NAME

login_close, **login_getcapbool**, **login_getcaplist**, **login_getcapnum**, **login_getcapstr**, **login_getcapsize**, **login_getcaptime**, **login_getclass**, **login_getclassbyname**, **login_getpwclass**, **login_getstyle**, **login_getuserclass**, **login_setcryptfmt** - functions for accessing the login class capabilities database

LIBRARY

System Utilities Library (libutil, -lutil)

SYNOPSIS

#include <sys/types.h>
#include <login_cap.h>

void
login_close(*login_cap_t *lc*);

*login_cap_t **
login_getclassbyname(*const char *nam, const struct passwd *pwd*);

*login_cap_t **
login_getclass(*const char *nam*);

*login_cap_t **
login_getpwclass(*const struct passwd *pwd*);

*login_cap_t **
login_getuserclass(*const struct passwd *pwd*);

*const char **
login_getcapstr(*login_cap_t *lc, const char *cap, const char *def, const char *error*);

*const char ***
login_getcaplist(*login_cap_t *lc, const char *cap, const char *chars*);

*const char **
login_getpath(*login_cap_t *lc, const char *cap, const char *error*);

rlim_t
login_getcaptime(*login_cap_t *lc, const char *cap, rlim_t def, rlim_t error*);

rlim_t

login_getcapnum(*login_cap_t *lc, const char *cap, rlim_t def, rlim_t error*);

rlim_t
login_getcapsize(*login_cap_t *lc, const char *cap, rlim_t def, rlim_t error*);

int
login_getcapbool(*login_cap_t *lc, const char *cap, int def*);

*const char **
login_getstyle(*login_cap_t *lc, const char *style, const char *auth*);

*const char **
login_setcryptfmt(*login_cap_t *lc, const char *def, const char *error*);

DESCRIPTION

These functions represent a programming interface to the login classes database provided in login.conf(5). This database contains capabilities, attributes and default environment and accounting settings for users and programs running as specific users, as determined by the login class field within entries in */etc/master.passwd*.

Entries in login.conf(5) consist of colon ':' separated fields, the first field in each record being one or more identifiers for the record (which must be unique for the entire database), each separated by a '|', and may optionally include a description as the last 'name'. Remaining fields in the record consist of keyword/data pairs. Long lines may be continued with a backslash within empty entries, with the second and subsequent lines optionally indented for readability. This is similar to the format used in termcap(5), except that keywords are not limited to two significant characters, and are usually longer for improved readability. As with termcap entries, multiple records can be linked together (one record including another) using a field containing 'tc=<*recordid*>'. The result is that the entire record referenced by <*recordid*> replaces the *tc=* field at the point at which it occurs. See getcap(3) for further details on the format and use of a capabilities database.

The **login_cap** interface provides a convenient means of retrieving login class records with all *tc=* references expanded. A program will typically call one of **login_getclass**(), **login_getpwclass**(), **login_getuserclass**() or **login_getclassbyname**() according to its requirements. Each of these functions returns a login capabilities structure, *login_cap_t*, which may subsequently be used to interrogate the database for specific values using the rest of the API. Once the *login_cap_t* is of no further use, the **login_close**() function should be called to free all resources used.

The structure of *login_cap_t* is defined in <*login_cap.h*>, as:

```
typedef struct {
        char *lc_class;
        char *lc_cap;
        char *lc_style;
} login_cap_t;
```

The *lc_class* member contains a pointer to the name of the login class retrieved. This may not necessarily be the same as the one requested, either directly via **login_getclassbyname**(), or indirectly via a user's login record using **login_getpwclass**(), by class name using **login_getclass**(). If the referenced user has no login class specified in */etc/master.passwd*, the class name is NULL or an empty string. If the class specified does not exist in the database, each of these functions will search for a record with an id of 'default', with that name returned in the *lc_class* field. In addition, if the referenced user has a UID of 0 (normally, 'root', although the user name is not considered) then **login_getpwclass**() will search for a record with an id of 'root' before it searches for the record with the id of 'default'.

The *lc_cap* field is used internally by the library to contain the expanded login capabilities record. Programs with unusual requirements may wish to use this with the lower-level **getcap**() style functions to access the record directly.

