complex.h>

double complex casinh(double complex z);
float complex casinhf(float complex z);
long double complex casinhl(long double complex z);
```

catan, catanf, catanl

Complex arc tangent functions

```
#include <complex.h>

double complex catan(double complex z);
float complex catanf(float complex z);
long double complex catanl(long double complex z);
```

catanh, catanhf, catanhl

Complex arc hyperbolic tangent functions

```
#include <complex.h>

double complex catanh(double complex z);
float complex catanhf(float complex z);
long double complex catanhl(long double complex z);
```

catclose

Close a message catalog descriptor

XSI

```
#include <nl_types.h>

int catclose(nl_catd catd);
```

catgets

Read a program message

XSI
```
#include <nl_types.h>

char *catgets(nl_catd catd, int set_id, int msg_id, const char *s);
```

catopen

Open a message catalog

XSI
```
#include <nl_types.h>

nl_catd catopen(const char *name, int oflag);
```

cbrt, cbrtf, cbrtl

Cube root functions

```
#include <math.h>

double cbrt(double x);
float cbrtf(float x);
long double cbrtl(long double x);
```

ccos, ccosf, ccosl

Complex cosine functions

```
#include <complex.h>

double complex ccos(double complex z);
float complex ccosf(float complex z);
long double complex ccosl(long double complex z);
```

ccosh, ccoshf, ccoshl

Complex hyperbolic cosine functions

```
#include <complex.h>

double complex ccosh(double complex z);
float complex ccoshf(float complex z);
long double complex ccoshl(long double complex z);
```

ceil, ceilf, ceill

Ceiling value function

```
#include <math.h>

double ceil(double x);
float ceilf(float x);
long double ceill(long double x);
```

cexp, cexpf, cexpl

Complex exponential functions

```
#include <complex.h>

double complex cexp(double complex z);
float complex cexpf(float complex z);
long double complex cexpl(long double complex z);
```

cfgetispeed

Get input baud rate

```
#include <termios.h>

speed_t cfgetispeed(const struct termios *termios_p);
```

cfgetospeed

Get output baud rate

```
#include <termios.h>

speed_t cfgetospeed(const struct termios *termios_p);
```

cfsetispeed

Set input baud rate

```
#include <termios.h>

int cfsetispeed(struct termios *termios_p, speed_t speed);
```

cfsetospeed

Set output baud rate

```
#include <termios.h>

int cfsetospeed(struct termios *termios_p, speed_t speed);
```

chdir

Change working directory

```
#include <unistd.h>

int chdir(const char *path);
```

chmod

Change mode of a file

```
#include <sys/stat.h>

int chmod(const char *path, mode_t mode);
```

chown

Change owner and group of a file

```
#include <unistd.h>

int chown(const char *path, uid_t owner, gid_t group);
```

cimag, cimagf, cimagl

Complex imaginary functions

```
#include <complex.h>

double cimag(double complex z);
float cimagf(float complex z);
long double cimagl(long double complex z);
```

clearerr

Clear indicators on a stream

```
#include <stdio.h>

void clearerr(FILE *stream);
```

clock

Report CPU time used

```
#include <time.h>

clock_t clock(void);
```

clock_getcpuclockid

Access a process CPU-time clock (**ADVANCED REALTIME**)

CPT

```
#include <time.h>

int clock_getcpuclockid(pid_t pid, clockid_t *clock_id);
```

clock_getres, clock_gettime, clock_settime

Clock and timer functions (**REALTIME**)

TMR
```
#include <time.h>

int clock_getres(clockid_t clock_id, struct timespec *res);
int clock_gettime(clockid_t clock_id, struct timespec *tp);
int clock_settime(clockid_t clock_id, const struct timespec *tp);
```

clock_nanosleep

High resolution sleep with specifiable clock (**ADVANCED REALTIME**)

CS
```
#include <time.h>

int clock_nanosleep(clockid_t clock_id, int flags,
    const struct timespec *rqtp, struct timespec *rmtp);
```

clog, clogf, clogl

Complex natural logarithm functions

```
#include <complex.h>

double complex clog(double complex z);
float complex clogf(float complex z);
long double complex clogl(long double complex z);
```

close

Close a file descriptor

```
#include <unistd.h>

int close(int fildes);
```

closedir

Close a directory stream

```
#include <dirent.h>

int closedir(DIR *dirp);
```

closelog, openlog, setlogmask, syslog

Control system log

XSI
```
#include <syslog.h>

void closelog(void);
void openlog(const char *ident, int logopt, int facility);
int setlogmask(int maskpri);
void syslog(int priority, const char *message, ... /* arguments */);
```

confstr

Get configurable variables

```
#include <unistd.h>

size_t confstr(int name, char *buf, size_t len);
```

conj, conjf, conjl

Complex conjugate functions

```
#include <complex.h>

double complex conj(double complex z);
float complex conjf(float complex z);
long double complex conjl(long double complex z);
```

connect

Connect a socket

```
#include <sys/socket.h>

int connect(int socket, const struct sockaddr *address,
    socklen_t address_len);
```

copysign, copysignf, copysignl

Number manipulation function

```
#include <math.h>

double copysign(double x, double y);
float copysignf(float x, float y);
long double copysignl(long double x, long double y);
```

cos, cosf, cosl

Cosine function

```
#include <math.h>

double cos(double x);
float cosf(float x);
long double cosl(long double x);
```

cosh, coshf, coshl

Hyperbolic cosine functions

```
#include <math.h>

double cosh(double x);
float coshf(float x);
long double coshl(long double x);
```

cpow, cpowf, cpowl

Complex power functions

```
#include <complex.h>

double complex cpow(double complex x, double complex y);
float complex cpowf(float complex x, float complex y);
long double complex cpowl(long double complex x,
    long double complex y);
```

cproj, cprojf, cprojl

Complex projection functions

```
#include <complex.h>

double complex cproj(double complex z);
float complex cprojf(float complex z);
long double complex cprojl(long double complex z);
```

creal, crealf, creall

Complex real functions

```
#include <complex.h>

double creal(double complex z);
float crealf(float complex z);
long double creall(long double complex z);
```

creat

Create a new file or rewrite an existing one

OH

```
#include <sys/stat.h>
#include <fcntl.h>

int creat(const char *path, mode_t mode);
```

crypt

String encoding function (**CRYPT**)

XSI

```
#include <unistd.h>

char *crypt(const char *key, const char *salt);
```

csin, csinf, csinl

Complex sine functions

```
#include <complex.h>

double complex csin(double complex z);
float complex csinf(float complex z);
long double complex csinl(long double complex z);
```

csinh, csinhf, csinhl

Complex hyperbolic sine functions

```
#include <complex.h>

double complex csinh(double complex z);
float complex csinhf(float complex z);
long double complex csinhl(long double complex z);
```

csqrt, csqrtf, csqrtl

Complex square root functions

```
#include <complex.h>

double complex csqrt(double complex z);
float complex csqrtf(float complex z);
long double complex csqrtl(long double complex z);
```

ctan, ctanf, ctanl

Complex tangent functions

```
#include <complex.h>

double complex ctan(double complex z);
float complex ctanf(float complex z);
long double complex ctanl(long double complex z);
```

ctanh, ctanhf, ctanhl

Complex hyperbolic tangent functions

```
#include <complex.h>

double complex ctanh(double complex z);
float complex ctanhf(float complex z);
long double complex ctanhl(long double complex z);
```

ctermid

Generate a pathname for controlling terminal

CX
```
#include <stdio.h>

char *ctermid(char *s);
```

ctime, ctime_r

Convert a time value to date and time string

```
#include <time.h>

char *ctime(const time_t *clock);
```
TSF
```
char *ctime_r(const time_t *clock, char *buf);
```

dbm_clearerr, dbm_close, dbm_delete, dbm_error, dbm_fetch, dbm_firstkey, dbm_nextkey, dbm_open, dbm_store

Database functions

XSI
```
#include <ndbm.h>

int dbm_clearerr(DBM *db);
void dbm_close(DBM *db);
int dbm_delete(DBM *db, datum key);
int dbm_error(DBM *db);
datum dbm_fetch(DBM *db, datum key);
datum dbm_firstkey(DBM *db);
datum dbm_nextkey(DBM *db);
DBM *dbm_open(const char *file, int open_flags, mode_t file_mode);
int dbm_store(DBM *db, datum key, datum content, int store_mode);
```

difftime

Compute the difference between two calendar time values

```
#include <time.h>

double difftime(time_t time1, time_t time0);
```

dirname

Report the parent directory name of a file pathname

XSI
```
#include <libgen.h>

char *dirname(char *path);
```

div

Compute the quotient and remainder of an integer division

```
#include <stdlib.h>

div_t div(int numer, int denom);
```

dlclose

Close a *dlopen*( ) object

XSI

```
#include <dlfcn.h>

int dlclose(void *handle);
```

dlerror

Get diagnostic information

XSI

```
#include <dlfcn.h>

char *dlerror(void);
```

dlopen

Gain access to an executable object file

XSI

```
#include <dlfcn.h>

void *dlopen(const char *file, int mode);
```

dlsym

Obtain the address of a symbol from a *dlopen*( ) object

XSI

```
#include <dlfcn.h>

void *dlsym(void *restrict handle, const char *restrict name);
```

drand48, erand48, jrand48, lcong48, lrand48, mrand48, nrand48, seed48, srand48

Generate uniformly distributed pseudo-random numbers

XSI

```
#include <stdlib.h>

double drand48(void);
double erand48(unsigned short xsubi[3]);
long jrand48(unsigned short xsubi[3]);
void lcong48(unsigned short param[7]);
long lrand48(void);
long mrand48(void);
long nrand48(unsigned short xsubi[3]);
unsigned short *seed48(unsigned short seed16v[3]);
void srand48(long seedval);
```

dup, dup2

Duplicate an open file descriptor

```
#include <unistd.h>

int dup(int fildes);
int dup2(int fildes, int fildes2);
```

ecvt, fcvt, gcvt

Convert a floating-point number to a string (**LEGACY**)

XSI

```
#include <stdlib.h>

char *ecvt(double value, int ndigit, int *restrict decpt,
    int *restrict sign);
char *fcvt(double value, int ndigit, int *restrict decpt,
    int *restrict sign);
char *gcvt(double value, int ndigit, char *buf);
```

encrypt

Encoding function (**CRYPT**)

XSI

```
#include <unistd.h>

void encrypt(char block[64], int edflag);
```

endgrent, getgrent, setgrent

Group database entry functions

XSI

```
#include <grp.h>

void endgrent(void);
struct group *getgrent(void);
void setgrent(void);
```

endhostent, gethostent, sethostent

Network host database functions

```
#include <netdb.h>

void endhostent(void);
struct hostent *gethostent(void);
void sethostent(int stayopen);
```

endnetent, getnetbyaddr, getnetbyname, getnetent, setnetent

Network database functions

```
#include <netdb.h>

void endnetent(void);
struct netent *getnetbyaddr(uint32_t net, int type);
struct netent *getnetbyname(const char *name);
struct netent *getnetent(void);
void setnetent(int stayopen);
```

endprotoent, getprotobyname, getprotobynumber, getprotoent, setprotoent

Network protocol database functions

```
#include <netdb.h>

void endprotoent(void);
struct protoent *getprotobyname(const char *name);
struct protoent *getprotobynumber(int proto);
struct protoent *getprotoent(void);
void setprotoent(int stayopen);
```

endpwent, getpwent, setpwent

User database functions

XSI

```
#include <pwd.h>

void endpwent(void);
struct passwd *getpwent(void);
void setpwent(void);
```

endservent, getservbyname, getservbyport, getservent, setservent

Network services database functions

```
#include <netdb.h>

void endservent(void);
struct servent *getservbyname(const char *name, const char *proto);
struct servent *getservbyport(int port, const char *proto);
struct servent *getservent(void);
void setservent(int stayopen);
```

endutxent, getutxent, getutxid, getutxline, pututxline, setutxent

User accounting database functions

XSI

```
#include <utmpx.h>

void endutxent(void);
struct utmpx *getutxent(void);
struct utmpx *getutxid(const struct utmpx *id);
struct utmpx *getutxline(const struct utmpx *line);
struct utmpx *pututxline(const struct utmpx *utmpx);
```

```
void setutxent(void);
```

erf, erff, erfl

Error functions

```
#include <math.h>

double erf(double x);
float erff(float x);
long double erfl(long double x);
```

erfc, erfcf, erfcl

Complementary error functions

```
#include <math.h>

double erfc(double x);
float erfcf(float x);
long double erfcl(long double x);
```

errno

Error return value

```
#include <errno.h>
```

environ, execl, execv, execle, execve, execlp, execvp

Execute a file

```
#include <unistd.h>

extern char **environ;
int execl(const char *path, const char *arg0, ... /*, (char *)0 */);
int execv(const char *path, char *const argv[]);
int execle(const char *path, const char *arg0, ... /*,
    (char *)0, char *const envp[]*/);
int execve(const char *path, char *const argv[], char *const envp[]);
int execlp(const char *file, const char *arg0, ... /*, (char *)0 */);
int execvp(const char *file, char *const argv[]);
```

exit, _Exit, _exit

Terminate a process

```
#include <stdlib.h>

void exit(int status);
void _Exit(int status);

#include <unistd.h>

void _exit(int status);
```

exp, expf, expl

Exponential function

```
#include <math.h>

double exp(double x);
float expf(float x);
long double expl(long double x);
```

exp2, exp2f, exp2l

Exponential base 2 functions

```
#include <math.h>

double exp2(double x);
float exp2f(float x);
long double exp2l(long double x);
```

expm1, expm1f, expm1l

Compute exponential functions

```
#include <math.h>

double expm1(double x);
float expm1f(float x);
long double expm1l(long double x);
```

fabs, fabsf, fabsl

Absolute value function

```
#include <math.h>

double fabs(double x);
float fabsf(float x);
long double fabsl(long double x);
```

fattach

Attach a STREAMS-based file descriptor to a file in the file system name space (**STREAMS**)

XSR

```
#include <stropts.h>

int fattach(int fildes, const char *path);
```

fchdir

Change working directory

XSI
```
#include <unistd.h>

int fchdir(int fildes);
```

fchmod

Change mode of a file

```
#include <sys/stat.h>

int fchmod(int fildes, mode_t mode);
```

fchown

Change owner and group of a file

```
#include <unistd.h>

int fchown(int fildes, uid_t owner, gid_t group);
```

fclose

Close a stream

```
#include <stdio.h>

int fclose(FILE *stream);
```

fcntl

File control

OH
```
#include <unistd.h>
#include <fcntl.h>

int fcntl(int fildes, int cmd, ...);
```

fdatasync

Synchronize the data of a file (**REALTIME**)

SIO
```
#include <unistd.h>

int fdatasync(int fildes);
```

fdetach

Detach a name from a STREAMS-based file descriptor (**STREAMS**)

XSR
```
#include <stropts.h>

int fdetach(const char *path);
```

fdim, fdimf, fdiml

Compute positive difference between two floating-point numbers

```
#include <math.h>

double fdim(double x, double y);
float fdimf(float x, float y);
long double fdiml(long double x, long double y);
```

fdopen

Associate a stream with a file descriptor

CX
```
#include <stdio.h>

FILE *fdopen(int fildes, const char *mode);
```

feclearexcept

Clear floating-point exception

```
#include <fenv.h>

int feclearexcept(int excepts);
```

fegetenv, fesetenv

Get and set current floating-point environment

```
#include <fenv.h>

int fegetenv(fenv_t *envp);
int fesetenv(const fenv_t *envp);
```

fegetexceptflag, fesetexceptflag

Get and set floating-point status flags

```
#include <fenv.h>

int fegetexceptflag(fexcept_t *flagp, int excepts);
int fesetexceptflag(const fexcept_t *flagp, int excepts);
```

fegetround, fesetround

Get and set current rounding direction

```
#include <fenv.h>

int fegetround(void);
int fesetround(int round);
```

feholdexcept

Save current floating-point environment

```
#include <fenv.h>

int feholdexcept(fenv_t *envp);
```

feof

Test end-of-file indicator on a stream

```
#include <stdio.h>

int feof(FILE *stream);
```

feraiseexcept

Raise floating-point exception

```
#include <fenv.h>

int feraiseexcept(int excepts);
```

ferror

Test error indicator on a stream

```
#include <stdio.h>

int ferror(FILE *stream);
```

fetestexcept

Test floating-point exception flags

```
#include <fenv.h>

int fetestexcept(int excepts);
```

feupdateenv

Update floating-point environment

```
#include <fenv.h>

int feupdateenv(const fenv_t *envp);
```

fflush

Flush a stream

```
#include <stdio.h>
int fflush(FILE *stream);
```

ffs

Find first set bit

XSI

```
#include <strings.h>
int ffs(int i);
```

fgetc

Get a byte from a stream

```
#include <stdio.h>
int fgetc(FILE *stream);
```

fgetpos

Get current file position information

```
#include <stdio.h>
int fgetpos(FILE *restrict stream, fpos_t *restrict pos);
```

fgets

Get a string from a stream

```
#include <stdio.h>
char *fgets(char *restrict s, int n, FILE *restrict stream);
```

fgetwc

Get a wide-character code from a stream

```
#include <stdio.h>
#include <wchar.h>
wint_t fgetwc(FILE *stream);
```

fgetws

Get a wide-character string from a stream

```
#include <stdio.h>
#include <wchar.h>

wchar_t *fgetws(wchar_t *restrict ws, int n,
    FILE *restrict stream);
```

fileno

Map a stream pointer to a file descriptor

CX

```
#include <stdio.h>

int fileno(FILE *stream);
```

flockfile, ftrylockfile, funlockfile

Stdio locking functions

TSF

```
#include <stdio.h>

void flockfile(FILE *file);
int ftrylockfile(FILE *file);
void funlockfile(FILE *file);
```

floor, floorf, floorl

Floor function

```
#include <math.h>

double floor(double x);
float floorf(float x);
long double floorl(long double x);
```

fma, fmaf, fmal

Floating-point multiply-add

