

GNU libiberty

---

Phil Edwards et al.

---

Copyright © 2001-2025 Free Software Foundation, Inc.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover Texts, and with no Back-Cover Texts. A copy of the license is included in the section entitled “GNU Free Documentation License”.

# Table of Contents

<b>1</b>	<b>Using</b>	<b>1</b>
<b>2</b>	<b>Overview</b>	<b>2</b>
2.1	Supplemental Functions	2
2.2	Replacement Functions	2
2.2.1	Memory Allocation	2
2.2.2	Exit Handlers	2
2.2.3	Error Reporting	2
2.3	Extensions	3
2.3.1	Obstacks	3
2.3.1.1	Creating Obstacks	3
2.3.1.2	Preparing for Using Obstacks	3
2.3.1.3	Allocation in an Obstack	5
2.3.1.4	Freeing Objects in an Obstack	6
2.3.1.5	Obstack Functions and Macros	6
2.3.1.6	Growing Objects	7
2.3.1.7	Extra Fast Growing Objects	8
2.3.1.8	Status of an Obstack	10
2.3.1.9	Alignment of Data in Obstacks	10
2.3.1.10	Obstack Chunks	11
2.3.1.11	Summary of Obstack Macros	11
<b>3</b>	<b>Function, Variable, and Macro Listing</b>	<b>14</b>
<b>Appendix A</b>	<b>Licenses</b>	<b>38</b>
A.1	GNU LESSER GENERAL PUBLIC LICENSE	38
A.1.1	Preamble	38
A.1.2	TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION	39
A.1.3	How to Apply These Terms to Your New Libraries	46
A.2	BSD	47
<b>Index</b>		<b>48</b>

# 1 Using

To date, `libiberty` is generally not installed on its own. It has evolved over years but does not have its own version number nor release schedule.

Possibly the easiest way to use `libiberty` in your projects is to drop the `libiberty` code into your project's sources, and to build the library along with your own sources; the library would then be linked in at the end. This prevents any possible version mismatches with other copies of `libiberty` elsewhere on the system.

Passing `--enable-install-libiberty` to the `configure` script when building `libiberty` causes the header files and archive library to be installed when `make install` is run. This option also takes an (optional) argument to specify the installation location, in the same manner as `--prefix`.

For your own projects, an approach which offers stability and flexibility is to include `libiberty` with your code, but allow the end user to optionally choose to use a previously-installed version instead. In this way the user may choose (for example) to install `libiberty` as part of GCC, and use that version for all software built with that compiler. (This approach has proven useful with software using the GNU `readline` library.)

Making use of `libiberty` code usually requires that you include one or more header files from the `libiberty` distribution. (They will be named as necessary in the function descriptions.) At link time, you will need to add `-liberty` to your link command invocation.

## 2 Overview

Functions contained in `libiberty` can be divided into three general categories.

### 2.1 Supplemental Functions

Certain operating systems do not provide functions which have since become standardized, or at least common. For example, the Single Unix Specification Version 2 requires that the `basename` function be provided, but an OS which predates that specification might not have this function. This should not prevent well-written code from running on such a system.

Similarly, some functions exist only among a particular “flavor” or “family” of operating systems. As an example, the `bzero` function is often not present on systems outside the BSD-derived family of systems.

Many such functions are provided in `libiberty`. They are quickly listed here with little description, as systems which lack them become less and less common. Each function *foo* is implemented in *foo.c* but not declared in any `libiberty` header file; more comments and caveats for each function’s implementation are often available in the source file. Generally, the function can simply be declared as `extern`.

### 2.2 Replacement Functions

Some functions have extremely limited implementations on different platforms. Other functions are tedious to use correctly; for example, proper use of `malloc` calls for the return value to be checked and appropriate action taken if memory has been exhausted. A group of “replacement functions” is available in `libiberty` to address these issues for some of the most commonly used subroutines.

All of these functions are declared in the `libiberty.h` header file. Many of the implementations will use preprocessor macros set by GNU Autoconf, if you decide to make use of that program. Some of these functions may call one another.

#### 2.2.1 Memory Allocation

The functions beginning with the letter ‘x’ are wrappers around standard functions; the functions provided by the system environment are called and their results checked before the results are passed back to client code. If the standard functions fail, these wrappers will terminate the program. Thus, these versions can be used with impunity.

