

C Standard Libraries

1 Abstract

This file lists the contents of the C standard libraries that are part of the ANSI standard. Very brief descriptions are given for each feature; see the man-pages for more information.

The library facilities are divided into groups according to their functionality. The header file that is necessary to get access to the facilities are given in the head of each section (together with the flag that must be given to the C compiler, when applicable).

Every group includes information about the following:

- Functions available (the function prototype is given).
- Constants defined in the header file (constants are C macros that take no arguments).
- Macros defined in the header file.
- Types defined in the header file (including structures).
- Global variables that are available to the programmer.

Note that some often used constants are defined in several header files.

(This list is based on chapters 10 - 20 of the book "C, a Reference Manual" by Samuel P. Harbison and Guy L. Steele jr. (3rd ed., Prentice Hall 1991).)

Contents

1	Abstract	1
2	Common definitions (<code>stddef.h</code>)	3
3	Input/output (<code>stdio.h</code>)	4
4	General utilities (<code>stdlib.h</code>)	7
5	String handling (<code>string.h</code>)	9
6	Character handling (<code>ctype.h</code>)	11
7	Date and time (<code>time.h</code>)	12
8	Mathematics (<code>math.h</code> , <code>-lm</code>)	13
9	Errors (<code>errno.h</code>)	14
10	cs (<code>assert.h</code>)	15
11	arguments (<code>stdarg.h</code>)	16
12	Signal handling (<code>signal.h</code>)	17
13	Non-local jumps (<code>setjmp.h</code>)	18
14	Localization (<code>locale.h</code>)	19
15	Facilities	20

2 Common definitions (stddef.h)

Functions:

None

Constants:

NULL

Null pointer.

Macros:

offsetof(struct, member)

Offset (in bytes) of member in struct.

Types:

ptrdiff_t
size_t
wchar_t

Type of subtraction of two pointers.
Type of result of sizeof operator.
Type for 'wide characters'.

Global variables:

None

3 Input/output (stdio.h)

Functions:

```
int remove(const char *filename);
int rename(const char *oldname,
           const char *newname);
FILE *tmpfile(void);
char *tmpnam(char *buf);
int fclose(FILE *stream);
int fflush(FILE *stream);
FILE *fopen(const char *filename,
            const char *mode);
FILE *freopen(const char *filename,
              const char *mode, FILE *stream);
void setbuf(FILE *stream, char *buf);
```

Removes a file.
Renames file oldname to newname.

Creates a new temporary file.
Closes stream.

Empties stream's buffers.
Opens filename in a certain mode (e.g. "r" for reading, "w" for writing).

Opens filename in a certain mode and reassociates stream with it.

Tells system to not buffer stream if buf is NULL; otherwise use strategy -IOFBF with buf as buffer and size BUFSIZE.

Tells system to use buffering strategy bufmode for stream, using buffer buf of size size.

Formatted output to stream according to format.

Formatted input from stream according to format.

Formatted output to stdout according to format.

Formatted input from stdin according to format.

Formatted output to string s according to format.

Formatted input from string s according to format.

Like printf(), except that extra arguments are taken from arg.

Like fprintf(), except that extra arguments are taken from arg.

Like sprintf(), except that extra arguments are taken from arg.

Reads next character from stream.

Reads a line from stream (at most n-1 characters) and puts in string s.

Writes character c on stream.

Writes string s on stream.

Like fgetc(), but may be a macro.

Reads next character from stdin.

Reads a line from stdin and puts in string s.

Like fputc(), but may be a macro.

Writes character c on stdout.

Writes string s on stdout.

```

int ungetc(int c, FILE *stream);
size_t fread(void *data, size_t elt_size,
             size_t count, FILE *stream);

size_t fwrite(const void *data,
              size_t elt_size, size_t count,
              FILE *stream);
int fgetpos(FILE *stream, fpos_t *pos);

int fseek(FILE *stream, long int offset,
          int wherefrom);

int fsetpos(FILE *stream,
            const fpos_t *pos);
long int ftell(FILE *stream);
void rewind(FILE *stream);

int feof(FILE *stream);

int ferror(FILE *stream);
void clearerr(FILE *stream);
void perror(const char *s);

```

Types:

FILE
fpos_t
size_t

Type for streams.
Type for file positions.
Type of result of sizeof operator.

Global variables:

stdin
stdout
stderr

Standard input stream ('normal input').
Standard output stream ('normal output').
Stream for error messages.

