NAME

malloc, **mallocarray**, **free**, **zfree**, **realloc**, **reallocf**, **malloc_usable_size**, **malloc_aligned**, **malloc_exec**, **MALLOC_DECLARE**, **MALLOC_DEFINE**, **malloc_domainset**, **malloc_domainset_aligned**, **malloc_domainset_exec**, **mallocarray_domainset** - kernel memory management routines

SYNOPSIS

#include <sys/types.h>
#include <sys/malloc.h>

*void **
malloc(*size_t size, struct malloc_type *type, int flags*);

*void **
mallocarray(*size_t nmemb, size_t size, struct malloc_type *type, int flags*);

void
free(*void *addr, struct malloc_type *type*);

void
zfree(*void *addr, struct malloc_type *type*);

*void **
realloc(*void *addr, size_t size, struct malloc_type *type, int flags*);

*void **
reallocf(*void *addr, size_t size, struct malloc_type *type, int flags*);

size_t
malloc_usable_size(*const void *addr*);

*void **
malloc_aligned(*size_t size, size_t align, struct malloc_type *type, int flags*);

*void **
malloc_exec(*size_t size, struct malloc_type *type, int flags*);

MALLOC_DECLARE(*type*);

#include <sys/param.h>
#include <sys/malloc.h>

#include <sys/kernel.h>

MALLOC_DEFINE(*type*, *shortdesc*, *longdesc*);

#include <sys/param.h>
#include <sys/domainset.h>
#include <sys/malloc.h>

*void **
malloc_domainset(*size_t size*, *struct malloc_type *type*, *struct domainset *ds*, *int flags*);

*void **
malloc_domainset_aligned(*size_t size*, *size_t align*, *struct malloc_type *type*, *struct domainset *ds*, *int flags*);

*void **
malloc_domainset_exec(*size_t size*, *struct malloc_type *type*, *struct domainset *ds*, *int flags*);

*void **
mallocarray_domainset(*size_t nmemb*, *size_t size*, *struct malloc_type *type*, *struct domainset *ds*, *int flags*);

DESCRIPTION

The **malloc**() function allocates uninitialized memory in kernel address space for an object whose size is specified by *size*.

The **malloc_domainset**() variant allocates memory from a specific numa(4) domain using the specified domain selection policy. See domainset(9) for some example policies.

The **malloc_aligned**() and **malloc_domainset_aligned**() variants return allocations aligned as specified by *align*, which must be non-zero, a power of two, and less than or equal to the page size.

Both **malloc_exec**() and **malloc_domainset_exec**() can be used to return executable memory. Not all platforms enforce a distinction between executable and non-executable memory.

The **mallocarray**() function allocates uninitialized memory in kernel address space for an array of *nmemb* entries whose size is specified by *size*.

The **mallocarray_domainset**() variant allocates memory from a specific numa(4) domain using the specified domain selection policy. See domainset(9) for some example policies.

The **free**() function releases memory at address *addr* that was previously allocated by **malloc**() for re-use. The memory is not zeroed. If *addr* is NULL, then **free**() does nothing.

Like **free**(), the **zfree**() function releases memory at address *addr* that was previously allocated by **malloc**() for re-use. However, **zfree**() will zero the memory before it is released.

The **realloc**() function changes the size of the previously allocated memory referenced by *addr* to *size* bytes. The contents of the memory are unchanged up to the lesser of the new and old sizes. Note that the returned value may differ from *addr*. If the requested memory cannot be allocated, NULL is returned and the memory referenced by *addr* is valid and unchanged. If *addr* is NULL, the **realloc**() function behaves identically to **malloc**() for the specified size.

The **reallocf**() function is identical to **realloc**() except that it will free the passed pointer when the requested memory cannot be allocated.

The **malloc_usable_size**() function returns the usable size of the allocation pointed to by *addr*. The return value may be larger than the size that was requested during allocation.

Unlike its standard C library counterpart (malloc(3)), the kernel version takes two more arguments. The *flags* argument further qualifies **malloc**()'s operational characteristics as follows:

M_ZERO
: Causes the allocated memory to be set to all zeros.

M_NODUMP
: For allocations greater than page size, causes the allocated memory to be excluded from kernel core dumps.

M_NOWAIT
: Causes **malloc**(), **realloc**(), and **reallocf**() to return NULL if the request cannot be immediately fulfilled due to resource shortage. Note that M_NOWAIT is required when running in an interrupt context.

M_WAITOK
: Indicates that it is OK to wait for resources. If the request cannot be immediately fulfilled, the current process is put to sleep to wait for resources to be released by other processes. The **malloc**(), **mallocarray**(), **realloc**(), and **reallocf**() functions cannot return NULL if M_WAITOK is specified. If the multiplication of *nmemb* and *size* would cause an integer overflow, the **mallocarray**() function induces a panic.

M_USE_RESERVE

Indicates that the system can use its reserve of memory to satisfy the request. This option should only be used in combination with M_NOWAIT when an allocation failure cannot be tolerated by the caller without catastrophic effects on the system.

Exactly one of either M_WAITOK or M_NOWAIT must be specified.

The *type* argument is used to perform statistics on memory usage, and for basic sanity checks. It can be used to identify multiple allocations. The statistics can be examined by 'vmstat -m'.

A *type* is defined using *struct malloc_type* via the **MALLOC_DECLARE**() and **MALLOC_DEFINE**() macros.

```
/* sys/something/foo_extern.h */

MALLOC_DECLARE(M_FOOBUF);

/* sys/something/foo_main.c */

MALLOC_DEFINE(M_FOOBUF, "foobuffers", "Buffers to foo data into the ether");

/* sys/something/foo_subr.c */

...
buf = malloc(sizeof(*buf), M_FOOBUF, M_NOWAIT);
```

In order to use **MALLOC_DEFINE**(), one must include <*sys/param.h*> (instead of <*sys/types.h*>) and <*sys/kernel.h*>.

CONTEXT

malloc(), **realloc**() and **reallocf**() may not be called from fast interrupts handlers. When called from threaded interrupts, *flags* must contain M_NOWAIT.

malloc(), **realloc**() and **reallocf**() may sleep when called with M_WAITOK. **free**() never sleeps. However, **malloc**(), **realloc**(), **reallocf**() and **free**() may not be called in a critical section or while holding a spin lock.

Any calls to **malloc**() (even with M_NOWAIT) or **free**() when holding a vnode(9) interlock, will cause a LOR (Lock Order Reversal) due to the intertwining of VM Objects and Vnodes.

IMPLEMENTATION NOTES

The memory allocator allocates memory in chunks that have size a power of two for requests up to the size of a page of memory. For larger requests, one or more pages is allocated. While it should not be relied upon, this information may be useful for optimizing the efficiency of memory use.

RETURN VALUES

The **malloc**(), **realloc**(), and **reallocf**() functions return a kernel virtual address that is suitably aligned for storage of any type of object, or NULL if the request could not be satisfied (implying that M_NOWAIT was set).

DIAGNOSTICS

A kernel compiled with the INVARIANTS configuration option attempts to detect memory corruption caused by such things as writing outside the allocated area and imbalanced calls to the **malloc**() and **free**() functions. Failing consistency checks will cause a panic or a system console message.

SEE ALSO

numa(4), vmstat(8), contigmalloc(9), domainset(9), memguard(9), vnode(9)

HISTORY

zfree() first appeared in FreeBSD 13.0.