# Мини-HOWTO: Восстановление удаленных файлов с файловой системы Ext2fs в Linux

**Aaron Crane**

aaronc@pobox.com

**Перевод: Денис Дементьев, SWSoft Pte Ltd.**

Версия 1.3, 2 Февраля 1999 года

Представьте себе следующую картину. Последние три дня вы не ели, не спали, даже не принимали душ. И, наконец, ваши усилия вознаграждены: вы закончили программу, которая принесет вам всемирную известность и славу. Все, что вам осталось сделать - это запаковать ее архиватором tar и поместить на Metalab. Да... и удалить все копии старых файлов, созданные Emacs. Итак, вы набиваете на клавиатуре `rm * ~`. Лишний пробел в вашей команде вы замечаете слишком поздно. Вы только что удалили вашу супер программу! Но помощь у вас под рукой. Этот документ рассказывает, как можно восстановить удаленные файлы с файловой системы ext2 (Second Extended File System). Может быть, несмотря ни на что, вам удастся выпустить вашу программу...

---

# 1. Введение

Этот мини-Howto предоставляет вам возможность восстановить удаленные с файловой системы ext2 файлы. В нем также содержатся некоторые советы о том, как можно избежать удаления файлов.

В первую очередь, этот документ предназначался для людей, случайно, если так можно выразиться, пострадавших от `rm`. Впрочем, надеемся, что остальные пользователи также прочитают его. Никогда нельзя быть уверенным во всем - может быть однажды информация из этого документа вам очень пригодится.

Мы предполагаем, что вы знакомы с файловыми системами UNIX. Впрочем, мы надеемся, что он вполне доступен большинству пользователей Linux. Если вы начинающий пользователь, учтите, что восстановление удаленных файлов под Linux все же требует некоторых технических знаний и упорства.

Вы не сможете восстановить удаленные файлы с файловой системы ext2, как минимум, без прав доступа для чтения к непосредственно к устройству (device), на котором хранился файл. Подразумевается, что вы должны иметь права root, хотя некоторые дистрибутивы (например Debian GNU/Linux) имеют группу `disk`, члены которой имеют доступ к этим устройствам. Также вам потребуется утилита `debugfs` из пакета `e2fsprogs`, который должен быть установлен.

Почему мы написали этот документ? Потому, что имеем большой опыт случайного и непоправимого использования команды `rm -r` с правами root. Ваш покорный слуга однажды удалил около 97 необходимых JPEG файлов, которые невозможно было найти в других источников. Но, при помощи некоторых полезных подсказок (см. раздел Разд. 15) и большого терпения, я восстановил 91 файл в первозданном виде. Пять из

оставшихся мне удалось восстановить частично (по крайней мере было понятно, что это за картинки). Лишь один файл невозможно было просмотреть, но, я уверен, что потерял не более 1024 байт информации (это очевидно из заголовка файла).

Ниже будет объяснено, какой процент восстановления удаленных файлов, вы можете ожидать, в зависимости от обстоятельств.

---

## 1.1. Версии документа

Предыдущие опубликованные версии документа:

- версия 1.0, 18 января 1997
- версия 1.1, 23 июля 1997 (см. раздел Разд. 1.1.1)
- версия 1.2, 4 августа 1997 (см. раздел Разд. 1.1.2)
- версия 1.3, 2 февраля 1999 (см. раздел Разд. 1.1.3)

---

## 1.2. Где можно найти этот документ

Последнюю версию этого документа всегда можно найти на сайте Linux Documentation Project (и зеркалах).

Последняя версия также лежит на моем сайте в нескольких форматах:

- SGML. Это исходный текст SGML, написанный с использованием пакета SGML Tools.
- HTML. Это HTML, автоматически созданный из SGML.
- Текст. Это простой текст, созданный из SGML.

---

# 2. Несколько советов

Жизненно необходимо помнить, что при восстановлении удаленных файлов Linux не похож на MS-DOS. В MS-DOS (и его внебрачном потомке Window 95) в большинстве случаев довольно легко восстановить файл, так как в комплект поставки "операционной системы" (я употребляю этот термин в широком смысле) входит утилита, автоматизирующая значительную часть процесса. Linux - совсем другой случай.

Итак, правило N 1 (главная заповедь, если хотите):

"*ХРАНИТЕ КОПИИ*"

неважно чего. Позаботьтесь об имеющейся у вас информации. Возможно, вы храните переписку, адреса, программы, документы за несколько лет на вашем компьютере. Подумайте о том, как перевернется ваша жизнь, если произойдет непоправимый сбой диска или, не дай бог, к вам в систему проникнет злоумышленник и повредит диски. И это не так уж невероятно - я общался с людьми, оказавшимися в такой ситуации. Я призываю здравомыслящих пользователей Linux пойти и купить устройство резервного копирования, составить приличное расписание копирования данных на него и *строго придерживаться этого*. Я, например, использую запасной диск на второй машине и периодически сбрасываю на него по сети мой домашний каталог. Для дополнительной информации, по составлению расписания резервного копирования, читайте Frisch (1995) (см. раздел Разд. 15).