Constants:

```

_IOFBF
_IOLBF
_IONBF
BUFSIZ
EOF
FILENAME_MAX
FOPEN_MAX
L_tmpnam
TMP_MAX
SEEK_CUR
SEEK_END
SEEK_SET
NULL

```

Puts character c back on stream
 Reads count elements of size elt_size from stream and puts them in array data.
 Writes count elements of size elt_size from array data to stream.
 Stores the 'current position' of stream in pos.
 Sets the current position of stream to offset characters from beginning of file, end of file or current position (wherefrom should be one of SEEK_SET, SEEK_END or SEEK_CUR)
 Sets the current position of stream to the position that pos points to.
 Returns current position in stream.
 Sets current position of stream to the beginning of file.
 True if current position of stream is at the end of file.
 Returns error status of stream
 Resets the error status of stream.
 Writes s followed by an error message depending on errno on stderr.
 Buffering strategy: Full buffering.
 Buffering strategy: Buffer one line.
 Buffering strategy: No buffering.
 Buffer size
 End of file.
 Maximum length of a filename.
 Maximum number of streams that can be open simultaneously.
 Maximum length of a temporary file name (from tmpnam()).
 Number of successive calls to tmpnam() that generates unique names.
 fseek(): seek from current position.
 fseek(): seek from end of file.
 fseek(): seek from beginning of file.
 Null pointer.

Macros:

None

4 General utilities (stdlib.h)

Functions:

```
double atof(const char *str);
int atoi(const char *str);
long int atol(const char *str);
double strtod(const char *str,
             char **ptr);

long int strtol(const char *str,
               char **ptr, int base);

unsigned long int strtoul(const char *str,
                        char **ptr, int base);
int rand(void);

void srand(unsigned int seed);
void *malloc(size_t size);

void *calloc(size_t elt_count,
             size_t elt_size);
void free(void *ptr);
void *realloc(void *ptr, size_t size);
void exit(int status);

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

char *getenv(const char *name);

int system(const char *command);
void *bsearch(const void *key,
             const void *base, size_t count,
             size_t size, int (*compar)
              (const void *the_key,
               const void *a_value));
void qsort(void *base, size_t count,
           size_t size, int (*compar)
            (const void *element1,
             const void *element2));

int abs(int x);
long int labs(long int x);
ldiv_t div(int n, int d);

ldiv_t ldiv(long int n, long int d);
```

the division of n by d .
Returns the size of multibyte character s , which is not more than n characters.
Converts multibyte characters, to a wide character pointed to by pwc .
Converts the wide character $wchar$ to multibyte character s .
Converts n multibyte characters in s to wide characters, and stores them in the array $pwcs$.
Converts n wide characters in $pwcs$ to multibyte characters, and stores them in the array s .

```
int mblen(const char *s, size_t n);
int mbtowc(wchar_t *pwc, const char *s,
           size_t n);
int wctomb(char *s, wchar_t wchar);
size_t mbstowcs(wchar_t *pwc,
                const char *s, size_t n);
size_t wcstombs(char *s,
                const wchar_t *pwc, size_t n);
```

Constants:

```
EXIT_FAILURE
EXIT_SUCCESS
MB_CUR_MAX
NULL
RAND_MAX
```

Return value of program if it failed.
Return value of program if it succeeded.
Maximum length of a multibyte character.
Null pointer.
Maximum value returned by `rand()`.

Macros:

```
None
```

Types:

```
div_t
ldiv_t
size_t
wchar_t
```

Structure containing quotient and remainder (integers).
Structure containing quotient and remainder (long integers).
Type of result of sizeof operator.
Type for 'wide characters'.

Global variables:

```
None
```

Converts string str to double.
Converts string str to integer.
Converts string str to long.
Converts string str to double, and sets ptr to point to next character in str after the number.
Converts string str to integer, and sets ptr to point to next character in str after the number. The number in str should be radix base.
Returns a pseudo random number between 0 and `RAND_MAX`. (This function may not be as good as it ought to. It may often be necessary to use some other random number generator, e.g. `drand48`.)
Initializes `rand()` with seed.
Allocates memory for an object of size $size$.
Allocates memory for elt_count elements of size elt_size .
Deallocates memory pointed to by ptr .
Changes the size of the allocated memory pointed to by ptr to $size$.
Terminate the program and return status.
Aborts the program.
Tells the system to call the function $func$ when the program terminates.
Returns the value of environment variable name.
Execute command in the shell.
Searches an array (which must be sorted) of count elements (of size $size$) starting at base for key.
The function $compar$ is used to compare elements with the key.
Sorts an array of count elements (of size $size$) starting at base. The function $compar$ gives the ordering.
Returns the absolute value of x .
Returns the absolute value of x .
Returns the quotient and remainder of the division of n by d .
Returns the quotient and remainder of

