NAME

ucred, **crget**, **crhold**, **crfree**, **crcopy**, **crdup**, **cru2x** - functions related to user credentials

SYNOPSIS

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

*struct ucred **
crget(*void*);

*struct ucred **
crhold(*struct ucred *cr*);

void
crfree(*struct ucred *cr*);

void
crcopy(*struct ucred *dest*, *struct ucred *src*);

*struct ucred **
crcopysafe(*struct proc *p*, *struct ucred *cr*);

*struct ucred **
crdup(*struct ucred *cr*);

void
crsetgroups(*struct ucred *cr*, *int ngrp*, *gid_t *groups*);

void
cru2x(*struct ucred *cr*, *struct xucred *xcr*);

DESCRIPTION

The **ucred** family of functions is used to manage user credential structures (*struct ucred*) within the kernel.

The **crget**() function allocates memory for a new structure, sets its reference count to 1, and initializes its lock.

The **crhold**() function increases the reference count on the credential.

The **crfree**() function decreases the reference count on the credential. If the count drops to 0, the storage for the structure is freed.

The **crcopy**() function copies the contents of the source (template) credential into the destination template. The *uidinfo* structure within the destination is referenced by calling uihold(9).

The **crcopysafe**() function copies the current credential associated with the process *p* into the newly allocated credential *cr*. The process lock on *p* must be held and will be dropped and reacquired as needed to allocate group storage space in *cr*.

The **crdup**() function allocates memory for a new structure and copies the contents of *cr* into it. The actual copying is performed by **crcopy**().

The **crsetgroups**() function sets the *cr_groups* and *cr_ngroups* variables and allocates space as needed. It also truncates the group list to the current maximum number of groups. No other mechanism should be used to modify the *cr_groups* array except for updating the primary group via assignment to *cr_groups[0]*.

The **cru2x**() function converts a *ucred* structure to an *xucred* structure. That is, it copies data from *cr* to *xcr*; it ignores fields in the former that are not present in the latter (e.g., *cr_uidinfo*), and appropriately sets fields in the latter that are not present in the former (e.g., *cr_version*).

RETURN VALUES

crget(), **crhold**(), **crdup**(), and **crcopysafe**() all return a pointer to a *ucred* structure.

USAGE NOTES

As of FreeBSD 5.0, the *ucred* structure contains extensible fields. This means that the correct protocol must always be followed to create a fresh and writable credential structure: new credentials must always be derived from existing credentials using **crget**(), **crcopy**(), and **crcopysafe**().

In the common case, credentials required for access control decisions are used in a read-only manner. In these circumstances, the thread credential *td_ucred* should be used, as it requires no locking to access safely, and remains stable for the duration of the call even in the face of a multi-threaded application changing the process credentials from another thread.

During a process credential update, the process lock must be held across check and update, to prevent race conditions. The process credential, *td->td_proc->p_ucred*, must be used both for check and update. If a process credential is updated during a system call and checks against the thread credential are to be made later during the same system call, the thread credential must also be refreshed from the process credential so as to prevent use of a stale value. To avoid this scenario, it is recommended that system

calls updating the process credential be designed to avoid other authorization functions.

If temporarily elevated privileges are required for a thread, the thread credential can be replaced for the duration of an activity, or for the remainder of the system call. However, as a thread credential is often shared, appropriate care should be taken to make sure modifications are made to a writable credential through the use of **crget**() and **crcopy**().

Caution should be exercised when checking authorization for a thread or process perform an operation on another thread or process. As a result of temporary elevation, the target thread credential should *never* be used as the target credential in an access control decision: the process credential associated with the thread, *td->td_proc->p_ucred*, should be used instead. For example, p_candebug(9) accepts a target process, not a target thread, for access control purposes.

SEE ALSO

uihold(9)

AUTHORS

This manual page was written by Chad David <*davidc@acns.ab.ca*>.