Что делать, если у вас нет возможности копировать данные?

Попробуйте запретить запись в важные файлы: запрет доступа на запись заставит `rm` запрашивать подтверждение перед удалением. (Впрочем, я заметил, что если я удаляю каталог со всеми его подкаталогами командой `rm -r`, то на первом или втором запросе подтверждения я прерываю команду и запускаю ее снова уже как `rm -rf`.)

Хорошим способом для некоторых файлов может послужить создание ссылки (hard link) на них в скрытом каталоге. Я слышал однажды историю о системном администраторе, который имел привычку случайно затирать `/etc/passwd` (что соответственно приводило систему в нерабочее состояние). Возможный способ исправить это - сделать (пользователем root) что-то типа:

```
# mkdir /.backup
# ln /etc/passwd /.backup
```

Теперь, чтобы удалить полностью содержимое файла, требуются некоторые дополнительные усилия: если вы наберете

```
# rm /etc/passwd
```

тогда

```
# ln /.backup/passwd /etc
```

восстановит его. Конечно, это не спасет в случае перезаписи файла, так что все равно делайте копии.

На файловой системе ext2 можно для защиты использовать атрибуты ext2. Эти атрибуты устанавливаются командой `chattr`. Есть атрибут `append-only' (только добавление): в файл - с этим атрибутом можно добавлять данные, но он не может быть удален, и его содержимое не может быть перезаписано. Если этот атрибут установить на каталог, то файлы можно изменять, как обычно, но они не могут быть удалены. Атрибут `append-only' устанавливается командой

```
$ chattr +a FILE...
```

Есть также атрибут `immutable' - устанавливать или снимать его может только root. Файл или каталог с этим атрибутом не может быть изменен, удален, переименован, на него нельзя создать ссылку (hard link). Он устанавливается следующим образом:

```
# chattr +i FILE...
```

Кроме того, ext2fs поддерживает атрибут `undeletable' (неудаляемый) (`+u` в `chattr`). При удалении файла с этим атрибутом, он реально не удаляется, а перемещается в "безопасное место", откуда его можно позже удалить. К сожалению, эта возможность пока еще не реализована; хотя интерес к ее реализации проявлялся, пока (насколько я знаю) она отсутствует во всех доступных ядрах.

Есть сторонники того, чтобы сделать `rm` псевдонимом (alias) или функцией для `rm -i` (с этим ключом rm запрашивает подтверждение для *каждого* удаляемого файла). В дистрибутиве Red Hat это сделано для всех пользователей, включая root. Лично я не делаю этого, так как не выношу программы, не работающие без дополнительной поддержки. Кроме того, тут кроется еще одна проблема - рано или поздно вам придется работать в однопользовательском режиме, использовать другой shell или даже другой компьютер, где отсутствует ваша функция `rm -i`. Если вы привыкли к подтверждениям, то легко можете забыть, где вы находитесь, и указать слишком много файлов для удаления. Таким образом, различные скрипты и программы, заменяющие `rm`, по моему мнению, очень опасны.

Несколько лучшим решением может быть использование пакета, удаляющего файлы с возможностью восстановления другой командой (не `rm`). Для более подробной информации смотрите Peek, et al (1993) (см. раздел Разд. 15). Минусом этого решения является то, что пользователи могут привыкнуть беззаботно удалять файлы, вместо внимательного подхода к удалению, часто требуемого на большинстве систем Unix.

---

# 3. Сколько процентов данных я смогу восстановить?

Это зависит от многих обстоятельств. Одной из трудностей с восстановлением файлов, на такой высококачественной многозадачной многопользовательской операционной системе, как Linux, является то, что вы никогда не знаете, что кто-то хочет записать что-нибудь на диск. Когда операционной системе дается команда удалить файл, она предполагает, что блоки этого файла могут быть использованы, если нужно место для нового файла. (Это конкретный пример общего принципа для Unix-подобных систем: ядро и связанные с ним инструменты предполагают, что пользователи не идиоты.) В общем случае, чем более загружена ваша машина, тем меньше у вас шансов восстановить файлы.

На сложность восстановления файлов также может влиять фрагментация диска. Если раздел, содержащий удаленные файлы, сильно фрагментирован, то вам сложно будет восстановить файл целиком.

Если ваша машина, подобно моей, фактически является однопользовательской рабочей станцией, и в момент удаления файлов вы не использовали интенсивно диск, то можете ожидать примерно указанный мною ранее процент. Я восстановил около 94% данных (не текстовых, заметьте). Если вы получите 80 или более процентов, то можете быть вполне довольны собой.

