NAME

aligned_alloc − allocate aligned memory

LIBRARY

Standard C library (*libc*, *−lc*)

SYNOPSIS

```
#include <stdlib.h>

void *aligned_alloc(size_t alignment, size_t size);
```

Feature Test Macro Requirements for glibc (see **feature_test_macros**(7)):

aligned_alloc():
_ISOC11_SOURCE

DESCRIPTION

aligned_alloc() allocates *size* bytes and returns a pointer to the allocated memory. The memory address will be a multiple of *alignment*, which must be a power of two. This address can later be successfully passed to **free**(3).

The memory is not zeroed.

RETURN VALUE

aligned_alloc() returns a pointer to the allocated memory on success. On error, NULL is returned, and *errno* is set to indicate the error.

ERRORS

EINVAL
The *alignment* argument was not a power of two.

ENOMEM
Out of memory.

ATTRIBUTES

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

Interface	Attribute	Value
aligned_alloc()	Thread safety	MT-Safe

STANDARDS

C23, POSIX.1-2024.

HISTORY

glibc 2.16. C11, POSIX.1-2024.

C11

In C11, the specification of this function had several issues ⟨https://port70.net/~nsz/c/c11/n1570.html#7.22.3.1p2⟩.

- *size* had to be a multiple of *alignment*. Otherwise, the behavior was undefined.
- If *alignment* was not a power of two, the behavior was undefined.

DR460 ⟨https://www.open−std.org/jtc1/sc22/wg14/www/docs/summary.htm#dr_460⟩ reported both cases of UB as unnecessarily dangerous, and fixed them with a Technical Corrigendum that transformed them into errors.

N2072 ⟨https://www.open−std.org/jtc1/sc22/wg14/www/docs/n2072.htm⟩ reported that the requirement that *size* is a multiple of *alignment* is superfluous, and removed it with a Technical Corrigendum.

C17 incorporates both technical corrigenda. The API has been stable since C17.

glibc initially implemented it as silently aligning as *stdc_bit_ceil(alignment)* instead of *alignment*. Since glibc 2.38, it implements the C17 specification.

Some implementations, such as FreeBSD/jemalloc, implement the C17 specification, even though their documentation claims having undefined behavior.

Some implementations, such as OpenBSD, implement C11 amended with DR460, even though their documentation claims having undefined behavior.

No known implementations have exploited the undefined behavior in a more dangerous way. This function should be safe to use.

NOTES

On many systems there are alignment restrictions, for example, on buffers used for direct block device I/O. POSIX specifies the *pathconf(path,_PC_REC_XFER_ALIGN)* call that tells what alignment is needed. Now one can use **aligned_alloc**() to satisfy this requirement.

The glibc **malloc**(3) always returns 8-byte aligned memory addresses, so this function is needed only if you require larger alignment values.

SEE ALSO

brk(2), **getpagesize**(2), **free**(3), **malloc**(3)