#### 2.2.2 Exit Handlers

The existence and implementation of the `atexit` routine varies amongst the flavors of Unix. `libiberty` provides an unvarying dependable implementation via `xatexit` and `xexit`.

#### 2.2.3 Error Reporting

These are a set of routines to facilitate programming with the system `errno` interface. The `libiberty` source file `strerror.c` contains a good deal of documentation for these functions.



Also, if the source file uses the macro `obstack_init`, it must declare or define two macros that will be called by the obstack library. One, `obstack_chunk_alloc`, is used to allocate the chunks of memory into which objects are packed. The other, `obstack_chunk_free`, is used to return chunks when the objects in them are freed. These macros should appear before any use of obstacks in the source file.

Usually these are defined to use `malloc` via the intermediary `xmalloc` (see Section “Unconstrained Allocation” in *The GNU C Library Reference Manual*). This is done with the following pair of macro definitions:

```
#define obstack_chunk_alloc xmalloc
#define obstack_chunk_free free
```

Though the memory you get using obstacks really comes from `malloc`, using obstacks is faster because `malloc` is called less often, for larger blocks of memory. See Section 2.3.1.10 [Obstack Chunks], page 11, for full details.

At run time, before the program can use a `struct obstack` object as an obstack, it must initialize the obstack by calling `obstack_init` or one of its variants, `obstack_begin`, `obstack_specify_allocation`, or `obstack_specify_allocation_with_arg`.

```
int obstack_init (struct obstack *obstack_ptr) [Function]
    Initialize obstack obstack_ptr for allocation of objects. This macro calls the obstack's obstack_chunk_alloc function. If allocation of memory fails, the function pointed to by obstack_alloc_failed_handler is called. The obstack_init macro always returns 1 (Compatibility notice: Former versions of obstack returned 0 if allocation failed).
```

Here are two examples of how to allocate the space for an obstack and initialize it. First, an obstack that is a static variable:

```
static struct obstack myobstack;
...
obstack_init (&myobstack);
```

Second, an obstack that is itself dynamically allocated:

```
struct obstack *myobstack_ptr
    = (struct obstack *) xmalloc (sizeof (struct obstack));

obstack_init (myobstack_ptr);
```

```
int obstack_begin (struct obstack *obstack_ptr, size_t chunk_size) [Function]
    Like obstack_init, but specify chunks to be at least chunk_size bytes in size.
```

```
int obstack_specify_allocation (struct obstack *obstack_ptr, size_t chunk_size, size_t alignment, void (*chunkfun) (size_t), void (*freefun) (void *)) [Function]
    Like obstack_init, specifying chunk size, chunk alignment, and memory allocation functions. A chunk_size or alignment of zero results in the default size or alignment respectively being used.
```

```
int obstack_specify_allocation_with_arg (struct obstack      [Function]
    *obstack_ptr, size_t chunk_size, size_t alignment, void
    *(*chunkfun) (void *, size_t), void (*freefun) (void *, void
    *), void *arg)
```

Like `obstack_specify_allocation`, but specifying memory allocation functions that take an extra first argument, *arg*.

```
obstack_alloc_failed_handler [Variable]
```

The value of this variable is a pointer to a function that `obstack` uses when `obstack_chunk_alloc` fails to allocate memory. The default action is to print a message and abort. You should supply a function that either calls `exit` (see Section “Program Termination” in *The GNU C Library Reference Manual*) or `longjmp` and doesn’t return.

```
void my_obstack_alloc_failed (void)
...
obstack_alloc_failed_handler = &my_obstack_alloc_failed;
```

### 2.3.1.3 Allocation in an Obstack

The most direct way to allocate an object in an obstack is with `obstack_alloc`, which is invoked almost like `malloc`.

```
void * obstack_alloc (struct obstack *obstack_ptr, size_t      [Function]
    size)
```

This allocates an uninitialized block of *size* bytes in an obstack and returns its address. Here *obstack\_ptr* specifies which obstack to allocate the block in; it is the address of the `struct obstack` object which represents the obstack. Each obstack macro requires you to specify an *obstack\_ptr* as the first argument.

This macro calls the obstack’s `obstack_chunk_alloc` function if it needs to allocate a new chunk of memory; it calls `obstack_alloc_failed_handler` if allocation of memory by `obstack_chunk_alloc` failed.