```
#include <math.h>

double fma(double x, double y, double z);
float fmaf(float x, float y, float z);
long double fmal(long double x, long double y, long double z);
```

fmax, fmaxf, fmaxl

Determine maximum numeric value of two floating-point numbers

```
#include <math.h>

double fmax(double x, double y);
float fmaxf(float x, float y);
long double fmaxl(long double x, long double y);
```

fmin, fminf, fminl

Determine minimum numeric value of two floating-point numbers

```
#include <math.h>

double fmin(double x, double y);
float fminf(float x, float y);
long double fminl(long double x, long double y);
```

fmod, fmodf, fmodl

Floating-point remainder value function

```
#include <math.h>

double fmod(double x, double y);
float fmodf(float x, float y);
long double fmodl(long double x, long double y);
```

fmtmsg

Display a message in the specified format on standard error and/or a system console

XSI

```
#include <fmtmsg.h>

int fmtmsg(long classification, const char *label, int severity,
    const char *text, const char *action, const char *tag);
```

fnmatch

Match a filename or a pathname

```
#include <fnmatch.h>

int fnmatch(const char *pattern, const char *string, int flags);
```

fopen

Open a stream

```
#include <stdio.h>

FILE *fopen(const char *restrict filename, const char *restrict mode);
```

fork

Create a new process

```
#include <unistd.h>

pid_t fork(void);
```

fpathconf, pathconf

Get configurable pathname variables

```
#include <unistd.h>

long fpathconf(int fildes, int name);
long pathconf(const char *path, int name);
```

fpclassify

Classify real floating type

```
#include <math.h>

int fpclassify(real-floating x);
```

fprintf, printf, snprintf, sprintf

Print formatted output

```
#include <stdio.h>

int fprintf(FILE *restrict stream, const char *restrict format, ...);
int printf(const char *restrict format, ...);
int snprintf(char *restrict s, size_t n,
    const char *restrict format, ...);
int sprintf(char *restrict s, const char *restrict format, ...);
```

fputc

Put a byte on a stream

```
#include <stdio.h>

int fputc(int c, FILE *stream);
```

fputs

Put a string on a stream

```
#include <stdio.h>

int fputs(const char *restrict s, FILE *restrict stream);
```

fputwc

Put a wide-character code on a stream

```
#include <stdio.h>
#include <wchar.h>

wint_t fputwc(wchar_t wc, FILE *stream);
```

fputws

Put a wide-character string on a stream

```
#include <stdio.h>
#include <wchar.h>

int fputws(const wchar_t *restrict ws, FILE *restrict stream);
```

fread

Binary input

```
#include <stdio.h>

size_t fread(void *restrict ptr, size_t size, size_t nitems,
    FILE *restrict stream);
```

free

Free allocated memory

```
#include <stdlib.h>

void free(void *ptr);
```

freeaddrinfo, getaddrinfo

Get address information

```
#include <sys/socket.h>
#include <netdb.h>

void freeaddrinfo(struct addrinfo *ai);
int getaddrinfo(const char *restrict nodename,
    const char *restrict servname,
    const struct addrinfo *restrict hints,
    struct addrinfo **restrict res);
```

freopen

Open a stream

```
#include <stdio.h>

FILE *freopen(const char *restrict filename, const char *restrict mode,
    FILE *restrict stream);
```

frexp, frexpf, frexpl

Extract mantissa and exponent from a double precision number

```
#include <math.h>

double frexp(double num, int *exp);
float frexpf(float num, int *exp);
long double frexpl(long double num, int *exp);
```

fscanf, scanf, sscanf

Convert formatted input

```
#include <stdio.h>

int fscanf(FILE *restrict stream, const char *restrict format, ... );
int scanf(const char *restrict format, ... );
int sscanf(const char *restrict s, const char *restrict format, ... );
```

fseek, fseeko

Reposition a file-position indicator in a stream

```
#include <stdio.h>

int fseek(FILE *stream, long offset, int whence);
```
CX
```
int fseeko(FILE *stream, off_t offset, int whence);
```

fsetpos

Set current file position

```
#include <stdio.h>

int fsetpos(FILE *stream, const fpos_t *pos);
```

fstat

Get file status

```
#include <sys/stat.h>

int fstat(int fildes, struct stat *buf);
```

fstatvfs, statvfs

Get file system information

XSI
```
#include <sys/statvfs.h>

int fstatvfs(int fildes, struct statvfs *buf);
int statvfs(const char *restrict path, struct statvfs *restrict buf);
```

fsync

Synchronize changes to a file

FSC
```
#include <unistd.h>

int fsync(int fildes);
```

ftell, ftello

Return a file offset in a stream

```
#include <stdio.h>

long ftell(FILE *stream);
```
CX
```
off_t ftello(FILE *stream);
```

ftime

Get date and time (**LEGACY**)

XSI
```
#include <sys/timeb.h>

int ftime(struct timeb *tp);
```

ftok

Generate an IPC key

XSI
```
#include <sys/ipc.h>

key_t ftok(const char *path, int id);
```

ftruncate

Truncate a file to a specified length

```
#include <unistd.h>

int ftruncate(int fildes, off_t length);
```

ftw

Traverse (walk) a file tree

XSI
```
#include <ftw.h>

int ftw(const char *path, int (*fn)(const char *,
    const struct stat *ptr, int flag), int ndirs);
```

fwide

Set stream orientation

```
#include <stdio.h>
#include <wchar.h>

int fwide(FILE *stream, int mode);
```

fwprintf, swprintf, wprintf

Print formatted wide-character output

```
#include <stdio.h>
#include <wchar.h>

int fwprintf(FILE *restrict stream, const wchar_t *restrict format, ...);
int swprintf(wchar_t *restrict ws, size_t n,
    const wchar_t *restrict format, ...);
int wprintf(const wchar_t *restrict format, ...);
```

fwrite

Binary output

```
#include <stdio.h>

size_t fwrite(const void *restrict ptr, size_t size, size_t nitems,
    FILE *restrict stream);
```

fwscanf, swscanf, wscanf

Convert formatted wide-character input

```
#include <stdio.h>
#include <wchar.h>

int fwscanf(FILE *restrict stream, const wchar_t *restrict format, ... );
int swscanf(const wchar_t *restrict ws,
    const wchar_t *restrict format, ... );
int wscanf(const wchar_t *restrict format, ... );
```

gai_strerror

Address and name information error description

```
#include <netdb.h>

const char *gai_strerror(int ecode);
```

getc

Get a byte from a stream

```
#include <stdio.h>

int getc(FILE *stream);
```

getc_unlocked, getchar_unlocked, putc_unlocked, putchar_unlocked

Stdio with explicit client locking

TSF

```
#include <stdio.h>

int getc_unlocked(FILE *stream);
int getchar_unlocked(void);
int putc_unlocked(int c, FILE *stream);
int putchar_unlocked(int c);
```

getchar

Get a byte from a stdin stream

```
#include <stdio.h>

int getchar(void);
```

getcontext, setcontext

Get and set current user context

XSI

```
#include <ucontext.h>

int getcontext(ucontext_t *ucp);
int setcontext(const ucontext_t *ucp);
```

getcwd

Get the pathname of the current working directory

```
#include <unistd.h>

char *getcwd(char *buf, size_t size);
```

getdate

Convert user format date and time

XSI

```
#include <time.h>

struct tm *getdate(const char *string);
```

getegid

Get the effective group ID

```
#include <unistd.h>

gid_t getegid(void);
```

getenv

Get value of an environment variable

```
#include <stdlib.h>

char *getenv(const char *name);
```

geteuid

Get the effective user ID

```
#include <unistd.h>

uid_t geteuid(void);
```

getgid

Get the real group ID

```
#include <unistd.h>

gid_t getgid(void);
```

getgrgid, getgrgid_r

Get group database entry for a group ID

```
#include <grp.h>

struct group *getgrgid(gid_t gid);
TSF int getgrgid_r(gid_t gid, struct group *grp, char *buffer,
        size_t bufsize, struct group **result);
```

getgrnam, getgrnam_r

```
Search group database for a name
#include <grp.h>

struct group *getgrnam(const char *name);
TSF int getgrnam_r(const char *name, struct group *grp, char *buffer,
        size_t bufsize, struct group **result);
```

getgroups

Get supplementary group IDs

```
#include <unistd.h>

int getgroups(int gidsetsize, gid_t grouplist[]);
```

gethostbyaddr, gethostbyname

Network host database functions

```
#include <netdb.h>
```

OB

```
struct hostent *gethostbyaddr(const void *addr, socklen_t len,
    int type);
struct hostent *gethostbyname(const char *name);
```

gethostid

Get an identifier for the current host

XSI

```
#include <unistd.h>

long gethostid(void);
```

gethostname

Get name of current host

```
#include <unistd.h>

int gethostname(char *name, size_t namelen);
```

getitimer, setitimer

Get and set value of interval timer

XSI

```
#include <sys/time.h>

int getitimer(int which, struct itimerval *value);
int setitimer(int which, const struct itimerval *restrict value,
    struct itimerval *restrict ovalue);
```

getlogin, getlogin_r

Get login name

```
#include <unistd.h>

char *getlogin(void);
```

TSF

```
int getlogin_r(char *name, size_t namesize);
```

getmsg, getpmsg

Receive next message from a STREAMS file (**STREAMS**)

XSR
```
#include <stropts.h>

int getmsg(int fildes, struct strbuf *restrict ctlptr,
    struct strbuf *restrict dataptr, int *restrict flagsp);
int getpmsg(int fildes, struct strbuf *restrict ctlptr,
    struct strbuf *restrict dataptr, int *restrict bandp,
    int *restrict flagsp);
```

getnameinfo

Get name information

```
#include <sys/socket.h>
#include <netdb.h>

int getnameinfo(const struct sockaddr *restrict sa, socklen_t salen,    char *restri
    socklen_t servicelen, int flags);
```

getopt, optarg, opterr, optind, optopt

Command option parsing

```
#include <unistd.h>

int getopt(int argc, char *const argv[], const char *optstring);
extern char *optarg;
extern int optind, opterr, optopt;
```

getpeername

Get the name of the peer socket

```
#include <sys/socket.h>

int getpeername(int socket, struct sockaddr *restrict address,
    socklen_t *restrict address_len);
```

getpgid

Get the process group ID for a process

XSI
```
#include <unistd.h>

pid_t getpgid(pid_t pid);
```

getpgrp

Get the process group ID of the calling process

```
#include <unistd.h>

pid_t getpgrp(void);
```

getpid

Get the process ID

```
#include <unistd.h>

pid_t getpid(void);
```

getppid

Get the parent process ID

```
#include <unistd.h>

pid_t getppid(void);
```

getpriority, setpriority

Get and set the nice value

XSI
```
#include <sys/resource.h>

int getpriority(int which, id_t who);
int setpriority(int which, id_t who, int value);
```

getpwnam, getpwnam_r

Search user database for a name

```
#include <pwd.h>

struct passwd *getpwnam(const char *name);
```

TSF
```
int getpwnam_r(const char *name, struct passwd *pwd, char *buffer,
    size_t bufsize, struct passwd **result);
```

getpwuid, getpwuid_r

Search user database for a user ID

```
#include <pwd.h>

struct passwd *getpwuid(uid_t uid);
```

TSF
```
int getpwuid_r(uid_t uid, struct passwd *pwd, char *buffer,
    size_t bufsize, struct passwd **result);
```

getrlimit, setrlimit

Control maximum resource consumption

XSI
```
#include <sys/resource.h>

int getrlimit(int resource, struct rlimit *rlp);
int setrlimit(int resource, const struct rlimit *rlp);
```

getrusage

Get information about resource utilization

XSI
```
#include <sys/resource.h>

int getrusage(int who, struct rusage *r_usage);
```

gets

Get a string from a stdin stream

```
#include <stdio.h>

char *gets(char *s);
```

getsid

Get the process group ID of a session leader

XSI
```
#include <unistd.h>

pid_t getsid(pid_t pid);
```

getsockname

Get the socket name

```
#include <sys/socket.h>

int getsockname(int socket, struct sockaddr *restrict address,
    socklen_t *restrict address_len);
```

getsockopt

Get the socket options

```
#include <sys/socket.h>

int getsockopt(int socket, int level, int option_name,
    void *restrict option_value, socklen_t *restrict option_len);
```

getsubopt

Parse suboption arguments from a string

XSI
```
#include <stdlib.h>

int getsubopt(char **optionp, char *const *tokens, char **valuep);
```

gettimeofday

Get the date and time

XSI
```
#include <sys/time.h>

int gettimeofday(struct timeval *restrict tp, void *restrict tzp);
```

getuid

Get a real user ID

```
#include <unistd.h>

uid_t getuid(void);
```

getwc

Get a wide character from a stream

```
#include <stdio.h>
#include <wchar.h>

wint_t getwc(FILE *stream);
```

getwchar

Get a wide character from a stdin stream

```
#include <wchar.h>

wint_t getwchar(void);
```

getwd

Get the current working directory pathname (**LEGACY**)

XSI
```
#include <unistd.h>

char *getwd(char *path_name);
```

glob, globfree

Generate pathnames matching a pattern

```
#include <glob.h>

int glob(const char *restrict pattern, int flags,
    int(*errfunc)(const char *epath, int eerrno),
    glob_t *restrict pglob);
void globfree(glob_t *pglob);
```

gmtime, gmtime_r

Convert a time value to a broken-down UTC time

```
#include <time.h>

struct tm *gmtime(const time_t *timer);
```
TSF
```
struct tm *gmtime_r(const time_t *restrict timer,
    struct tm *restrict result);
```

grantpt

Grant access to the slave pseudo-terminal device

XSI
```
#include <stdlib.h>

int grantpt(int fildes);
```

h_errno

Error return value for network database operations

OB
```
#include <netdb.h>
```

hcreate, hdestroy, hsearch

Manage hash search table

XSI
```
#include <search.h>

int hcreate(size_t nel);
void hdestroy(void);
ENTRY *hsearch(ENTRY item, ACTION action);
```

htonl, htons, ntohl, ntohs

Convert values between host and network byte order

```
#include <arpa/inet.h>

uint32_t htonl(uint32_t hostlong);
uint16_t htons(uint16_t hostshort);
uint32_t ntohl(uint32_t netlong);
uint16_t ntohs(uint16_t netshort);
```

hypot, hypotf, hypotl

Euclidean distance function

```
#include <math.h>

double hypot(double x, double y);
float hypotf(float x, float y);
long double hypotl(long double x, long double y);
```

iconv

Codeset conversion function

XSI

```
#include <iconv.h>

size_t iconv(iconv_t cd, char **restrict inbuf,
    size_t *restrict inbytesleft, char **restrict outbuf,
    size_t *restrict outbytesleft);
```

iconv_close

Codeset conversion deallocation function

XSI

```
#include <iconv.h>

int iconv_close(iconv_t cd);
```

iconv_open

Codeset conversion allocation function

XSI

```
#include <iconv.h>

iconv_t iconv_open(const char *tocode, const char *fromcode);
```

if_freenameindex

Free memory allocated by *if_nameindex*( )

```
#include <net/if.h>

void if_freenameindex(struct if_nameindex *ptr);
```

if_indextoname

Map a network interface index to its corresponding name

```
#include <net/if.h>

char *if_indextoname(unsigned ifindex, char *ifname);
```

if_nameindex

Return all network interface names and indexes

```
#include <net/if.h>

struct if_nameindex *if_nameindex(void);
```

if_nametoindex

Map a network interface name to its corresponding index

```
#include <net/if.h>

unsigned if_nametoindex(const char *ifname);
```

ilogb, ilogbf, ilogbl

Return an unbiased exponent

```
#include <math.h>

int ilogb(double x);
int ilogbf(float x);
int ilogbl(long double x);
```

imaxabs

Return absolute value

```
#include <inttypes.h>

intmax_t imaxabs(intmax_t j);
```

imaxdiv

Return quotient and remainder

```
#include <inttypes.h>

imaxdiv_t imaxdiv(intmax_t numer, intmax_t denom);
```

index

Character string operations (**LEGACY**)

XSI

```
#include <strings.h>

char *index(const char *s, int c);
```

inet_addr, inet_ntoa

IPv4 address manipulation

```
#include <arpa/inet.h>

in_addr_t inet_addr(const char *cp);
char *inet_ntoa(struct in_addr in);
```

inet_ntop, inet_pton

Convert IPv4 and IPv6 addresses between binary and text form

```
#include <arpa/inet.h>

const char *inet_ntop(int af, const void *restrict src,
    char *restrict dst, socklen_t size);
int inet_pton(int af, const char *restrict src, void *restrict dst);
```

initstate, random, setstate, srandom

Pseudo-random number functions

XSI

```
#include <stdlib.h>

char *initstate(unsigned seed, char *state, size_t size);
long random(void);
char *setstate(const char *state);
void srandom(unsigned seed);
```

insque, remque

Insert or remove an element in a queue

XSI

```
#include <search.h>

void insque(void *element, void *pred);
void remque(void *element);
```

ioctl

Control a STREAMS device (**STREAMS**)

XSR
```
#include <stropts.h>
int ioctl(int fildes, int request, ... /* arg */);
```

isalnum

Test for an alphanumeric character

```
#include <ctype.h>
int isalnum(int c);
```

isalpha

Test for an alphabetic character

```
#include <ctype.h>
int isalpha(int c);
```

isascii

Test for a 7-bit US-ASCII character

XSI
```
#include <ctype.h>
int isascii(int c);
```

isastream

Test a file descriptor (**STREAMS**)

XSR
```
#include <stropts.h>
int isastream(int fildes);
```

isatty

Test for a terminal device

```
#include <unistd.h>
int isatty(int fildes);
```

isblank

Test for a blank character

```
#include <ctype.h>
int isblank(int c);
```

iscntrl

Test for a control character

```
#include <ctype.h>
int iscntrl(int c);
```

isdigit

Test for a decimal digit

