ссылка - Man Page

свяжите один файл с другим файлом

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

```
#include <unistd.h>

int link (const char *path1, const char *path2);

#включить <fcntl.h>

int linkat(int fd1, const char *path1, int fd2,
const char *path2, флаг int);
```

Описание

Функция *link*() создает новую ссылку (запись каталога) для существующего файла *path1*.

Аргумент *path1* указывает на путь к существующему файлу. Аргумент *path2* указывает на путь, именующий новую запись каталога, которая будет создана. Функция *link*() должна атомарно создать новую ссылку для существующего файла, а количество ссылок в файле должно быть увеличено на единицу.

ссылка - Man Page

directories.

Если *path1* называет символическую ссылку, то определяется реализацией, следует ли *link*() за символической ссылкой или создает новую ссылку на саму символическую ссылку.

После успешного завершения *link*() помечает для обновления последнюю метку времени изменения статуса файла. Кроме того, для обновления должны быть помечены метки времени последнего изменения данных и последнего изменения состояния файла каталога, содержащего новую запись.

В случае *сбоя функции link*() ссылка не создается, а количество ссылок в файле остается неизменным.

Реализация может потребовать, чтобы вызывающий процесс имел разрешение на доступ к существующему файлу.

Функция *linkat*() должна быть эквивалентна функции *link*(), за исключением того, что символические ссылки должны обрабатываться так, как указано значением *flag* (см. Ниже), и за исключением случая, когда *path1* или *path2* или оба являются относительными путями. В этом случае относительный путь *path1* интерпретируется относительно каталога, связанного с файловым дескриптором *fd1*, вместо текущего рабочего каталога и аналогично для *path2* и файлового дескриптора *fd2*Если режим доступа описания открытого файла, связанного с файловым 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 будет выполнять проверку.

Значения для *флага* строятся побитовым включением ИЛИ из флагов из следующего списка, определенного в *<fcntl.h>*:

AT_SYMLINK_FOLLOW

ссылка - Man Page

Если *linkat*() передает специальное значение AT_FDCWD в параметре *fd1* или *fd2*, для соответствующего аргумента path будет использоваться текущий рабочий каталог. Если оба *fd1* и *fd2* имеют значение AT_FDCWD , поведение должно быть идентичным вызову *link*() , за исключением того, что символические ссылки должны обрабатываться так, как указано значением *flag*.

If the AT_SYMLINK_FOLLOW flag is clear in the *flag* argument and the *path1* argument names a symbolic link, a new link is created for the symbolic link *path1* and not its target.

Return Value

Upon successful completion, these functions shall return 0. Otherwise, these functions shall return -1 and set *errno* to indicate the error.

Errors

These functions shall fail if:

EACCES

A component of either path prefix denies search permission, or the requested link requires writing in a directory that denies write permission, or the calling process does not have permission to access the existing file and this is required by the implementation.

EEXIST

The *path2* argument resolves to an existing directory entry or refers to a symbolic link.

ссылка - Man Page

the *path1* or *path2* argument.

EMLINK

Количество ссылок на файл с именем *path1* будет превышать {LINK_MAX}.

RU - <url>

Длина компонента пути больше {NAME_MAX}.

ENOENT

Компонент с префиксом path не существует; файл с именем *path1* не существует; или *path1* или *path2* указывает на пустую строку.

ЭНОЕНТ или **ЭНОТДИР**

Аргумент *path1* называет существующий файл, не относящийся к каталогу, а аргумент *path2* содержит по крайней мере один символ, не являющийся <косой чертой>, и заканчивается одним или несколькими завершающими символами <косой черты>. Если *path2* без завершающих символов <slash> назовет существующий файл, **ошибка [ENOENT]** не возникнет.

ENOSPC

Каталог, содержащий ссылку, не может быть расширен.

ENOTDIR

Компонент либо с префиксом path называет существующий файл, который не является ни каталогом, ни символической ссылкой на каталог, либо аргумент *path1* содержит по крайней мере один символ, не являющийся <slash>, и заканчивается одним или несколькими конечными символами <slash>, а последний компонент pathname называет существующий файл, который не является ни каталогом, ни символической ссылкой на каталог.каталог или символическая ссылка на каталог, или аргумент *path1* называет существующий файл, не являющийся каталогом, а аргумент *path2*

ссылка - Man Page

несколькими конечными символами <slash> .

ЭПЕРМ

Файл с именем *path1* является каталогом, и либо вызывающий процесс не имеет соответствующих привилегий, либо реализация запрещает использование *link*() в каталогах.

EROFS

Запрашиваемая ссылка требует записи в каталог в файловой системе, доступной только для чтения.

EXDEV

Ссылка с именем *path2* и файл с именем *path1* находятся в разных файловых системах, и реализация не поддерживает связи между файловыми системами.

EXDEV

path1 относится к именованному ПОТОКУ.

Функция *linkat*() завершится ошибкой, если:

EACCES

Режим доступа к описанию открытого файла, связанного с *fd1* или *fd2*, не является O_SEARCH, и разрешения каталога, лежащего *в основе fd1* или *fd2*, соответственно, не разрешают поиск в каталоге.