For example, here is a function that allocates a copy of a string *str* in a specific obstack, which is in the variable `string_obstack`:

```
struct obstack string_obstack;

char *
copystring (char *string)
{
    size_t len = strlen (string) + 1;
    char *s = (char *) obstack_alloc (&string_obstack, len);
    memcpy (s, string, len);
    return s;
}
```

To allocate a block with specified contents, use the macro `obstack_copy`.

```
void * obstack_copy (struct obstack *obstack_ptr, void      [Function]
    *address, size_t size)
```

This allocates a block and initializes it by copying *size* bytes of data starting at *address*. It calls `obstack_alloc_failed_handler` if allocation of memory by `obstack_chunk_alloc` failed.







```
void obstack_ptr_grow_fast (struct obstack *obstack_ptr,      [Function]
                           void *data)
```

`obstack_ptr_grow_fast` adds `sizeof (void *)` bytes containing the value of `data` to the growing object in obstack `obstack_ptr`.

```
void obstack_int_grow_fast (struct obstack *obstack_ptr,      [Function]
                           int data)
```

`obstack_int_grow_fast` adds `sizeof (int)` bytes containing the value of `data` to the growing object in obstack `obstack_ptr`.

```
void obstack_blank_fast (struct obstack *obstack_ptr,         [Function]
                        size_t size)
```

`obstack_blank_fast` adds `size` bytes to the growing object in obstack `obstack_ptr` without initializing them.

When you check for space using `obstack_room` and there is not enough room for what you want to add, the fast growth macros are not safe. In this case, simply use the corresponding ordinary growth macro instead. Very soon this will copy the object to a new chunk; then there will be lots of room available again.

So, each time you use an ordinary growth macro, check afterward for sufficient space using `obstack_room`. Once the object is copied to a new chunk, there will be plenty of space again, so the program will start using the fast growth macros again.

Here is an example:

```
void
add_string (struct obstack *obstack, const char *ptr, size_t len)
{
    while (len > 0)
    {
        size_t room = obstack_room (obstack);
        if (room == 0)
        {
            /* Not enough room. Add one character slowly,
               which may copy to a new chunk and make room. */
            obstack_1grow (obstack, *ptr++);
            len--;
        }
        else
        {
            if (room > len)
                room = len;
            /* Add fast as much as we have room for. */
            len -= room;
            while (room-- > 0)
                obstack_1grow_fast (obstack, *ptr++);
        }
    }
}
```

You can use `obstack_blank_fast` with a “negative” size argument to make the current object smaller. Just don’t try to shrink it beyond zero length—there’s no telling what will happen if you do that. Earlier versions of obstacks allowed you to use `obstack_blank` to shrink objects. This will no longer work.



### 2.3.1.10 Obstack Chunks

Obstacks work by allocating space for themselves in large chunks, and then parceling out space in the chunks to satisfy your requests. Chunks are normally 4096 bytes long unless you specify a different chunk size. The chunk size includes 8 bytes of overhead that are not actually used for storing objects. Regardless of the specified size, longer chunks will be allocated when necessary for long objects.

The obstack library allocates chunks by calling the function `obstack_chunk_alloc`, which you must define. When a chunk is no longer needed because you have freed all the objects in it, the obstack library frees the chunk by calling `obstack_chunk_free`, which you must also define.

These two must be defined (as macros) or declared (as functions) in each source file that uses `obstack_init` (see Section 2.3.1.1 [Creating Obstacks], page 3). Most often they are defined as macros like this:

```
#define obstack_chunk_alloc malloc
#define obstack_chunk_free free
```

Note that these are simple macros (no arguments). Macro definitions with arguments will not work! It is necessary that `obstack_chunk_alloc` or `obstack_chunk_free`, alone, expand into a function name if it is not itself a function name.

If you allocate chunks with `malloc`, the chunk size should be a power of 2. The default chunk size, 4096, was chosen because it is long enough to satisfy many typical requests on the obstack yet short enough not to waste too much memory in the portion of the last chunk not yet used.

`size_t obstack_chunk_size (struct obstack *obstack_ptr)` [Macro]

This returns the chunk size of the given obstack.

Since this macro expands to an lvalue, you can specify a new chunk size by assigning it a new value. Doing so does not affect the chunks already allocated, but will change the size of chunks allocated for that particular obstack in the future. It is unlikely to be useful to make the chunk size smaller, but making it larger might improve efficiency if you are allocating many objects whose size is comparable to the chunk size. Here is how to do so cleanly:

```
if (obstack_chunk_size (obstack_ptr) < new-chunk-size)
    obstack_chunk_size (obstack_ptr) = new-chunk-size;
```

### 2.3.1.11 Summary of Obstack Macros

Here is a summary of all the macros associated with obstacks. Each takes the address of an obstack (`struct obstack *`) as its first argument.

