NAME

refcount, **refcount_init**, **refcount_acquire**, **refcount_release** - manage a simple reference counter

SYNOPSIS

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

void
refcount_init(*volatile u_int *count, u_int value*);

u_int
refcount_load(*volatile u_int *count*);

u_int
refcount_acquire(*volatile u_int *count*);

bool
refcount_acquire_checked(*volatile u_int *count*);

bool
refcount_acquire_if_not_zero(*volatile u_int *count*);

bool
refcount_release(*volatile u_int *count*);

bool
refcount_release_if_last(*volatile u_int *count*);

bool
refcount_release_if_not_last(*volatile u_int *count*);

DESCRIPTION

The **refcount** functions provide an API to manage a simple reference counter. The caller provides the storage for the counter in an unsigned integer. A pointer to this integer is passed via *count*. Usually the counter is used to manage the lifetime of an object and is stored as a member of the object.

Currently all functions are implemented as static inline.

The **refcount_init**() function is used to set the initial value of the counter to *value*. It is normally used when creating a reference-counted object.

The **refcount_load**() function returns a snapshot of the counter value. This value may immediately become out-of-date in the absence of external synchronization. **refcount_load**() should be used instead of relying on the properties of the *volatile* qualifier.

The **refcount_acquire**() function is used to acquire a new reference. It returns the counter value before the new reference was acquired. The caller is responsible for ensuring that it holds a valid reference while obtaining a new reference. For example, if an object is stored on a list and the list holds a reference on the object, then holding a lock that protects the list provides sufficient protection for acquiring a new reference.

The **refcount_acquire_checked**() variant performs the same operation as **refcount_acquire**(), but additionally checks that the *count* value does not overflow as result of the operation. It returns true if the reference was sucessfully obtained, and false if it was not, due to the overflow.

The **refcount_acquire_if_not_zero**() function is yet another variant of **refcount_acquire**(), which only obtains the reference when some reference already exists. In other words, **count* must be already greater than zero for the function to succeed, in which case the return value is true, otherwise false is returned.

The **refcount_release**() function is used to release an existing reference. The function returns true if the reference being released was the last reference; otherwise, it returns false.

The **refcount_release_if_last**() and **refcount_release_if_not_last**() functions are variants of **refcount_release**() which only drop the reference when it is or is not the last reference, respectively. In other words, **refcount_release_if_last**() returns true when **count* is equal to one, in which case it is decremented to zero. Otherwise, **count* is not modified and the function returns false. Similarly, **refcount_release_if_not_last**() returns true when **count* is greater than one, in which case **count* is decremented. Otherwise, if **count* is equal to one, the reference is not released and the function returns false.

Note that these routines do not provide any inter-CPU synchronization or data protection for managing the counter. The caller is responsible for any additional synchronization needed by consumers of any containing objects. In addition, the caller is also responsible for managing the life cycle of any containing objects including explicitly releasing any resources when the last reference is released.

The **refcount_release**() unconditionally executes a release fence (see atomic(9)) before releasing the reference, which synchronizes with an acquire fence executed right before returning the true value. This ensures that the destructor, supposedly executed by the caller after the last reference was dropped, sees all updates done during the lifetime of the object.

RETURN VALUES

The **refcount_release** function returns true when releasing the last reference and false when releasing any other reference.

HISTORY

These functions were introduced in FreeBSD 6.0.