opendir() — Open a directory

Last Updated: 2021-03-03

Standards

Standards / Extensions	C or C++	Dependencies
POSIX.1 XPG4 XPG4.2 Single UNIX Specification, Version 3	both	

Format

```
#define _POSIX_SOURCE
#include <dirent.h>

DIR *opendir(const char *dirname);
```

General description

Opens a directory so that it can be read with readdir() or __readdir2(). *dirname* is a string giving the name of the directory you want to open. The first readdir() or __readdir2() call reads the first entry in the directory.

Returned value

If successful, opendir() returns a pointer to a DIR object. This object describes the directory and is used in subsequent operations on the directory, in the same way that FILE objects are used in file I/O operations.

If unsuccessful, opendir() returns a NULL pointer and sets errno to one of the following values:

Error Code
   Description
EACCES
   The process does not have permission to search some component of *dirname*, or it does not have read permission on the directory itself.
ELOOP
   A loop exists in the symbolic links. This error is issued if more than POSIX_SYMLOOP (defined in the limits.h header file) symbolic links are encountered during resolution of the *dirname* argument.
EMFILE
   The process has too many other file descriptors already open.
ENAMETOOLONG
   dirname is longer than **PATH_MAX** characters, or some component of *dirname* is longer than **NAME_MAX** characters while _POSIX_NO_TRUNC is in effect. For symbolic links, the length of the pathname string substituted for a

symbolic link exceeds **PATH_MAX.** The **PATH_MAX** and **NAME_MAX** values can be determined using pathconf().

ENFILE

The entire system has too many other file descriptors already open.

ENOENT

The directory *dirname* does not exist.

ENOMEM

There is not enough storage available to open the directory.

ENOTDIR

Some component of the *dirname* pathname is not a directory.

Example

CELEBO01

```
/* CELEBO01

   This example opens a directory.

 */
#define _POSIX_SOURCE
#include <dirent.h>
#include <errno.h>
#include <sys/stat.h>
#include <sys/types.h>
#undef _POSIX_SOURCE
#include <stdio.h>

void traverse(char *fn, int indent) {
  DIR *dir;
  struct dirent *entry;
  int count;
  char path[1025];
  struct stat info;
```

```
    for (count=0; count<indent; count++) printf("  ");
    printf("%s\n", fn);

    if ((dir = opendir(fn)) == NULL)
      perror("opendir() error");
    else {
      while ((entry = readdir(dir)) != NULL) {
        if (entry->d_name[0] != '.') {
          strcpy(path, fn);
          strcat(path, "/");
          strcat(path, entry->d_name);
          if (stat(path, &info) != 0)
            fprintf(stderr, "stat() error on %s: %s\n", path,
                    strerror(errno));
          else if (S_ISDIR(info.st_mode))
                 traverse(path, indent+1);
        }
      }
      closedir(dir);
    }
  }

  main() {
    puts("Directory structure:");
    traverse("/etc", 0);
  }
```

Output

```
Directory structure:
/etc
  /etc/samples
    /etc/samples/IBM
  /etc/IBM
```

Related information

- dirent.h — POSIX directory access
- stdio.h — Standard input and output
- sys/types.h — typedef symbols and structures
- closedir() — Close a directory
- __opendir2() — Open a directory
- readdir() — Read an entry from a directory
- rewinddir() — Reposition a directory stream to the beginning
- seekdir() — Set position of directory stream
- telldir() — Current location of directory stream

Parent topic:

→ Library functions

Previous
open() — Open a file

Next
__opendir2() — Open a directory