`int obstack_init (struct obstack *obstack_ptr)`

Initialize use of an obstack. See Section 2.3.1.1 [Creating Obstacks], page 3.

`int obstack_begin (struct obstack *obstack_ptr, size_t chunk_size)`

Initialize use of an obstack, with an initial chunk of *chunk\_size* bytes.



- `void obstack_blank_fast (struct obstack *obstack-ptr, size_t size)`  
Add *size* uninitialized bytes to a growing object without checking that there is enough room. See Section 2.3.1.7 [Extra Fast Growing], page 8.
- `void obstack_1grow_fast (struct obstack *obstack-ptr, char data-char)`  
Add one byte containing *data-char* to a growing object without checking that there is enough room. See Section 2.3.1.7 [Extra Fast Growing], page 8.
- `size_t obstack_room (struct obstack *obstack-ptr)`  
Get the amount of room now available for growing the current object. See Section 2.3.1.7 [Extra Fast Growing], page 8.
- `size_t obstack_alignment_mask (struct obstack *obstack-ptr)`  
The mask used for aligning the beginning of an object. This is an lvalue. See Section 2.3.1.9 [Obstacks Data Alignment], page 10.
- `size_t obstack_chunk_size (struct obstack *obstack-ptr)`  
The size for allocating chunks. This is an lvalue. See Section 2.3.1.10 [Obstack Chunks], page 11.
- `void *obstack_base (struct obstack *obstack-ptr)`  
Tentative starting address of the currently growing object. See Section 2.3.1.8 [Status of an Obstack], page 10.
- `void *obstack_next_free (struct obstack *obstack-ptr)`  
Address just after the end of the currently growing object. See Section 2.3.1.8 [Status of an Obstack], page 10.















ISPUNCT (c)	[Extension]
ISSPACE (c)	[Extension]
ISUPPER (c)	[Extension]
ISXDIGIT (c)	[Extension]

These twelve macros are defined by `safe-ctype.h`. Each has the same meaning as the corresponding macro (with name in lowercase) defined by the standard header `ctype.h`. For example, `ISALPHA` returns true for alphabetic characters and false for others. However, there are two differences between these macros and those provided by `ctype.h`:

- These macros are guaranteed to have well-defined behavior for all values representable by `signed char` and `unsigned char`, and for `EOF`.
- These macros ignore the current locale; they are true for these fixed sets of characters:

```
ALPHA      A-Za-z
ALNUM      A-Za-z0-9
BLANK      space tab
CNTRL      !PRINT
DIGIT      0-9
GRAPH      ALNUM || PUNCT
LOWER      a-z
PRINT      GRAPH || space
PUNCT      ~!@#$%^&*()_-=+[{]}\|;:','<.>/?
SPACE      space tab \n \r \f \v
UPPER      A-Z
XDIGIT     0-9A-Fa-f
```

Note that, if the host character set is ASCII or a superset thereof, all these macros will return false for all values of `char` outside the range of 7-bit ASCII. In particular, both `ISPRINT` and `ISCNTRL` return false for characters with numeric values from 128 to 255.

ISIDNUM (c)	[Extension]
ISIDST (c)	[Extension]
IS_VSPACE (c)	[Extension]
IS_NVSPACE (c)	[Extension]
IS_SPACE_OR_NUL (c)	[Extension]
IS_ISOBASIC (c)	[Extension]

These six macros are defined by `safe-ctype.h` and provide additional character classes which are useful when doing lexical analysis of C or similar languages. They are true for the following sets of characters:

```
IDNUM      A-Za-z0-9_
IDST       A-Za-z_
VSPACE     \r \n
NVSPACE    space tab \f \v \0
SPACE_OR_NUL  VSPACE || NVSPACE
ISOBASIC   VSPACE || NVSPACE || PRINT
```



**int memcmp** (const void \**x*, const void \**y*, size\_t *count*) [Supplemental]  
 Compares the first *count* bytes of two areas of memory. Returns zero if they are the same, a value less than zero if *x* is lexically less than *y*, or a value greater than zero if *x* is lexically greater than *y*. Note that lexical order is determined as if comparing unsigned char arrays.

