open - Man Page

открыть файл

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.

Краткое описание

```
#include <sys/stat.h>
#include <fcntl.h>

int open(const char *path, int oflag, ...);
int openat(int fd, const char *path, int oflag, ...);
```

Описание

Функция *open*() устанавливает соединение между файлом и файловым дескриптором. Он должен создать описание открытого файла, которое ссылается на файл, и дескриптор файла, который ссылается на это описание открытого файла. Файловый дескриптор используется другими функциями ввода-вывода для ссылки на этот файл. *Аргумент* path указывает на путь к файлу.

Функция *open*() возвращает файловый дескриптор для именованного файла, выделенного, как описано в *разделе 2.14*, *Распределение файловых дескрипторов*. Описание открытого файла является новым, и поэтому дескриптор файла не должен делиться им с каким-либо другим процессом в

open - Man Page

установлен в *oflag*.

Смещение файла, используемое для обозначения текущей позиции в файле, должно быть установлено на начало файла.

Флаги состояния файла и режимы доступа к файлу описания открытого файла устанавливаются в соответствии со значением *oflag*.

Значения для *oflag* строятся с помощью побитового включения ИЛИ флагов из следующего списка, определенного в *<fcntl.h>*. Приложения должны указывать ровно одно из первых пяти значений (режимов доступа к файлам) ниже в значении *oflag*:

O_EXEC

Открыть только для выполнения (файлы, не входящие в каталог). Результат не определен, если этот флаг применяется к каталогу.

O_RDONLY

Открыть только для чтения.

O_RDWR

Открыть для чтения и записи. Результат не определен, если этот флаг применяется к FIFO.

O_SEARCH

Откройте каталог только для поиска. Результат не определен, если этот флаг применяется к файлу, не относящемуся к каталогу.

O_WRONLY

Открыть только для записи.

Можно использовать любую комбинацию из следующих:

O_APPEND

open - Man Page

O_CLOEXEC

If set, the FD_CLOEXEC flag for the new file descriptor shall be set.

O_CREAT

Если файл существует, этот флаг не имеет никакого эффекта, кроме как указано в разделе O_EXCL ниже. В противном случае, если O_DIRECTORY не установлен, файл должен быть создан как обычный файл; идентификатор пользователя файла должен быть установлен на эффективный идентификатор пользователя процесса; идентификатор группы файла должен быть установлен на идентификатор группы родительского каталога файла или на эффективный идентификатор группыпроцесса; и биты разрешения доступа (см. *<sys/stat.h>*) режима файла должны быть установлены в значение аргумента, следующего за аргументом oflag, принятым как тип **mode_t** модифицировано следующим образом: побитовое И выполняется на битах режима файла и соответствующих битах в дополнении к маске создания режима файла процесса. Таким образом, все биты в файловом режиме, для которых установлен соответствующий бит в маске создания файлового режима, очищаются. Когда установлены биты, отличные от битов разрешения файла, эффект не определен. Аргумент, следующий за *oflag* аргумент не влияет на то, открыт ли файл для чтения, записи или для обоих. Реализации должны обеспечивать способ инициализации идентификатора группы файла в идентификатор группы родительского каталога. Реализации могут, но не должны, предоставлять определенный реализацией способ инициализации идентификатора группы файла в эффективный идентификатор группы вызывающего процесса.

O_DIRECTORY

Если *путь* преобразуется в файл, не относящийся к каталогу, выполните сбой и установите *errno* в **[ENOTDIR]**.

open - Man Page

завершаться в соответствии с завершением целостности синхронизированных данных ввода-вывода.

O_EXCL

Если O_CREAT и O_EXCL установлены, *open*() завершится ошибкой, если файл существует. Проверка существования файла и создание файла, если он не существует, должны быть атомарными по отношению к другим потокам, выполняющим *open*(), называющим одно и то же имя файла в том же каталоге с набором O_EXCL и O_CREAT . Если заданы O_EXCL и O_CREAT, а *путь* именует символическую ссылку, *open*() завершится ошибкой и установит *errno* в **[EEXIST]**, независимо от содержимого символической ссылки. Если O_EXCL установлен, а O_CREAT не установлен, результат не определен.