---

# 4. Итак, как мне восстановить файл?

Для восстановления данных нужно найти их на устройстве с разделом (raw partition device) и сделать так, чтобы они снова были видны операционной системе. Для этого есть два способа: первый - изменить существующую файловую систему так, что в удаленных inode был снят флаг удаления, после чего надеяться, что данные волшебным образом окажутся на месте. Другой способ, намного более безопасный, но медленный - выяснить, где именно в разделе лежат данные, после чего записать их в новый файл на другой файловой системе.

До начала восстановления файлов нужно кое-что сделать; смотрите разделы Разд. 5, Разд. 6 и Разд. 7. Чтобы узнать, как восстановить файлы, смотрите разделы Разд. 8, Разд. 9, Разд. 10 и Разд. 11.

---

# 5. Отключение файловой системы

Независимо от того, какой способ вы выберете, первое, что нужно сделать - отключить файловую систему, содержащую удаленные файлы. Я настоятельно рекомендую вам не пытаться работать с подключенной файловой системой. Отключение файловой системы должно быть произведено, *как можно скорее*, после того, как вы удалили файлы; чем скорее вы это сделаете, тем меньше шансов, что данные будут перезаписаны.

Простейший способ сделать это (предполагаем, что удаленные файлы находились в файловой системе `/usr`) - дать команду

```
# umount /usr
```

Если вы хотите оставить доступ к файловой системе `/usr`, перемонтируйте ее только для чтения:

```
# mount -o ro,remount /usr
```

Если удаленные файлы находились в корневой файловой системе, то вам понадобится ключ `-n` для предотвращения попытки mount записать информацию в файл `/etc/mtab`:

```
# mount -n -o ro,remount /
```

Возможно, какой-то процесс использует эту файловую систему (что вызовет, при попытке отключения, ошибку `Resource busy' (ресурс занят)). Есть программа, посылающая сигналы любому процессу, использующему указанный файл или точку монтирования - это: `fuser`. Для раздела `/usr` попробуйте следующую команду:

```
# fuser -v -m /usr
```

Она выведет список процессов. Предполагая, что ни один из них не является жизненно необходимым, дайте команду

```
# fuser -k -v -m /usr
```

которая пошлет сигнал `SIGKILL` (что гарантировано завершит процесс), или команду

```
# fuser -k -TERM -v -m /usr
```

которая пошлет сигнал SIGTERM (что обычно приводит к нормальному (чистому) завершению работы процесса).

---

# 6. Подготовка к непосредственному изменению inode

Хотите мой совет? Не делайте так. Не думаю, что мудро играть с файловой системой на достаточно низком уровне. Недостатком этого способа является и то, что вы гарантировано восстановите лишь первые 12 блоков каждого файла. Поэтому, если ваш файл был длиннее, воспользуйтесь другим способом. (Впрочем, посмотрите раздел Разд. 12 для дополнительной информации.)

Если вы считаете, что для вас этот способ подходит, то я советую скопировать данные этого раздела в файл на другом разделе, используя обратную петлю:

```
# cp /dev/hda5 /root/working
# mount -t ext2 -o loop /root/working /mnt
```

(Примечание. Устаревшие версии `mount` могут не работать с файлами. Если ваш `mount` не работает, я настоятельно рекомендую поставить последнюю версию, по крайней мере 2.7, так как очень старые версии имеют серьезные ошибки, связанные с правами доступа).

Использование обратной петли означает, что, даже если вы полностью уничтожите файловую систему, то вам достаточно заново скопировать раздел в файл и начать сначала.

---

# 7. Подготовка к записи данных в другое место

Если вы решили воспользоваться этим способом, убедитесь, что у вас имеется раздел, куда вы можете записать новые копии восстанавливаемых файлов. Надеюсь, ваша система имеет несколько разделов: возможно это корневой, `/usr` и `/home`. Если у вас есть такой выбор, проблем не должно возникнуть: просто создайте на одном из них новый каталог.

Если у вас только один (корневой) раздел, и вы все держите на нем, то дела обстоят хуже. Может быть у вас найдется раздел MS-DOS или Windows, который можно использовать? Или в вашем ядре (или в модуле) есть поддержка RAM-диска (диска в памяти)? Чтобы использовать RAM-диск (ядро должно быть версии старше 1.3.48), дайте команды:

```
# dd if=/dev/zero of=/dev/ram0 bs=1k count=2048
# mke2fs -v -m 0 /dev/ram0 2048
# mount -t ext2 /dev/ram0 /mnt
```

Они создадут 2МБ RAM-диск и подключат его к `/mnt`.

