NAME

libcapsicum — library interface to file descriptor lists

LIBRARY

library “libcapsicum”

SYNOPSIS

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

struct lc_fdlist *
lc_fdlist_new(void);

struct lc_fdlist *
lc_fdlist_global(void);

struct lc_fdlist *
lc_fdlist_dup(struct lc_fdlist *lfp);

void
lc_fdlist_free(struct lc_fdlist *lfp);

int
lc_fdlist_add(struct lc_fdlist *lfp, const char *subsystem,
    const char *classname, const char *name, int fd);

int
lc_fdlist_addcap(struct lc_fdlist *lfp, const char *subsystem,
    const char *classname, const char *name, int fd, cap_rights_t rights);

int
lc_fdlist_append(struct lc_fdlist *to, struct lc_fdlist *from);

int
lc_fdlist_getentry(struct lc_fdlist *lfp, char **subsystem,
    char **classname, char **name, int *fdp, int *pos);

int
lc_fdlist_lookup(struct lc_fdlist *lfp, const char *subsystem,
    const char *classname, const char **name, int *fdp, int *pos);
```

DESCRIPTION

These **libcapsicum** library routines create, manage, and destroy file descriptor lists. File descriptor lists are used by **libcapsicum** to describe sets of rights that should be delegated to newly created sandboxes, as well as binding them to names so that sandboxed code can look up file descriptors provided by code in the host without using hard-coded file descriptor numbers. This is necessary because file descriptors may not be the same in the host and sandbox environments. **libcapsicum** will arrange for all necessary name and descriptor information to be available in the sandbox, and file descriptor numbers returned in the sandbox are with respect to the sandbox’s file descriptor assignments.

Note that the file descriptor list code is not aware of any changes in file descriptor status that may happen as a result of application behavior, such as calls to open(2), dup(2,) or close(2.) As such, applications must update any file descriptor lists referring to manipulated descriptors if the descriptor list will later be queried for them, or used in creating a new sandbox.

File descriptor list creation and destruction

These functions create, duplicate, and free file descriptor lists:

lc_fdlist_new() allocates a new file descriptor list containing no file descriptor registrations. Sandboxed code may also use **lc_fdlist_global**() to query the global file descriptor list passed in when the sandbox was created.

lc_fdlist_dup() duplicates an existing file descriptor list, creating a new list with identical entries. Once duplicated, the lists may diverge; this allows the creation of a template list for a class of sandbox, followed by duplication and customization for a specific sandbox instance.

lc_fdlist_free() frees an existing file descriptor list; note that this does not close or otherwise modify file descriptors described by the list.

File descriptor list entries

Each file descriptor list entry is described by a three-part character string namespace:

subsystem Application or library name, globally unique in order to prevent collisions between software components in the same host/sandbox pair.

classname An application-specific or library-specific name, intended to reflect a specific software component within that application or library.

name A per-subsystem, per-class namespace, which might contain file names or other specific object instance description.

These functions insert and look up file descriptor list entries:

lc_fdlist_add() adds a file descriptor, *fd*, with the three-part name *subsystem*, *classname*, and *name* to the file descriptor list *lfp*. **lc_fdlist_add**() is identical except that it further registers a capability mask to apply to the descriptor during sandbox creation, avoiding the need for separate calls to .Xr cap_new in application code.

lc_fdlist_lookup() looks up a file descriptor using the three-part name *subsystem*, *classname*, and *name* from the file descriptor list *lfp*. **lc_fdlist_getentry**() may be used to iterate through all descriptors in the list.

RETURN VALUES

The **lc_fdlist_new**(), **lc_flist_global**(), and **lc_fdlist_dup**() functions return a pointer to the desired file descriptor list if successful; otherwise the value NULL is returned and the global variable *errno* is set to indicate the error.

The **lc_fdlist_add**(), **lc_fdlist_addcap**(), and **lc_fdlist_lookup**() functions return the value 0 if successful; otherwise the value −1 is returned and the global variable *errno* is set to indicate the error.

SEE ALSO

cap_new(2), close(2), dup(2), open(2), libcapsicum(3), libcapsicum_host(3), libcapsicum_sandbox(3),

HISTORY

Support for capabilities and capabilities mode was developed as part of the TrustedBSD Project.

BUGS

WARNING: THIS IS EXPERIMENTAL SECURITY SOFTWARE THAT MUST NOT BE RELIED ON IN PRODUCTION SYSTEMS. IT WILL BREAK YOUR SOFTWARE IN NEW AND UNEXPECTED WAYS.

AUTHORS

These functions were created by Jonathan Anderson at the University of Cambridge Computer Laboratory.