The *lc_style* field is set by the **login_getstyle**() function to the authorisation style, according to the requirements of the program handling a login itself.

The **login_getclassbyname**() function is the basic means to get a *login_cap_t* object. It accepts two arguments: the first one, *name*, is the record identifier of the record to be retrieved; the second, *pwd*, is an optional pointer to a *passwd* structure. First of all, its arguments are used by the function to choose between system and user modes of operation. When in system mode, only the system login class database is used. When in user mode, the supplemental login class database in the user's home directory is allowed to override settings from the system database in a limited way as noted below. To minimize security implications, user mode is entered by **login_getclassbyname**() if and only if *name* is LOGIN_MECLASS ('me') and *pwd* is not NULL. Otherwise system mode is chosen.

In system mode, any record in the system database */etc/login.conf* can be accessed, and a fallback to the default record is provided as follows. If *name* is NULL, an empty string, or a class that does not exist in the login class database, then the LOGIN_DEFCLASS record ('default') is returned instead.

In user mode, only the LOGIN_MECLASS record ('me') is accessed and no fallback to the 'default' record is provided. The directory specified by *pwd->pw_dir* is searched for a login database file called *.login_conf*, and only the 'me' capability record contained within it may override the system record with the same name while other records are ignored. Using this scheme, an application can explicitly allow users to override a selected subset of login settings. To do so, the application should obtain two

login_cap_t objects, one in user mode and the other in system mode, and then query the user object before the system object for login parameters that are allowed to be overridden by the user. For example, the user's *.login_conf* can provide a convenient way for a user to set up their preferred login environment before the shell is invoked on login if supported by login(1).

Note that access to the */etc/login.conf* and *.login_conf* files will only be performed subject to the security checks documented in _secure_path(3) for the uids 0 and *pwd->pw_uid* respectively.

If the specified record is NULL, empty or does not exist, and the system has no 'default' record available to fall back to, there is a memory allocation error or for some reason cgetent(3) is unable to access the login capabilities database, this function returns NULL.

The functions **login_getclass**(), **login_getpwclass**() and **login_getuserclass**() retrieve the applicable login class record for the user's passwd entry or class name by calling **login_getclassbyname**(). On failure, NULL is returned. The difference between these functions is that **login_getuserclass**() includes the user's overriding *.login_conf* that exists in the user's home directory, and **login_getpwclass**() and **login_getclass**() restrict lookup only to the system login class database in */etc/login.conf*. As explained earlier, **login_getpwclass**() differs from **login_getclass**() in that it allows the default class for a super-user as 'root' if none has been specified in the password database. Otherwise, if the passwd pointer is NULL, or the user record has no login class, then the system 'default' entry is retrieved. Essentially, **login_getclass**(*name*) is equivalent to **login_getclassbyname**(*name, NULL*) and **login_getuserclass**(*pwd*) to **login_getclassbyname**(*LOGIN_MECLASS, pwd*).

Once a program no longer wishes to use a *login_cap_t* object, **login_close**() may be called to free all resources used by the login class. The **login_close**() function may be passed a NULL pointer with no harmful side-effects.

The remaining functions may be used to retrieve individual capability records. Each function takes a *login_cap_t* object as its first parameter, a capability tag as the second, and remaining parameters being default and error values that are returned if the capability is not found. The type of the additional parameters passed and returned depend on the *type* of capability each deals with, be it a simple string, a list, a time value, a file or memory size value, a path (consisting of a colon-separated list of directories) or a boolean flag. The manpage for login.conf(5) deals in specific tags and their type.

Note that with all functions in this group, you should not call free(3) on any pointers returned. Memory allocated during retrieval or processing of capability tags is automatically reused by subsequent calls to functions in this group, or deallocated on calling **login_close**().

login_getcapstr() This function returns a simple string capability. If the string is not found, then the value in *def* is returned as the default value, or if an error occurs, the value in the

error parameter is returned.

login_getcaplist() This function returns the value corresponding to the named capability tag as a list of values in a NULL terminated array. Within the login class database, some tags are of type *list*, which consist of one or more comma- or space separated values. Usually, this function is not called directly from an application, but is used indirectly via **login_getstyle**().

login_getpath() This function returns a list of directories separated by colons ‘:’. Capability tags for which this function is called consist of a list of directories separated by spaces.

login_getcaptime()

This function returns a *time value* associated with a particular capability tag with the value expressed in seconds (the default), minutes, hours, days, weeks or (365 day) years or any combination of these. A suffix determines the units used: ‘S’ for seconds, ‘M’ for minutes, ‘H’ for hours, ‘D’ for days, ‘W’ for weeks and ‘Y’ for 365 day years. Case of the units suffix is ignored.