Небольшое предупреждение: если вы используете `kerneld` (или его замену `kmod` в ядрах версии 2.2.x и в поздних 2.1.x) для автоматической загрузки и выгрузки модулей ядра, то не отключайте RAM-диск, пока не скопируете все файлы на постоянный носитель. Иначе, при отключении диска, `kerneld` предполагает, что он может выгрузить модуль, а если это происходит, то освободившаяся память может быть использована другими частями ядра, и вы потеряете все результаты нескольких часов нудной работы по восстановлению своих данных.

Если у вас есть дисковод Zip, Jazz, LS-120 или что-нибудь подобное, то возможно он будет хорошим выбором для раздела с восстановленными данными. В противном случае, вам не останется ничего, кроме флоппи-дисков.

Еще одна вещь, которая вам понадобиться - программа, которая может читать данные из середины раздела устройства. В крайнем случае, для этого подойдет `dd`, но если вам нужно будет прочитать, скажем, область с 600МБ по 800МБ, то первые 600МБ все равно будут прочитаны `dd`, хотя и проигнорированы. Это займет довольно много времени, даже на быстрых дисках. В моем случае, для работы с разделом была написана специальная программа. Она называется `fsgrab`; исходный текст вы можете найти на моем сайте или на Metalab (и зеркалах). Далее в этом документе предполагается наличие `fsgrab`.

Если размер файлов, подлежащих восстановлению, не превышал 12 блоков (один блок обычно равен одному килобайту), то `fsgrab` вам не понадобится.

Если у вас нет `fsgrab`, и вы не хотите скачивать и собирать его, то можете просто переводить команды для `fsgrab` в команды для `dd`. Если команда для `fsgrab` выглядит так:

```
fsgrab -c count -s skip device
```

то соответствующая команда для `dd` будет выглядеть так:

```
dd bs=1k if=device count=count skip=skip
```

Я должен предупредить, что хотя `fsgrab` нормально работал у меня, я не даю никаких гарантий его нормального функционирования. Он был написан очень быстро и не очень аккуратно - лишь бы работало. Для дополнительной информации см раздел `No Warranty' (без гарантии) в файле `COPYING` (GNU General Public Licence).

---

# 8. Поиск удаленных inodes

Следующий шаг - выяснить, какие именно inode были удалены. Это можно сделать с помощью `debugfs`. Запустите `debugfs`, указав имя устройства с удаленными файлами:

```
# debugfs /dev/hda5
```

Если хотите непосредственно изменять inode, то укажите ключ `-w` для разрешения записи в файловую систему:

```
# debugfs -w /dev/hda5
```

Команда `lsdel` программы `debugfs` предназначена для поиска удаленных inode. При появлении приглашения, введите ее:

```
debugfs:  lsdel
```

После долгого скрипения диском, вашей любимой программе просмотра текста (переменная `$PAGER`) будет передан длинный список, который нужно сохранить. Если вы используете `less`, наберите `-o` с именем файла. В противном случае, вам придется перенаправлять вывод. Можно сделать так:

```
debugfs:  quit
# echo lsdel | debugfs /dev/hda5 > lsdel.out
```

Теперь вам предстоит, основываясь на времени удаления, размере, типе, числовых значениях прав доступа и владельца, определить, какие из удаленных inode вам нужны. Если вам повезет, то вы сможете быстро найти их по времени удаления. Иначе придется очень тщательно копаться в этом списке.

Советую, если есть такая возможность, распечатать список indode, которые вы хотите восстановить. Это сильно упрощает жизнь.

---

# 9. Получение подробной информации об удаленных inode

`debugfs` имеет команду `stat`, выводящую подробную информацию об inode. Выполните ее для всех inode, подлежащих восстановлению. Например, если вам нужно восстановить inode 148003, наберите:

```
debugfs:  stat <148003>
```

```
Inode: 148003   Type: regular    Mode:  0644   Flags: 0x0   Version: 1
User:   503   Group:   100   Size: 6065
File ACL: 0    Directory ACL: 0
Links: 0   Blockcount: 12
Fragment:  Address: 0    Number: 0    Size: 0
ctime: 0x31a9a574 -- Mon May 27 13:52:04 1996
atime: 0x31a21dd1 -- Tue May 21 20:47:29 1996
mtime: 0x313bf4d7 -- Tue Mar  5 08:01:27 1996
dtime: 0x31a9a574 -- Mon May 27 13:52:04 1996
BLOCKS:
594810 594811 594814 594815 594816 594817
TOTAL: 6
```

Если восстанавливать нужно много, можно автоматизировать эту работу. Предполагая, что список удаленных inode, сформированный командой `lsdel`, находится в файле `lsdel.out`, можно сделать так:

```
# cut -c1-6 lsdel.out | grep "[0-9]" | tr -d " " > inodes
```

Новый файл `inodes` содержит номера inode, подлежащих восстановлению, по одной в строке. Он нам пригодится для следующей команды:

```
# sed 's/^.*$/stat <\0>/' inodes | debugfs /dev/hda5 > stats
```

и файл `stats` содержит данные всех команд `stat`.

