NAME

alloca − allocate memory that is automatically freed

SYNOPSIS

#include <alloca.h>

void *alloca(size_t *size***);**

DESCRIPTION

The **alloca**() function allocates *size* bytes of space in the stack frame of the caller. This temporary space is automatically freed when the function that called **alloca**() returns to its caller.

RETURN VALUE

The **alloca**() function returns a pointer to the beginning of the allocated space. If the allocation causes stack overflow, program behavior is undefined.

ATTRIBUTES

For an explanation of the terms used in this section, see **attributes**(7).

Interface	Attribute	Value
alloca()	Thread safety	MT-Safe

CONFORMING TO

This function is not in POSIX.1.

There is evidence that the **alloca**() function appeared in 32V, PWB, PWB.2, 3BSD, and 4BSD. There is a man page for it in 4.3BSD. Linux uses the GNU version.

NOTES

The **alloca**() function is machine- and compiler-dependent. For certain applications, its use can improve efficiency compared to the use of **malloc**(3) plus **free**(3). In certain cases, it can also simplify memory deallocation in applications that use **longjmp**(3) or **siglongjmp**(3). Otherwise, its use is discouraged.

Because the space allocated by **alloca**() is allocated within the stack frame, that space is automatically freed if the function return is jumped over by a call to **longjmp**(3) or **siglongjmp**(3).

The space allocated by **alloca**() is *not* automatically deallocated if the pointer that refers to it simply goes out of scope.

Do not attempt to **free**(3) space allocated by **alloca**()!

Notes on the GNU version

Normally, **gcc**(1) translates calls to **alloca**() with inlined code. This is not done when either the *−ansi*, *−std=c89*, *−std=c99*, or the *−std=c11* option is given **and** the header *<alloca.h>* is not included. Otherwise, (without an −ansi or −std=c* option) the glibc version of *<stdlib.h>* includes *<alloca.h>* and that contains the lines:

```
#ifdef  __GNUC__
#define alloca(size)   __builtin_alloca (size)
#endif
```

with messy consequences if one has a private version of this function.

The fact that the code is inlined means that it is impossible to take the address of this function, or to change its behavior by linking with a different library.

The inlined code often consists of a single instruction adjusting the stack pointer, and does not check for stack overflow. Thus, there is no NULL error return.

BUGS

There is no error indication if the stack frame cannot be extended. (However, after a failed allocation, the program is likely to receive a **SIGSEGV** signal if it attempts to access the unallocated space.)

On many systems **alloca**() cannot be used inside the list of arguments of a function call, because the stack space reserved by **alloca**() would appear on the stack in the middle of the space for the function arguments.

SEE ALSO

brk(2), **longjmp**(3), **malloc**(3)

COLOPHON

This page is part of release 5.02 of the Linux *man-pages* project. A description of the project, information about reporting bugs, and the latest version of this page, can be found at https://www.kernel.org/doc/man−pages/.