EBADF

Аргумент *path1* или *path2* не указывает абсолютный путь, а аргумент *fd1* или *fd2*, соответственно, не является ни AT_FDCWD, ни допустимым файловым дескриптором, открытым для чтения или поиска.

ENOTDIR

ссылка - Man Page

файлом, не относящимся к каталогу.

Эти функции могут выйти из строя, если:

ELOOP

Во время разрешения аргумента path1 или path2 было обнаружено более {SYMLOOP_MAX} символических ссылок.

RU - <url>

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

Функция *linkat*() может завершиться ошибкой, если:

EINVAL

Значение аргумента *flag* недопустимо.

Следующие разделы являются информативными.

Примеры

Создание ссылки на файл

В следующем примере показано, как создать ссылку на файл с именем **/home/cnd/mod1**, создав новую запись каталога с именем **/modules/pass1**.

```
#include <unistd.h>

char *path1 = "/home/cnd/mod1";
char *path2 = "/modules/pass1";
```

ссылка - Man Page

Создание ссылки на файл в программе

В следующем примере программы функция link() **связывает файл /etc/**passwd (определенный как **PASSWDFILE**) с файлом **/etc/opasswd** (определенным как **SAVEFILE**), который используется для сохранения текущего файла пароля. Затем, после удаления текущего файла пароля (определенного как **PASSWDFILE**), новый файл пароля снова сохраняется как текущий файл пароля с помощью функции *link*() .

```
#включить <unistd.h>

#определить ФАЙЛ БЛОКИРОВКИ "/etc/ptmp"
#определить PASSWDFILE "/etc/passwd"
#определить ФАЙЛ СОХРАНЕНИЯ "/etc/opasswd"
...
/* Сохранить текущий файл пароля */
ссылка (PASSWDFILE, SAVEFILE);

/* Удалить текущий файл пароля. */
unlink (PASSWDFILE);

/ * Сохраните новый файл пароля как текущий файл пароля.
link (LOCKFILE,PASSWDFILE);
```

Application Usage

Некоторые реализации допускают ссылки между файловыми системами.

Если *path1* ссылается на символическую ссылку, разработчики приложений должны использовать *linkat*() с соответствующими флагами, чтобы выбрать, следует ли разрешать символическую ссылку.

ссылка - Man Page

Ссылка на каталог ограничена суперпользователем в большинстве исторических реализаций, поскольку эта возможность может создавать циклы в файловой иерархии или иным образом повреждать файловую систему. Этот том POSIX.1-2017 продолжает эту философию, запрещая *link*() и *unlink*() делать это. Другие функции могут сделать это, если разработчик разработал такое расширение.

Некоторые исторические реализации позволяют связывать файлы в разных файловых системах. Формулировка была добавлена, чтобы явно разрешить это необязательное поведение.

Исключение для ссылок между файловыми системами предназначено только для ссылок, которые программно неотличимы от “жестких” ссылок.

Цель функции *linkat*() - связать файлы в каталогах, отличных от текущего рабочего каталога, без воздействия условий гонки. Любая часть пути к файлу может быть изменена параллельно вызову *link*(), что приведет к неопределенному поведению. Открыв файловый дескриптор для каталога как существующего файла, так и целевого местоположения и используя функцию *linkat*(), можно гарантировать, что оба имени файлов находятся в нужных каталогах.

Флаг AT_SYMLINK_FOLLOW позволяет реализовать оба распространенных поведения функции *link*(). Спецификация POSIX требует, чтобы, если *path1* является символической ссылкой, создавалась новая ссылка для цели символической ссылки. Многие системы по умолчанию или в качестве альтернативы предоставляют механизм, позволяющий избежать неявного поиска символической ссылки и создать новую ссылку для самой символической ссылки.

Более ранние версии этого стандарта указывали только функцию *link*() и требовали, чтобы она вела себя как *linkat*() с флагом AT_SYMLINK_FOLLOW. Однако в исторической практике ядер SVR4 и Linux *функция link*() вела

ссылка - Man Page

использовалась, то не соответствовала стандарту (например, не будучи атомарным, или неправильно разыменовывая символическую ссылку). Поскольку приложения не могут полагаться на *link*(), следуя ссылкам на практике, *linkat*была добавлена функция (), принимающая флаг для указания желаемого поведения приложения.

Будущие направления

Нет.

См. Также

переименовать(), символическая ссылка(), отменить ссылку()

Объем базовых определений POSIX.1-2017, <fcntl.h>, <unistd.h>

Авторские права

Части этого текста перепечатаны и воспроизведены в электронном виде из IEEE Std 1003.1-2017, Standard for Information Technology -- Portable Operating System Interface (POSIX), The Open Group Base Specifications Issue 7, 2018 Edition, Copyright (C) 2018 by the Institute of Electrical and Electronics Engineers, Inc and The Open GroupГруппа. В случае любого несоответствия между этой версией и исходным стандартом IEEE и Open Group исходный стандарт IEEE и Open Group является документом рефери. Оригинальный стандарт можно получить онлайн по адресу http://www.opengroup.org/unix/online.html .

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 - Man Page

fstatvfs (3p), ссылка (1p), ln (1p), открыть (3p), переименовать (3p), символическая ссылка (3p), unistd.h(0p), unlink (3p).

2017 IEEE/The Open Group POSIX Programmer's Manual