```
#include <ctype.h>
int isdigit(int c);
```

isfinite

Test for finite value

```
#include <math.h>
int isfinite(real-floating x);
```

isgraph

Test for a visible character

```
#include <ctype.h>
int isgraph(int c);
```

isgreater

Test if x greater than y

```
#include <math.h>
int isgreater(real-floating x, real-floating y);
```

isgreaterequal

Test if x greater than or equal to y

```
#include <math.h>
int isgreaterequal(real-floating x, real-floating y);
```

isinf

Test for infinity

```
#include <math.h>
int isinf(real-floating x);
```

isless

Test if x is less than y

```
#include <math.h>
int isless(real-floating x, real-floating y);
```

islessequal

Test if x is less than or equal to y

```
#include <math.h>
int islessequal(real-floating x, real-floating y);
```

islessgreater

Test if x is less than or greater than y

```
#include <math.h>
int islessgreater(real-floating x, real-floating y);
```

islower

Test for a lowercase letter

```
#include <ctype.h>
int islower(int c);
```

isnan

Test for a NaN

```
#include <math.h>
int isnan(real-floating x);
```

isnormal

Test for a normal value

```
#include <math.h>
int isnormal(real-floating x);
```

isprint

Test for a printable character

```
#include <ctype.h>
int isprint(int c);
```

ispunct

Test for a punctuation character

```
#include <ctype.h>
int ispunct(int c);
```

isspace

Test for a white-space character

```
#include <ctype.h>
int isspace(int c);
```

isunordered

Test if arguments are unordered

```
#include <math.h>
int isunordered(real-floating x, real-floating y);
```

isupper

Test for an uppercase letter

```
#include <ctype.h>
int isupper(int c);
```

iswalnum

Test for an alphanumeric wide-character code

```
#include <wctype.h>
int iswalnum(wint_t wc);
```

iswalpha

Test for an alphabetic wide-character code

```
#include <wctype.h>
int iswalpha(wint_t wc);
```

iswblank

Test for a blank wide-character code

```
#include <wctype.h>
int iswblank(wint_t wc);
```

iswcntrl

Test for a control wide-character code

```
#include <wctype.h>
int iswcntrl(wint_t wc);
```

iswctype

Test character for a specified class

```
#include <wctype.h>
int iswctype(wint_t wc, wctype_t charclass);
```

iswdigit

Test for a decimal digit wide-character code

```
#include <wctype.h>
int iswdigit(wint_t wc);
```

iswgraph

Test for a visible wide-character code

```
#include <wctype.h>
int iswgraph(wint_t wc);
```

iswlower

Test for a lowercase letter wide-character code

```
#include <wctype.h>
int iswlower(wint_t wc);
```

iswprint

Test for a printable wide-character code

```
#include <wctype.h>
int iswprint(wint_t wc);
```

iswpunct

Test for a punctuation wide-character code

```
#include <wctype.h>
int iswpunct(wint_t wc);
```

iswspace

Test for a white-space wide-character code

```
#include <wctype.h>
int iswspace(wint_t wc);
```

iswupper

Test for an uppercase letter wide-character code

```
#include <wctype.h>
int iswupper(wint_t wc);
```

iswxdigit

Test for a hexadecimal digit wide-character code

```
#include <wctype.h>
int iswxdigit(wint_t wc);
```

isxdigit

Test for a hexadecimal digit

```
#include <ctype.h>
int isxdigit(int c);
```

j0, j1, jn

Bessel functions of the first kind

XSI

```
#include <math.h>

double j0(double x);
double j1(double x);
double jn(int n, double x);
```

kill

Send a signal to a process or a group of processes

CX
```
#include <signal.h>

int kill(pid_t pid, int sig);
```

killpg

Send a signal to a process group

XSI
```
#include <signal.h>

int killpg(pid_t pgrp, int sig);
```

labs, llabs

Return a long integer absolute value

```
#include <stdlib.h>

long labs(long i);
long long llabs(long long i);
```

lchown

Change the owner and group of a symbolic link

XSI
```
#include <unistd.h>

int lchown(const char *path, uid_t owner, gid_t group);
```

ldexp, ldexpf, ldexpl

Load exponent of a floating-point number

```
#include <math.h>

double ldexp(double x, int exp);
float ldexpf(float x, int exp);
long double ldexpl(long double x, int exp);
```

ldiv, lldiv

Compute quotient and remainder of a long division

```
#include <stdlib.h>

ldiv_t ldiv(long numer, long denom);
lldiv_t lldiv(long long numer, long long denom);
```

lgamma, lgammaf, lgammal

Log gamma function

```
#include <math.h>

double lgamma(double x);
float lgammaf(float x);
long double lgammal(long double x);
XSI extern int signgam;
```

link

Link to a file

```
#include <unistd.h>

int link(const char *path1, const char *path2);
```

lio_listio

List directed I/O (**REALTIME**)

```
AIO #include <aio.h>

int lio_listio(int mode, struct aiocb *restrict const list[restrict],
    int nent, struct sigevent *restrict sig);
```

listen

Listen for socket connections and limit the queue of incoming connections

```
#include <sys/socket.h>

int listen(int socket, int backlog);
```

llrint, llrintf, llrintl

Round to nearest integer value using current rounding direction

```
#include <math.h>

long long llrint(double x);
long long llrintf(float x);
long long llrintl(long double x);
```

llround, llroundf, llroundl

Round to nearest integer value

```
#include <math.h>

long long llround(double x);
long long llroundf(float x);
long long llroundl(long double x);
```

localeconv

Return locale-specific information

```
#include <locale.h>

struct lconv *localeconv(void);
```

localtime, localtime_r

Convert a time value to a broken-down local time

```
#include <time.h>

struct tm *localtime(const time_t *timer);
```
TSF
```
struct tm *localtime_r(const time_t *restrict timer,
    struct tm *restrict result);
```

lockf

Record locking on files

XSI
```
#include <unistd.h>

int lockf(int fildes, int function, off_t size);
```

log, logf, logl

Natural logarithm function

```
#include <math.h>

double log(double x);
float logf(float x);
long double logl(long double x);
```

log10, log10f, log10l

Base 10 logarithm function

```
#include <math.h>

double log10(double x);
float log10f(float x);
long double log10l(long double x);
```

log1p, log1pf, log1pl

Compute a natural logarithm

```
#include <math.h>

double log1p(double x);
float log1pf(float x);
long double log1pl(long double x);
```

log2, log2f, log2l

Compute base 2 logarithm functions

```
#include <math.h>

double log2(double x);
float log2f(float x);
long double log2l(long double x);
```

logb, logbf, logbl

Radix-independent exponent

```
#include <math.h>

double logb(double x);
float logbf(float x);
long double logbl(long double x);
```

longjmp

Non-local goto

```
#include <setjmp.h>

void longjmp(jmp_buf env, int val);
```

lrint, lrintf, lrintl

Round to nearest integer value using current rounding direction

```
#include <math.h>

long lrint(double x);
long lrintf(float x);
long lrintl(long double x);
```

lround, lroundf, lroundl

Round to nearest integer value

```
#include <math.h>

long lround(double x);
long lroundf(float x);
long lroundl(long double x);
```

lsearch, lfind

Linear search and update

XSI

```
#include <search.h>

void *lsearch(const void *key, void *base, size_t *nelp, size_t width,
    int (*compar)(const void *, const void *));
void *lfind(const void *key, const void *base, size_t *nelp,
    size_t width, int (*compar)(const void *, const void *));
```

lseek

Move the read/write file offset

```
#include <unistd.h>

off_t lseek(int fildes, off_t offset, int whence);
```

lstat

Get symbolic link status

```
#include <sys/stat.h>

int lstat(const char *restrict path, struct stat *restrict buf);
```

makecontext, swapcontext

Manipulate user contexts

XSI

```
#include <ucontext.h>

void makecontext(ucontext_t *ucp, void (*func)(void),
    int argc, ...);
int swapcontext(ucontext_t *restrict oucp,
    const ucontext_t *restrict ucp);
```

malloc

A memory allocator

```
#include <stdlib.h>

void *malloc(size_t size);
```

mblen

Get number of bytes in a character

```
#include <stdlib.h>

int mblen(const char *s, size_t n);
```

mbrlen

Get number of bytes in a character (restartable)

```
#include <wchar.h>

size_t mbrlen(const char *restrict s, size_t n,
    mbstate_t *restrict ps);
```

mbrtowc

Convert a character to a wide-character code (restartable)

```
#include <wchar.h>

size_t mbrtowc(wchar_t *restrict pwc, const char *restrict s,
    size_t n, mbstate_t *restrict ps);
```

mbsinit

Determine conversion object status

```
#include <wchar.h>

int mbsinit(const mbstate_t *ps);
```

mbsrtowcs

Convert a character string to a wide-character string (restartable)

```
#include <wchar.h>

size_t mbsrtowcs(wchar_t *restrict dst, const char **restrict src,
    size_t len, mbstate_t *restrict ps);
```

mbstowcs

Convert a character string to a wide-character string

```
#include <stdlib.h>

size_t mbstowcs(wchar_t *restrict pwcs, const char *restrict s,
    size_t n);
```

mbtowc

Convert a character to a wide-character code

```
#include <stdlib.h>

int mbtowc(wchar_t *restrict pwc, const char *restrict s, size_t n);
```

memccpy

Copy bytes in memory

XSI

```
#include <string.h>

void *memccpy(void *restrict s1, const void *restrict s2,
    int c, size_t n);
```

memchr

Find byte in memory

```
#include <string.h>
void *memchr(const void *s, int c, size_t n);
```

memcmp

Compare bytes in memory

```
#include <string.h>
int memcmp(const void *s1, const void *s2, size_t n);
```

memcpy

Copy bytes in memory

```
#include <string.h>
void *memcpy(void *restrict s1, const void *restrict s2, size_t n);
```

memmove

Copy bytes in memory with overlapping areas

```
#include <string.h>
void *memmove(void *s1, const void *s2, size_t n);
```

memset

Set bytes in memory

```
#include <string.h>
void *memset(void *s, int c, size_t n);
```

mkdir

Make a directory

```
#include <sys/stat.h>
int mkdir(const char *path, mode_t mode);
```

mkfifo

Make a FIFO special file

```
#include <sys/stat.h>
int mkfifo(const char *path, mode_t mode);
```

mknod

Make a directory, a special file, or a regular file

XSI
```
#include <sys/stat.h>

int mknod(const char *path, mode_t mode, dev_t dev);
```

mkstemp

Make a unique filename

XSI
```
#include <stdlib.h>

int mkstemp(char *template);
```

mktemp

Make a unique filename (**LEGACY**)

XSI
```
#include <stdlib.h>

char *mktemp(char *template);
```

mktime

Convert broken-down time into time since the Epoch

```
#include <time.h>

time_t mktime(struct tm *timeptr);
```

mlock, munlock

Lock or unlock a range of process address space (**REALTIME**)

MLR
```
#include <sys/mman.h>

int mlock(const void *addr, size_t len);
int munlock(const void *addr, size_t len);
```

mlockall, munlockall

Lock/unlock the address space of a process (**REALTIME**)

ML
```
#include <sys/mman.h>

int mlockall(int flags);
int munlockall(void);
```

mmap

Map pages of memory

MC3
```
#include <sys/mman.h>

void *mmap(void *addr, size_t len, int prot, int flags,
    int fildes, off_t off);
```

modf, modff, modfl

Decompose a floating-point number

```
#include <math.h>

double modf(double x, double *iptr);
float modff(float value, float *iptr);
long double modfl(long double value, long double *iptr);
```

mprotect

Set protection of memory mapping

MPR
```
#include <sys/mman.h>

int mprotect(void *addr, size_t len, int prot);
```

mq_close

Close a message queue (**REALTIME**)

MSG
```
#include <mqueue.h>

int mq_close(mqd_t mqdes);
```

mq_getattr

Get message queue attributes (**REALTIME**)

MSG
```
#include <mqueue.h>

int mq_getattr(mqd_t mqdes, struct mq_attr *mqstat);
```

mq_notify

Notify process that a message is available (**REALTIME**)

MSG
```
#include <mqueue.h>

int mq_notify(mqd_t mqdes, const struct sigevent *notification);
```

mq_open

Open a message queue (**REALTIME**)

MSG
```
#include <mqueue.h>

mqd_t mq_open(const char *name, int oflag, ...);
```

mq_receive, mq_timedreceive

Receive a message from a message queue (**REALTIME**)

MSG
```
#include <mqueue.h>

ssize_t mq_receive(mqd_t mqdes, char *msg_ptr, size_t msg_len,
    unsigned *msg_prio);
```

MSG TMO
```
#include <mqueue.h>
#include <time.h>

ssize_t mq_timedreceive(mqd_t mqdes, char *restrict msg_ptr,
    size_t msg_len, unsigned *restrict msg_prio,
    const struct timespec *restrict abs_timeout);
```

mq_send, mq_timedsend

Send a message to a message queue (**REALTIME**)

MSG
```
#include <mqueue.h>

int mq_send(mqd_t mqdes, const char *msg_ptr, size_t msg_len,
    unsigned msg_prio);
```

MSG TMO
```
#include <mqueue.h>
#include <time.h>

int mq_timedsend(mqd_t mqdes, const char *msg_ptr, size_t msg_len,
    unsigned msg_prio, const struct timespec *abs_timeout);
```

mq_setattr

Set message queue attributes (**REALTIME**)

MSG
```
#include <mqueue.h>

int mq_setattr(mqd_t mqdes, const struct mq_attr *restrict mqstat,
    struct mq_attr *restrict omqstat);
```

mq_unlink

Remove a message queue (**REALTIME**)

MSG
```
#include <mqueue.h>

int mq_unlink(const char *name);
```

msgctl

XSI message control operations

XSI
```
#include <sys/msg.h>

int msgctl(int msqid, int cmd, struct msqid_ds *buf);
```

msgget

Get the XSI message queue identifier

XSI
```
#include <sys/msg.h>

int msgget(key_t key, int msgflg);
```

msgrcv

XSI message receive operation

XSI
```
#include <sys/msg.h>

ssize_t msgrcv(int msqid, void *msgp, size_t msgsz, long msgtyp,
    int msgflg);
```

msgsnd

XSI message send operation

XSI
```
#include <sys/msg.h>

int msgsnd(int msqid, const void *msgp, size_t msgsz, int msgflg);
```

msync

Synchronize memory with physical storage

MF SIO
```
#include <sys/mman.h>

int msync(void *addr, size_t len, int flags);
```

munmap

Unmap pages of memory

MC3

```
#include <sys/mman.h>

int munmap(void *addr, size_t len);
```

nan, nanf, nanl

Return quiet NaN

```
#include <math.h>

double nan(const char *tagp);
float nanf(const char *tagp);
long double nanl(const char *tagp);
```

nanosleep

High resolution sleep (**REALTIME**)

TMR

```
#include <time.h>

int nanosleep(const struct timespec *rqtp, struct timespec *rmtp);
```

nearbyint, nearbyintf, nearbyintl

Floating-point rounding functions

```
#include <math.h>

double nearbyint(double x);
float nearbyintf(float x);
long double nearbyintl(long double x);
```

nextafter, nextafterf, nextafterl, nexttoward, nexttowardf, nexttowardl

Next representable floating-point number

```
#include <math.h>

double nextafter(double x, double y);
float nextafterf(float x, float y);
long double nextafterl(long double x, long double y);
double nexttoward(double x, long double y);
float nexttowardf(float x, long double y);
long double nexttowardl(long double x, long double y);
```

nftw

Walk a file tree

XSI
```
#include <ftw.h>

int nftw(const char *path, int (*fn)(const char *,
    const struct stat *, int, struct FTW *), int depth, int flags);
```

nice

Change the nice value of a process

XSI
```
#include <unistd.h>

int nice(int incr);
```

nl_langinfo

Language information

XSI
```
#include <langinfo.h>

char *nl_langinfo(nl_item item);
```

open

Open a file

OH
```
#include <sys/stat.h>
#include <fcntl.h>

int open(const char *path, int oflag, ... );
```

opendir

Open a directory

```
#include <dirent.h>

DIR *opendir(const char *dirname);
```

pause

Suspend the thread until a signal is received

```
#include <unistd.h>

int pause(void);
```

pclose

Close a pipe stream to or from a process

CX
```
#include <stdio.h>
int pclose(FILE *stream);
```

perror

Write error messages to standard error

```
#include <stdio.h>
void perror(const char *s);
```

pipe

Create an interprocess channel

```
#include <unistd.h>
int pipe(int fildes[2]);
```

poll

Input/output multiplexing

XSI
```
#include <poll.h>
int poll(struct pollfd fds[], nfds_t nfds, int timeout);
```

popen

Initiate pipe streams to or from a process

CX
```
#include <stdio.h>
FILE *popen(const char *command, const char *mode);
```

posix_fadvise

File advisory information (**ADVANCED REALTIME**)

ADV
```
#include <fcntl.h>
int posix_fadvise(int fd, off_t offset, size_t len, int advice);
```

posix_fallocate

File space control (**ADVANCED REALTIME**)

ADV
```
#include <fcntl.h>

int posix_fallocate(int fd, off_t offset, size_t len);
```

posix_madvise

Memory advisory information and alignment control (**ADVANCED REALTIME**)

ADV
```
#include <sys/mman.h>

int posix_madvise(void *addr, size_t len, int advice);
```

posix_mem_offset

Find offset and length of a mapped typed memory block (**ADVANCED REALTIME**)

TYM
```
#include <sys/mman.h>

int posix_mem_offset(const void *restrict addr, size_t len,
    off_t *restrict off, size_t *restrict contig_len,
    int *restrict fildes);
```

posix_memalign

Aligned memory allocation (**ADVANCED REALTIME**)

ADV
```
#include <stdlib.h>

int posix_memalign(void **memptr, size_t alignment, size_t size);
```

posix_openpt

Open a pseudo-terminal device

XSI
```
#include <stdlib.h>
#include <fcntl.h>

int posix_openpt(int oflag);
```

posix_spawn, posix_spawnp

Spawn a process (**ADVANCED REALTIME**)

SPN
```
#include <spawn.h>

int posix_spawn(pid_t *restrict pid, const char *restrict path,
    const posix_spawn_file_actions_t *file_actions,
    const posix_spawnattr_t *restrict attrp,
    char *const argv[restrict], char *const envp[restrict]);
int posix_spawnp(pid_t *restrict pid, const char *restrict file,
```