---

# 10. Восстановление блоков данных

Эта часть может быть очень легкой и довольно сложной, в зависимости от того, занимал ли удаленный файл больше 12 блоков или нет.

---

## 10.1. Короткие файлы

Если файл занимал не больше 12 блоков, то все их номера хранятся непосредственно в inode. Вы можете просмотреть их командой `stat`. Более того, `debugfs` имеет команду, позволяющую восстановить файл автоматически. Например:

```
debugfs:  stat <148003>
Inode: 148003   Type: regular    Mode:  0644   Flags: 0x0   Version: 1
User:   503   Group:   100   Size: 6065
File ACL: 0    Directory ACL: 0
Links: 0   Blockcount: 12
Fragment:  Address: 0    Number: 0    Size: 0
ctime: 0x31a9a574 -- Mon May 27 13:52:04 1996
atime: 0x31a21dd1 -- Tue May 21 20:47:29 1996
mtime: 0x313bf4d7 -- Tue Mar  5 08:01:27 1996
dtime: 0x31a9a574 -- Mon May 27 13:52:04 1996
BLOCKS:
594810 594811 594814 594815 594816 594817
TOTAL: 6
```

Файл занимает 6 блоков. Так как 6 меньше 12, даем команду `debugfs` записать файл в новое место, например `/mnt/recovered.000`:

```
debugfs:  dump <148003> /mnt/recovered.000
```

Также можно это сделать с помощью `fsgrab`. Пример:

```
# fsgrab -c 2 -s 594810 /dev/hda5 > /mnt/recovered.000
# fsgrab -c 4 -s 594814 /dev/hda5 >> /mnt/recovered.000
```

В обоих случаях (как `debugfs`, так и `fsgrab`) в конце `/mnt/recovered.000` будет мусор. Впрочем, это не так уж важно. Самый простой способ убрать его - использовать значение поля `Size` из inode с ключом `bs` команды `dd`:

```
# dd count=1 if=/mnt/recovered.000 of=/mnt/resized.000 bs=6065
```

Конечно существует вероятность, что один или несколько блоков вашего файла были перезаписаны. Если это так, то вам не повезло - данные этого блока навсегда потеряны.

---

## 10.2. Длинные файлы

Сложнее обстоит дело с файлами длиной более 12 блоков. Необходимо пояснить, как устроена файловая система UNIX. Данные файла хранятся в так называемых "блоках". Эти блоки пронумерованы. Для каждого файла также имеется "inode" (от английского information node), где хранится информация о владельце, правах, типе файла и т. п. Также как и блоки, inode пронумерованы (хотя нумерация ведется независимо от блоков). Каталоги файловой системы содержат имя файла и номер inode.

Кроме того, для того, чтобы ядро знало, где искать данные, соответствующие элементу каталога (файлу), в inode следующим образом размещается информация о блоках с данными файла:

- Номера первых 12 блоков хранятся непосредственно в inode; их еще иногда называют *блоками с прямой адресацией (direct blocks)*.
- В inode хранится номер блока, в котором хранятся номера еще 256 блоков данных. Иногда его называют *блок косвенной адресации (indirect block)*.
- В inode хранится номер блока, в котором хранятся 256 номеров блоков косвенной адресации. Иногда этот блок называют *блоком двойной косвенной адресации (doubly indirect block)*.
- В inode хранится номер блока, в котором хранятся 256 номеров блоков двойной косвенной адресации. Его называют *блоком тройной косвенной адресации (triply indirect block)*.

Прочтите еще раз: я знаю, что это непросто, но также и очень важно.

Версии ядра до 2.0.36 включительно при удалении файла обнуляют блоки косвенной адресации (а также блоки двойной косвенной адресации и т. д.). Так что если ваш файл был длинной более 12 блоков, нет никакой гарантии, что вы сможете выявить даже номера блоков с данными, не говоря уже о самих данных.

Единственный способ, который я нашел - это предположить, что файл не был фрагментирован; если был, то у вас проблемы. Если же предполагать, что файл не был фрагментирован, то, в зависимости от количества блоков с данными файла, возможно следующее расположение блоков, описывающих местоположение файла:

0 - 12

: Номера блоков хранятся в inode, как описано выше.

13 - 268

После блоков с прямой адресацией идет блок косвенной адресации и далее 256 блоков с данными.

269 - 65804

Как и в прошлом случае, в начале 12 блоков с прямой адресацией, блок косвенной адресации и 256 блоков данных. Далее блок двойной косвенной адресации, за ним 256 групп блоков, состоящих из одного блока косвенной адресации и 256 блоков данных.

