NAME

new_unrhdr, **clean_unrhdr**, **clear_unrhdr**, **delete_unrhdr**, **alloc_unr**, **alloc_unr_specific**, **free_unr**, **create_iter_unr**, **next_iter_unr**, **free_iter_unr** - kernel unit number allocator

SYNOPSIS

#include <sys/systm.h>

*struct unrhdr **
new_unrhdr(*int low, int high, struct mtx *mutex*);

void
clean_unrhdr(*struct unrhdr *uh*);

void
clean_unrhdrl(*struct unrhdr *uh*);

void
clear_unrhdr(*struct unrhdr *uh*);

void
delete_unrhdr(*struct unrhdr *uh*);

int
alloc_unr(*struct unrhdr *uh*);

int
alloc_unrl(*struct unrhdr *uh*);

int
alloc_unr_specific(*struct unrhdr *uh, u_int item*);

void
free_unr(*struct unrhdr *uh, u_int item*);

*void **
create_iter_unr(*struct unrhdr *uh*);

int
next_iter_unr(*void *handle*);

void
free_iter_unr(*void *handle*);

DESCRIPTION

The kernel unit number allocator is a generic facility, which allows to allocate unit numbers within a specified range.

new_unrhdr(*low*, *high*, *mutex*)

Initialize a new unit number allocator entity. The *low* and *high* arguments specify minimum and maximum number of unit numbers. There is no cost associated with the range of unit numbers, so unless the resource really is finite, INT_MAX can be used. If *mutex* is not NULL, it is used for locking when allocating and freeing units. If the passed value is the token *UNR_NO_MTX*, then no locking is applied internally. Otherwise, internal mutex is used.

clear_unrhdr(*uh*)

Clear all units from the specified unit number allocator entity. This function resets the entity as if it were just initialized with **new_unrhdr**().

delete_unrhdr(*uh*)

Delete specified unit number allocator entity. This function frees the memory associated with the entity, it does not free any units. To free all units use **clear_unrhdr**().

clean_unrhdr(*uh*)

Freeing unit numbers might result in some internal memory becoming unused. There are **unit** allocator consumers that cannot tolerate taking malloc(9) locks to free the memory, while having their unit mutex locked. For this reason, free of the unused memory after delete is postponed until the consumer can afford calling into the malloc(9) subsystem. Call **clean_unrhdr**(*uh*) to do the cleanup. In particular, this needs to be done before freeing a unr, if a deletion of units could have been performed.

clean_unrhdrl()

Same as **clean_unrhdr**(), but assumes that the unr mutex is already owned, if any.

alloc_unr(*uh*)

Return a new unit number. The lowest free number is always allocated. This function does not allocate memory and never sleeps, however it may block on a mutex. If no free unit numbers are left, -1 is returned.

alloc_unrl(*uh*)

Same as **alloc_unr**() except that mutex is assumed to be already locked and thus is not used.

alloc_unr_specific(*uh*, *item*)

Allocate a specific unit number. This function allocates memory and thus may sleep. The allocated unit number is returned on success. If the specified number is already allocated or out of the range, -1 is returned.

free_unr(*uh*, *item*)

Free a previously allocated unit number. This function may require allocating memory, and thus it can sleep. There is no pre-locked variant.

ITERATOR INTERFACE

The **unr** facility provides an interface to iterate over all allocated units for the given unrhdr. Iterators are identified by an opaque handle. More than one iterators can operate simultaneously; the iterator position data is recorded only in the iterator handle.

Consumers must ensure that the unit allocator is not modified between calls to the iterator functions. In particular, the internal allocator mutex cannot provide consistency, because it is acquired and dropped inside the **next_iter_unr**() function. If the allocator was modified, it is safe to free the iterator with **free_iter_unr**() method nevertheless.

create_iter_unr(*uh*)

Create an iterator. Return the handle that should be passed to other iterator functions.

next_iter_unr(*handle*)

Return the value of the next unit. Units are returned in ascending order. A return value of -1 indicates the end of iteration, in which case -1 is returned for all future calls.

free_iter_unr(*handle*)

Free the iterator, handle is no longer valid.

CODE REFERENCES

The above functions are implemented in *sys/kern/subr_unit.c*.

HISTORY

Kernel unit number allocator first appeared in FreeBSD 6.0.

AUTHORS

Kernel unit number allocator was written by Poul-Henning Kamp. This manpage was written by Gleb Smirnoff.