```
    const posix_spawn_file_actions_t *file_actions,
    const posix_spawnattr_t *restrict attrp,
    char *const argv[restrict], char *const envp[restrict]);
```

posix_spawn_file_actions_addclose, posix_spawn_file_actions_addopen

Add close or open action to spawn file actions object (**ADVANCED REALTIME**)

SPN
```
#include <spawn.h>

int posix_spawn_file_actions_addclose(posix_spawn_file_actions_t *
    file_actions, int fildes);
int posix_spawn_file_actions_addopen(posix_spawn_file_actions_t *restrict
    file_actions, int fildes, const char *restrict path,
    int oflag, mode_t mode);
```

posix_spawn_file_actions_adddup2

Add dup2 action to spawn file actions object (**ADVANCED REALTIME**)

SPN
```
#include <spawn.h>

int posix_spawn_file_actions_adddup2(posix_spawn_file_actions_t *
    file_actions, int fildes, int newfildes);
```

posix_spawn_file_actions_destroy, posix_spawn_file_actions_init

Destroy and initialize spawn file actions object (**ADVANCED REALTIME**)

SPN
```
#include <spawn.h>

int posix_spawn_file_actions_destroy(posix_spawn_file_actions_t *
    file_actions);
int posix_spawn_file_actions_init(posix_spawn_file_actions_t *
    file_actions);
```

posix_spawnattr_destroy, posix_spawnattr_init

Destroy and initialize spawn attributes object (**ADVANCED REALTIME**)

SPN
```
#include <spawn.h>

int posix_spawnattr_destroy(posix_spawnattr_t *attr);
int posix_spawnattr_init(posix_spawnattr_t *attr);
```

posix_spawnattr_getflags, posix_spawnattr_setflags

Get and set spawn-flags attribute of spawn attributes object (**ADVANCED REALTIME**)

SPN
```
#include <spawn.h>

int posix_spawnattr_getflags(const posix_spawnattr_t *restrict attr,
    short *restrict flags);
int posix_spawnattr_setflags(posix_spawnattr_t *attr, short flags);
```

posix_spawnattr_getpgroup, posix_spawnattr_setpgroup

Get and set spawn-pgroup attribute of spawn attributes object (**ADVANCED REALTIME**)

SPN
```
#include <spawn.h>

int posix_spawnattr_getpgroup(const posix_spawnattr_t *restrict attr,
    pid_t *restrict pgroup);
int posix_spawnattr_setpgroup(posix_spawnattr_t *attr, pid_t pgroup);
```

posix_spawnattr_getschedparam, posix_spawnattr_setschedparam

Get and set spawn-schedparam attribute of spawn attributes object (**ADVANCED REALTIME**)

SPN PS
```
#include <spawn.h>
#include <sched.h>

int posix_spawnattr_getschedparam(
    const posix_spawnattr_t *restrict attr,
    struct sched_param *restrict schedparam);
int posix_spawnattr_setschedparam(posix_spawnattr_t *restrict attr,
    const struct sched_param *restrict schedparam);
```

posix_spawnattr_getschedpolicy, posix_spawnattr_setschedpolicy

Get and set spawn-schedpolicy attribute of spawn attributes object (**ADVANCED REALTIME**)

SPN PS
```
#include <spawn.h>
#include <sched.h>

int posix_spawnattr_getschedpolicy(
    const posix_spawnattr_t *restrict attr,
    int *restrict schedpolicy);
int posix_spawnattr_setschedpolicy(posix_spawnattr_t *attr,
    int schedpolicy);
```

posix_spawnattr_getsigdefault, posix_spawnattr_setsigdefault

Get and set spawn-sigdefault attribute of spawn attributes object (**ADVANCED REALTIME**)

SPN
```
#include <signal.h>
#include <spawn.h>
```

```
int posix_spawnattr_getsigdefault(
    const posix_spawnattr_t *restrict attr,
    sigset_t *restrict sigdefault);
int posix_spawnattr_setsigdefault(posix_spawnattr_t *restrict attr,
    const sigset_t *restrict sigdefault);
```

posix_spawnattr_getsigmask, posix_spawnattr_setsigmask

Get and set spawn-sigmask attribute of spawn attributes object (**ADVANCED REALTIME**)

SPN
```
#include <signal.h>
#include <spawn.h>
```

```
int posix_spawnattr_getsigmask(const posix_spawnattr_t *restrict attr,
    sigset_t *restrict sigmask);
int posix_spawnattr_setsigmask(posix_spawnattr_t *restrict attr,
    const sigset_t *restrict sigmask);
```

posix_trace_attr_destroy, posix_trace_attr_init

Trace stream attributes object destroy and initialization (**TRACING**)

TRC
```
#include <trace.h>
```

```
int posix_trace_attr_destroy(trace_attr_t *attr);
int posix_trace_attr_init(trace_attr_t *attr);
```

posix_trace_attr_getclockres, posix_trace_attr_getcreatetime, posix_trace_attr_getgenversion, posix_trace_attr_getname, posix_trace_attr_setname

Retrieve and set information about a trace stream (**TRACING**)

TRC
```
#include <time.h>
#include <trace.h>
```

```
int posix_trace_attr_getclockres(const trace_attr_t *attr,
    struct timespec *resolution);
int posix_trace_attr_getcreatetime(const trace_attr_t *attr,
    struct timespec *createtime);
```

```
#include <trace.h>
```

```
int posix_trace_attr_getgenversion(const trace_attr_t *attr,
    char *genversion);
int posix_trace_attr_getname(const trace_attr_t *attr,
    char *tracename);
int posix_trace_attr_setname(trace_attr_t *attr,
    const char *tracename);
```

posix_trace_attr_getinherited, posix_trace_attr_getlogfullpolicy, posix_trace_attr_getstreamfullpolicy, posix_trace_attr_setinherited, posix_trace_attr_setlogfullpolicy, posix_trace_attr_setstreamfullpolicy

Retrieve and set the behavior of a trace stream (**TRACING**)

```
TRC     #include <trace.h>

TRC TRI int posix_trace_attr_getinherited(const trace_attr_t *restrict attr,
            int *restrict inheritancepolicy);
TRC TRL int posix_trace_attr_getlogfullpolicy(const trace_attr_t *restrict attr,
            int *restrict logpolicy);
TRC     int posix_trace_attr_getstreamfullpolicy(const trace_attr_t *attr,
            int *streampolicy);
TRC TRI int posix_trace_attr_setinherited(trace_attr_t *attr,
            int inheritancepolicy);
TRC TRL int posix_trace_attr_setlogfullpolicy(trace_attr_t *attr,
            int logpolicy);
TRC     int posix_trace_attr_setstreamfullpolicy(trace_attr_t *attr,
            int streampolicy);
```

posix_trace_attr_getlogsize, posix_trace_attr_getmaxdatasize, posix_trace_attr_getmaxsystemeventsize, posix_trace_attr_getmaxuserevent size, posix_trace_attr_getstreamsize, posix_trace_attr_setlogsize, posix_trace_attr_setmaxdatasize, posix_trace_attr_setstreamsize

Retrieve and set trace stream size attributes (**TRACING**)

```
TRC     #include <sys/types.h>
        #include <trace.h>

TRC TRL int posix_trace_attr_getlogsize(const trace_attr_t *restrict attr,
            size_t *restrict logsize);
TRC     int posix_trace_attr_getmaxdatasize(const trace_attr_t *restrict attr,
            size_t *restrict maxdatasize);
        int posix_trace_attr_getmaxsystemeventsize(
            const trace_attr_t *restrict attr,
            size_t *restrict eventsize);
        int posix_trace_attr_getmaxuserevent size(
            const trace_attr_t *restrict attr,
            size_t data_len, size_t *restrict eventsize);
        int posix_trace_attr_getstreamsize(const trace_attr_t *restrict attr,
            size_t *restrict streamsize);
TRC TRL int posix_trace_attr_setlogsize(trace_attr_t *attr,
            size_t logsize);
TRC     int posix_trace_attr_setmaxdatasize(trace_attr_t *attr,
            size_t maxdatasize);
        int posix_trace_attr_setstreamsize(trace_attr_t *attr,
            size_t streamsize);
```

posix_trace_clear

Clear trace stream and trace log (**TRACING**)

TRC
```
#include <sys/types.h>
#include <trace.h>

int posix_trace_clear(trace_id_t trid);
```

posix_trace_close, posix_trace_open, posix_trace_rewind

Trace log management (**TRACING**)

TRC TRL
```
#include <trace.h>

int posix_trace_close(trace_id_t trid);
int posix_trace_open(int file_desc, trace_id_t *trid);
int posix_trace_rewind(trace_id_t trid);
```

posix_trace_create, posix_trace_create_withlog, posix_trace_flush, posix_trace_shutdown

Trace stream initialization, flush, and shutdown from a process (**TRACING**)

TRC
```
#include <sys/types.h>
#include <trace.h>

int posix_trace_create(pid_t pid,
    const trace_attr_t *restrict attr,
    trace_id_t *restrict trid);
```
TRC TRL
```
int posix_trace_create_withlog(pid_t pid,
    const trace_attr_t *restrict attr, int file_desc,
    trace_id_t *restrict trid);
int posix_trace_flush(trace_id_t trid);
```
TRC
```
int posix_trace_shutdown(trace_id_t trid);
```

posix_trace_event, posix_trace_eventid_open

Trace functions for instrumenting application code (**TRACING**)

TRC
```
#include <sys/types.h>
#include <trace.h>

void posix_trace_event(trace_event_id_t event_id,
    const void *restrictdata_ptr, size_t data_len);
int posix_trace_eventid_open(const char *restrict event_name,
    trace_event_id_t *restrict event_id);
```

posix_trace_eventid_equal, posix_trace_eventid_get_name, posix_trace_trid_eventid_open

Manipulate trace event type identifier (**TRACING**)

TRC
```
#include <trace.h>
```

```
int posix_trace_eventid_equal(trace_id_t trid, trace_event_id_t event1,
    trace_event_id_t event2);
int posix_trace_eventid_get_name(trace_id_t trid,
    trace_event_id_t event, char *event_name);
```
TRC TEF
```
int posix_trace_trid_eventid_open(trace_id_t trid,
    const char *restrict event_name,
    trace_event_id_t *restrict event);
```

posix_trace_eventset_add, posix_trace_eventset_del, posix_trace_eventset_empty, posix_trace_eventset_fill, posix_trace_eventset_ismember

Manipulate trace event type sets (**TRACING**)

TRC TEF
```
#include <trace.h>
```

```
int posix_trace_eventset_add(trace_event_id_t event_id,
    trace_event_set_t *set);
int posix_trace_eventset_del(trace_event_id_t event_id,
    trace_event_set_t *set);
int posix_trace_eventset_empty(trace_event_set_t *set);
int posix_trace_eventset_fill(trace_event_set_t *set, int what);
int posix_trace_eventset_ismember(trace_event_id_t event_id,
    const trace_event_set_t *restrict set,
    int *restrict ismember);
```

posix_trace_eventtypelist_getnext_id, posix_trace_eventtypelist_rewind

Iterate over a mapping of trace event types (**TRACING**)

TRC
```
#include <trace.h>
```

```
int posix_trace_eventtypelist_getnext_id(trace_id_t trid,
    trace_event_id_t *restrict event, int *restrict unavailable);
int posix_trace_eventtypelist_rewind(trace_id_t trid);
```

posix_trace_get_attr, posix_trace_get_status

Retrieve the trace attributes or trace statuses (**TRACING**)

TRC
```
#include <trace.h>
```

```
int posix_trace_get_attr(trace_id_t trid, trace_attr_t *attr);
int posix_trace_get_status(trace_id_t trid,
    struct posix_trace_status_info *statusinfo);
```

posix_trace_get_filter, posix_trace_set_filter

Retrieve and set filter of an initialized trace stream (**TRACING**)

```
TRC TEF #include <trace.h>

int posix_trace_get_filter(trace_id_t trid, trace_event_set_t *set);
int posix_trace_set_filter(trace_id_t trid,
    const trace_event_set_t *set, int how);
```

posix_trace_getnext_event, posix_trace_timedgetnext_event, posix_trace_trygetnext_event

Retrieve a trace event (**TRACING**)

```
TRC     #include <sys/types.h>
        #include <trace.h>

        int posix_trace_getnext_event(trace_id_t trid,
            struct posix_trace_event_info *restrict event,
            void *restrict data, size_t num_bytes,
            size_t *restrict data_len, int *restrict unavailable);
TRC TMO int posix_trace_timedgetnext_event(trace_id_t trid,
            struct posix_trace_event_info *restrict event,
            void *restrict data, size_t num_bytes,
            size_t *restrict data_len, int *restrict unavailable,
            const struct timespec *restrict abs_timeout);
TRC     int posix_trace_trygetnext_event(trace_id_t trid,
            struct posix_trace_event_info *restrict event,
            void *restrict data, size_t num_bytes,
            size_t *restrict data_len, int *restrict unavailable);
```

posix_trace_start, posix_trace_stop

Trace start and stop (**TRACING**)

```
TRC     #include <trace.h>

        int posix_trace_start(trace_id_t trid);
        int posix_trace_stop (trace_id_t trid);
```

posix_typed_mem_get_info

Query typed memory information (**ADVANCED REALTIME**)

```
TYM     #include <sys/mman.h>

        int posix_typed_mem_get_info(int fildes,
            struct posix_typed_mem_info *info);
```

posix_typed_mem_open

Open a typed memory object (**ADVANCED REALTIME**)

TYM
```
#include <sys/mman.h>

int posix_typed_mem_open(const char *name, int oflag, int tflag);
```

pow, powf, powl

Power function

```
#include <math.h>

double pow(double x, double y);
float powf(float x, float y);
long double powl(long double x, long double y);
```

pselect, select

Synchronous I/O multiplexing

```
#include <sys/select.h>

int pselect(int nfds, fd_set *restrict readfds,
    fd_set *restrict writefds, fd_set *restrict errorfds,
    const struct timespec *restrict timeout,
    const sigset_t *restrict sigmask);
int select(int nfds, fd_set *restrict readfds,
    fd_set *restrict writefds, fd_set *restrict errorfds,
    struct timeval *restrict timeout);
void FD_CLR(int fd, fd_set *fdset);
int FD_ISSET(int fd, fd_set *fdset);
void FD_SET(int fd, fd_set *fdset);
void FD_ZERO(fd_set *fdset);
```

pthread_atfork

Register fork handlers

THR
```
#include <pthread.h>

int pthread_atfork(void (*prepare)(void), void (*parent)(void),
    void (*child)(void));
```

pthread_attr_destroy, pthread_attr_init

Destroy and initialize threads attributes object

THR
```
#include <pthread.h>

int pthread_attr_destroy(pthread_attr_t *attr);
int pthread_attr_init(pthread_attr_t *attr);
```

pthread_attr_getdetachstate, pthread_attr_setdetachstate

Get and set detachstate attribute

THR
```
#include <pthread.h>

int pthread_attr_getdetachstate(const pthread_attr_t *attr,
    int *detachstate);
int pthread_attr_setdetachstate(pthread_attr_t *attr, int detachstate);
```

pthread_attr_getguardsize, pthread_attr_setguardsize

Get and set the thread guardsize attribute

XSI
```
#include <pthread.h>

int pthread_attr_getguardsize(const pthread_attr_t *restrict attr,
    size_t *restrict guardsize);
int pthread_attr_setguardsize(pthread_attr_t *attr,
    size_t guardsize);
```

pthread_attr_getinheritsched, pthread_attr_setinheritsched

Get and set inheritsched attribute (**REALTIME THREADS**)

THR TPS
```
#include <pthread.h>

int pthread_attr_getinheritsched(const pthread_attr_t *restrict attr,
    int *restrict inheritsched);
int pthread_attr_setinheritsched(pthread_attr_t *attr,
    int inheritsched);
```

pthread_attr_getschedparam, pthread_attr_setschedparam

Get and set schedparam attribute

THR
```
#include <pthread.h>

int pthread_attr_getschedparam(const pthread_attr_t *restrict attr,
    struct sched_param *restrict param);
int pthread_attr_setschedparam(pthread_attr_t *restrict attr,
    const struct sched_param *restrict param);
```

pthread_attr_getschedpolicy, pthread_attr_setschedpolicy

Get and set schedpolicy attribute (**REALTIME THREADS**)

THR TPS
```
#include <pthread.h>

int pthread_attr_getschedpolicy(const pthread_attr_t *restrict attr,
    int *restrict policy);
int pthread_attr_setschedpolicy(pthread_attr_t *attr, int policy);
```

pthread_attr_getscope, pthread_attr_setscope

Get and set contentionscope attribute (**REALTIME THREADS**)

THR TPS
```
#include <pthread.h>

int pthread_attr_getscope(const pthread_attr_t *restrict attr,
    int *restrict contentionscope);
int pthread_attr_setscope(pthread_attr_t *attr, int contentionscope);
```

pthread_attr_getstack, pthread_attr_setstack

Get and set stack attributes

THR
TSA TSS
```
#include <pthread.h>

int pthread_attr_getstack(const pthread_attr_t *restrict attr,
    void **restrict stackaddr, size_t *restrict stacksize);
int pthread_attr_setstack(pthread_attr_t *attr, void *stackaddr,
    size_t stacksize);
```

pthread_attr_getstackaddr, pthread_attr_setstackaddr

Get and set stackaddr attribute

THR TSA
OB
```
#include <pthread.h>