65805 и более

Расположение первых 65804 блоков указано выше. Далее один блок тройной косвенной адресации и 256 повторений групп "двойной косвенной адресации". Каждая такая группа состоит из блока двойной косвенной адресации, за которым идет 256 групп из одного блока косвенной адресации и 256 блоков данных.

Даже если номера блоков данных правильны, нет никакой гарантии, что данные в них не перезаписывались. К тому же, чем больше файл, тем меньше шансов, что он был записан без фрагментации (кроме некоторых особых случаев).

Заметьте, что я предполагал, что размер вашего блока 1024 байта, так как это стандартное значение. Если ваши блоки больше, некоторые числа, указанные выше, изменятся. В частности: так как номер блока занимает 4 байта, то количество номеров блоков, которые могут быть размещены в блоке косвенной адресации равно размер_блока/4. Так что везде, где выше встречается число 256, меняйте его на размер_блока/4. Количество требуемых для размещения файла блоков также нужно изменить.

Пример восстановления длинного файла:

```
debugfs:  stat <1387>
Inode: 148004   Type: regular    Mode:  0644   Flags: 0x0   Version: 1
User:   503   Group:   100   Size: 1851347
File ACL: 0    Directory ACL: 0
Links: 0   Blockcount: 3616
Fragment:  Address: 0    Number: 0    Size: 0
ctime: 0x31a9a574 -- Mon May 27 13:52:04 1996
atime: 0x31a21dd1 -- Tue May 21 20:47:29 1996
mtime: 0x313bf4d7 -- Tue Mar  5 08:01:27 1996
dtime: 0x31a9a574 -- Mon May 27 13:52:04 1996
BLOCKS:
8314 8315 8316 8317 8318 8319 8320 8321 8322 8323 8324 8325 8326 8583
TOTAL: 14
```

В данном случае шансы того, что файл не фрагментирован, довольно велики: первые 12 блоков, перечисленные в inode, (блоки с данными) идут подряд. Начнем с того, что восстановим их:

```
# fsgrab -c 12 -s 8314 /dev/hda5 > /mnt/recovered.001
```

Следующий блок указанный в inode (8326) - блок косвенной адресации, который мы можем игнорировать, так как предполагаем, что за ним идут блоки данных (с 8327 по 8582).

```
# fsgrab -c 256 -s 8327 /dev/hda5 >> /mnt/recovered.001
```

Последний блок, указанный в inode имеет номер 8583. Заметьте, если предположить, что файл не фрагментирован, то пока все нормально: последний блок данных, записанный нами имеет номер 8582, то есть 8327 + 255. Блок 8583 - блок двойной косвенной адресации, его можно игнорировать. За ним идут до 256 групп состоящих из блока косвенной адресации (который мы также игнорируем), и 256 блоков данных. Быстренько выполнив несложные арифметические подсчеты, выполняем следующие команды (заметьте, что мы пропускаем блок двойной косвенной адресации 8583 и следующий за ним (как мы надеемся) блок косвенной адресации 8584 и начинаем с блока 8525):

```
# fsgrab -c 256 -s 8585 /dev/hda5 >> /mnt/recovered.001
# fsgrab -c 256 -s 8842 /dev/hda5 >> /mnt/recovered.001
# fsgrab -c 256 -s 9099 /dev/hda5 >> /mnt/recovered.001
# fsgrab -c 256 -s 9356 /dev/hda5 >> /mnt/recovered.001
# fsgrab -c 256 -s 9613 /dev/hda5 >> /mnt/recovered.001
# fsgrab -c 256 -s 9870 /dev/hda5 >> /mnt/recovered.001
```

Итого мы записали 12 + (7 * 256) блоков, то есть 1804. В соответствие с результатом команды "stat" число блоков в файле было равно 3616; считается, что это блоки длиной 512 байт (пережиток с UNIX), поэтому, в действительности, нам надо 3616/2 = 1808 блоков длиной 1024 байт. То есть, нам нужно записать еще четыре блока. Последний записанный блок имел номер 10125. Также, как и раньше, пропускаем блок косвенной адресации (номер 10126) и записываем последние четыре блока:

```
# fsgrab -c 4 -s 10127 /dev/hda5 >> /mnt/recovered.001
```

В результате, если нам повезло, то файл полностью восстановлен.

---

# 11. Прямое редактирование inodes

Этот способ, на первый взгляд, намного проще. Однако, как уже говорилось, он подходит только для файлов длиной менее 12 блоков.