O_NOCTTY

Если set и *path* идентифицируют терминальное устройство, *open*() не приведет к тому, что терминальное устройство станет управляющим терминалом для процесса. Если *path* не идентифицирует терминальное устройство, O_NOCTTY игнорируется.

O_NOFOLLOW

Если *path* называет символическую ссылку, потерпите неудачу и установите *errno* в **[ELOOP].**

O_NONBLOCK

При открытии FIFO с набором O_RDONLY или O_WRONLY:

* Если O_NONBLOCK установлен, *open*() только для чтения вернется без задержки. Функция *open*() только для записи возвращает ошибку, если ни один процесс в данный момент не имеет файла, открытого для чтения.

*

open - Man Page

записи. *open*() только для записи блокирует вызывающий поток до тех пор, пока поток не откроет файл для чтения.

При открытии блочного специального или символьного специального файла, поддерживающего неблокирующее открытие:

* Если O_NONBLOCK установлен, функция *open*() должна вернуться без блокировки, чтобы устройство было готово или доступно. Последующее поведение устройства зависит от конкретного устройства.

* Если O_NONBLOCK пуст, функция *open*() блокирует вызывающий поток до тех пор, пока устройство не будет готово или доступно, прежде чем вернуться.

В противном случае флаг O_NONBLOCK не должен вызывать ошибку, но не указано, будут ли флаги состояния файла включать флаг O_NONBLOCK .

O_RSYNC

Операции ввода-вывода чтения в файловом дескрипторе должны выполняться с тем же уровнем целостности, что и флаги O_DSYNC и O_SYNC. Если O_DSYNC и O_RSYNC заданы в *oflag*, все операции ввода-вывода с файловым дескриптором должны быть завершены в соответствии с завершением целостности синхронизированных данных ввода-вывода. Если флаги O_SYNC и O_RSYNC установлены, все операции ввода-вывода с файловым дескриптором должны быть завершены в соответствии с завершением целостности синхронизированного файла ввода-вывода.

O_SYNC

open - Man Page

синхронизированного файла ввода-вывода.

Флаг O_SYNC должен поддерживаться для обычных файлов, даже если опция синхронизированного ввода и вывода не поддерживается.

O_TRUNC

Если файл существует и является обычным файлом, и файл успешно открыт O_RDWR или O_WRONLY, его длина должна быть усечена до 0, а режим и владелец должны быть неизменными. Это не должно влиять на специальные файлы FIFO или файлы терминальных устройств. Его влияние на другие типы файлов определяется реализацией. Результат использования O_TRUNC без O_RDWR или O_WRONLY не определен.

O_TTY_INIT

Если *path* идентифицирует терминальное устройство, отличное от псевдотерминального, устройство еще не открыто ни в одном процессе, и либо O_TTY_INIT установлен в *oflag*, либо O_TTY_INIT имеет нулевое значение, *open*() устанавливает любые нестандартные параметры терминала termios structure в состояние, которое обеспечивает соответствующее поведение; см.Объем базовых определений POSIX.1-2017, *Раздел 11.2*, *Параметры, которые можно установить*. Не указано, влияет ли O_TTY_INIT, если устройство уже открыто в любом процессе. Если *путь* определяет подчиненную сторону псевдотерминала, который еще не открыт ни в одном процессе, *откройте*() устанавливает любые нестандартные терминальные параметры структуры termios в состояние, обеспечивающее соответствующее поведение, независимо от того, установлен ли O_TTY_INIT . Если *path* не идентифицирует терминальное устройство, O_TTY_INIT игнорируется.

Если заданы O_CREAT и O_DIRECTORY, а запрошенный режим доступа не является ни O_WRONLY, ни O_RDWR, результат не указан.

open - Man Page

доступа к данным, последней модификации данных и последнего изменения состояния файла файла и метки времени последней модификации данных и последнего изменения состояния файла родительского каталога.