**void\* memcpy** (void \**out*, const void \**in*, size\_t *length*) [Supplemental]  
 Copies *length* bytes from memory region *in* to region *out*. Returns a pointer to *out*.

**void\* memmem** (const void \**haystack*, size\_t *haystack\_len*, const void \**needle*, size\_t *needle\_len*) [Supplemental]  
 Returns a pointer to the first occurrence of *needle* (length *needle\_len*) in *haystack* (length *haystack\_len*). Returns NULL if not found.

**void\* memmove** (void \**from*, const void \**to*, size\_t *count*) [Supplemental]  
 Copies *count* bytes from memory area *from* to memory area *to*, returning a pointer to *to*.

**void\* mempcpy** (void \**out*, const void \**in*, size\_t *length*) [Supplemental]  
 Copies *length* bytes from memory region *in* to region *out*. Returns a pointer to *out* + *length*.

**void\* memset** (void \**s*, int *c*, size\_t *count*) [Supplemental]  
 Sets the first *count* bytes of *s* to the constant byte *c*, returning a pointer to *s*.

**int mkstemps** (char \**pattern*, int *suffix\_len*) [Replacement]  
 Generate a unique temporary file name from *pattern*. *pattern* has the form:  
     *path/ccXXXXXXsuffix*  
*suffix\_len* tells us how long *suffix* is (it can be zero length). The last six characters of *pattern* before *suffix* must be 'XXXXXX'; they are replaced with a string that makes the filename unique. Returns a file descriptor open on the file for reading and writing.

**void pex\_free** (struct pex\_obj \**obj*) [Extension]  
 Clean up and free all data associated with *obj*. If you have not yet called **pex\_get\_times** or **pex\_get\_status**, this will try to kill the subprocesses.

**int pex\_get\_status** (struct pex\_obj \**obj*, int *count*, int \**vector*) [Extension]  
 Returns the exit status of all programs run using *obj*. *count* is the number of results expected. The results will be placed into *vector*. The results are in the order of the calls to **pex\_run**. Returns 0 on error, 1 on success.

**int pex\_get\_times** (struct pex\_obj \**obj*, int *count*, struct pex\_time \**vector*) [Extension]  
 Returns the process execution times of all programs run using *obj*. *count* is the number of results expected. The results will be placed into *vector*. The results are in the order of the calls to **pex\_run**. Returns 0 on error, 1 on success.









`char* rindex (const char *s, int c)` [Supplemental]  
 Returns a pointer to the last occurrence of the character *c* in the string *s*, or NULL if not found. The use of `rindex` is deprecated in new programs in favor of `strrchr`.

`int setenv (const char *name, const char *value, int overwrite)` [Supplemental]

`void unsetenv (const char *name)` [Supplemental]  
`setenv` adds *name* to the environment with value *value*. If the name was already present in the environment, the new value will be stored only if *overwrite* is nonzero. The companion `unsetenv` function removes *name* from the environment. This implementation is not safe for multithreaded code.

`void setproctitle (const char *fmt, ...)` [Supplemental]  
 Set the title of a process to *fmt*. *va* args not supported for now, but defined for compatibility with BSD.

`int signo_max (void)` [Extension]  
 Returns the maximum signal value for which a corresponding symbolic name or message is available. Note that in the case where we use the `sys_siglist` supplied by the system, it is possible for there to be more symbolic names than messages, or vice versa. In fact, the manual page for `psignal(3b)` explicitly warns that one should check the size of the table (`NSIG`) before indexing it, since new signal codes may be added to the system before they are added to the table. Thus `NSIG` might be smaller than value implied by the largest `signo` value defined in `<signal.h>`.  
 We return the maximum value that can be used to obtain a meaningful symbolic name or message.

`int sigsetmask (int set)` [Supplemental]  
 Sets the signal mask to the one provided in *set* and returns the old mask (which, for `liberty`'s implementation, will always be the value 1).

`const char * simple_object_attributes_compare` [Extension]  
`(simple_object_attributes *attrs1, simple_object_attributes *attrs2, int *err)`  
 Compare *attrs1* and *attrs2*. If they could be linked together without error, return NULL. Otherwise, return an error message and set *\*err* to an `errno` value or 0 if there is no relevant `errno`.