5 String handling (string.h)

Functions:

```
size_t strlen(const char *s);           Returns the number of characters in
                                       the string s.
int strcmp(const char *s1,             Lexicographically compares the two
const char *s2);                      strings s1 and s2.
int strncmp(const char *s1,           Lexicographically compares up to n
const char *s2, size_t n);           characters of string s1 with up to n
                                       characters of string s2.
char *strcpy(char *dest, const char *src); Copies the string src to the string
                                       dest.
char *strncpy(char *dest, const char *src, size_t n); Copies n characters from the string
                                       src to the string dest.
char *strcat(char *dest, const char *src); Appends the string src to the end of
                                       the string dest.
char *strncat(char *dest, const char *src, size_t n); Appends up to n characters from the
                                       string s to the end of the string dest.
char *strchr(const char *s, int c);   Searches the string s for character c.
                                       Returns pointer to first occurrence.
char *strrchr(const char *s, int c);  Searches the string s for character c.
                                       Returns pointer to last occurrence.
size_t strspn(const char *s,          Searches the string s for the first
const char *set);                    character that is not included in set.
                                       Returns the number of characters
                                       that were skipped.
size_t strscpn(const char *s,         Searches the string s for the first
const char *set);                    character that is included in set.
                                       Returns the number of characters
                                       that were skipped.
char *strpbrk(const char *s,          Like strcspn(), but returns a pointer
const char *set);                    to the first character found from set.
char *strstr(const char *src,        Locates the first occurrence of string
const char *sub);                    sub in the string src.
char *strtok(char *str, const char *set); Separates the string str into tokens
                                       separated by characters from set.
void *memcpy(void *dest, const void *src, size_t len); Copies len bytes from src to dest.
void *memmove(void *dest, const void *src, size_t len); Copies len bytes from src to dest
                                       (src and dest may overlap).
int memcmp(const void *ptr1,         Compares len bytes beginning at ptr1
const void *ptr2, size_t len);       with len bytes beginning at ptr2.
void *memchr(const void *ptr, int val, size_t len); Searches for val in the len bytes
                                       beginning at ptr.
void *memset(void *ptr, int val,     Copies val into len bytes starting
size_t len);                          at ptr.
char *strerror(int errnum);           Returns a pointer to an error message
                                       that explains error errnum.
int strcoll(const char *s1,          Compares the two strings s1 and s2
const char *s2);                     according to the current locale.
size_t strxfrm(char *dest,           Transforms the string src so that
```

```
const char *src, size_t len);
```

```
strcmp() can be used instead of
strcoll() and puts the result in dest.
```

Constants:

```
NULL
```

```
Null pointer.
```

Macros:

```
None
```

Types:

```
size_t
```

```
Type of result of sizeof operator.
```

Global variables:

```
None
```

6 Character handling (ctype.h)

Functions:

```
int isalnum(int c);
int isalpha(int c);
int iscntrl(int c);
int isdigit(int c);
int isgraph(int c);
int islower(int c);
int isprint(int c);
int ispunct(int c);
int isspace(int c);
int isupper(int c);
int isxdigit(int c);
int tolower(int c);
int toupper(int c);
```

True if c is alphanumeric character.
True if c is alphabetic character.
True if c is 'control character'.
True if c is decimal digit.
True if c is printable and not space.
True if c is lowercase letter.
True if c is printable.
True if c is 'punctuation character'.
True if c is 'whitespace character'.
True if c is uppercase letter.
True if c is hexadecimal digit.
Returns c converted to lowercase.
Returns c converted to uppercase.

Constants:

None

Macros:

None

Types:

None

Global variables:

None

7 Date and time (time.h)

Functions:

```
clock_t clock(void);
time_t time(time_t *tptr);
double difftime(time_t t1, time_t t0);
time_t mktime(struct tm *tmptr);
struct tm *gmtime(const time_t *);
struct tm *localtime(const time_t *);
char *asctime(const struct tm *ts);
char *ctime(const time_t *tmptr);
size_t strftime(char *s, size_t maxsize,
                const char *format,
                const struct tm *timeptr);
```

Returns the processor time used by the current process.
Returns the the current calendar time, and stores it at tptr.
Returns the number of seconds from time t0 to time t1.
Converts a time value from a tm structure to type time_t.
Converts a time value of type time_t to a tm structure (Greenwich mean time).
Converts a time value of type time_t to a tm structure (local time).
Returns a pointer to a string that contains time ts in printable form.
Returns a pointer to a string that contains time tmptr in printable form.
Stores the time value at timeptr in printable form in s according to format. At most maxsize characters are written in s.

Constants:

```
CLOCKS_PER_SEC
NULL
```

Number of time units per second.
Null pointer.

