# fdopendir - Man Page

*открыть каталог, связанный с файловым дескриптором*

## Пролог

Эта страница руководства является частью Руководства программиста POSIX. Реализация этого интерфейса в Linux может отличаться (обратитесь к соответствующей странице руководства Linux для получения подробной информации о поведении Linux), или интерфейс может быть не реализован в Linux.

## Краткий обзор

```
#включить <dirent.h>

DIR *fdopendir(int fd);
DIR *opendir(const char *dirname);
```

## Описание

Функция *fdopendir*() должна быть эквивалентна функции *opendir*(), за исключением того, что каталог задается файловым дескриптором, а не именем. Смещение файла, связанное с файловым дескриптором во время вызова, определяет, какие записи будут возвращены.

После успешного возврата из *fdopendir*() файловый дескриптор находится под контролем системы, и если какая-либо попытка закрыть файловый дескриптор или изменить состояние связанного описания, кроме как с помощью *closedir*(), *readdir*(), *readdir_r*(), *rewindir*(), или *seekdir*() , поведение не определено. При вызове *closedir*() файловый дескриптор должен быть закрыт.

# fdopendir - Man Page

Функция *opendir*() должна открыть поток каталогов, соответствующий каталогу, названному аргументом *dirname*. Поток каталогов располагается в первой записи. Если тип **DIR** реализован с использованием файлового дескриптора, приложения смогут открывать в общей сложности только файлы и каталоги {OPEN_MAX}.

Если тип **DIR** реализован с использованием файлового дескриптора, то дескриптор должен быть получен так, как если бы флаг O_DIRECTORY был передан в *open*().

## Возвращаемое значение

После успешного завершения эти функции должны возвращать указатель на объект типа **DIR**. В противном случае эти функции должны возвращать нулевой указатель и устанавливать *errno* для указания ошибки.

## Ошибки

Функция *fdopendir*() завершится ошибкой, если:

**EBADF**

Аргумент *fd* не является допустимым файловым дескриптором, открытым для чтения.

**ENOTDIR**

Дескриптор *fd* не связан с каталогом.

Функция *opendir*() завершится ошибкой, если:

**EACCES**

# **fdopendir** - Man Page

**ELOOP**

Существует цикл в символических ссылках, встречающихся при разрешении аргумента *dirname* .

**ENAMETOOLONG**

Длина компонента пути больше {NAME_MAX} .

**ENOENT**

Компонент *dirname* не называет существующий каталог, или *dirname* - это пустая строка.

**ЭНОТДИР**

Компонент *dirname* называет существующий файл, который не является ни каталогом, ни символической ссылкой на каталог.

Функция *opendir*() может выйти из строя, если:

**ELOOP**

При разрешении аргумента dirname было обнаружено более символических ссылок {SYMLOOP_MAX}.

**EMFILE**

Все файловые дескрипторы, доступные процессу, в настоящее время открыты.

**RU - <url>**

Длина пути превышает {PATH_MAX}, или разрешение пути символической ссылки дало промежуточный результат с длиной, превышающей {PATH_MAX} .

**ENFILE**

В настоящее время в системе открыто слишком много файлов.

# fdopendir - Man Page

## Примеры

### Открыть поток каталогов

Следующий фрагмент программы демонстрирует, как используется функция opendir().

```
#включить <dirent.h>
...
 DIR *dir;
struct dirent *dp;
...
 if ((dir = opendir (".")) == NULL) {
 perror ("Не может открыться");
 выход (1);
 }

 while ((dp = readdir (dir)) != NULL) {
...
```

### Найти и открыть файл

Следующая программа выполняет поиск по заданному каталогу в поисках файлов, имя которых не начинается с точки и размер которых больше 1 MiB.

```
#включить <stdio.h>
#включить <dirent.h>
#включить <fcntl.h>
#включить <sys/stat.h>
#включить <stdint.h>
#включить <stdlib.h>
#include <unistd.h>
```

# fdopendir - Man Page

```
`
 struct stat statbuf;
 DIR *d;
struct dirent *dp;
int dfd, ffd;

 if ((d = fdopendir((dfd = open("./tmp", O_RDONLY)))) ==
 fprintf(stderr, "Не удается открыть каталог ./tmp\n");
 выход (1);
 }
 while ((dp = readdir(d)) != NULL) {
 if (dp->d_name[0] == '.')
continue;
 / * здесь есть возможное условие гонки в виде файла
 * может быть переименован между readdir и open */
 if ((ffd = openat(dfd, dp->d_name, O_RDONLY)) == -1) {
 perror(dp->d_name);
 продолжение;
 }
if (fstat(ffd, &statbuf) == 0 && statbuf.st_size> (1024*
 /* нашел его ... */
 printf("%s: %jdK\n", dp->d_name,
(intmax_t)(statbuf.st_size / 1024));
 }
 закрыть (ffd);
 }
 closedir(d); // обратите внимание, что это неявно закры
return 0;
}
```

## Использование приложений

Функция *opendir*() должна использоваться в сочетании с *readdir*(), *closedir*() и *rewindir*() для изучения содержимого каталога (см. Раздел Примеры в *readdir*() ). Этот метод рекомендуется для переносимости.

# **fdopendir** - Man Page

Цель функции *fdopendir*() состоит в том, чтобы разрешить открытие файлов в каталогах, отличных от текущего рабочего каталога, без воздействия условий гонки. Любая часть пути к файлу может быть изменена параллельно вызову *opendir*(), что приведет к неопределенному поведению.

Основываясь на исторических реализациях, правила о файловых дескрипторах применяются и к потокам каталогов. Однако этот том POSIX.1-2017 не требует, чтобы поток каталогов был реализован с использованием файловых дескрипторов. В описании *closedir*() уточняется, что если для потока каталогов используется файловый дескриптор, то *closedir*() обязательно освобождает файловый дескриптор. Когда файловый дескриптор используется для реализации потока каталогов, он ведет себя так, как если бы FD_CLOEXEC был установлен для файлового дескриптора.

Записи каталога для dot и dot-dot являются необязательными. Этот том POSIX.1-2017 не дает возможности *априори проверить* их существование, поскольку переносимое приложение должно быть написано для поиска (и обычно игнорирования) этих записей. Написание кода, предполагающего, что они являются первыми двумя записями, не всегда работает, так как многие реализации позволяют им быть отличными от первых двух записей, а “нормальная” запись предшествует им. Существует незначительная ценность в предоставлении способа определить, что делает реализация, потому что код для работы с dot и dot-dot должен быть написан в любом случае и потому, что такой флаг добавит в список этих флагов (который сам по себе оказался нежелательным) и может быть злоупотреблен.

Поскольку структура и распределение буфера, если таковые имеются, для операций с каталогами определяются реализацией, этот том POSIX.1-2017 не предъявляет никаких требований к переносимости для ошибочных программных конструкций, ошибочных данных или использования неопределенных значений, таких как использование или ссылка *на значение*

# fdopendir - Man Page

## Будущие направления

Нет.

## См. Также

closedir(), dirfd(), фстатат(), Открыть(), readdir(), rewinddir(), символическая ссылка()

Объем базовых определений POSIX.1-2017, <dirent.h>, <sys_types.h>

## Авторские права



## Ссылка

# **fdopendir** - Man Page

(3p), символическая ссылка (3p), telldir (3p).

2017 IEEE/The Open Group POSIX Programmer's Manual