`simple_object_attributes * simple_object_fetch_attributes` [Extension]  
`(simple_object_read *simple_object, const char **errmsg, int *err)`  
 Fetch the attributes of *simple\_object*. The attributes are internal information such as the format of the object file, or the architecture it was compiled for. This information will persist until `simple_object_attributes_release` is called, even if *simple\_object* itself is released.

On error this returns NULL, sets *\*errmsg* to an error message, and sets *\*err* to an `errno` value or 0 if there is no relevant `errno`.



```
void simple_object_release_read (simple_object_read      [Extension]
                               *simple_object)
```

Release all resources associated with *simple\_object*. This does not close the file descriptor.

```
void simple_object_release_write (simple_object_write    [Extension]
                                 *simple_object)
```

Release all resources associated with *simple\_object*.

```
simple_object_write * simple_object_start_write          [Extension]
    (simple_object_attributes attrs, const char *segment_name,
     const char **errmsg, int *err)
```

Start creating a new object file using the object file format described in *attrs*. You must fetch attribute information from an existing object file before you can create a new one. There is currently no support for creating an object file de novo.

*segment\_name* is only used with Mach-O as found on Darwin aka Mac OS X. The parameter is required on that target. It means that all sections are created within the named segment. It is ignored for other object file formats.

On error *simple\_object\_start\_write* returns NULL, sets *\*ERRMSG* to an error message, and sets *\*err* to an errno value or 0 if there is no relevant errno.

```
const char * simple_object_write_add_data              [Extension]
    (simple_object_write *simple_object,
     simple_object_write_section *section, const void *buffer,
     size_t size, int copy, int *err)
```

Add data *buffer/size* to *section* in *simple\_object*. If *copy* is non-zero, the data will be copied into memory if necessary. If *copy* is zero, *buffer* must persist until *simple\_object\_write\_to\_file* is called. is released.

On success this returns NULL. On error this returns an error message, and sets *\*err* to an errno value or 0 if there is no relevant error.

```
simple_object_write_section *                           [Extension]
    simple_object_write_create_section (simple_object_write
    *simple_object, const char *name, unsigned int align, const
    char **errmsg, int *err)
```

Add a section to *simple\_object*. *name* is the name of the new section. *align* is the required alignment expressed as the number of required low-order 0 bits (e.g., 2 for alignment to a 32-bit boundary).

The section is created as containing data, readable, not writable, not executable, not loaded at runtime. The section is not written to the file until *simple\_object\_write\_to\_file* is called.

On error this returns NULL, sets *\*errmsg* to an error message, and sets *\*err* to an errno value or 0 if there is no relevant errno.





**char\* strrchr (const char \*s, int c)** [Supplemental]  
 Returns a pointer to the last occurrence of the character *c* in the string *s*, or NULL if not found. If *c* is itself the null character, the results are undefined.

**const char \* strsignal (int signo)** [Supplemental]  
 Maps an signal number to an signal message string, the contents of which are implementation defined. On systems which have the external variable **sys\_siglist**, these strings will be the same as the ones used by **psignal()**.  
 If the supplied signal number is within the valid range of indices for the **sys\_siglist**, but no message is available for the particular signal number, then returns the string 'Signal *num*', where *num* is the signal number.  
 If the supplied signal number is not a valid index into **sys\_siglist**, returns NULL.  
 The returned string is only guaranteed to be valid only until the next call to **strsignal**.

**const char\* strsigno (int signo)** [Extension]  
 Given an signal number, returns a pointer to a string containing the symbolic name of that signal number, as found in **<signal.h>**.  
 If the supplied signal number is within the valid range of indices for symbolic names, but no name is available for the particular signal number, then returns the string 'Signal *num*', where *num* is the signal number.  
 If the supplied signal number is not within the range of valid indices, then returns NULL.  
 The contents of the location pointed to are only guaranteed to be valid until the next call to **strsigno**.

**char\* strstr (const char \*string, const char \*sub)** [Supplemental]  
 This function searches for the substring *sub* in the string *string*, not including the terminating null characters. A pointer to the first occurrence of *sub* is returned, or NULL if the substring is absent. If *sub* points to a string with zero length, the function returns *string*.

**double strtod (const char \*string, char \*\*endptr)** [Supplemental]  
 This ISO C function converts the initial portion of *string* to a double. If *endptr* is not NULL, a pointer to the character after the last character used in the conversion is stored in the location referenced by *endptr*. If no conversion is performed, zero is returned and the value of *string* is stored in the location referenced by *endptr*.

