unlink (3p) — Linux manual page

PROLOG top

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

НАЗВАНИЕ вверху

unlink, unlink — удалить запись каталога

СИНОПСИС сверху

```
#включить <unistd.h>

int unlink(const char *path);

#включить <fcntl.h>
```

```
int unlink(int fd, const char *path, int flag);
```

DESCRIPTION top

Функция *unlink*() удаляет ссылку на файл. Если *путь*
для символической ссылки *функция unlink*() удаляет символическую ссылку
с именем *path* и не влияет на файл или каталог с именем
содержимого символической ссылки. В противном случае *unlink*()
удалит ссылку с именем пути, на которое указывает *path* и
должен уменьшить количество ссылок в файле, на который ссылается
ссылка.

Когда количество ссылок на файл становится равным 0 и ни один процесс
не открывает файл, пространство, занимаемое файлом, освобождается, и файл
больше не доступен. Если один или несколько процессов
открывают файл при удалении последней ссылки, ссылка должна быть
удалена до *возврата unlink*(), но удаление
содержимого файла должно быть отложено до тех пор, пока все ссылки на файл не будут
закрыты.

The аргумент не должен называть каталог, если процесс
не имеет соответствующих привилегий и реализация не поддерживает использование
unlink() для каталогов.

После успешного завершения *unlink*() отметит для обновления
последнюю модификацию данных и последние временные метки изменения состояния файла
родительского каталога. Кроме того, если количество ссылок в файле не равно 0,
последняя метка времени изменения состояния файла должна быть отмечена
для обновления.

Функция *unlink*() должна быть эквивалентна функции *unlink*() или
rmdir(), за исключением случая, когда *path* указывает
относительный путь. В этом случае удаляемая запись каталога
определяется относительно каталога, связанного с файловым

дескриптором *fd*, а не текущего рабочего каталога. 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 должна выполнять проверку.

Значения для *флага* строятся побитово-включительно ИЛИ из флагов из следующего списка, определенного в *<fcntl.h>*:

AT_REMOVEDIR
Удалите запись каталога, указанную *fd* и *path* как каталог, а не обычный файл.

Если *unlink*() передается специальное значение AT_FDCWD в *fd* параметр, должен использоваться текущий рабочий каталог, и поведение должно быть идентичным вызову *unlink*() или *rmdir*(). соответственно, в зависимости от того, установлен ли бит AT_REMOVEDIR во *флаге*.

ВОЗВРАЩАЕМОЕ ЗНАЧЕНИЕ top

После успешного завершения эти функции должны вернуть 0.
В противном случае эти функции должны возвращать -1 и устанавливать *errno* для указания ошибки. Если возвращается значение -1, именованный файл не изменяется.

ОШИБКИ топ

Эти функции должны завершиться неудачно и не должны отсоединять файл, если:

Отказано в разрешении поиска для компонента префикса пути или отказано в разрешении записи в каталоге , содержащем удаляемую запись каталога.

EBUSY Файл с именем аргумента *path* не может быть отключен
, потому что он используется системой или другим процессом
, и реализация считает это ошибкой.

ELOOP Цикл существует в символических ссылках, встречающихся при
разрешении аргумента *path*.

ENAMETOOLONG
Длина компонента пути больше
{NAME_MAX}.

ENOENT Компонент *path* не называет существующий файл или *путь*
это пустая строка.

ENOTDIR
Компонент префикса path называет существующий файл, который
не является ни каталогом, ни символической ссылкой на каталог,
ни *путем* аргумент содержит хотя бы один не-<косая черта>
символ и заканчивается одним или несколькими завершающими <slash>
символы и последний путь компонент именует
существующий файл, который не является ни каталогом, ни символической
ссылкой на каталог.

EPERM Файл с именем по *пути* является каталогом, и либо
вызывающий процесс не имеет соответствующих привилегий, либо
реализация запрещает использование *unlink*() в
каталогах.

