NAME

tmpnam, tmpnam_r, tempnam – create a name for a temporary file

SYNOPSIS

#include <stdio.h>

char *tmpnam(char ***_s_);**

char *tmpnam_r(char ***_s_);**

char *tempnam(const char ***_dir_, const char ***_pfx_**);**

MT-LEVEL

See the **NOTES** section of this page.

DESCRIPTION

These functions generate file names that can safely be used for a temporary file.

tmpnam( ) always generates a file name using the path-prefix defined as **P_tmpdir** in the **<stdio.h>** header. If *s* is **NULL**, **tmpnam( )** leaves its result in an internal static area and returns a pointer to that area. The next call to **tmpnam( )** will destroy the contents of the area. If *s* is not **NULL**, it is assumed to be the address of an array of at least **L_tmpnam** bytes, where **L_tmpnam** is a constant defined in **<stdio.h>**; **tmpnam( )** places its result in that array and returns *s*.

tmpnam_r( ) has the same functionality as **tmpnam( )** except that if *s* is a **NULL** pointer, the function returns **NULL.**

tempnam( ) allows the user to control the choice of a directory. The argument *dir* points to the name of the directory in which the file is to be created. If *dir* is **NULL** or points to a string that is not a name for an appropriate directory, the path-prefix defined as **P_tmpdir** in the **<stdio.h>** header is used. If that directory is not accessible, **/tmp** will be used as a last resort. This entire sequence can be up-staged by providing an environment variable **TMPDIR** in the user's environment, whose value is the name of the desired temporary-file directory.

Many applications prefer their temporary files to have certain favorite initial letter sequences in their names. Use the *pfx* argument for this. This argument may be **NULL** or point to a string of up to five characters to be used as the first few characters of the temporary-file name.

tempnam( ) uses **malloc**(3C) to get space for the constructed file name, and returns a pointer to this area. Thus, any pointer value returned from **tempnam( )** may serve as an argument to **free**(3C) (see **malloc**(3C)). If **tempnam( )** cannot return the expected result for any reason—for example, **malloc**(3C) failed—or none of the above mentioned attempts to find an appropriate directory was successful, a **NULL** pointer will be returned.

tempnam( ) fails if there is not enough space.

SEE ALSO

creat(2), **unlink**(2), **fopen**(3S), **free**(3C), **malloc**(3C), **mktemp**(3C), **tmpfile**(3S)

NOTES

The **tmpnam_r( )** interface is as proposed in the POSIX.4a Draft #6 document, and is subject to change to be compliant to the standard when it is accepted.

When compiling multi-thread applications, the **_REENTRANT** flag must be defined on the compile line. This flag should only be used in multi-thread applications.

These functions generate a different file name each time they are called.

Files created using these functions and either **fopen**(3S) or **creat**(2) are temporary only in the sense that they reside in a directory intended for temporary use, and their names are unique. It is the user's responsibility to remove the file when its use is ended.

If called more than **TMP_MAX** (defined in **<stdio.h>**) times in a single process, these functions start recycling previously used names.

Between the time a file name is created and the file is opened, it is possible for some other process to create a file with the same name. This can never happen if that other process is using these functions or **mktemp**(3C) and the file names are chosen to render duplication by other means unlikely.

tempnam( ) is safe in multi-thread applications. **tmpnam( )** is unsafe in multi-thread applications, **tmpnam_r( )** should be used instead.

On Solaris systems, the default value for **P_tmpdir** is **/var/tmp.**