int pthread_attr_getstackaddr(const pthread_attr_t *restrict attr,
    void **restrict stackaddr);
int pthread_attr_setstackaddr(pthread_attr_t *attr, void *stackaddr);
```

pthread_attr_getstacksize, pthread_attr_setstacksize

Get and set stacksize attribute

THR TSA
```
#include <pthread.h>

int pthread_attr_getstacksize(const pthread_attr_t *restrict attr,
    size_t *restrict stacksize);
int pthread_attr_setstacksize(pthread_attr_t *attr, size_t stacksize);
```

pthread_barrier_destroy, pthread_barrier_init

Destroy and initialize a barrier object (**ADVANCED REALTIME THREADS**)

THR BAR
```
#include <pthread.h>

int pthread_barrier_destroy(pthread_barrier_t *barrier);
int pthread_barrier_init(pthread_barrier_t *restrict barrier,
    const pthread_barrierattr_t *restrict attr, unsigned count);
```

pthread_barrier_wait

Synchronize at a barrier (**ADVANCED REALTIME THREADS**)

THR BAR
```
#include <pthread.h>

int pthread_barrier_wait(pthread_barrier_t *barrier);
```

pthread_barrierattr_destroy, pthread_barrierattr_init

Destroy and initialize barrier attributes object (**ADVANCED REALTIME THREADS**)

THR BAR
```
#include <pthread.h>

int pthread_barrierattr_destroy(pthread_barrierattr_t *attr);
int pthread_barrierattr_init(pthread_barrierattr_t *attr);
```

pthread_barrierattr_getpshared, pthread_barrierattr_setpshared

Get and set process-shared attribute of barrier attributes object (**ADVANCED REALTIME THREADS**)

THR
```
#include <pthread.h>
```

BAR TSH
```
int pthread_barrierattr_getpshared(
    const pthread_barrierattr_t *restrict attr,
    int *restrict pshared);
int pthread_barrierattr_setpshared(pthread_barrierattr_t *attr,
    int pshared);
```

pthread_cancel

Cancel execution of a thread

THR
```
#include <pthread.h>

int pthread_cancel(pthread_t thread);
```

pthread_cleanup_pop, pthread_cleanup_push

Establish cancelation handlers

THR
```
#include <pthread.h>

void pthread_cleanup_pop(int execute);
void pthread_cleanup_push(void (*routine)(void*), void *arg);
```

pthread_cond_broadcast, pthread_cond_signal

Broadcast or signal a condition

THR
```
#include <pthread.h>

int pthread_cond_broadcast(pthread_cond_t *cond);
int pthread_cond_signal(pthread_cond_t *cond);
```

pthread_cond_destroy, pthread_cond_init

Destroy and initialize condition variables

THR
```
#include <pthread.h>

int pthread_cond_destroy(pthread_cond_t *cond);
int pthread_cond_init(pthread_cond_t *restrict cond,
    const pthread_condattr_t *restrict attr);
pthread_cond_t cond = PTHREAD_COND_INITIALIZER;
```

pthread_cond_timedwait, pthread_cond_wait

Wait on a condition

THR
```
#include <pthread.h>

int pthread_cond_timedwait(pthread_cond_t *restrict cond,
    pthread_mutex_t *restrict mutex,
    const struct timespec *restrict abstime);
int pthread_cond_wait(pthread_cond_t *restrict cond,
    pthread_mutex_t *restrict mutex);
```

pthread_condattr_destroy, pthread_condattr_init

Destroy and initialize condition variable attributes object

THR
```
#include <pthread.h>

int pthread_condattr_destroy(pthread_condattr_t *attr);
int pthread_condattr_init(pthread_condattr_t *attr);
```

pthread_condattr_getclock, pthread_condattr_setclock

Get and set the clock selection condition variable attribute (**ADVANCED REALTIME**)

THR CS
```
#include <pthread.h>

int pthread_condattr_getclock(const pthread_condattr_t *restrict attr,
    clockid_t *restrict clock_id);
int pthread_condattr_setclock(pthread_condattr_t *attr,
    clockid_t clock_id);
```

pthread_condattr_getpshared, pthread_condattr_setpshared

Get and set the process-shared condition variable attributes

THR TSH

```
#include <pthread.h>

int pthread_condattr_getpshared(const pthread_condattr_t *restrict attr,
    int *restrict pshared);
int pthread_condattr_setpshared(pthread_condattr_t *attr,
    int pshared);
```

pthread_create

Thread creation

THR

```
#include <pthread.h>

int pthread_create(pthread_t *restrict thread,
    const pthread_attr_t *restrict attr,
    void *(*start_routine)(void*), void *restrict arg);
```

pthread_detach

Detach a thread

THR

```
#include <pthread.h>

int pthread_detach(pthread_t thread);
```

pthread_equal

Compare thread IDs

THR

```
#include <pthread.h>

int pthread_equal(pthread_t t1, pthread_t t2);
```

pthread_exit

Thread termination

THR

```
#include <pthread.h>

void pthread_exit(void *value_ptr);
```

pthread_getconcurrency, pthread_setconcurrency

Get and set level of concurrency

XSI

```
#include <pthread.h>

int pthread_getconcurrency(void);
int pthread_setconcurrency(int new_level);
```

pthread_getcpuclockid

Access a thread CPU-time clock (**ADVANCED REALTIME THREADS**)

THR TCT

```
#include <pthread.h>
#include <time.h>

int pthread_getcpuclockid(pthread_t thread_id, clockid_t *clock_id);
```

pthread_getschedparam, pthread_setschedparam

Dynamic thread scheduling parameters access (**REALTIME THREADS**)

THR TPS

```
#include <pthread.h>

int pthread_getschedparam(pthread_t thread, int *restrict policy,
    struct sched_param *restrict param);
int pthread_setschedparam(pthread_t thread, int policy,
    const struct sched_param *param);
```

pthread_getspecific, pthread_setspecific

Thread-specific data management

THR

```
#include <pthread.h>

void *pthread_getspecific(pthread_key_t key);
int pthread_setspecific(pthread_key_t key, const void *value);
```

pthread_join

Wait for thread termination

THR

```
#include <pthread.h>

int pthread_join(pthread_t thread, void **value_ptr);
```

pthread_key_create

Thread-specific data key creation

THR

```
#include <pthread.h>

int pthread_key_create(pthread_key_t *key, void (*destructor)(void*));
```

pthread_key_delete

Thread-specific data key deletion

THR

```
#include <pthread.h>

int pthread_key_delete(pthread_key_t key);
```

pthread_kill

Send a signal to a thread

THR

```
#include <signal.h>

int pthread_kill(pthread_t thread, int sig);
```

pthread_mutex_destroy, pthread_mutex_init

Destroy and initialize a mutex

THR

```
#include <pthread.h>

int pthread_mutex_destroy(pthread_mutex_t *mutex);
int pthread_mutex_init(pthread_mutex_t *restrict mutex,
    const pthread_mutexattr_t *restrict attr);
pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;
```

pthread_mutex_getprioceiling, pthread_mutex_setprioceiling

Get and set the priority ceiling of a mutex (**REALTIME THREADS**)

THR TPP

```
#include <pthread.h>

int pthread_mutex_getprioceiling(const pthread_mutex_t *restrict mutex,
    int *restrict prioceiling);
int pthread_mutex_setprioceiling(pthread_mutex_t *restrict mutex,
    int prioceiling, int *restrict old_ceiling);
```

pthread_mutex_lock, pthread_mutex_trylock, pthread_mutex_unlock

Lock and unlock a mutex

THR

```
#include <pthread.h>

int pthread_mutex_lock(pthread_mutex_t *mutex);
int pthread_mutex_trylock(pthread_mutex_t *mutex);
int pthread_mutex_unlock(pthread_mutex_t *mutex);
```

pthread_mutex_timedlock

Lock a mutex (**ADVANCED REALTIME**)

THR TMO

```
#include <pthread.h>
#include <time.h>

int pthread_mutex_timedlock(pthread_mutex_t *restrict mutex,
    const struct timespec *restrict abs_timeout);
```

pthread_mutexattr_destroy, pthread_mutexattr_init

Destroy and initialize mutex attributes object

THR

```
#include <pthread.h>

int pthread_mutexattr_destroy(pthread_mutexattr_t *attr);
int pthread_mutexattr_init(pthread_mutexattr_t *attr);
```

pthread_mutexattr_getprioceiling, pthread_mutexattr_setprioceiling

Get and set prioceiling attribute of mutex attributes object (**REALTIME THREADS**)

THR TPP

```
#include <pthread.h>

int pthread_mutexattr_getprioceiling(
    const pthread_mutexattr_t *restrict attr,
    int *restrict prioceiling);
int pthread_mutexattr_setprioceiling(pthread_mutexattr_t *attr,
    int prioceiling);
```

pthread_mutexattr_getprotocol, pthread_mutexattr_setprotocol

Get and set protocol attribute of mutex attributes object (**REALTIME THREADS**)

THR

```
#include <pthread.h>
```

TPP|TPI

```
int pthread_mutexattr_getprotocol(
    const pthread_mutexattr_t *restrict attr,
    int *restrict protocol);
int pthread_mutexattr_setprotocol(pthread_mutexattr_t *attr,
    int protocol);
```

pthread_mutexattr_getpshared, pthread_mutexattr_setpshared

Get and set process-shared attribute

THR TSH
```
#include <pthread.h>

int pthread_mutexattr_getpshared(
    const pthread_mutexattr_t *restrict attr,
    int *restrict pshared);
int pthread_mutexattr_setpshared(pthread_mutexattr_t *attr,
    int pshared);
```

pthread_mutexattr_gettype, pthread_mutexattr_settype

Get and set a mutex type attribute

XSI
```
#include <pthread.h>

int pthread_mutexattr_gettype(const pthread_mutexattr_t *restrict attr,
    int *restrict type);
int pthread_mutexattr_settype(pthread_mutexattr_t *attr, int type);
```

pthread_once

Dynamic package initialization

THR
```
#include <pthread.h>

int pthread_once(pthread_once_t *once_control,
    void (*init_routine)(void));
pthread_once_t once_control = PTHREAD_ONCE_INIT;
```

pthread_rwlock_destroy, pthread_rwlock_init

Destroy and initialize a read-write lock object

THR
```
#include <pthread.h>

int pthread_rwlock_destroy(pthread_rwlock_t *rwlock);
int pthread_rwlock_init(pthread_rwlock_t *restrict rwlock,
    const pthread_rwlockattr_t *restrict attr);
```

pthread_rwlock_rdlock, pthread_rwlock_tryrdlock

Lock a read-write lock object for reading

THR
```
#include <pthread.h>

int pthread_rwlock_rdlock(pthread_rwlock_t *rwlock);
int pthread_rwlock_tryrdlock(pthread_rwlock_t *rwlock);
```

pthread_rwlock_timedrdlock

Lock a read-write lock for reading

THR TMO
```
#include <pthread.h>
#include <time.h>

int pthread_rwlock_timedrdlock(pthread_rwlock_t *restrict rwlock,
    const struct timespec *restrict abs_timeout);
```

pthread_rwlock_timedwrlock

Lock a read-write lock for writing

THR TMO
```
#include <pthread.h>
#include <time.h>

int pthread_rwlock_timedwrlock(pthread_rwlock_t *restrict rwlock,
    const struct timespec *restrict abs_timeout);
```

pthread_rwlock_trywrlock, pthread_rwlock_wrlock

Lock a read-write lock object for writing

THR
```
#include <pthread.h>

int pthread_rwlock_trywrlock(pthread_rwlock_t *rwlock);
int pthread_rwlock_wrlock(pthread_rwlock_t *rwlock);
```

pthread_rwlock_unlock

Unlock a read-write lock object

THR
```
#include <pthread.h>

int pthread_rwlock_unlock(pthread_rwlock_t *rwlock);
```

pthread_rwlockattr_destroy, pthread_rwlockattr_init

Destroy and initialize read-write lock attributes object

THR
```
#include <pthread.h>

int pthread_rwlockattr_destroy(pthread_rwlockattr_t *attr);
int pthread_rwlockattr_init(pthread_rwlockattr_t *attr);
```

pthread_rwlockattr_getpshared, pthread_rwlockattr_setpshared

Get and set process-shared attribute of read-write lock attributes object

THR TSH
```
#include <pthread.h>

int pthread_rwlockattr_getpshared(
    const pthread_rwlockattr_t *restrict attr,
    int *restrict pshared);
int pthread_rwlockattr_setpshared(pthread_rwlockattr_t *attr,
    int pshared);
```

pthread_self

Get calling thread's ID

THR
```
#include <pthread.h>

pthread_t pthread_self(void);
```

pthread_setcancelstate, pthread_setcanceltype, pthread_testcancel

Set cancelability state

THR
```
#include <pthread.h>

int pthread_setcancelstate(int state, int *oldstate);
int pthread_setcanceltype(int type, int *oldtype);
void pthread_testcancel(void);
```

pthread_setschedprio

Dynamic thread scheduling parameters access (**REALTIME THREADS**)

THR TPS
```
#include <pthread.h>

int pthread_setschedprio(pthread_t thread, int prio);
```

pthread_sigmask, sigprocmask

Examine and change blocked signals

```
#include <signal.h>
```

THR
```
int pthread_sigmask(int how, const sigset_t *restrict set,
    sigset_t *restrict oset);
```
CX
```
int sigprocmask(int how, const sigset_t *restrict set,
    sigset_t *restrict oset);
```

pthread_spin_destroy, pthread_spin_init

Destroy or initialize a spin lock object (**ADVANCED REALTIME THREADS**)

THR SPI
```
#include <pthread.h>

int pthread_spin_destroy(pthread_spinlock_t *lock);
int pthread_spin_init(pthread_spinlock_t *lock, int pshared);
```

pthread_spin_lock, pthread_spin_trylock

Lock a spin lock object (**ADVANCED REALTIME THREADS**)

THR SPI
```
#include <pthread.h>

int pthread_spin_lock(pthread_spinlock_t *lock);
int pthread_spin_trylock(pthread_spinlock_t *lock);
```

pthread_spin_unlock

Unlock a spin lock object (**ADVANCED REALTIME THREADS**)

THR SPI
```
#include <pthread.h>

int pthread_spin_unlock(pthread_spinlock_t *lock);
```

ptsname

Get name of the slave pseudo-terminal device

XSI
```
#include <stdlib.h>

char *ptsname(int fildes);
```

putc

Put byte on a stream

```
#include <stdio.h>

int putc(int c, FILE *stream);
```

putchar

Put byte on stdout stream

```
#include <stdio.h>

int putchar(int c);
```

putenv

Change or add a value to environment

XSI
```
#include <stdlib.h>

int putenv(char *string);
```

putmsg, putpmsg

Send a message on a STREAM (**STREAMS**)

XSR
```
#include <stropts.h>

int putmsg(int fildes, const struct strbuf *ctlptr,
    const struct strbuf *dataptr, int flags);
int putpmsg(int fildes, const struct strbuf *ctlptr,
    const struct strbuf *dataptr, int band, int flags);
```

puts

Put a string on standard output

```
#include <stdio.h>

int puts(const char *s);
```

putwc

Put a wide character on a stream

```
#include <stdio.h>
#include <wchar.h>

wint_t putwc(wchar_t wc, FILE *stream);
```

putwchar

Put a wide character on stdout stream

```
#include <wchar.h>

wint_t putwchar(wchar_t wc);
```

qsort

Sort a table of data

```
#include <stdlib.h>

void qsort(void *base, size_t nel, size_t width,
    int (*compar)(const void *, const void *));
```

raise

Send a signal to the executing process

```
#include <signal.h>

int raise(int sig);
```

rand, rand_r, srand

Pseudo-random number generator

```
#include <stdlib.h>

int rand(void);
TSF int rand_r(unsigned *seed);
void srand(unsigned seed);
```

pread, read

Read from a file

```
#include <unistd.h>

XSI ssize_t pread(int fildes, void *buf, size_t nbyte, off_t offset);
ssize_t read(int fildes, void *buf, size_t nbyte);
```

readdir, readdir_r

Read directory

```
#include <dirent.h>

struct dirent *readdir(DIR *dirp);
TSF int readdir_r(DIR *restrict dirp, struct dirent *restrict entry,
    struct dirent **restrict result);
```

readlink

Read the contents of a symbolic link

```
#include <unistd.h>

ssize_t readlink(const char *restrict path, char *restrict buf,
    size_t bufsize);
```

readv

Read a vector

```
XSI #include <sys/uio.h>

ssize_t readv(int fildes, const struct iovec *iov, int iovcnt);
```

realloc

Memory reallocator

```
#include <stdlib.h>

void *realloc(void *ptr, size_t size);
```

realpath

Resolve a pathname

XSI

```
#include <stdlib.h>

char *realpath(const char *restrict file_name,
    char *restrict resolved_name);
```

recv

Receive a message from a connected socket

```
#include <sys/socket.h>

ssize_t recv(int socket, void *buffer, size_t length, int flags);
```

recvfrom

Receive a message from a socket

```
#include <sys/socket.h>

ssize_t recvfrom(int socket, void *restrict buffer, size_t length,
    int flags, struct sockaddr *restrict address,
    socklen_t *restrict address_len);
```

recvmsg

Receive a message from a socket

```
#include <sys/socket.h>

ssize_t recvmsg(int socket, struct msghdr *message, int flags);
```

regcomp, regerror, regexec, regfree

Regular expression matching

```
#include <regex.h>

int regcomp(regex_t *restrict preg, const char *restrict pattern,
    int cflags);
size_t regerror(int errcode, const regex_t *restrict preg,
    char *restrict errbuf, size_t errbuf_size);
int regexec(const regex_t *restrict preg, const char *restrict string,
    size_t nmatch, regmatch_t pmatch[restrict], int eflags);
void regfree(regex_t *preg);
```

remainder, remainderf, remainderl

Remainder function

```
#include <math.h>

double remainder(double x, double y);
float remainderf(float x, float y);
long double remainderl(long double x, long double y);
```

remove

Remove a file