EPERM или **EACCES**
Флаг S_ISVTX устанавливается в каталоге, содержащем
файл, на который ссылается аргумент *path*, и процесс
не удовлетворяет критериям, указанным в
томе базовых определений POSIX.1-2017, *Раздел 4.3*, *Защита каталогов*.

EROFS Разблокируемая запись каталога является частью файловой системы, доступной только для чтения
.

Функция *unlink*() завершится ошибкой, если:

EACCES Режим доступа к описанию открытого файла, связанного
с *fd* не является O_SEARCH, и разрешения
каталога, лежащего *в основе fd*, не разрешают поиск в каталоге.

EBADF Аргумент *path* не указывает абсолютный путь, а
аргумент *fd* не является ни AT_FDCWD, ни допустимым файловым
дескриптором, открытым для чтения или поиска.

ENOTDIR
Аргумент *path* не является абсолютным путем, а *fd* является файловым
дескриптором, связанным с файлом, не относящимся к каталогу.

EEXIST или **ENOTEMPTY**
Параметр *flag* имеет бит AT_REMOVEDIR и
путь аргумент называет каталог, который не является пустым
каталогом, или есть жесткие ссылки на каталог, отличный
от точки или одной записи в dot-dot.

ENOTDIR
Параметр *flag* имеет бит AT_REMOVEDIR и *путь*
не называет каталог.

Эти функции могут выйти из строя и не отсоединить файл, если:

EBUSY Файл с именем по *пути* является именованным ПОТОКОМ.

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

ENAMETOOLONG

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

ETXTBSY
Запись, которая должна быть отключена, является последней записью каталога
в выполняемом файле чистой процедуры (общий текст).

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

EINVAL Значение аргумента *flag* недопустимо.

Следующие разделы являются информативными.

ПРИМЕРЫ сверху

Удаление ссылки на файл

В следующем примере показано, как удалить ссылку на файл с именем
/home/cnd/mod1 удалив запись с именем **/modules/pass1**.

```
#включить <unistd.h>

char *path = "/modules/pass1";
int status;
...
status = unlink(путь);
```

Проверка ошибки

Следующий фрагмент примера создает временный
файл блокировки паролем с именем **LOCKFILE**, который определяется как **/etc/ptmp**, и получает
для него файловый дескриптор. Если файл не может быть открыт для записи,
снимите ссылку() используется для удаления связи между файловым дескриптором
и **ФАЙЛОМ БЛОКИРОВКИ**.

```
#включить <sys/types.h>
```

```
  #включить <stdio.h>
  #включить <fcntl.h>
  #включить <errno.h>
  #включить <unistd.h>
  #включить <sys/stat.h>

  #define LOCKFILE "/etc/ptmp"

  int pfd; /* Целое число для дескриптора файла, возвращаемого вызовом open. */
  FILE *fpfd; /* Указатель файла для использования в putpwent(). */
  ...
  /* Открыть файл блокировки паролем. Если он существует, это ошибка. */
 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. Повторите попытку позже.\n");
  выход(1);
  }

  /* Файл блокировки создан; перейдите к fdopen файла блокировки, чтобы
 можно было использовать putpwent().
  */
 if ((fpfd = fdopen(pfd, "w")) == NULL) {
  закрыть (pfd);
  unlink(LOCKFILE);
  выход(1);
  }
```