Для всех inode, подлежащих восстановлению, нужно установить счетчик ссылок в 1 и обнулить время удаления. Это можно сделать командой `mi` (modify inode - изменить inode) утилиты `debugfs`. Пример изменения inode 148003:

```
debugfs:  mi <148003>
                          Mode    [0100644]
                       User ID    [503]
                      Group ID    [100]
                          Size    [6065]
                 Creation time    [833201524]
             Modification time    [832708049]
                   Access time    [826012887]
                 Deletion time    [833201524] 0
                    Link count    [0] 1
                   Block count    [12]
                    File flags    [0x0]
                     Reserved1    [0]
                      File acl    [0]
                 Directory acl    [0]
              Fragment address    [0]
               Fragment number    [0]
                 Fragment size    [0]
               Direct Block #0    [594810]
               Direct Block #1    [594811]
               Direct Block #2    [594814]
               Direct Block #3    [594815]
               Direct Block #4    [594816]
               Direct Block #5    [594817]
               Direct Block #6    [0]
               Direct Block #7    [0]
               Direct Block #8    [0]
               Direct Block #9    [0]
              Direct Block #10    [0]
              Direct Block #11    [0]
                Indirect Block    [0]
         Double Indirect Block    [0]
         Triple Indirect Block    [0]
```

То есть, я обнулил время удаления (deletion time) и увеличил счетчик ссылок (link count). Для остальных полей я просто нажимал Enter. Довольно нудная работа, если нужно восстановить много файлов, но, я думаю, что вы справитесь. Если бы вы хотели чего попроще, то наверное поставили бы графическую "операционную систему" с ее хорошенькой "Корзиной".

Кстати, в предыдущем примере `mi` запрашивает значение поля "время создания" (`Creation time'). Обманывает вас! На самом деле вы не можете указать в файловой системе UNIX время создания файла. Поле `st_ctime` структуры `struct stat` описывает "время изменения inode", то есть время последней модификации содержимого inode. Все, пожалуй, хватит на сегодня уроков.

Более поздние версии `debugfs`, чем та, которую я использую, не включают некоторые поля из вышеуказанного примера ( `Reserved1` и (некоторые?) поля fragment).

После окончания модификации всех inode можно выйти из `debugfs` и дать команду:

```
# e2fsck -f /dev/hda5
```

Смысл в том, что хотя мы и восстановили файлы, но ни в одном каталоге нет на них ссылок. Утилита `e2fsck` определяет это и для каждого файла создает ссылку в каталоге файловой системы `/lost+found`. (То есть, если файловая система обычно монтируется в `/usr`, то при следующем монтировании восстановленные файлы будут лежать в `/usr/lost+found`.) Остается лишь определить, какое имя имел тот или иной файл, и поместить его туда, где он находился до удаления.

Во время работы `e2fsck` будет задавать вам различные вопросы, касающиеся ремонта файловой системы. Отвечайте "да" (yes) на все вопросы, связанные с "итоговой информацией" (summary information) и с исправленными вами inode. Остальное на ваше усмотрение, хотя, в общем случае, лучше отвечать "да" на все вопросы. По окончании работы `e2fsck` можно смонтировать файловую систему.

Вообще говоря, есть альтернатива помещению утилитой `e2fsck` файлов в каталог `/lost+found`: вы можете использовать `debugfs` для создания ссылки на inode. Для этого, после изменения inode, дайте команду `link`:

```
debugfs:  link <148003> foo.txt
```

Она создаст файл `foo.txt` в каталоге, который `debugfs` считает текущим. `foo.txt` - это ваш файл. Независимо от этого `e2fsck` нужно запустить для исправления итоговой информации и т. п.

---

# 12. Станет ли легче в будущем?

Да. Уже сейчас стало проще. Ядра версий 2.0.x обнуляют блоки косвенной адресации, но к ядрам более поздних версий (2.1.x и 2.2.x) это не относится. Я пишу это 2 февраля 1999 г., буквально через несколько дней после выхода в свет ядра версии 2.2.1. Думаю, что через месяц - другой появятся дистрибутивы Linux с этим ядром.

После того, как ограничение, связанное с обнулением блоков косвенной адресации, будет снято, большинство моих указаний по технике изменения inode отпадут за ненадобностью. Кроме того, можно будет как использовать команду `dump` утилиты `debugfs` для длинных файлов, так и с успехом пользоваться другими утилитами для восстановления файлов.

---

# 13. Есть ли какие-нибудь утилиты для автоматизации процесса?

Есть. К сожалению, при работе с ними проблема с восстановлением блоков косвенной адресации (также, как и при восстановлении файлов вручную) остается. Впрочем, если учесть, что эта проблема вскоре перестанет быть таковой, можно начинать искать такие утилиты.

Я написал на Perl утилиту для упрощения работы с `fsgrab` и назвал ее `e2recover`. Она выполняет большую часть работы по восстановлению файлов с блоками косвенной адресации и, вроде бы, нормально работает, если файлы не были фрагментированы. Кроме того, она корректно выставляет права (если это возможно, то и владельца) и длину восстановленных файлов.