Macros:

None

Types:

```
clock_t
time_t
size_t
struct tm
```

Type for processor time.
Type for calendar time.
Type of result of sizeof operator.
Structure for calendar time.

Global variables:

None

8 Mathematics (math.h, -lm)

Functions:

double acos(double x); Returns arc cosine of x.
double asin(double x); Returns arc sine of x.
double atan(double x); Returns arc tangent of x.
double atan2(double y, double x); Returns arc tangent of y/x.
double cos(double x); Returns cosine of x.
double sin(double x); Returns sine of x.
double tan(double x); Returns tangent of x.
double cosh(double x); Returns hyperbolic cosine of x.
double sinh(double x); Returns hyperbolic sine of x.
double tanh(double x); Returns hyperbolic tangent of x.
double exp(double x); Returns e to the power of x.
double frexp(double x, int *nptr); Splits x into fraction and exponent.
double ldexp(double x, int n); Returns x times 2ⁿ.
double log(double x); Returns the natural logarithm of x.
double log10(double x); Returns the base-10 logarithm of x.
double modf(double x, double *nptr); Splits x into fraction and integer.
double pow(double x, double y); Returns x to the power of y.
double sqrt(double x); Returns the square root of x (>= 0).
double ceil(double x); Returns floor of x rounded up to integer >= x.
double fabs(double x); Returns the absolute value of x.
double floor(double x); Returns largest integer <= x.
double fmod(double x, double y); Returns f so that x = ky+f, for some k.

Constants:

HUGE_VAL

Returned if value is too large to be represented.

Macros:

None

Types:

None

Global variables:

None

9 Errors (errno.h)

Functions:

None

Constants:

EDOM
ERANGE
...

Error 'argument not in domain'.
Error 'out of range'.
(Many other error codes often defined).

Macros:

None

Types:

None

Global variables:

errno

Latest error code.

10 Cs (assert.h)

Functions:

None

Constants:

NDEBUG

Define to turn off assert facility.

Macros:

assert(expression)

Check that expression is true (!= 0).

Types:

None

Global variables:

None

11 Arguments (stdarg.h)

Functions:

None

Constants:

None

Macros:

va_start(ap, last_fixed_arg)

Initializes the va_list ap to point to the next argument after last_fixed_arg

va_arg(ap, type)

Returns next argument which should be of type type.

va_end(ap)

Cleans up after last call to va_arg.

Types:

va_list

Local state variable used to traverse the arguments.

Global variables:

None

12 Signal handling (signal.h)

Functions:

```
void (*signal(int sig, void (*func) (int)))(int);  
    Associates signal handler func with  
    signal sig. Returns old handler.  
int raise(int sig);  
    Raises signal sig.
```

Constants:

```
SIG_DFL    Default handling of signals.  
SIG_ERR    Error in signal().  
SIG_IGN    Ignore signal.  
SIGABRT    Signal: abort.  
SIGFPE     Signal: division by zero.  
SIGILL     Signal: illegal instruction.  
SIGINT     Signal: interrupt.  
SIGSEGV    Signal: segmentation fault.  
SIGTERM    Signal: termination.  
...        (Many other signals often defined.)
```

Macros:

None

Types:

```
sig_atomic_t
```

Global variables:

None

13 Non-local jumps (setjmp.h)

Functions:

```
int setjmp(jmp_buf env);  
    Records caller's environment in 'jump  
    buffer' env and returns 0.  
void longjmp(jmp_buf env, int status);  
    Jumps to 'jump buffer' env and  
    returns status.
```

Constants:

None

Macros:

None

Types:

```
jmp_buf    'Jump buffer'
```

Global variables:

None

14 Localization (locale.h)

Functions:

```
char *setlocale(int category,  
               const char *locale);  
struct lconv *localeconv(void);
```

Change locale-specific behavior of category to locale.
Returns info. about how numeric and monetary values are handled in the current locale.

Constants:

```
LC_ALL  
LC_COLLATE  
LC_CTYPE  
LC_MONETARY  
LC_NUMERIC  
LC_TIME  
NULL
```

Category: all behavior.
Category: strcoll() and strxfrm().
Category: character functions.
Category: monetary information.
Category: numeric information.
Category: strftime().
Null pointer.

Macros:

None

Types:

```
struct lconv
```

Contains info. about how numeric and monetary values are handled in the current locale.

Global variables:

None

15 Facilities

Functions:

None

Constants:

```
--DATE--  
--FILE--  
--LINE--  
--TIME--  
--STDC--
```

String with date of compilation.
String with name of source file.
Integer with current line number.
String with time of compilation.
Defined in ANSI C compilers.

Macros:

None

Types:

None

Global variables:

None