NAME

mapmalloc, calloc, cfree, free, realloc, – memory allocator

SYNOPSIS

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

#include <stdlib.h>

void *****malloc(size_t** *size***);**

void *****calloc(size_t** *nelem***, size_t** *elsize***);**

void cfree(void *****ptr***, unsigned** *num***, unsigned** *size***);**

void free(void * *ptr***);**

void *****realloc(void** *****ptr***, size_t** *size***);**

MT-LEVEL

Safe

DESCRIPTION

The collection of **malloc** routines in this library use **mmap**(2) instead of **sbrk**(2) for acquiring new heap space. The routines in this library are intended to be used only if necessary, when applications must call **sbrk( ),** but need to call other library routines that might call **malloc.** The algorithms used by these routines are not sophisticated. There is no reclaiming of memory.

malloc( ) and **free( )** provide a simple general-purpose memory allocation package. **malloc( )** returns a pointer to a block of at least *size* bytes suitably aligned for any use.

The argument to **free( )** is a pointer to a block previously allocated by **malloc( )**, **calloc( )** or **realloc( )**. If *ptr* is a **NULL** pointer, no action occurs.

Undefined results will occur if the space assigned by **malloc( )** is overrun or if some random number is handed to **free( )**.

calloc( ) allocates space for an array of *nelem* elements of size *elsize*. The space is initialized to zeros.

realloc( ) changes the size of the block pointed to 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 *ptr* is **NULL**, **realloc( )** behaves like **malloc( )** for the specified size. If *size* is zero and *ptr* is not a null pointer, the object pointed to is freed.

Each of the allocation routines returns a pointer to space suitably aligned (after possible pointer coercion) for storage of any type of object.

malloc( ) and **realloc**( ) will fail if there is not enough available memory.

Entry points for **malloc_debug**( )**,** **mallocmap**( )**,** **mallopt**( )**,** **mallinfo**( )**,** **memalign**( )**,** and **valloc**( )**,** are empty routines, and are provided only to protect the user from mixing **malloc**( ) functions from different implementations.

RETURN VALUES

If there is no available memory, **malloc( )**, **realloc( )**, and **calloc( )** return a null pointer. When **realloc( )** returns **NULL**, the block pointed to by *ptr* is left intact. If *size*, *nelem*, or *elsize* is 0, a unique pointer to the arena is returned.

FILES **/usr/lib/libmapmalloc**

SEE ALSO **brk**(2), **getrlimit**(2), **mmap**(2), **realloc**(3C)