```
#include <stdio.h>

int remove(const char *path);
```

remquo, remquof, remquol

Remainder functions

```
#include <math.h>

double remquo(double x, double y, int *quo);
float remquof(float x, float y, int *quo);
long double remquol(long double x, long double y, int *quo);
```

rename

Rename a file

```
#include <stdio.h>

int rename(const char *old, const char *new);
```

rewind

Reset file position indicator in a stream

```
#include <stdio.h>

void rewind(FILE *stream);
```

rewinddir

Reset position of directory stream to the beginning of a directory

```
#include <dirent.h>

void rewinddir(DIR *dirp);
```

rindex

Character string operations (**LEGACY**)

XSI
```
#include <strings.h>

char *rindex(const char *s, int c);
```

rint, rintf, rintl

Round-to-nearest integral value

```
#include <math.h>

double rint(double x);
float rintf(float x);
long double rintl(long double x);
```

rmdir

Remove a directory

```
#include <unistd.h>

int rmdir(const char *path);
```

round, roundf, roundl

Round to nearest integer value in floating-point format

```
#include <math.h>

double round(double x);
float roundf(float x);
long double roundl(long double x);
```

scalb

Load exponent of a radix-independent floating-point number

OB XSI
```
#include <math.h>

double scalb(double x, double n);
```

scalbln, scalblnf, scalblnl, scalbn, scalbnf, scalbnl

Compute exponent using FLT_RADIX

```
#include <math.h>

double scalbln(double x, long n);
float scalblnf(float x, long n);
long double scalblnl(long double x, long n);
double scalbn(double x, int n);
float scalbnf(float x, int n);
long double scalbnl(long double x, int n);
```

sched_get_priority_max, sched_get_priority_min

Get priority limits (**REALTIME**)

PS
```
#include <sched.h>

int sched_get_priority_max(int policy);
int sched_get_priority_min(int policy);
```

sched_getparam

Get scheduling parameters (**REALTIME**)

PS
```
#include <sched.h>

int sched_getparam(pid_t pid, struct sched_param *param);
```

sched_getscheduler

Get scheduling policy (**REALTIME**)

PS
```
#include <sched.h>

int sched_getscheduler(pid_t pid);
```

sched_rr_get_interval

Get execution time limits (**REALTIME**)

PS
```
#include <sched.h>

int sched_rr_get_interval(pid_t pid, struct timespec *interval);
```

sched_setparam

Set scheduling parameters (**REALTIME**)

PS
```
#include <sched.h>

int sched_setparam(pid_t pid, const struct sched_param *param);
```

sched_setscheduler

Set scheduling policy and parameters (**REALTIME**)

PS
```
#include <sched.h>

int sched_setscheduler(pid_t pid, int policy,
    const struct sched_param *param);
```

sched_yield

Yield processor

PS|THR
```
#include <sched.h>
int sched_yield(void);
```

seekdir

Set position of directory stream

XSI
```
#include <dirent.h>
void seekdir(DIR *dirp, long loc);
```

sem_close

Close a named semaphore (**REALTIME**)

SEM
```
#include <semaphore.h>
int sem_close(sem_t *sem);
```

sem_destroy

Destroy an unnamed semaphore (**REALTIME**)

SEM
```
#include <semaphore.h>
int sem_destroy(sem_t *sem);
```

sem_getvalue

Get the value of a semaphore (**REALTIME**)

SEM
```
#include <semaphore.h>
int sem_getvalue(sem_t *restrict sem, int *restrict sval);
```

sem_init

Initialize an unnamed semaphore (**REALTIME**)

SEM
```
#include <semaphore.h>
int sem_init(sem_t *sem, int pshared, unsigned value);
```

sem_open

Initialize and open a named semaphore (**REALTIME**)

SEM
```
#include <semaphore.h>

sem_t *sem_open(const char *name, int oflag, ...);
```

sem_post

Unlock a semaphore (**REALTIME**)

SEM
```
#include <semaphore.h>

int sem_post(sem_t *sem);
```

sem_timedwait

Lock a semaphore (**ADVANCED REALTIME**)

SEM TMO
```
#include <semaphore.h>
#include <time.h>

int sem_timedwait(sem_t *restrict sem,
    const struct timespec *restrict abs_timeout);
```

sem_trywait, sem_wait

Lock a semaphore (**REALTIME**)

SEM
```
#include <semaphore.h>

int sem_trywait(sem_t *sem);
int sem_wait(sem_t *sem);
```

sem_unlink

Remove a named semaphore (**REALTIME**)

SEM
```
#include <semaphore.h>

int sem_unlink(const char *name);
```

semctl

XSI semaphore control operations

XSI
```
#include <sys/sem.h>

int semctl(int semid, int semnum, int cmd, ...);
```

semget

Get set of XSI semaphores

XSI
```
#include <sys/sem.h>

int semget(key_t key, int nsems, int semflg);
```

semop

XSI semaphore operations

XSI
```
#include <sys/sem.h>

int semop(int semid, struct sembuf *sops, size_t nsops);
```

send

Send a message on a socket

```
#include <sys/socket.h>

ssize_t send(int socket, const void *buffer, size_t length, int flags);
```

sendmsg

Send a message on a socket using a message structure

```
#include <sys/socket.h>

ssize_t sendmsg(int socket, const struct msghdr *message, int flags);
```

sendto

Send a message on a socket

```
#include <sys/socket.h>

ssize_t sendto(int socket, const void *message, size_t length,
    int flags, const struct sockaddr *dest_addr,
    socklen_t dest_len);
```

setbuf

Assign buffering to a stream

```
#include <stdio.h>

void setbuf(FILE *restrict stream, char *restrict buf);
```

setegid

Set effective group ID

```
#include <unistd.h>
int setegid(gid_t gid);
```

setenv

Add or change environment variable

CX

```
#include <stdlib.h>
int setenv(const char *envname, const char *envval, int overwrite);
```

seteuid

Set effective user ID

```
#include <unistd.h>
int seteuid(uid_t uid);
```

setgid

Set-group-ID

```
#include <unistd.h>
int setgid(gid_t gid);
```

setjmp

Set jump point for a non-local goto

```
#include <setjmp.h>
int setjmp(jmp_buf env);
```

setkey

Set encoding key (**CRYPT**)

XSI

```
#include <stdlib.h>
void setkey(const char *key);
```

setlocale

Set program locale

```
#include <locale.h>
char *setlocale(int category, const char *locale);
```

setpgid

Set process group ID for job control

```
#include <unistd.h>
int setpgid(pid_t pid, pid_t pgid);
```

setpgrp

Set process group ID

XSI

```
#include <unistd.h>
pid_t setpgrp(void);
```

setregid

Set real and effective group IDs

XSI

```
#include <unistd.h>
int setregid(gid_t rgid, gid_t egid);
```

setreuid

Set real and effective user IDs

XSI

```
#include <unistd.h>
int setreuid(uid_t ruid, uid_t euid);
```

setsid

Create session and set process group ID

```
#include <unistd.h>
pid_t setsid(void);
```

setsockopt

Set the socket options

```
#include <sys/socket.h>

int setsockopt(int socket, int level, int option_name,
    const void *option_value, socklen_t option_len);
```

setuid

Set user ID

```
#include <unistd.h>

int setuid(uid_t uid);
```

setvbuf

Assign buffering to a stream

```
#include <stdio.h>

int setvbuf(FILE *restrict stream, char *restrict buf, int type,
    size_t size);
```

shm_open

Open a shared memory object (**REALTIME**)

SHM

```
#include <sys/mman.h>

int shm_open(const char *name, int oflag, mode_t mode);
```

shm_unlink

Remove a shared memory object (**REALTIME**)

SHM

```
#include <sys/mman.h>

int shm_unlink(const char *name);
```

shmat

XSI shared memory attach operation

XSI

```
#include <sys/shm.h>

void *shmat(int shmid, const void *shmaddr, int shmflg);
```

shmctl

XSI shared memory control operations

XSI
```
#include <sys/shm.h>

int shmctl(int shmid, int cmd, struct shmid_ds *buf);
```

shmdt

XSI shared memory detach operation

XSI
```
#include <sys/shm.h>

int shmdt(const void *shmaddr);
```

shmget

Get XSI shared memory segment

XSI
```
#include <sys/shm.h>

int shmget(key_t key, size_t size, int shmflg);
```

shutdown

Shut down socket send and receive operations

```
#include <sys/socket.h>

int shutdown(int socket, int how);
```

sigaction

Examine and change signal action

CX
```
#include <signal.h>

int sigaction(int sig, const struct sigaction *restrict act,
    struct sigaction *restrict oact);
```

sigaddset

Add a signal to a signal set

CX
```
#include <signal.h>

int sigaddset(sigset_t *set, int signo);
```

sigaltstack

Set and get signal alternate stack context

XSI

```
#include <signal.h>

int sigaltstack(const stack_t *restrict ss, stack_t *restrict oss);
```

sigdelset

Delete a signal from a signal set

CX

```
#include <signal.h>

int sigdelset(sigset_t *set, int signo);
```

sigemptyset

Initialize and empty a signal set

CX

```
#include <signal.h>

int sigemptyset(sigset_t *set);
```

sigfillset

Initialize and fill a signal set

CX

```
#include <signal.h>

int sigfillset(sigset_t *set);
```

sighold, sigignore, sigpause, sigrelse, sigset

Signal management

XSI

```
#include <signal.h>

int sighold(int sig);
int sigignore(int sig);
int sigpause(int sig);
int sigrelse(int sig);
void (*sigset(int sig, void (*disp)(int)))(int);
```

siginterrupt

Allow signals to interrupt functions

XSI
```
#include <signal.h>
int siginterrupt(int sig, int flag);
```

sigismember

Test for a signal in a signal set

CX
```
#include <signal.h>
int sigismember(const sigset_t *set, int signo);
```

siglongjmp

Non-local goto with signal handling

CX
```
#include <setjmp.h>
void siglongjmp(sigjmp_buf env, int val);
```

signal

Signal management

```
#include <signal.h>
void (*signal(int sig, void (*func)(int)))(int);
```

signbit

Test sign

```
#include <math.h>
int signbit(real-floating x);
```

sigpending

Examine pending signals

CX
```
#include <signal.h>
int sigpending(sigset_t *set);
```

sigqueue

Queue a signal to a process (**REALTIME**)

RTS
```
#include <signal.h>

int sigqueue(pid_t pid, int signo, const union sigval value);
```

sigsetjmp

Set jump point for a non-local goto

CX
```
#include <setjmp.h>

int sigsetjmp(sigjmp_buf env, int savemask);
```

sigsuspend

Wait for a signal

CX
```
#include <signal.h>

int sigsuspend(const sigset_t *sigmask);
```

sigtimedwait, sigwaitinfo

Wait for queued signals (**REALTIME**)

RTS
```
#include <signal.h>

int sigtimedwait(const sigset_t *restrict set,
    siginfo_t *restrict info,
    const struct timespec *restrict timeout);
int sigwaitinfo(const sigset_t *restrict set,
    siginfo_t *restrict info);
```

sigwait

Wait for queued signals

CX
```
#include <signal.h>

int sigwait(const sigset_t *restrict set, int *restrict sig);
```

sin, sinf, sinl

Sine function

```
#include <math.h>

double sin(double x);
float sinf(float x);
long double sinl(long double x);
```

sinh, sinhf, sinhl

Hyperbolic sine function

```
#include <math.h>

double sinh(double x);
float sinhf(float x);
long double sinhl(long double x);
```

sleep

Suspend execution for an interval of time

```
#include <unistd.h>

unsigned sleep(unsigned seconds);
```

sockatmark

Determine whether a socket is at the out-of-band mark

```
#include <sys/socket.h>

int sockatmark(int s);
```

socket

Create an endpoint for communication

```
#include <sys/socket.h>

int socket(int domain, int type, int protocol);
```

socketpair

Create a pair of connected sockets

```
#include <sys/socket.h>

int socketpair(int domain, int type, int protocol,
    int socket_vector[2]);
```

sqrt, sqrtf, sqrtl

Square root function

```
#include <math.h>

double sqrt(double x);
float sqrtf(float x);
long double sqrtl(long double x);
```

stat

Get file status

```
#include <sys/stat.h>

int stat(const char *restrict path, struct stat *restrict buf);
```

stderr, stdin, stdout

Standard I/O streams

```
#include <stdio.h>

extern FILE *stderr, *stdin, *stdout;
```

strcasecmp, strncasecmp

Case-insensitive string comparisons

XSI

```
#include <strings.h>

int strcasecmp(const char *s1, const char *s2);
int strncasecmp(const char *s1, const char *s2, size_t n);
```

strcat

Concatenate two strings

```
#include <string.h>

char *strcat(char *restrict s1, const char *restrict s2);
```

strchr

String scanning operation

```
#include <string.h>

char *strchr(const char *s, int c);
```

strcmp

Compare two strings

```
#include <string.h>

int strcmp(const char *s1, const char *s2);
```

strcoll

String comparison using collating information

```
#include <string.h>

int strcoll(const char *s1, const char *s2);
```

strcpy

Copy a string

```
#include <string.h>

char *strcpy(char *restrict s1, const char *restrict s2);
```

strcspn

Get length of a complementary substring

```
#include <string.h>

size_t strcspn(const char *s1, const char *s2);
```

strdup

Duplicate a string

XSI

```
#include <string.h>

char *strdup(const char *s1);
```

strerror, strerror_r

Get error message string

```
#include <string.h>

char *strerror(int errnum);
```

TSF

```
int strerror_r(int errnum, char *strerrbuf, size_t buflen);
```

strfmon

Convert monetary value to a string

XSI

```
#include <monetary.h>

ssize_t strfmon(char *restrict s, size_t maxsize,
    const char *restrict format, ...);
```

strftime

Convert date and time to a string

```
#include <time.h>

size_t strftime(char *restrict s, size_t maxsize,
    const char *restrict format, const struct tm *restrict timeptr);
```

strlen

Get string length

```
#include <string.h>

size_t strlen(const char *s);
```

strncat

Concatenate a string with part of another

```
#include <string.h>

char *strncat(char *restrict s1, const char *restrict s2, size_t n);
```

strncmp

Compare part of two strings

```
#include <string.h>

int strncmp(const char *s1, const char *s2, size_t n);
```

strncpy

Copy part of a string

```
#include <string.h>

char *strncpy(char *restrict s1, const char *restrict s2, size_t n);
```

strpbrk

Scan string for byte

```
#include <string.h>

char *strpbrk(const char *s1, const char *s2);
```

strptime

Date and time conversion

XSI

```
#include <time.h>

char *strptime(const char *restrict buf, const char *restrict format,
    struct tm *restrict tm);
```

strrchr

String scanning operation

```
#include <string.h>

char *strrchr(const char *s, int c);
```

strspn

Get length of a substring

```
#include <string.h>

size_t strspn(const char *s1, const char *s2);
```

strstr

Find a substring

```
#include <string.h>

char *strstr(const char *s1, const char *s2);
```

strtod, strtof, strtold

Convert string to a double-precision number

```
#include <stdlib.h>

double strtod(const char *restrict nptr, char **restrict endptr);
float strtof(const char *restrict nptr, char **restrict endptr);
long double strtold(const char *restrict nptr, char **restrict endptr);
```

strtoimax, strtoumax

Convert string to integer type

```
#include <inttypes.h>

intmax_t strtoimax(const char *restrict nptr, char **restrict endptr,
    int base);
uintmax_t strtoumax(const char *restrict nptr, char **restrict endptr,
    int base);
```

strtok, strtok_r

Split string into tokens

```
#include <string.h>

char *strtok(char *restrict s1, const char *restrict s2);
```
TSF
```
char *strtok_r(char *restrict s, const char *restrict sep,
    char **restrict lasts);
```

strtol, strtoll

Convert string to a long integer

```
#include <stdlib.h>

long strtol(const char *restrict str, char **restrict endptr, int base);
long long strtoll(const char *restrict str, char **restrict endptr,
    int base)
```

strtoul, strtoull

Convert string to an unsigned long

```
#include <stdlib.h>

unsigned long strtoul(const char *restrict str,
    char **restrict endptr, int base);
unsigned long long strtoull(const char *restrict str,
    char **restrict endptr, int base);
```

strxfrm

String transformation

```
#include <string.h>

size_t strxfrm(char *restrict s1, const char *restrict s2, size_t n);
```

swab

Swap bytes

XSI
```
#include <unistd.h>

void swab(const void *restrict src, void *restrict dest,
    ssize_t nbytes);
```

symlink

Make symbolic link to a file

```
#include <unistd.h>

int symlink(const char *path1, const char *path2);
```

sync

Schedule file system updates

XSI
```
#include <unistd.h>

void sync(void);
```

sysconf

Get configurable system variables

```
#include <unistd.h>

long sysconf(int name);
```

system

Issue a command

```
#include <stdlib.h>

int system(const char *command);
```

tan, tanf, tanl

Tangent function

```
#include <math.h>

double tan(double x);
float tanf(float x);
long double tanl(long double x);
```

tanh, tanhf, tanhl

Hyperbolic tangent functions

```
#include <math.h>

double tanh(double x);
float tanhf(float x);
long double tanhl(long double x);
```

tcdrain

Wait for transmission of output

```
#include <termios.h>

int tcdrain(int fildes);
```

tcflow

Suspend or restart the transmission or reception of data

```
#include <termios.h>

int tcflow(int fildes, int action);
```

tcflush

Flush non-transmitted output data, non-read input data, or both

```
#include <termios.h>

int tcflush(int fildes, int queue_selector);
```

tcgetattr

Get the parameters associated with the terminal

```
#include <termios.h>

int tcgetattr(int fildes, struct termios *termios_p);
```

tcgetpgrp

Get the foreground process group ID

```
#include <unistd.h>

pid_t tcgetpgrp(int fildes);
```

tcgetsid

Get process group ID for session leader for controlling terminal

XSI
```
#include <termios.h>

pid_t tcgetsid(int fildes);
```

tcsendbreak

Send a ‘‘break’’ for a specific duration

