NAME

cap_copy_ext, cap_size, cap_copy_int − capability state external representation translation

SYNOPSIS

```
#include <sys/capability.h>

ssize_t cap_size(cap_t cap_p);
ssize_t cap_copy_ext(void *ext_p, cap_t cap_p, ssize_t size);
cap_t cap_copy_int(const void * ext_p);
cap_t cap_copy_int_check(const void *cap_ext, ssize_t length);
```

Link with −*lcap*.

DESCRIPTION

These functions translate between internal and external representations of a capability state. The external representation is an exportable, contiguous, persistent representation of a capability state in user-managed space. The internal representation is managed by the capability functions in working storage.

cap_size() returns the total length (in bytes) that the capability state in working storage identified by *cap_p* would require when converted by **cap_copy_ext**(). This function is used primarily to determine the amount of buffer space that must be provided to the **cap_copy_ext**() function in order to hold the capability data record created from *cap_p*.

cap_copy_ext() copies a capability state in working storage, identified by *cap_p*, from system-managed space to user-managed space (pointed to by *ext_p*) and returns the length of the resulting data record. The size parameter represents the maximum size, in bytes, of the resulting data record. The **cap_copy_ext**() function will do any conversions necessary to convert the capability state from the undefined internal format to an exportable, contiguous, persistent data record. It is the responsibility of the user to allocate a buffer large enough to hold the copied data. The buffer length required to hold the copied data may be obtained by a call to the **cap_size**() function.

cap_copy_int() copies a capability state from a capability data record in user-managed space to a new capability state in working storage, allocating any memory necessary, and returning a pointer to the newly created capability state. The function initializes the capability state and then copies the capability state from the record pointed to by *ext_p* into the capability state, converting, if necessary, the data from a contiguous, persistent format to an opaque, internal format. Once copied into internal format, the object can be manipulated by the capability state manipulation functions (see**cap_clear**(3)). Note that the record pointed to by *ext_p* must have been obtained from a previous, successful call to **cap_copy_ext**() for this function to work successfully. The caller should free any releasable memory, when the capability state in working storage is no longer required, by calling **cap_free**() with the *cap_t* as an argument.

cap_copy_int_check() performs the same operation as **cap_copy_int**() but additionally checks that the provided external data's size is not larger than the noted length.

RETURN VALUE

cap_size() returns the length required to hold a capability data record on success, and −1 on failure.

cap_copy_ext() returns the number of bytes placed in the user managed space pointed to by *ext_p* on success, and −1 on failure.

cap_copy_int() and **cap_copy_int_check**() return a pointer to the newly created capability state in working storage on success, and NULL on failure.

On failure, **errno** is set to **EINVAL**, **ENOMEM**, or **ERANGE**.

CONFORMING TO

These functions are specified in the withdrawn POSIX.1e draft specification.

SEE ALSO

libcap(3), **cap_clear**(3), **cap_from_text**(3), **cap_get_file**(3), **cap_get_proc**(3), **cap_init**(3), **capabilities**(7)