Если задано значение O_TRUNC и файл существовал ранее, после успешного завершения *open*() отметит для обновления последнюю модификацию данных и последние временные метки изменения статуса файла файла.

Если установлены флаги O_SYNC и O_DSYNC, эффект будет таким, как если бы был установлен только флаг O_SYNC.

Если *path* ссылается на файл STREAMS, *oflag* может быть построен из O_NONBLOCK или O_RDONLY , O_WRONLY или O_RDWR . Другие значения флагов не применимы к устройствам STREAMS и не влияют на них. Значение O_NONBLOCK влияет на работу драйверов ПОТОКОВ и некоторых функций, применяемых к файловым дескрипторам, связанным с файлами ПОТОКОВ. Для драйверов ПОТОКОВ реализация O_NONBLOCK зависит от устройства.

Приложение должно убедиться, что оно указывает флаг O_TTY_INIT при первом открытии терминального устройства с момента загрузки системы или с момента закрытия устройства процессом, который в последний раз его открывал. Приложению не нужно указывать флаг O_TTY_INIT при открытии псевдотерминалов. Если *path* называет главную сторону псевдотерминального устройства, то не указано, блокирует ли *open*() подчиненную сторону, чтобы ее нельзя было открыть. Соответствующие приложения должны вызывать *unlockpt*() перед открытием подчиненной стороны.

Наибольшее значение, которое может быть корректно представлено в объекте типа **off_t**, должно быть установлено как максимальное смещение в описании открытого файла.

Функция *openat*() должна быть эквивалентна функции *open*(), за исключением случая, когда *path* указывает относительный путь. В этом случае файл, который нужно открыть, определяется относительно каталога,

open - Man Page

дескриптором, не является O_SEARCH, функция должна проверить, разрешен
ли поиск в каталоге с использованием текущих разрешений каталога, лежащего в основе файлового дескриптора. Если режим доступа O_SEARCH, функция не должна выполнять проверку.

Параметр *oflag* и необязательный четвертый параметр точно соответствуют параметрам *open*().

Если *openat*() передается специальное значение AT_FDCWD в параметре *fd* , должен использоваться текущий рабочий каталог, и поведение должно быть идентичным вызову *open*() .

Возвращаемое значение

После успешного завершения эти функции должны открыть файл и вернуть неотрицательное целое число, представляющее дескриптор файла. В противном случае эти функции должны возвращать -1 и устанавливать *errno* для указания ошибки. Если возвращается значение -1, файлы не создаются и не изменяются.

Ошибки

Эти функции не будут работать, если:

EACCES

Отказано в разрешении поиска для компонента префикса пути, или файл существует, и разрешения, указанные *oflag*, отклонены, или файл не существует, и разрешение на запись отклонено для родительского каталога создаваемого файла, или указан O_TRUNC, и разрешение на запись отклонено.

EEXIST

O_CREAT и O_EXCL установлены, и именованный файл существует.

open - Man Page

EINVAL

Реализация не поддерживает синхронизированный ввод-вывод для этого файла.

EIO Аргумент *path* называет файл STREAMS, и во время *open*() произошло зависание или ошибка.

EISDIR

Именованный файл является каталогом, а *oflag* включает O_WRONLY или O_RDWR или включает O_CREAT без O_DIRECTORY .

ELOOP

Цикл существует в символических ссылках, встречающихся при разрешении аргумента *path* , или был указан O_NOFOLLOW, и аргумент *path* называет символическую ссылку.

EMFILE

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

RU - <url>

Длина компонента пути больше {NAME_MAX} .

ENFILE

Максимально допустимое количество файлов, открытых в данный момент в системе.

ENOENT

O_CREAT не установлен, и компонент *path* не называет существующий файл, или O_CREAT установлен, и компонент префикса path *path* не называет существующий файл, или *path* указывает на пустую строку.

ENOENT или **ENOTDIR**

open - Man Page

несколькими конечными символами <slash> . Если *путь* без завершающих символов <slash> будет называть существующий файл, **ошибка [ENOENT]** не должна возникать.