```
#include <termios.h>

int tcsendbreak(int fildes, int duration);
```

tcsetattr

Set the parameters associated with the terminal

```
#include <termios.h>

int tcsetattr(int fildes, int optional_actions,
    const struct termios *termios_p);
```

tcsetpgrp

Set the foreground process group ID

```
#include <unistd.h>

int tcsetpgrp(int fildes, pid_t pgid_id);
```

tdelete, tfind, tsearch, twalk

Manage a binary search tree

XSI
```
#include <search.h>

void *tdelete(const void *restrict key, void **restrict rootp,
    int(*compar)(const void *, const void *));
void *tfind(const void *key, void *const *rootp,
    int(*compar)(const void *, const void *));
void *tsearch(const void *key, void **rootp,
    int (*compar)(const void *, const void *));
void twalk(const void *root,
    void (*action)(const void *, VISIT, int));
```

telldir

Current location of a named directory stream

XSI
```
#include <dirent.h>

long telldir(DIR *dirp);
```

tempnam

Create a name for a temporary file

XSI
```
#include <stdio.h>

char *tempnam(const char *dir, const char *pfx);
```

tgamma, tgammaf, tgammal

Compute gamma function

```
#include <math.h>

double tgamma(double x);
float tgammaf(float x);
long double tgammal(long double x);
```

time

Get time

```
#include <time.h>

time_t time(time_t *tloc);
```

timer_create

Create a per-process timer (**REALTIME**)

TMR
```
#include <signal.h>
#include <time.h>

int timer_create(clockid_t clockid, struct sigevent *restrict evp,
    timer_t *restrict timerid);
```

timer_delete

Delete a per-process timer (**REALTIME**)

TMR
```
#include <time.h>

int timer_delete(timer_t timerid);
```

timer_getoverrun, timer_gettime, timer_settime

Per-process timers (**REALTIME**)

TMR
```
#include <time.h>

int timer_getoverrun(timer_t timerid);
int timer_gettime(timer_t timerid, struct itimerspec *value);
int timer_settime(timer_t timerid, int flags,
    const struct itimerspec *restrict value,
    struct itimerspec *restrict ovalue);
```

times

Get process and waited-for child process times

```
#include <sys/times.h>

clock_t times(struct tms *buffer);
```

tmpfile

Create a temporary file

```
#include <stdio.h>

FILE *tmpfile(void);
```

tmpnam

Create a name for a temporary file

```
#include <stdio.h>

char *tmpnam(char *s);
```

toascii

Translate integer to a 7-bit ASCII character

XSI
```
#include <ctype.h>

int toascii(int c);
```

tolower

Transliterate uppercase characters to lowercase

```
#include <ctype.h>

int tolower(int c);
```

toupper

Transliterate lowercase characters to uppercase

```
#include <ctype.h>

int toupper(int c);
```

towctrans

Wide-character transliteration

```
#include <wctype.h>

wint_t towctrans(wint_t wc, wctrans_t desc);
```

towlower

Transliterate uppercase wide-character code to lowercase

```
#include <wctype.h>

wint_t towlower(wint_t wc);
```

towupper

Transliterate lowercase wide-character code to uppercase

```
#include <wctype.h>

wint_t towupper(wint_t wc);
```

trunc, truncf, truncl

Round to truncated integer value

```
#include <math.h>

double trunc(double x);
float truncf(float x);
long double truncl(long double x);
```

truncate

Truncate a file to a specified length

XSI

```
#include <unistd.h>

int truncate(const char *path, off_t length);
```

ttyname, ttyname_r

Find pathname of a terminal

```
#include <unistd.h>

char *ttyname(int fildes);
TSF int ttyname_r(int fildes, char *name, size_t namesize);
```

daylight, timezone, tzname, tzset

Set timezone conversion information

```
#include <time.h>

XSI extern int daylight;
    extern long timezone;
CX  extern char *tzname[2];
    void tzset(void);
```

ualarm

Set the interval timer

```
OB XSI #include <unistd.h>

useconds_t ualarm(useconds_t useconds, useconds_t interval);
```

ulimit

Get and set process limits

```
XSI #include <ulimit.h>

long ulimit(int cmd, ...);
```

umask

Set and get file mode creation mask

```
#include <sys/stat.h>

mode_t umask(mode_t cmask);
```

uname

Get name of current system

```
#include <sys/utsname.h>

int uname(struct utsname *name);
```

ungetc

Push byte back into input stream

```
#include <stdio.h>

int ungetc(int c, FILE *stream);
```

ungetwc

Push wide-character code back into input stream

```
#include <stdio.h>
#include <wchar.h>

wint_t ungetwc(wint_t wc, FILE *stream);
```

unlink

Remove a directory entry

```
#include <unistd.h>

int unlink(const char *path);
```

unlockpt

Unlock a pseudo-terminal master/slave pair

XSI

```
#include <stdlib.h>

int unlockpt(int fildes);
```

unsetenv

Remove environment variable

CX

```
#include <stdlib.h>

int unsetenv(const char *name);
```

usleep

Suspend execution for an interval

OB XSI

```
#include <unistd.h>

int usleep(useconds_t useconds);
```

utime

Set file access and modification times

```
#include <utime.h>

int utime(const char *path, const struct utimbuf *times);
```

utimes

Set file access and modification times (**LEGACY**)

XSI

```
#include <sys/time.h>

int utimes(const char *path, const struct timeval times[2]);
```

va_arg, va_copy, va_end, va_start

Handle variable argument list

```
#include <stdarg.h>

type va_arg(va_list ap, type);
void va_copy(va_list dest, va_list src);
void va_end(va_list ap);
void va_start(va_list ap, argN);
```

vfork

Create new process; share virtual memory

OB XSI

```
#include <unistd.h>

pid_t vfork(void);
```

vfprintf, vprintf, vsnprintf, vsprintf

Format output of a stdarg argument list

```
#include <stdarg.h>
#include <stdio.h>

int vfprintf(FILE *restrict stream, const char *restrict format,
    va_list ap);
int vprintf(const char *restrict format, va_list ap);
int vsnprintf(char *restrict s, size_t n, const char *restrict format,
    va_list ap);
int vsprintf(char *restrict s, const char *restrict format, va_list ap);
```

vfscanf, vscanf, vsscanf

Format input of a stdarg list

```
#include <stdarg.h>
#include <stdio.h>

int vfscanf(FILE *restrict stream, const char *restrict format,
    va_list arg);
int vscanf(const char *restrict format, va_list arg);
int vsscanf(const char *restrict s, const char *restrict format,
    va_list arg);
```

vfwprintf, vswprintf, vwprintf

Wide-character formatted output of a stdarg argument list

```
#include <stdarg.h>
#include <stdio.h>
#include <wchar.h>

int vfwprintf(FILE *restrict stream, const wchar_t *restrict format,
    va_list arg);
int vswprintf(wchar_t *restrict ws, size_t n,
    const wchar_t *restrict format, va_list arg);
int vwprintf(const wchar_t *restrict format, va_list arg);
```

vfwscanf, vswscanf, vwscanf

Wide-character formatted input of a stdarg list

```
#include <stdarg.h>
#include <stdio.h>
#include <wchar.h>

int vfwscanf(FILE *restrict stream, const wchar_t *restrict format,
    va_list arg);
int vswscanf(const wchar_t *restrict ws, const wchar_t *restrict format,
    va_list arg);
int vwscanf(const wchar_t *restrict format, va_list arg);
```

wait, waitpid

Wait for a child process to stop or terminate

```
#include <sys/wait.h>

pid_t wait(int *stat_loc);
pid_t waitpid(pid_t pid, int *stat_loc, int options);
```

waitid

Wait for a child process to change state

XSI
```
#include <sys/wait.h>

int waitid(idtype_t idtype, id_t id, siginfo_t *infop, int options);
```

wcrtomb

Convert a wide-character code to a character (restartable)

```
#include <stdio.h>

size_t wcrtomb(char *restrict s, wchar_t wc, mbstate_t *restrict ps);
```

wcscat

Concatenate two wide-character strings

```
#include <wchar.h>

wchar_t *wcscat(wchar_t *restrict ws1, const wchar_t *restrict ws2);
```

wcschr

Wide-character string scanning operation

```
#include <wchar.h>

wchar_t *wcschr(const wchar_t *ws, wchar_t wc);
```

wcscmp

Compare two wide-character strings

```
#include <wchar.h>

int wcscmp(const wchar_t *ws1, const wchar_t *ws2);
```

wcscoll

Wide-character string comparison using collating information

```
#include <wchar.h>

int wcscoll(const wchar_t *ws1, const wchar_t *ws2);
```

wcscpy

Copy a wide-character string

```
#include <wchar.h>

wchar_t *wcscpy(wchar_t *restrict ws1, const wchar_t *restrict ws2);
```

wcscspn

Get length of a complementary wide substring

```
#include <wchar.h>

size_t wcscspn(const wchar_t *ws1, const wchar_t *ws2);
```

wcsftime

Convert date and time to a wide-character string

```
#include <wchar.h>

size_t wcsftime(wchar_t *restrict wcs, size_t maxsize,
    const wchar_t *restrict format, const struct tm *restrict timeptr);
```

wcslen

Get wide-character string length

```
#include <wchar.h>

size_t wcslen(const wchar_t *ws);
```

wcsncat

Concatenate a wide-character string with part of another

```
#include <wchar.h>

wchar_t *wcsncat(wchar_t *restrict ws1, const wchar_t *restrict ws2,
    size_t n);
```

wcsncmp

Compare part of two wide-character strings

```
#include <wchar.h>

int wcsncmp(const wchar_t *ws1, const wchar_t *ws2, size_t n);
```

wcsncpy

Copy part of a wide-character string

```
#include <wchar.h>

wchar_t *wcsncpy(wchar_t *restrict ws1, const wchar_t *restrict ws2,
    size_t n);
```

wcspbrk

Scan wide-character string for a wide-character code

```
#include <wchar.h>

wchar_t *wcspbrk(const wchar_t *ws1, const wchar_t *ws2);
```

wcsrchr

Wide-character string scanning operation

```
#include <wchar.h>

wchar_t *wcsrchr(const wchar_t *ws, wchar_t wc);
```

wcsrtombs

Convert a wide-character string to a character string (restartable)

```
#include <wchar.h>

size_t wcsrtombs(char *restrict dst, const wchar_t **restrict src,
    size_t len, mbstate_t *restrict ps);
```

wcsspn

Get length of a wide substring

```
#include <wchar.h>

size_t wcsspn(const wchar_t *ws1, const wchar_t *ws2);
```

wcsstr

Find a wide-character substring

```
#include <wchar.h>

wchar_t *wcsstr(const wchar_t *restrict ws1, const wchar_t *restrict ws2);
```

wcstod, wcstof, wcstold

Convert a wide-character string to a double-precision number

```
#include <wchar.h>

double wcstod(const wchar_t *restrict nptr, wchar_t **restrict endptr);
float wcstof(const wchar_t *restrict nptr, wchar_t **restrict endptr);
long double wcstold(const wchar_t *restrict nptr,
    wchar_t **restrict endptr);
```

wcstoimax, wcstoumax

Convert wide-character string to integer type

```
#include <stddef.h>
#include <inttypes.h>

intmax_t wcstoimax(const wchar_t *restrict nptr,
    wchar_t **restrict endptr, int base);
uintmax_t wcstoumax(const wchar_t *restrict nptr,
    wchar_t **restrict endptr, int base);
```

wcstok

Split wide-character string into tokens

```
#include <wchar.h>

wchar_t *wcstok(wchar_t *restrict ws1, const wchar_t *restrict ws2,
    wchar_t **restrict ptr);
```

wcstol, wcstoll

Convert a wide-character string to a long integer

```
#include <wchar.h>

long wcstol(const wchar_t *restrict nptr, wchar_t **restrict endptr,
    int base);
long long wcstoll(const wchar_t *restrict nptr,
    wchar_t **restrict endptr, int base);
```

wcstombs

Convert a wide-character string to a character string

```
#include <stdlib.h>

size_t wcstombs(char *restrict s, const wchar_t *restrict pwcs,
    size_t n);
```

wcstoul, wcstoull

Convert a wide-character string to an unsigned long

```
#include <wchar.h>

unsigned long wcstoul(const wchar_t *restrict nptr,
    wchar_t **restrict endptr, int base);
unsigned long long wcstoull(const wchar_t *restrict nptr,
    wchar_t **restrict endptr, int base);
```

wcswcs

Find a wide substring (**LEGACY**)

XSI
```
#include <wchar.h>

wchar_t *wcswcs(const wchar_t *ws1, const wchar_t *ws2);
```

wcswidth

Number of column positions of a wide-character string

XSI
```
#include <wchar.h>

int wcswidth(const wchar_t *pwcs, size_t n);
```

wcsxfrm

Wide-character string transformation

```
#include <wchar.h>

size_t wcsxfrm(wchar_t *restrict ws1, const wchar_t *restrict ws2,
    size_t n);
```

wctob

Wide-character to single-byte conversion

```
#include <stdio.h>
#include <wchar.h>

int wctob(wint_t c);
```

wctomb

Convert a wide-character code to a character

```
#include <stdlib.h>

int wctomb(char *s, wchar_t wchar);
```

wctrans

Define character mapping

```
#include <wctype.h>

wctrans_t wctrans(const char *charclass);
```

wctype

Define character class

```
#include <wctype.h>

wctype_t wctype(const char *property);
```

wcwidth

Number of column positions of a wide-character code

XSI

```
#include <wchar.h>

int wcwidth(wchar_t wc);
```

wmemchr

Find a wide character in memory

```
#include <wchar.h>

wchar_t *wmemchr(const wchar_t *ws, wchar_t wc, size_t n);
```

wmemcmp

Compare wide characters in memory

```
#include <wchar.h>

int wmemcmp(const wchar_t *ws1, const wchar_t *ws2, size_t n);
```

wmemcpy

Copy wide characters in memory

```
#include <wchar.h>

wchar_t *wmemcpy(wchar_t *restrict ws1, const wchar_t *restrict ws2,
    size_t n);
```

wmemmove

Copy wide characters in memory with overlapping areas

```
#include <wchar.h>

wchar_t *wmemmove(wchar_t *ws1, const wchar_t *ws2, size_t n);
```

wmemset

Set wide characters in memory

```
#include <wchar.h>

wchar_t *wmemset(wchar_t *ws, wchar_t wc, size_t n);
```

wordexp, wordfree

Perform word expansions

```
#include <wordexp.h>

int wordexp(const char *restrict words, wordexp_t *restrict pwordexp,
    int flags);
void wordfree(wordexp_t *pwordexp);
```

pwrite, write

Write on a file

```
#include <unistd.h>

ssize_t pwrite(int fildes, const void *buf, size_t nbyte,
    off_t offset);
ssize_t write(int fildes, const void *buf, size_t nbyte);
```

writev

Write a vector

```
#include <sys/uio.h>

ssize_t writev(int fildes, const struct iovec *iov, int iovcnt);
```

y0, y1, yn

Bessel functions of the second kind

```
#include <math.h>

double y0(double x);
double y1(double x);
double yn(int n, double x);
```

Chapter 2

Utilities Reference

This chapter contains a brief reference for each interface defined in XCU, Issue 6.

admin

Create and administer SCCS files (**DEVELOPMENT**)

XSI
```
admin –i[name] [–n] [–a login] [–d flag] [–e login] [–f flag] [–m mrlist]
    [–r rel] [–t[name] [–y[comment]] newfile
admin –n[–a login] [–d flag] [–e login] [–f flag] [–m mrlist] [–t[name]]
    [–y[comment]] newfile ...
admin [–a login] [–d flag] [–m mrlist] [–r rel] [–t[name]] file ...
admin –h file ...
admin –z file ...
```

alias

Define or display aliases

UP
```
alias [alias-name[=string] ...]
```

ar

Create and maintain library archives

XSI
```
ar –m [–v] archive file ...
ar –m –a[–v] posname archive file ...
ar –m –b[–v] posname archive file ...
ar –m –i[–v] posname archive file ...
```

XSI
```
ar –p[–v] [–s] archive [file ...]
```

XSI
```
ar –q[–cv] archive file ...
```

```
ar –r[–cuv] archive file ...
```

XSI
```
ar –r –a[–cuv] posname archive file ...
ar –r –b[–cuv] posname archive file ...
ar –r –i[–cuv] posname archive file ...
```

XSI
```
ar –t[–v] [–s] archive [file ...]
```

XSI
```
ar –x[–v] [–sCT] archive [file ...]
```

asa

Interpret carriage-control characters

FR `asa [ file ... ]`

at

Execute commands at a later time

UP `at [–m] [–f file] [–q queuename] –t time_arg`

`at [–m] [–f file] [–q queuename] timespec ...`

`at –r at_job_id ...`

`at –l –q queuename`

`at –l [at_job_id ...]`

awk

Pattern scanning and processing language

`awk [–F ERE] [–v assignment] ... program [argument ...]`

`awk [–F ERE] –f progfile ... [–v assignment] ...[argument ...]`

basename

Return non-directory portion of a pathname

`basename string [suffix]`

batch

Schedule commands to be executed in a batch queue

UP `batch`

bc

Arbitrary-precision arithmetic language

`bc [–l] [file ...]`

bg

Run jobs in the background

UP `bg [job_id ...]`

c99

Compile standard C programs

CD
```
c99 [–c] [–D name[=value]]... [–E] [–g] [–I directory] ... [–L directory]
    ... [–o outfile] [–Ooptlevel] [–s] [–U name]... operand ...
```

cal

Print a calendar

XSI
```
cal [[month] year ]
```

cat

Concatenate and print files

```
cat [–u] [file ...]
```

cd

Change the working directory

```
cd [–L |–P] [directory]
```

```
cd –
```

cflow

Generate a C-language flowgraph (**DEVELOPMENT**)

XSI
```
cflow [–r] [–d num] [–D name[=def]] ... [–i incl] [–I dir] ...
    [–U dir] ... file ...
```

chgrp

Change the file group ownership