Замена файлов

Следующий пример фрагмента использует *unlink*() для удаления ссылок на файлы, чтобы их можно было заменить новыми версиями файлов. Первый вызов удаляет ссылку на **БЛОКИРОВКА** ФАЙЛА при возникновении ошибки. Последовательные вызовы удаляют ссылки на **SAVEFILE** и **PASSWDFILE**, чтобы можно было создавать новые ссылки, а затем удаляют ссылку на **LOCKFILE** когда он больше не нужен.

```
  #включить <sys/types.h>
  #включить <stdio.h>
  #включить <fcntl.h>
  #включить <errno.h>
  #включить <unistd.h>
  #включить <sys/stat.h>

  #define LOCKFILE "/etc/ptmp"
  #define PASSWDFILE "/etc/passwd"
  #define SAVEFILE "/etc/opasswd"
  ...
  /* Если никаких изменений не произошло, предположите ошибку и оставьте passwd без изменений. */
 if (!valid_change) {
 fprintf(stderr, "Не удалось изменить пароль для пользователя %s \n", user);
  unlink(LOCKFILE);
  выход(1);
  }

  /* Изменение прав доступа к новому файлу паролей. */
 chmod(LOCKFILE, S_IRUSR | S_IRGRP | S_IROTH);

  /* Удалить сохраненный файл пароля. */
 unlink(SAVEFILE);

  /* Сохранить текущий файл пароля. */
 ссылка (PASSWDFILE, SAVEFILE);

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

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

  /* Удалить файл блокировки. */
 unlink(LOCKFILE);
```

```
выход(0);
```

ИСПОЛЬЗОВАНИЕ ПРИЛОЖЕНИЙ сверху

Приложения должны использовать *rmdir*() для удаления каталога.

ОБОСНОВАНИЕ сверху

Во многих
исторических реализациях отключение ссылки на каталог ограничено суперпользователем по причинам, указанным в *link*() (см. Также *rename*()).

Значение **[EBUSY]** в исторических реализациях
`точка монтирования занята". Поскольку этот том POSIX.1-2017 не охватывает
концепции системного администрирования монтирования и размонтирования,
описание ошибки было изменено на "ресурс занят".
(Это значение используется некоторыми драйверами устройств, когда второй
процесс пытается открыть устройство эксклюзивного использования.) Формулировка
также предназначена для того, чтобы позволить реализациям отказаться от удаления
каталога, если он является корневым или текущим рабочим каталогом любого
процесса.

Разработчики стандарта рассмотрели TR 24715-2006 и отметили, что
Реализации, соответствующие LSB, могут возвращать **[EISDIR]** вместо
[EPERM] при отключении ссылки на каталог. Изменение, позволяющее разрешить такое
поведение, путем изменения требования к **[EPERM]** на **[EPERM]** или
[EISDIR] был рассмотрен, но решил не делать этого, поскольку он нарушит
существующие строго соответствующие и соответствующие приложения.
Приложения, написанные для переносимости как на POSIX.1-2008, так и на
LSB, должны быть готовы обрабатывать любой код ошибки.

Цель *unlink*функция () заключается в удалении
записей каталога в каталогах, отличных от текущего рабочего каталога

, без воздействия условий гонки. Любая часть пути
к файлу может быть изменена параллельно вызову *unlink*(),
что приведет к неопределенному поведению. Открыв файловый дескриптор
для целевого каталога и используя функцию *unlink*(), можно
гарантировать, что удаленная запись каталога расположена
относительно нужного каталога.

FUTURE DIRECTIONS top

Нет.

СМ. ТАКЖЕ top

close(3p), link(3p), remove(3p), rename(3p), rmdir(3p),
symlink(3p)

Базовый том определений POSIX.1-2017, *Раздел 4.3*,
Защита каталогов, fcntl.h(0p), unistd.h (0p)

COPYRIGHT top

, скорее всего, были допущены во время преобразования
исходных файлов в формат man-страницы. Чтобы сообщить о таких ошибках, см.
https://www.kernel.org/doc/man-pages/reporting_bugs.html .

IEEE/The Open Group 2017 UNLINK(3P)

Pages that refer to this page: unistd.h(0p), cp(1p), ln(1p), rm(1p), rmdir(1p), unlink(1p), close(3p), fstatvfs(3p), link(3p), posix_fallocate(3p), remove(3p), rename(3p), rmdir(3p), symlink(3p), tempnam(3p), tmpfile(3p), tmpnam(3p)