NAME

mktemp - make temporary file name (unique)

LIBRARY

Standard C Library (libc, -lc)

SYNOPSIS

#include <stdlib.h>

*char **
mktemp(*char *template*);

int
mkstemp(*char *template*);

int
mkostemp(*char *template*, *int oflags*);

int
mkostemps(*char *template*, *int suffixlen*, *int oflags*);

int
mkostempsat(*int dfd*, *char *template*, *int suffixlen*, *int oflags*);

*char **
mkdtemp(*char *template*);

#include <unistd.h>

int
mkstemps(*char *template*, *int suffixlen*);

DESCRIPTION

The **mktemp**() function takes the given file name template and overwrites a portion of it to create a file name. This file name is guaranteed not to exist at the time of function invocation and is suitable for use by the application. The template may be any file name with some number of ‘Xs’ appended to it, for example */tmp/temp.XXXXXX*. The trailing ‘Xs’ are replaced with a unique alphanumeric combination. The number of unique file names **mktemp**() can return depends on the number of ‘Xs’ provided; six ‘Xs’ will result in **mktemp**() selecting one of 56800235584 (62 ** 6) possible temporary file names.

The **mkstemp**() function makes the same replacement to the template and creates the template file, mode 0600, returning a file descriptor opened for reading and writing. This avoids the race between testing for a file's existence and opening it for use.

The **mkostemp**() function is like **mkstemp**() but allows specifying additional open(2) flags (defined in <*fcntl.h*>). The permitted flags are O_APPEND, O_DIRECT, O_SHLOCK, O_EXLOCK, O_SYNC and O_CLOEXEC.

The **mkstemps**() and **mkostemps**() functions act the same as **mkstemp**() and **mkostemp**() respectively, except they permit a suffix to exist in the template. The template should be of the form */tmp/tmpXXXXXXsuffix*. The **mkstemps**() and **mkostemps**() function are told the length of the suffix string.

The **mkostempsat**() function acts the same as **mkostemps**() but takes an additional directory descriptor as a parameter. The temporary file is created relative to the corresponding directory, or to the current working directory if the special value AT_FDCWD is specified. If the template path is an absolute path, the *dfd* parameter is ignored and the behavior is identical to **mkostemps**().

The **mkdtemp**() function makes the same replacement to the template as in **mktemp**() and creates the template directory, mode 0700.

RETURN VALUES

The **mktemp**() and **mkdtemp**() functions return a pointer to the template on success and NULL on failure. The **mkstemp**(), **mkostemp**() **mkstemps**() and **mkostemps**() functions return -1 if no suitable file could be created. If either call fails an error code is placed in the global variable *errno*.

ERRORS

The **mkstemp**(), **mkostemp**(), **mkstemps**(), **mkostemps**() and **mkdtemp**() functions may set *errno* to one of the following values:

[ENOTDIR] The pathname portion of the template is not an existing directory.

The **mkostemp**() and **mkostemps**() functions may also set *errno* to the following value:

[EINVAL] The *oflags* argument is invalid.

The **mkstemp**(), **mkostemp**(), **mkstemps**(), **mkostemps**() and **mkdtemp**() functions may also set *errno* to any value specified by the stat(2) function.

The **mkstemp**(), **mkostemp**(), **mkstemps**() and **mkostemps**() functions may also set *errno* to any value

specified by the open(2) function.

The **mkdtemp**() function may also set *errno* to any value specified by the mkdir(2) function.

NOTES

A common problem that results in a core dump is that the programmer passes in a read-only string to **mktemp**(), **mkstemp**(), **mkstemps**() or **mkdtemp**(). This is common with programs that were developed before ISO/IEC 9899:1990 ("ISO C90") compilers were common. For example, calling **mkstemp**() with an argument of "/tmp/tempfile.XXXXXX" will result in a core dump due to **mkstemp**() attempting to modify the string constant that was given.

The **mkdtemp**(), **mkstemp**() and **mktemp**() function prototypes are also available from <*unistd.h*>.

SEE ALSO

chmod(2), getpid(2), mkdir(2), open(2), stat(2)

STANDARDS

The **mkstemp**() and **mkdtemp**() functions are expected to conform to IEEE Std 1003.1-2008 ("POSIX.1"). The **mktemp**() function is expected to conform to IEEE Std 1003.1-2001 ("POSIX.1") and is not specified by IEEE Std 1003.1-2008 ("POSIX.1"). The **mkostemp**(), **mkstemps**(), **mkostemps**() and **mkostempsat**() functions do not conform to any standard.

HISTORY

A **mktemp**() function appeared in Version 7 AT&T UNIX. The **mkstemp**() function appeared in 4.4BSD. The **mkdtemp**() function first appeared in OpenBSD 2.2, and later in FreeBSD 3.2. The **mkstemps**() function first appeared in OpenBSD 2.4, and later in FreeBSD 3.4. The **mkostemp**() and **mkostemps**() functions appeared in FreeBSD 10.0. The **mkostempsat**() function appeared in FreeBSD 13.0.

BUGS

This family of functions produces filenames which can be guessed, though the risk is minimized when large numbers of ‘Xs’ are used to increase the number of possible temporary filenames. This makes the race in **mktemp**(), between testing for a file’s existence (in the **mktemp**() function call) and opening it for use (later in the user application) particularly dangerous from a security perspective. Whenever it is possible, **mkstemp**(), **mkostemp**() or **mkostempsat**() should be used instead, since they do not have the race condition. If **mkstemp**() cannot be used, the filename created by **mktemp**() should be created using the O_EXCL flag to open(2) and the return status of the call should be tested for failure. This will ensure that the program does not continue blindly in the event that an attacker has already created the file with the intention of manipulating or reading its contents.