Time values are normally used for setting resource, accounting and session limits. If supported by the operating system and compiler (which is true of FreeBSD), the value returned is a *quad* (*long long*), of type *rlim_t*. A value ‘inf’ or ‘infinity’ may be used to express an infinite value, in which case RLIM_INFINITY is returned.

login_getcapnum()

This function returns a numeric value for a tag, expressed either as ‘tag=<value>’ or the standard **cgetnum**() format ‘tag#<value>’. The first format should be used in preference to the second, the second format is provided for compatibility and consistency with the getcap(3) database format where numeric types use the ‘#’ as the delimiter for numeric values. If in the first format, then the value given may be ‘inf’ or ‘infinity’ which results in a return value of RLIM_INFINITY. If the given capability tag cannot be found, the *def* parameter is returned, and if an error occurs, the *error* parameter is returned.

login_getcapsize()

login_getcapsize() returns a value representing a size (typically, file or memory) which may be expressed as bytes (the default), 512 byte blocks, kilobytes, megabytes, gigabytes, and on systems that support the *long long* type, terabytes. The suffix used determines the units, and multiple values and units may be used in combination (e.g. 1m500k = 1.5 megabytes). A value with no suffix is interpreted as bytes, ‘B’ as 512-byte blocks, ‘K’ as kilobytes, ‘M’ as megabytes, ‘G’ as gigabytes and ‘T’ as

terabytes. Case is ignored. The error value is returned if there is a login capabilities database error, if an invalid suffix is used, or if a numeric value cannot be interpreted.

login_getcapbool()

This function returns a boolean value tied to a particular flag. It returns 0 if the given capability tag is not present or is negated by the presence of a ‘tag@’ (see getcap(3) for more information on boolean flags), and returns 1 if the tag is found.

login_getstyle() This function is used by the login authorisation system to determine the style of login available in a particular case. The function accepts three parameters, the *lc* entry itself and two optional parameters, and authorisation type *auth* and *style*, and applies these to determine the authorisation style that best suites these rules.

- If *auth* is neither NULL nor an empty string, look for a tag of type ‘auth-<*auth*>’ in the capability record. If not present, then look for the default tag *auth*=.
- If no valid authorisation list was found from the previous step, then default to ‘passwd’ as the authorisation list.
- If *style* is not NULL or empty, look for it in the list of authorisation methods found from the previous step. If *style* is NULL or an empty string, then default to ‘passwd’ authorisation.
- If *style* is found in the chosen list of authorisation methods, then return that, otherwise return NULL.

This scheme allows the administrator to determine the types of authorisation methods accepted by the system, depending on the means by which the access occurs. For example, the administrator may require skey or kerberos as the authentication method used for access to the system via the network, and standard methods via direct dialup or console logins, significantly reducing the risk of password discovery by "snooping" network packets.

login_setcryptfmt()

The **login_setcryptfmt**() function is used to set the crypt(3) format using the *passwd_format* configuration entry. If no entry is found, *def* is taken to be used as the fallback. If calling crypt_set_format(3) on the specifier fails, *error* is returned to indicate this.

SEE ALSO

login(1), crypt(3), getcap(3), login_class(3), login.conf(5), termcap(5)

HISTORY

The functions **login_close**(), **login_getcapbool**(), **login_getcaplist**(), **login_getcapnum**(), **login_getcapstr**(), **login_getcapsize**(), **login_getcaptime**(), **login_getclass**(), **login_getclassbyname**(), **login_getpwclass**(), **login_getstyle**(), **login_getuserclass**() and **login_setcryptfmt**() first appeared in FreeBSD 2.1.5.