```
chgrp [–hR] group file ...
```

```
chgrp –R [–H | –L | –P ] group file ...
```

chmod

Change the file modes

```
chmod [–R] mode file ...
```

chown

Change the file ownership

```
chown [–hR] owner[:group] file ...
chown –R [–H | –L | –P ] owner[:group] file ...
```

cksum

Write file checksums and sizes

```
cksum [file ...]
```

cmp

Compare two files

```
cmp [ –l | –s ] file1 file2
```

comm

Select or reject lines common to two files

```
comm [–123] file1 file2
```

command

Execute a simple command

```
command [–p] command_name [argument ...]
```

UP

```
command [ –v | –V ] command_name
```

compress

Compress data

XSI

```
compress [–fv][–b bits][file ...]
compress [–cfv][–b bits][file]
```

cp

Copy files

```
cp [–fip] source_file target_file
cp [–fip] source_file ... target
cp –R [–H | –L | –P][–fip] source_file ... target
```

OB

```
cp -r [–H | –L | –P][–fip] source_file ... target
```

crontab

Schedule periodic background work

UP
```
crontab [file]
crontab [ –e | –l | –r ]
```

csplit

Split files based on context

UP
```
csplit [–ks] [–f prefix] [–n number] file arg1 ...argn
```

ctags

Create a tags file (**DEVELOPMENT**, **FORTRAN**)

UP
```
ctags [–a] [–f tagsfile] pathname ...
ctags –x pathname ...
```

cut

Cut out selected fields of each line of a file

```
cut –b list [–n] [file ...]
cut –c list [file ...]
cut –f list [–d delim] [–s] [file ...]
```

cxref

Generate a C-language program cross-reference table (**DEVELOPMENT**)

XSI
```
cxref [–cs] [–o file] [–w num] [–D name[=def]] ... [–I dir] ...
    [–U name] ... file ...
```

date

Write the date and time

```
date [–u] [+format]
```

XSI
```
date [–u] mmddhhmm[[cc]yy]
```

dd

Convert and copy a file

```
dd [operand ...]
```

delta

Make a delta (change) to an SCCS file (**DEVELOPMENT**)

XSI

```
delta [-nps][-g list][-m mrlist][-r SID][-y[comment]] file...
```

df

Report free disk space

UP XSI

```
df [-k][-P|-t][file...]
```

diff

Compare two files

```
diff [-c| -e| -f| -C n][-br] file1 file2
```

dirname

Return the directory portion of pathname

```
dirname string
```

du

Estimate file space usage

UP

```
du [-a | -s][-kx][-H | -L][file ...]
```

echo

Write arguments to standard output

```
echo [string ...]
```

ed

Edit text

```
ed [-p string][-s][file]
```

env

Set the environment for command invocation

```
env [–i] [name=value]... [utility [argument...]]
```

ex

Text editor

UP

```
ex [–rR] [–l] [–s | –v] [–c command] [–t tagstring] [–w size] [file ...]
```

expand

Convert tabs to spaces

UP

```
expand [–t tablist] [file ...]
```

expr

Evaluate arguments as an expression

```
expr operand
```

false

Return false value

```
false
```

fc

Process the command history list

UP

```
fc [–r] [–e editor] [first [last]]
fc –l [–nr] [first [last]]
fc –s [old=new] [first]
```

fg

Run jobs in the foreground

UP

```
fg [job_id]
```

file

Determine file type

UP
```
file [–dh] [–M file] [–m file] file ...
file –i [–h] file ...
```

find

Find files

```
find [–H | –L] path ... [operand_expression ...]
```

fold

Filter for folding lines

```
fold [–bs] [–w width] [file...]
```

fort77

FORTRAN compiler (**FORTRAN**)

FD
```
fort77 [–c] [–g] [–L directory]... [–O optlevel] [–o outfile] [–s] [–w]
    operand...
```

fuser

List process IDs of all processes that have one or more files open

XSI
```
fuser [ –cfu ] file ...
```

gencat

Generate a formatted message catalog

XSI
```
gencat catfile msgfile...
```

get

Get a version of an SCCS file (**DEVELOPMENT**)

XSI
```
get [–begkmnlLpst] [–c cutoff] [–i list] [–r SID] [–x list] file...
```

getconf

Get configuration values

```
getconf [ –v specification ] system_var
getconf [ –v specification ] path_var pathname
```

getopts

Parse utility options

```
getopts optstring name [arg...]
```

grep

Search a file for a pattern

```
grep [–E| –F][–c| –l| –q][–insvx] –e pattern_list...
    [–f pattern_file]...[file...]
grep [–E| –F][–c| –l| –q][–insvx][–e pattern_list]...
    –f pattern_file...[file...]
grep [–E| –F][–c| –l| –q][–insvx] pattern_list[file...]
```

hash

Remember or report utility locations

XSI

```
hash [utility...]
hash –r
```

head

Copy the first part of files

```
head [–n number][file...]
```

iconv

Codeset conversion

```
iconv [–cs] –f frommap –t tomap [file ...]
iconv –f fromcode [–cs] [–t tocode [file ...]
iconv –t tocode [–cs] [–f fromcode] [file ...]
iconv –l
```

id

Return user identity

```
id [user]
id –G[–n] [user]
id –g[–nr] [user]
id –u[–nr] [user]
```

ipcrm

Remove an XSI message queue, semaphore set, or shared memory segment identifier

XSI
```
ipcrm [ –q msgid | –Q msgkey | –s semid | –S semkey |
    –m shmid | –M shmkey ] ...
```

ipcs

Report XSI interprocess communication facilities status

XSI
```
ipcs [–qms] [–a | –bcopt]
```

jobs

Display status of jobs in the current session

UP
```
jobs [–l| –p] [job_id...]
```

join

Relational database operator

```
join [–a file_number | –v file_number] [–e string] [–o list] [–t char]
    [–1 field] [–2 field] file1 file2
```

kill

Terminate or signal processes

```
kill –s signal_name pid ...
kill –l [exit_status]
```

XSI
```
kill [–signal_name] pid ...
kill [–signal_number] pid ...
```

lex

Generate programs for lexical tasks (**DEVELOPMENT**)

CD `lex [-t] [-n|-v] [file ...]`

link

Call *link*( ) function

XSI `link file1 file2`

ln

Link files

`ln [-fs] source_file target_file`

`ln [-fs] source_file ... target_dir`

locale

Get locale-specific information

`locale [-a| -m]`

`locale [-ck] name...`

localedef

Define locale environment

`localedef [-c] [-f charmap] [-i sourcefile] [-u code_set_name] name`

logger

Log messages

`logger string ...`

logname

Return the user's login name

`logname`

lp

Send files to a printer

`lp [-c] [-d dest] [-n copies] [-msw] [-o option]... [-t title] [file...]`

ls

List directory contents

XSI `ls [–CFRacdilqrtu1][–H | –L ][–fgmnopsx][*file*...]`

m4

Macro processor (**DEVELOPMENT**)

XSI `m4 [–s][–D *name*[=*val*]]...[–U *name*]... *file*...`

mailx

Process messages

make

Maintain, update, and regenerate groups of programs (**DEVELOPMENT**)

SD `make [–einpqrst][–f *makefile*]...[ –k| –S][*macro*=*value*]...`
`[*target_name*...]`

man

Display system documentation

`man [–k] *name*...`

mesg

Permit or deny messages

UP `mesg [y|n]`

mkdir

Make directories

`mkdir [–p][–m *mode*] *dir*...`

mkfifo

Make FIFO special files

`mkfifo [–m *mode*] *file*...`

more

Display files on a page-by-page basis

UP
```
more [–ceisu] [–n number] [–p command] [–t tagstring] [file ...]
```

mv

Move files

```
mv [–fi] source_file target_file
mv [–fi] source_file... target_file
```

newgrp

Change to a new group

UP
```
newgrp [–l] [group
```

nice

Invoke a utility with an altered nice value

UP
```
nice [–n increment] utility [argument...]
```

nl

Line numbering filter

XSI
```
nl [–p] [–b type] [–d delim] [–f type] [–h type] [–i incr] [–l num] [–n format]
    [–s sep] [–v startnum] [–w width] [file]
```

nm

Write the name list of an object file (**DEVELOPMENT**)

UP SD XSI
```
nm [–APv] [–efox] [ –g| –u] [–t format] file...
```

nohup

Invoke a utility immune to hangups

```
nohup utility [argument...]
```

od

Dump files in various formats

```
od [–v] [–A address_base] [–j skip] [–N count] [–t type_string]...
    [file...]
```

XSI
```
od [–bcdosx] [file] [[+]offset[.][b]]
```

paste

Merge corresponding or subsequent lines of files

```
paste [–s] [–d list] file...
```

patch

Apply changes to files

UP
```
patch [–blNR] [ –c| –e| –n] [–d dir] [–D define] [–i patchfile]
    [–o outfile] [–p num] [–r rejectfile] [file]
```

pathchk

Check pathnames

```
pathchk [–p] pathname...
```

pax

Portable archive interchange

```
pax [–cdnv] [–H|–L] [–f archive] [–s replstr]...[pattern...]
```

```
pax –r[–cdiknuv] [–H|–L] [–f archive] [–o options]...[–p string]...
    [–s replstr]...[pattern...]
```

```
pax –w[–dituvX] [–H|–L] [–b blocksize] [[–a] [–f archive] [–o options]...
    [–s replstr]...[–x format] [file...]
```

```
pax –r –w[–diklntuvX] [–H|–L] [–p string]...[–s replstr]...
    [file...] directory
```

pr

Print files

XSI
```
pr [+page] [–column] [–adFmrt] [–e[char] [gap]] [–h header] [–i[char] [gap]]
    [–l lines] [–n[char] [width]] [–o offset] [–s[char]] [–w width] [–fp]
    [file...]
```

printf

Write formatted output

```
printf format[argument...]
```

prs

Print an SCCS file (**DEVELOPMENT**)

XSI
```
prs [-a] [-d dataspec] [-r[SID]] file...
```

XSI
```
prs [ -e| -l] -c cutoff [-d dataspec] file...
```

XSI
```
prs [ -e| -l] -r[SID] [-d dataspec]file...
```

ps

Report process status

UP XSI
```
ps [-aA] [-defl] [-G grouplist] [-o format]...[-p proclist] [-t termlist]
[-U userlist] [-g grouplist] [-n namelist] [-u userlist]
```

pwd

Return working directory name

```
pwd [-L | -P ]
```

qalter

Alter batch job

BE
```
qalter [-a date_time] [-A account_string] [-c interval] [-e path_name]
    [-h hold_list] [-j join_list] [-k keep_list] [-l resource_list]
    [-m mail_options] [-M mail_list] [-N name] [-o path_name]
    [-p priority] [-r y|n] [-S path_name_list] [-u user_list]
    job_identifier ...
```

qdel

Delete batch jobs

BE
```
qdel job_identifier ...
```

qhold

Hold batch jobs

BE
```
qhold [–h hold_list] job_identifier ...
```

qmove

Move batch jobs

BE
```
qmove destination job_identifier ...
```

qmsg

Send message to batch jobs

BE
```
qmsg [–E] [–O] message_string job_identifier ...
```

qrerun

Rerun batch jobs

BE
```
qrerun job_identifier ...
```

This utility is part of the Batch Environment Services and Utilities option and may not be available on all implementations.

qrls

Release batch jobs

BE
```
qrls [–h hold_list] job_identifier ...
```

qselect

Select batch jobs

BE
```
qselect [–a [op]date_time] [–A account_string] [–c [op]interval]
    [–h hold_list] [–l resource_list] [–N name] [–p [op]priority]
    [–q destination] [–r y|n] [–s states] [–u user_list]
```

qsig

Signal batch jobs

BE
```
qsig [–s signal] job_identifier ...
```

qstat

Show status of batch jobs

BE

```
qstat [–f] job_identifier ...
qstat –Q [–f] destination ...
qstat –B [–f] server_name ...
```

qsub

Submit a script

BE

```
qsub [–a date_time] [–A account_string] [–c interval]
    [–C directive_prefix] [–e path_name] [–h] [–j join_list] [–k keep_list]
    [–m mail_options] [–M mail_list] [–N name]
    [–o path_name] [–p priority] [–q destination] [–r y|n]
    [–S path_name_list] [–u user_list] [–v variable_list] [–V]
    [–z] [script]
```

read

Read a line from standard input

```
read [–r] var...
```

renice

Set nice values of running processes

UP

```
renice –n increment [–g | –p | –u] ID ...
```

rm

Remove directory entries

```
rm [–fiRr] file...
```

rmdel

Remove a delta from an SCCS file (**DEVELOPMENT**)

XSI

```
rmdel –r SID file...
```

rmdir

Remove directories

```
rmdir [–p] dir...
```

sact

Print current SCCS file-editing activity (**DEVELOPMENT**)

XSI

```
sact file...
```

sccs

Front end for the SCCS subsystem (**DEVELOPMENT**)

XSI

```
sccs [–r] [–d path] [–p path] command [options...] [operands...]
```

sed

Stream editor

```
sed [–n] script[file...]
sed [–n] [–e script]... [–f script_file]... [file...]
```

sh

Shell, the standard command language interpreter

```
sh [–abCefhimnuvx] [–o option] [+abCefhmnuvx] [+o option]
    [command_file [argument...]]
sh –c [–abCefhimnuvx] [–o option] [+abCefhimnuvx] [+o option] command_string
    [command_name [argument...]]
sh –s [–abCefhimnuvx] [–o option] [+abCefhimnuvx] [+o option] [argument]
```

sleep

Suspend execution for an interval

```
sleep time
```

sort

Sort, merge, or sequence check text files

```
sort [–m] [–o output] [–bdfinru] [–t char] [–k keydef]... [file...]
sort –c [–bdfinru] [–t char] [–k keydef] [file]
```

split

Split files into pieces

UP

```
split [–l line_count] [–a suffix_length] [file [name]]
split –b n[k|m] [–a suffix_length] [file [name]]
```

strings

Find printable strings in files

UP

```
strings [–a] [–t format] [–n number] [file...]
```

strip

Remove unnecessary information from executable files (**DEVELOPMENT**)

SD

```
strip file...
```

stty

Set the options for a terminal

```
stty [ –a| –g]
stty operands
```

tabs

Set terminal tabs

UP XSI

```
tabs [ –n| –a| –a2| –c| –c2| –c3| –f| –p| –s| –u] [+m[n]] [–T type]
tabs [–T type] [ +[n]] n1[,n2,...]
```

tail

Copy the last part of a file

```
tail [–f] [ –c number| –n number] [file]
```

talk

Talk to another user

UP

```
talk address [terminal]
```

tee

Duplicate standard input

```
tee [–ai][file...]
```

test

Evaluate expression

```
test [expression]
[ [expression] ]
```

time

Time a simple command

UP

```
time [–p] utility [argument...]
```

touch

Change file access and modification times

```
touch [–acm][ –r ref_file| –t time] file...
```

tput

Change terminal characteristics

UP

```
tput [–T type] operand...
```

tr

Translate characters

```
tr [–c | –C][–s] string1 string2
tr –s [–c | –C] string1
tr –d [–c | –C] string1
tr –ds [–c | –C] string1 string2
```

true

Return true value

```
true
```

tsort

Topological sort

XSI `tsort [file]`

tty

Return user's terminal name

`tty`

type

Write a description of command type

XSI `type name...`

ulimit

Set or report file size limit

XSI `ulimit [-f] [blocks]`

umask

Get or set the file mode creation mask

`umask [-S] [mask]`

unalias

Remove alias definitions

UP `unalias alias-name...`

`unalias -a`

uname

Return system name

`uname [-snrvma]`

uncompress

Expand compressed data

XSI `uncompress [–cfv] [file...]`

unexpand

Convert spaces to tabs

UP `unexpand [ –a| –t tablist] [file...]`

unget

Undo a previous get of an SCCS file (**DEVELOPMENT**)

XSI `unget [–ns] [–r SID] file...`

uniq

Report or filter out repeated lines in a file

`uniq [–c|–d|–u] [–f fields] [–s char] [input_file [output_file]]`

unlink

Call the *unlink*( ) function

XSI `unlink file`

uucp

System-to-system copy

XSI `uucp [–cCdfjmr] [–n user] source-file... destination-file`

uudecode

Decode a binary file

UP `uudecode [–o outfile] [file]`

uuencode

Encode a binary file

UP

```
uuencode [–m] [file] decode_pathname
```

uustat

uucp status inquiry and job control

XSI

```
uustat [ –q| –k jobid| –r jobid]
uustat [–s system] [–u user]
```

uux

Remote command execution

XSI

```
uux [–np] command–string
uux [–jnp] command–string
```

val

Validate SCCS files (**DEVELOPMENT**)

XSI

```
val –
val [–s] [–m name] [–r SID] [–y type] file...
```

vi

Screen-oriented (visual) display editor

UP

```
vi [–rR] [–l] [–c command] [–t tagstring] [–w size] [file ...]
```

wait

Await process completion

```
wait [pid...]
```

wc

Word, line, and byte or character count

```
wc [–c|–m] [–lw] [file...]
```

what

Identify SCCS files (**DEVELOPMENT**)

XSI
```
what [–s] file...
```

who

Display who is on the system

UP
```
who [–mTu]
```

XSI
```
who [–mu]–s[–bHlprt][file]
who [–mTu][–abdHlprt][file]
who –q [file]
who am i
who am I
```

write

Write to another user

UP
```
write user_name [terminal]
```

xargs

Construct argument lists and invoke utility

XSI
```
xargs [–t][–p]][–E eofstr][–I replstr][–L number][–n number [–x]]
    [–s size][utility [argument...]]
```

yacc

Yet another compiler compiler (**DEVELOPMENT**)

CD
```
yacc [–dltv][–b file_prefix][–p sym_prefix] grammar
```

zcat

Expand and concatenate data

XSI
```
zcat [file...]
```

Index