## NAME

bsdmalloc, malloc, free, realloc – memory allocator

## SYNOPSIS

**cc** [ *flag* ... ] *file* ... **–lbsdmalloc** [ *library* ... ]

**char** *****malloc(** *size***)**
**unsigned** *size***;**

**int free(** *ptr***)**
**char** * *ptr***;**

**char** *****realloc(** *ptr*, *size***)**
**char** **ptr***;**
**unsigned** size**;**

## DESCRIPTION

These routines provide a general-purpose memory allocation package. They maintain a table of free blocks for efficient allocation and coalescing of free storage. When there is no suitable space already free, the allocation routines call **sbrk**(2) to get more memory from the system. Each of the allocation routines returns a pointer to space suitably aligned for storage of any type of object. Each returns a NULL pointer if the request cannot be completed (see **DIAGNOSTICS**).

**malloc( )** returns a pointer to a block of at least *size* bytes, which is appropriately aligned.

**free( )** releases a previously allocated block. Its argument is a pointer to a block previously allocated by **malloc( )** or **realloc( )**.

**realloc( )** changes the size of the block referenced by *ptr* to *size* bytes and returns a pointer to the (possibly moved) block. The contents will be unchanged up to the lesser of the new and old sizes. If unable to honor a reallocation request, **realloc( )** leaves its first argument unaltered. For backwards compatibility, **realloc( )** accepts a pointer to a block freed since the most recent call to **malloc( )** or **realloc( )**.

## RETURN VALUES

**malloc( )** and **realloc( )** return a **NULL** pointer if there is not enough available memory. When **realloc( )** returns **NULL**, the block pointed to by *ptr* is left intact.

## ERRORS

If **malloc( )** or **realloc( )** returns unsuccessfully, **errno** will be set to indicate the following:

**ENOMEM** *size* bytes of memory exceeds the physical limits of your system, and cannot be allocated.

**EAGAIN** There is not enough memory available AT THIS POINT IN TIME to allocate *size* bytes of memory; but the application could try again later.

## SEE ALSO

**brk**(2), **malloc**(3C), **malloc**(3X), **mapmalloc**(3X)

## WARNINGS

Use of libbsdmalloc renders an application non-SCD compliant.

libbsdmalloc routines are incompatible with the memory allocation routines in the standard C-library (libc): **malloc**(3C), **alloca**(3C), **calloc**(3C), **free**(3C), **memalign**(3C), **realloc**(3C), and **valloc**(3C).

**NOTES**

Using **realloc( )** with a block freed before the most recent call to **malloc( )** or **realloc( )** will result in an error.

**malloc( )** and **realloc( )** return a non-NULL pointer if *size* is 0. These pointers should not be dereferenced.

Always cast the value returned by **malloc( )** and **realloc( )**.

Comparative Features of **bsdmalloc( ), malloc**(3X), and **malloc**(3C)**:**

- The **bsdmalloc( )** routines afford better performance, but are space-inefficient.
- The **malloc**(3X) routines are space-efficient, but have slower performance.
- The standard, fully SCD-compliant **malloc**(3C) routines are a trade-off between performance and space-efficiency.

**free( )** does not set **errno.**