**int strtouerrno (const char \*name)** [Extension]  
 Given the symbolic name of a error number (e.g., EACCES), map it to an errno value. If no translation is found, returns 0.

**long int strtol (const char \*string, char \*\*endptr, int base)** [Supplemental]

**unsigned long int strtoul (const char \*string, char \*\*endptr, int base)** [Supplemental]

The **strtoul** function converts the string in *string* to a long integer value according to the given *base*, which must be between 2 and 36 inclusive, or be the special value







`void xmalloc_set_program_name (const char *name)` [Replacement]  
You can use this to set the name of the program used by `xmalloc_failed` when printing a failure message.

`void* xmemdup (void *input, size_t copy_size, size_t alloc_size)` [Replacement]  
Duplicates a region of memory without fail. First, *alloc\_size* bytes are allocated, then *copy\_size* bytes from *input* are copied into it, and the new memory is returned. If fewer bytes are copied than were allocated, the remaining memory is zeroed.

`void* xrealloc (void *ptr, size_t size)` [Replacement]  
Reallocate memory without fail. This routine functions like `realloc`, but will behave the same as `xmalloc` if memory cannot be found.

`char* xstrdup (const char *s)` [Replacement]  
Duplicates a character string without fail, using `xmalloc` to obtain memory.

`char* xstrerror (int errnum)` [Replacement]  
Behaves exactly like the standard `strerror` function, but will never return a NULL pointer.

`char* xstrndup (const char *s, size_t n)` [Replacement]  
Returns a pointer to a copy of *s* with at most *n* characters without fail, using `xmalloc` to obtain memory. The result is always NUL terminated.

`char* xvasprintf (const char *format, va_list args)` [Replacement]  
Print to allocated string without fail. If `xvasprintf` fails, this will print a message to `stderr` (using the name set by `xmalloc_set_program_name`, if any) and then call `xexit`.

## Appendix A Licenses

### A.1 GNU LESSER GENERAL PUBLIC LICENSE

Version 2.1, February 1999

Copyright © 1991-2025 Free Software Foundation, Inc.  
51 Franklin Street - Fifth Floor, Boston, MA 02110-1301, USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

[This is the first released version of the Lesser GPL. It also counts as the successor of the GNU Library Public License, version 2, hence the version number 2.1.]

#### A.1.1 Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public Licenses are intended to guarantee your freedom to share and change free software—to make sure the software is free for all its users.

This license, the Lesser General Public License, applies to some specially designated software—typically libraries—of the Free Software Foundation and other authors who decide to use it. You can use it too, but we suggest you first think carefully about whether this license or the ordinary General Public License is the better strategy to use in any particular case, based on the explanations below.

When we speak of free software, we are referring to freedom of use, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish); that you receive source code or can get it if you want it; that you can change the software and use pieces of it in new free programs; and that you are informed that you can do these things.

To protect your rights, we need to make restrictions that forbid distributors to deny you these rights or to ask you to surrender these rights. These restrictions translate to certain responsibilities for you if you distribute copies of the library or if you modify it.

For example, if you distribute copies of the library, whether gratis or for a fee, you must give the recipients all the rights that we gave you. You must make sure that they, too, receive or can get the source code. If you link other code with the library, you must provide complete object files to the recipients, so that they can relink them with the library after making changes to the library and recompiling it. And you must show them these terms so they know their rights.

We protect your rights with a two-step method: (1) we copyright the library, and (2) we offer you this license, which gives you legal permission to copy, distribute and/or modify the library.

To protect each distributor, we want to make it very clear that there is no warranty for the free library. Also, if the library is modified by someone else and passed on, the recipients should know that what they have is not the original version, so that the original author's reputation will not be affected by problems that might be introduced by others.

Finally, software patents pose a constant threat to the existence of any free program. We wish to make sure that a company cannot effectively restrict the users of a free program by obtaining a restrictive license from a patent holder. Therefore, we insist that any patent license obtained for a version of the library must be consistent with the full freedom of use specified in this license.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License. This license, the GNU Lesser General Public License, applies to certain designated libraries, and is quite different from the ordinary General Public License. We use this license for certain libraries in order to permit linking those libraries into non-free programs.

When a program is linked with a library, whether statically or using a shared library, the combination of the two is legally speaking a combined work, a derivative of the original library. The ordinary General Public License therefore permits such linking only if the entire combination fits its criteria of freedom. The Lesser General Public License permits more lax criteria for linking other code with the library.