ENOSR

Аргумент *path* называет файл на основе ПОТОКОВ, и система не может выделить ПОТОК.

ENOSPC

Каталог или файловая система, которые будут содержать новый файл, не могут быть расширены, файл не существует, и указан O_CREAT .

ENOTDIR

A component of the path prefix names an existing file that is neither a directory nor a symbolic link to a directory; or O_CREAT and O_EXCL are not specified, the *path* argument contains at least one non-<slash> character and ends with one or more trailing <slash> characters, and the last pathname component names an existing file that is neither a directory nor a symbolic link to a directory; or O_DIRECTORY was specified and the *path* argument resolves to a non-directory file.

ENXIO

O_NONBLOCK is set, the named file is a FIFO, O_WRONLY is set, and no process has the file open for reading.

ENXIO

The named file is a character special or block special file, and the device associated with this special file does not exist.

EOVERFLOW

The named file is a regular file and the size of the file cannot be represented correctly in an object of type **off_t**.

open - Man Page

O_WRONLY, O_RDWR, O_CREAT (if the file does not exist), or
O_TRUNC is set in the *oflag* argument.

The *openat*() function shall fail if:

EACCES

The access mode of the open file description associated with *fd* is not O_SEARCH and the permissions of the directory underlying *fd* do not permit directory searches.

EBADF

The *path* argument does not specify an absolute path and the *fd* argument is neither AT_FDCWD nor a valid file descriptor open for reading or searching.

ENOTDIR

The *path* argument is not an absolute path and *fd* is a file descriptor associated with a non-directory file.

These functions may fail if:

EAGAIN

Аргумент *path* называет подчиненную сторону псевдотерминального устройства, которое заблокировано.

EINVAL

Значение аргумента *oflag* недопустимо.

ELOOP

Во время разрешения аргумента path было обнаружено более {SYMLOOP_MAX} символических ссылок.

RU - <url>

open - Man Page

превышающей {PATH_MAX} .

ENOMEM

Аргумент *path* называет файл STREAMS, и система не может выделить ресурсы.

EOPNOTSUPP

Аргумент *path* называет сокет.

ETXTBSY

Файл представляет собой чистый файл процедуры (общий текст), который выполняется, а *oflag* – O_WRONLY или O_RDWR .

Следующие разделы являются информативными.

Примеры

Открытие файла для записи владельцем

В следующем примере файл **/tmp/file открывается**либо путем его создания (если он еще не существует), либо путем усечения его длины до 0 (если он существует). В первом случае, если вызов создает новый файл, биты разрешения доступа в файловом режиме файла устанавливаются так, чтобы разрешать чтение и запись владельцем и разрешать чтение только членами группы и другими.

Если вызов *open*() успешен, файл открывается для записи.

```
#include <fcntl.h>
...
int fd;
mode_t mode = S_IRUSR | S_IWUSR | S_IRGRP | S_IROTH;
char *pathname = "/tmp/file";
```

open - Man Page

Открытие файла с помощью проверки существования

В следующем примере используется функция *open*(), чтобы попытаться создать файл **LOCKFILE** и открыть его для записи. Поскольку функция *open*() указывает флаг O_EXCL, вызов завершается неудачно, если файл уже существует. В этом случае программа предполагает, что кто-то еще обновляет файл пароля и завершает работу.

```
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>

#define LOCKFILE "/etc/ptmp"
...
int pfd; /* Целое число для файлового дескриптора, возвр
...
if ((pfd = open(LOCKFILE, O_WRONLY | O_CREAT | O_EXCL,
 S_IRUSR | S_IWUSR | S_IRGRP | S_IROTH)) == -1)
{
 fprintf(stderr, "Не удается открыть /etc/ptmp. Повторит
 exit(1);
}
...
```

