NAME

tempnam − create a name for a temporary file

LIBRARY

Standard C library (*libc*, *−lc*)

SYNOPSIS

#include <stdio.h>

char *tempnam(const char ***dir*, const char ***pfx***);**

Feature Test Macro Requirements for glibc (see **feature_test_macros**(7)):

tempnam():
  Since glibc 2.19:
    _DEFAULT_SOURCE
  glibc 2.19 and earlier:
    _BSD_SOURCE || _SVID_SOURCE

DESCRIPTION

Never use this function. Use **mkstemp**(3) or **tmpfile**(3) instead.

The **tempnam**() function returns a pointer to a string that is a valid filename, and such that a file with this name did not exist when **tempnam**() checked. The filename suffix of the pathname generated will start with *pfx* in case *pfx* is a non-NULL string of at most five bytes. The directory prefix part of the pathname generated is required to be "appropriate" (often that at least implies writable).

Attempts to find an appropriate directory go through the following steps:

a) In case the environment variable **TMPDIR** exists and contains the name of an appropriate directory, that is used.

b) Otherwise, if the *dir* argument is non-NULL and appropriate, it is used.

c) Otherwise, *P_tmpdir* (as defined in *<stdio.h>*) is used when appropriate.

d) Finally an implementation-defined directory may be used.

The string returned by **tempnam**() is allocated using **malloc**(3) and hence should be freed by **free**(3).

RETURN VALUE

On success, the **tempnam**() function returns a pointer to a unique temporary filename. It returns NULL if a unique name cannot be generated, with *errno* set to indicate the error.

ERRORS

ENOMEM
  Allocation of storage failed.

ATTRIBUTES

For an explanation of the terms used in this section, see **attributes**(7).

Interface	Attribute	Value
tempnam()	Thread safety	MT-Safe env

STANDARDS

SVr4, 4.3BSD, POSIX.1-2001. POSIX.1-2008 marks **tempnam**() as obsolete.

NOTES

Although **tempnam**() generates names that are difficult to guess, it is nevertheless possible that between the time that **tempnam**() returns a pathname, and the time that the program opens it, another program might create that pathname using **open**(2), or create it as a symbolic link. This can lead to security holes. To avoid such possibilities, use the **open**(2) **O_EXCL** flag to open the pathname. Or better yet, use **mkstemp**(3) or **tmpfile**(3).

SUSv2 does not mention the use of **TMPDIR**; glibc will use it only when the program is not set-user-ID.

On SVr4, the directory used under **d)** is */tmp* (and this is what glibc does).

Because it dynamically allocates memory used to return the pathname, **tempnam**() is reentrant, and thus thread safe, unlike **tmpnam**(3).

The **tempnam**() function generates a different string each time it is called, up to **TMP_MAX** (defined in <*stdio.h*>) times. If it is called more than **TMP_MAX** times, the behavior is implementation defined.

tempnam() uses at most the first five bytes from *pfx*.

The glibc implementation of **tempnam**() fails with the error **EEXIST** upon failure to find a unique name.

BUGS

The precise meaning of "appropriate" is undefined; it is unspecified how accessibility of a directory is determined.

SEE ALSO

mkstemp(3), **mktemp**(3), **tmpfile**(3), **tmpnam**(3)