We call this license the *Lesser* General Public License because it does *Less* to protect the user's freedom than the ordinary General Public License. It also provides other free software developers *Less* of an advantage over competing non-free programs. These disadvantages are the reason we use the ordinary General Public License for many libraries. However, the Lesser license provides advantages in certain special circumstances.

For example, on rare occasions, there may be a special need to encourage the widest possible use of a certain library, so that it becomes a de-facto standard. To achieve this, non-free programs must be allowed to use the library. A more frequent case is that a free library does the same job as widely used non-free libraries. In this case, there is little to gain by limiting the free library to free software only, so we use the Lesser General Public License.

In other cases, permission to use a particular library in non-free programs enables a greater number of people to use a large body of free software. For example, permission to use the GNU C Library in non-free programs enables many more people to use the whole GNU operating system, as well as its variant, the GNU/Linux operating system.

Although the Lesser General Public License is *Less* protective of the users' freedom, it does ensure that the user of a program that is linked with the Library has the freedom and the wherewithal to run that program using a modified version of the Library.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a "work based on the library" and a "work that uses the library". The former contains code derived from the library, whereas the latter must be combined with the library in order to run.

### **A.1.2 TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION**

0. This License Agreement applies to any software library or other program which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Lesser General Public License (also called "this License"). Each licensee is addressed as "you".

A “library” means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

The “Library”, below, refers to any such software library or work which has been distributed under these terms. A “work based on the Library” means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term “modification”.)

“Source code” for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

1. You may copy and distribute verbatim copies of the Library’s complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:
  - a. The modified work must itself be a software library.
  - b. You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.
  - c. You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.
  - d. If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful.

(For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Library.

In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy.

This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

4. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.

If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.

5. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a “work that uses the Library”. Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

However, linking a “work that uses the Library” with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a “work that uses the library”. The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.

When a “work that uses the Library” uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work

can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)

Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

6. As an exception to the Sections above, you may also combine or link a “work that uses the Library” with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer’s own use and reverse engineering for debugging such modifications.

You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

- a. Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable “work that uses the Library”, as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)
- b. Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (1) uses at run time a copy of the library already present on the user’s computer system, rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if the user installs one, as long as the modified version is interface-compatible with the version that the work was made with.
- c. Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.
- d. If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.
- e. Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

For an executable, the required form of the “work that uses the Library” must include any data and utility programs needed for reproducing the executable from it. However,

as a special exception, the materials to be distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

7. You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:
  - a. Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.
  - b. Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.
8. You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.
9. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.
10. Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties with this License.
11. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way

you could satisfy both it and this License would be to refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

12. If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Library under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

13. The Free Software Foundation may publish revised and/or new versions of the Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library specifies a version number of this License which applies to it and “any later version”, you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.

14. If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

#### NO WARRANTY

15. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE LIBRARY “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE LIBRARY IS WITH

YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

**END OF TERMS AND CONDITIONS**

### A.1.3 How to Apply These Terms to Your New Libraries

If you develop a new library, and you want it to be of the greatest possible use to the public, we recommend making it free software that everyone can redistribute and change. You can do so by permitting redistribution under these terms (or, alternatively, under the terms of the ordinary General Public License).

To apply these terms, attach the following notices to the library. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the “copyright” line and a pointer to where the full notice is found.

```
one line to give the library's name and an idea of what it does.
Copyright (C) year  name of author
```

```
This library is free software; you can redistribute it and/or modify it
under the terms of the GNU Lesser General Public License as published by
the Free Software Foundation; either version 2.1 of the License, or (at
your option) any later version.
```

```
This library is distributed in the hope that it will be useful, but
WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
Lesser General Public License for more details.
```

```
You should have received a copy of the GNU Lesser General Public
License along with this library; if not, write to the Free Software
Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301,
USA.
```

Also add information on how to contact you by electronic and paper mail.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a “copyright disclaimer” for the library, if necessary. Here is a sample; alter the names:

```
Yoyodyne, Inc., hereby disclaims all copyright interest in the library
`Frob' (a library for tweaking knobs) written by James Random Hacker.
```

```
signature of Ty Coon, 1 April 1990
Ty Coon, President of Vice
```

That’s all there is to it!

## A.2 BSD

Copyright © 1990 Regents of the University of California. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. [rescinded 22 July 1999]
4. Neither the name of the University nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.