Открытие файла для записи

Следующий пример открывает файл для записи, создавая файл, если он еще не существует. Если файл существует, система обрезает его до нуля байтов.

```
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
```

open - Man Page

```
char pathname[PATH_MAX+1];
...
if ((pfd = open(pathname, O_WRONLY | O_CREAT | O_TRUNC,
 S_IRUSR | S_IWUSR | S_IRGRP | S_IROTH)) == -1)
{
 perror("Не удается открыть выходной файл\n"); exit(1);
}
...
```

Использование приложений

POSIX.1-2008 не требует, чтобы параметры терминала автоматически устанавливались в любое состояние при первом открытии или сбрасывались после последнего закрытия. Для несоответствующего приложения возможно оставить терминальное устройство в состоянии, когда следующий процесс, использующий это устройство, находит его в несоответствующем состоянии, но не имеет возможности определить это. Чтобы убедиться, что устройство установлено в соответствующее начальное состояние, приложения, которые выполняют первое открытие терминала (кроме псевдотерминала), должны использовать флаг O_TTY_INIT для установки параметров, связанных с терминалом, в соответствующее состояние.

За исключением случаев, указанных в этом томе POSIX.1-2017, флаги, разрешенные в *oflag* они не являются взаимоисключающими и могут использоваться одновременно. Не все комбинации флагов имеют смысл. Например, использование O_SEARCH | O_CREAT успешно откроет уже существующий каталог для поиска, но если нет существующего файла с таким именем, то не указано, будет ли создан обычный файл. Аналогично, если файловый дескриптор, не относящийся к каталогу, успешно возвращен, не указано, будет ли этот дескриптор иметь разрешения на выполнение, как если бы O_EXEC (обратите внимание, что не указано, имеют ли O_EXEC и O_SEARCH одинаковое значение).

open - Man Page

Some implementations permit opening FIFOs with O_RDWR. Since FIFOs could be implemented in other ways, and since two file descriptors can be used to the same effect, this possibility is left as undefined.

See *getgroups*() about the group of a newly created file.

Использование *open*() для создания обычного файла предпочтительнее использования *creat*(), поскольку последний избыточен и включен только по историческим причинам.

Использование флага O_TRUNC в FIFO и каталогах (pipes cannot be *open*()-ed) должно быть допустимо без неожиданных побочных эффектов (например, *creat*() в FIFO не должен удалять данные). Поскольку специальные файлы терминала могут иметь данные типа вперед, хранящиеся в буфере, O_TRUNC не должен влиять на их содержимое, особенно если программа, которая обычно открывает обычный файл, должна открывать текущий управляющий терминал вместо этого. Другие типы файлов, особенно определяемые реализацией, остаются определяемыми реализацией.

POSIX.1-2008 позволяет **возвращать [EACCES]** для условий, отличных от явно перечисленных.

Флаг O_NOCTTY был добавлен, чтобы позволить приложениям избежать непреднамеренного получения управляющего терминала в качестве побочного эффекта открытия файла терминала. Этот том POSIX.1-2017 не определяет, как приобретается управляющий терминал, но он позволяет реализации обеспечить это на *open*(), если флаг O_NOCTTY не установлен и другие условия, указанные в базовом томе определений POSIX.1-2017, *глава 11*, *Общий интерфейс терминала*, выполнены.

В исторических реализациях значение O_RDONLY равно нулю. Из-за этого невозможно обнаружить наличие O_RDONLY и другой опции. Будущие

open - Man Page

```
    O_RDONLY | O_WRONLY == O_RDWR
```

O_EXEC и O_SEARCH задаются как два из пяти режимов доступа к файлам. Поскольку O_EXEC не применяется к каталогам, а O_SEARCH применяется только к каталогам, их значения не должны быть разными. Поскольку O_RDONLY исторически имел нулевое значение, реализации не могут различать O_SEARCH и O_SEARCH | O_RDONLY , а также O_EXEC .

In general, the *open*() function follows the symbolic link if *path* names a symbolic link. However, the *open*() function, when called with O_CREAT and O_EXCL, is required to fail with **[EEXIST]** if *path* names an existing symbolic link, even if the symbolic link refers to a nonexistent file. This behavior is required so that privileged applications can create a new file in a known location without the possibility that a symbolic link might cause the file to be created in a different location.