Изначально, я написал `e2recover` для будущего полного Howto, поэтому документация по ней появится лишь в нем. Впрочем, если желаете, можете скачать ее с моей странички, а также вскоре с Metalab.

Scott D. Heavner - автор `lde`, редактора дисков Linux (Linux Disk Editor). Эта утилита может быть использована и как редактор диска, и как замена `debugfs` для файловых систем ext2 и minix, и даже для xia (хотя поддержка xia убрана из ядер версий 2.1.x и 2.2.x ). Она имеет некоторые полезные возможности для облегчения восстановления, например, просмотр списка блоков файла и поиск по диску. Кроме того, в нее включена полезная документация по основам файловых систем, руководство по восстановлению файлов. `lde` версии 2.4 можно найти на Metalab и зеркалах или на страничке автора.

Еще один вариант предлагается GNU Midnight Commander, `mc`. Это полноэкранный менеджер файлов, основанный AFAIK (насколько я знаю) на программе под MS-DOS, известной как "NC". `mc` поддерживает мышь для консоли и xterm, и доступ к виртуальным файловым системам, что позволяет, например, войти в tar архив, как в обычный каталог. Среди прочих виртуальных файловых систем есть и одна для восстановления файлов с ext2. Звучит удобно, но должен признаться, что сам я эту программу не использовал - я предпочитаю старый добрый sh.

Для использования возможности восстановления файлов нужно при настройке программы перед сборкой указать опцию `--with-ext2undel`. Кроме того, понадобятся библиотеки из пакета `e2fsprogs`. Версия программы, включаемая в Debian GNU/Linux, собрана с поддержкой восстановления, это же относится и к другим дистрибутивам. После сборки программы наберите `cd undel:/dev/hda5`, и вы получите "каталог" с удаленными файлами. Как и большинство других утилит восстановления файлов, mc плохо обрабатывает файлы, содержавшие блоки косвенной адресации - обычно просто восстанавливаются первые 12 блоков.

Очередная версия может быть найдена на ftp сайте Midnight Commander.

---

# 14. Отзывы и предложения

Я планирую обновлять этот документ, если появится что-то интересное, и найдется свободное время. Это означает, что я совсем не против каких-либо комментариев от читателей. Можно ли написать понятней? Может быть что-то как-то можно сделать проще? Или появилась новая утилита, позволяющая автоматизировать процесс? Ну и все такое. Если у вас есть, что сказать про этот документ или про `fsgrab`, или `e2recover`, черкните мне пару строчек на адрес `aaronc@pobox.com`.

---

# 15. Ссылки и благодарности

""Если я видел дальше других, то потому, что стоял на плечах гигантов." (Исаак Ньютон)"

Этот Мини-Howto был изначально основан на статье Robin Glover swrglovr@met.rdg.ac.uk из конференции comp.os.linux.misc. Хотелось бы поблагодарить его за милостивое разрешение использовать его идеи в этом Мини-Howto.

Хотелось бы также воспользоваться возможностью и еще раз поблагодарить всех, кто написал мне по поводу этого документа.

Кое-какая литература:


- *Frisch*, ϕleen (1995), *Essential System Administration*, second edition, O'Reilly and Associates, Inc., ISBN: 1-56592-127-5.

- *Garfinkel*, Simson, Daniel *Weise* and Steven *Strassmann* (1994), *The Unix-Haters Handbook*, IDG Books, ISBN: 1-56884-203-1. Большая часть этой книги - это дикие визги тех, кто считает, что *ux* ОС лучше, чем Unix. Остальное практически не представляет интереса, так как в GNU все значительно подробнее и проще. Однако, среди всего этого мусора есть и хорошие вещи: например, рассуждения о том, как проще удалить файл в Unix.

- *Glover*, Robin (31 Jan 1996), *HOW-TO : undelete linux files (ext2fs/debugfs)*, comp.os.linux.misc Usenet posting.

- *Peek*, Jerry, Tim *O'Reilly*, Mike *Loukides* et al (1993), *UNIX Power Tools*, O'Reilly and Associates, Inc./Random House, Inc., ISBN: 0-679-79073-X. Second edition, 1998.


---

# 16. Права

Все торговые марки - собственность их уважаемых владельцев. В частности:

- *MS-DOS* и *Windows* - торговые марки Microsoft.

- *UNIX* - торговая марка Open Group.

- *Linux* - торговая марка Linus Torvalds в США и некоторых других странах.

The Linux HOWTO Coordinator is currently Tim Bynum linux-howto@metalab.unc.edu.

---

# 17. Авторские права



Этот документ является частью проекта Linux HOWTO.