NAME

vmem **-** general purpose resource allocator

SYNOPSIS

#include <sys/vmem.h>

*vmem_t **
vmem_create(*const char *name*, *vmem_addr_t base*, *vmem_size_t size*, *vmem_size_t quantum*, *vmem_size_t qcache_max*, *int flags*);

int
vmem_add(*vmem_t *vm*, *vmem_addr_t addr*, *vmem_size_t size*, *int flags*);

int
vmem_xalloc(*vmem_t *vm*, *const vmem_size_t size*, *vmem_size_t align*, *const vmem_size_t phase*, *const vmem_size_t nocross*, *const vmem_addr_t minaddr*, *const vmem_addr_t maxaddr*, *int flags*, *vmem_addr_t *addrp*);

void
vmem_xfree(*vmem_t *vm*, *vmem_addr_t addr*, *vmem_size_t size*);

int
vmem_alloc(*vmem_t *vm*, *vmem_size_t size*, *int flags*, *vmem_addr_t *addrp*);

void
vmem_free(*vmem_t *vm*, *vmem_addr_t addr*, *vmem_size_t size*);

void
vmem_destroy(*vmem_t *vm*);

DESCRIPTION

The **vmem** is a general purpose resource allocator. Despite its name, it can be used for arbitrary resources other than virtual memory.

vmem_create() creates a new vmem arena.

name The string to describe the vmem.

base The start address of the initial span. Pass 0 if no initial span is required.

size	The size of the initial span. Pass 0 if no initial span is required.
quantum	The smallest unit of allocation.
qcache_max	The largest size of allocations which can be served by quantum cache. It is merely a hint and can be ignored.
flags	malloc(9) wait flag.

vmem_add() adds a span of size *size* starting at *addr* to the arena. Returns 0 on success, ENOMEM on failure. *flags* is malloc(9) wait flag.

vmem_xalloc() allocates a resource from the arena.

vm	The arena which we allocate from.
size	Specify the size of the allocation.
align	If zero, don’t care about the alignment of the allocation. Otherwise, request a resource segment starting at offset *phase* from an *align* aligned boundary.
phase	See the above description of *align*. If *align* is zero, *phase* should be zero. Otherwise, *phase* should be smaller than *align*.
nocross	Request a resource which doesn’t cross *nocross* aligned boundary.
minaddr	Specify the minimum address which can be allocated, or VMEM_ADDR_MIN if the caller does not care.
maxaddr	Specify the maximum address which can be allocated, or VMEM_ADDR_MAX if the caller does not care.
flags	A bitwise OR of an allocation strategy and a malloc(9) wait flag. The allocation strategy is one of:

M_FIRSTFIT
Prefer allocation performance.

M_BESTFIT
: Prefer space efficiency.

M_NEXTFIT
: Perform an address-ordered search for free addresses, beginning where the previous search ended.

addrp On success, if *addrp* is not NULL, **vmem_xalloc**() overwrites it with the start address of the allocated span.

vmem_xfree() frees resource allocated by **vmem_xalloc**() to the arena.

vm The arena which we free to.

addr The resource being freed. It must be the one returned by **vmem_xalloc**(). Notably, it must not be the one from **vmem_alloc**(). Otherwise, the behaviour is undefined.

size The size of the resource being freed. It must be the same as the *size* argument used for **vmem_xalloc**().

vmem_alloc() allocates a resource from the arena.

vm The arena which we allocate from.

size Specify the size of the allocation.

flags A bitwise OR of an **vmem** allocation strategy flag (see above) and a malloc(9) sleep flag.

addrp
: On success, if *addrp* is not NULL, **vmem_alloc**() overwrites it with the start address of the allocated span.

vmem_free() frees resource allocated by **vmem_alloc**() to the arena.

vm The arena which we free to.

addr The resource being freed. It must be the one returned by **vmem_alloc**(). Notably, it must not be the one from **vmem_xalloc**(). Otherwise, the behaviour is undefined.

size The size of the resource being freed. It must be the same as the *size* argument used for

vmem_alloc().

vmem_destroy() destroys a vmem arena.

vm The vmem arena being destroyed. The caller should ensure that no one will use it anymore.

RETURN VALUES

vmem_create() returns a pointer to the newly allocated vmem_t. Otherwise, it returns NULL.

On success, **vmem_xalloc**() and **vmem_alloc**() return 0. Otherwise, ENOMEM is returned.

CODE REFERENCES

The **vmem** subsystem is implemented within the file *sys/kern/subr_vmem.c*.

SEE ALSO

malloc(9)

Jeff Bonwick and Jonathan Adams, "Magazines and Vmem: Extending the Slab Allocator to Many CPUs and Arbitrary Resources", *2001 USENIX Annual Technical Conference*, 2001.

HISTORY

The **vmem** allocator was originally implemented in NetBSD. It was introduced in FreeBSD 10.0.

AUTHORS

Original implementation of **vmem** was written by YAMAMOTO Takashi. The FreeBSD port was made by Jeff Roberson.

BUGS

vmem relies on malloc(9), so it cannot be used as early during system bootstrap.