For example, a privileged application that must create a file with a predictable name in a user-writable directory, such as the user's home directory, could be compromised if the user creates a symbolic link with that name that refers to a nonexistent file in a system directory. If the user can influence the contents of a file, the user could compromise the system by creating a new system configuration or spool file that would then be interpreted by the system. The test for a symbolic link which refers to a nonexisting file must be atomic with the creation of a new file.

Кроме того, функция *open*() отказывается открывать не каталоги, если установлен флаг O_DIRECTORY. Это позволяет избежать условий гонки, при которых пользователь может скомпрометировать систему, заменив жесткую ссылку на конфиденциальный файл (например, устройство или FIFO) во время работы привилегированного приложения, где открытие файла даже для доступа на чтение может иметь нежелательные побочные эффекты.

open - Man Page

которых пользователь может скомпрометировать систему, заменив символическую ссылку на конфиденциальный файл (например, устройство) во время работы привилегированного приложения, где открытие файла даже для доступа на чтение может иметь нежелательные побочные эффекты.

Стандарт POSIX.1-1990 требовал, чтобы идентификатор группы вновь созданного файла был установлен на идентификатор группы его родительского каталога или на эффективный идентификатор группы процесса создания. FIPS 151-2 требовал, чтобы реализации обеспечивали способ установки идентификатора группы в идентификатор группы содержащего каталога, но не запрещали реализации, также поддерживающие способ установки идентификатора группы в эффективный идентификатор группы процесса создания. Соответствующие приложения не должны предполагать, какой идентификатор группы будет использоваться. Если это имеет значение, приложение может использовать *chown*() установить идентификатор группы после создания файла или определить, при каких условиях реализация установит нужный идентификатор группы.

Цель функции *openat*() - разрешить открытие файлов в каталогах, отличных от текущего рабочего каталога, без воздействия условий гонки. Любая часть пути к файлу может быть изменена параллельно вызову *open*(), что приведет к неопределенному поведению. Открыв файловый дескриптор для целевого каталога и используя функцию *openat*(), можно гарантировать, что открытый файл находится относительно нужного каталога. Некоторые реализации используют функцию *openat*() и для других целей. В некоторых случаях, если *oflag* параметр имеет бит O_XATTR, возвращаемый дескриптор файла обеспечивает доступ к расширенным атрибутам. Эта функциональность здесь не стандартизирована.

Будущие направления

Нет.

open - Man Page

chmod(), close(), creat(), dirfd(), dup(), exec, fcntl(), fdopendir(), link(), lseek(), mkdtemp(), mknod(), read(), symlink(), umask(), unlockpt(), write()

The Base Definitions volume of POSIX.1-2017, *Chapter 11*, *General Terminal Interface*, <fcntl.h>, <sys_stat.h>, <sys_types.h>

Copyright

Любые типографские ошибки или ошибки форматирования, которые появляются на этой странице, скорее всего, были введены во время преобразования исходных файлов в формат man page. Чтобы сообщить о таких ошибках, см. https://www.kernel.org/doc/man-pages/reporting_bugs.html .

Ссылка на

aio_fsync (3p), chmod (3p), закрыть (3p), cp (1p), creat (3p), dbm_clearerr (3p), dirfd (3p), dup (3p), exec (3p), fchmod (3p), fcntl (3p), fcntl.h (0p), fdatasync (3p), fdopen (3p), fdopendir (3p),

open - Man Page

posix_fallocate(3p), posix_openpt (3p), posix_spawn (3p), posix_spawn_file_actions_addclose(3p), ptsname (3p), read (3p), sh (1p), stropts.h (0p), символическая ссылка (3p), tempnam (3p), tmpnam (3p), truncate (3p), umask (3p), unlockpt (3p), запись (3p).

2017 IEEE/The Open Group POSIX Programmer's Manual

Главная Блог О нас