Мини-HOWTO: Восстановление структуры каталогов файловой системы Ext2fs

Tomas Ericsson

tomase@matematik.su.se

Перевод: Станислав Рогин, SWSoft Pte Ltd.

История изменений

Издание версия 0.1	7 сентября 2000 года	Под редакцией: T.E.
Исходная версия.		

Этот документ является дополнением к *Мини-HOWTO: Восстановление файлов в Ext2fs* автора Aaron Crane. Я настоятельно рекомендую вам изучить его перед прочтением этого документа.

Ниже я опишу простой способ восстановления целых стуктур каталогов, случайно удаленных командой *rm -rf*.

---

1. Введение

1.1. Ответственность

Автор этого документа не несет никакой ответственности за прямые или косвенные последствия использования этого документа. Вы можете использовать информацию из этого документа только на свой страх и риск.

1.2. License

1.3. Авторские права

1.4. Отзывы

Я приветствую любые отзывы. Особый интерес для меня представляют указания на ошибки в этом документе. Если то, что я написал в в этом документе, кому-то пригодится, то я буду рад об этом услышать.

1.5. Новые версии этого документа

Самую свежую версию этого документа можно найти по адресу http://www.matematik.su.se/~tomase/ext2fs-undeletion/

1.6. Благодарности

Спасибо за исправления всем, кто мне помог, а в особенности:

- Gabriel Kihlman
- Richard Nyberg
- Mats Oldin
- Tobias Westerblom

1.7. Причины написания этого документа

Этот текст был написан вследствие проблем с восстановлением файлов, возникших у меня не так давно. Я перенес командой move с одного диска на другой несколько каталогов. Проблема состояла в том, что данные на втором диске были испорчены после переноса.

Поэтому мне понадобилось восстановить на первом диске исходные перенесенные каталоги . Мне надо было вернуть к жизни около 40000 файлов, и мне не очень хотелось восстанавливать их по одному вручную.

Мне надо было вернуть всю структуру каталогов. То же самое случилось бы, если бы я использовал для этих каталогов команду *rm -rf*.

---

2. Обязательные условия

Очень важно, чтобы вы сразу отключили необходимый раздел командой umount, не производя с ним больше никаких операций. Если вы после удаления необходимых файлов запишите на этот раздел что-нибудь, то ваши шансы на успешное восстановление файлов резко уменьшаются.

Также вам понадобится достаточно новая версия ядра, потому что ядра 2.0.x и ниже очищают блоки косвенной адресации, что не позволит вам восстановить файлы длиной больше 12 блоков.

Я опишу лишь один метод восстановления, и не уделю большого внимания ошибкам, которые могут возникнуть в процессе восстановления. Если вы решите, что какой-то из нижеописанных шагов привел к ошибке, то я не рекомендую вам продолжать следовать этим советам.

---

3. Приготовления

Отключите раздел, на котором находились удаленные файлы. Назовем этот раздел /dev/hdx1:

```
# umount /dev/hdx1
```

Узнайте размер /dev/hdx1 в блоках командой:

```
# fdisk -l /dev/hdx
```

Теперь для дальнейшей работы вам понадобится дополнительный раздел того же размера, что и /dev/hdx1. Предположим, что у вас есть пустой жесткий диск /dev/hdy:

```
# fdisk /dev/hdy
```

Создайте на нем раздел того же размера, что и /dev/hdx1. Здесь и ниже *размер* - это размер раздела /dev/hdx1 в блоках (каждый блок - это 1024 байта), который вы узнали ранее.

Я использую *fdisk* версии 2.10f. Если вы используете другую версию *fdisk*, то работа с ней может различаться.

```
   fdisk: n        <- Создать новый раздел.
   fdisk: p        <- Основной раздел.
   fdisk:          <- Просто нажмите Enter, чтобы использовать предложенный начальный
цилиндр.
   fdisk: +размерK <- создайте раздел, имеющий тот же размер, что и /dev/hdx1.
   fdisk: w        <- Запишите таблицу разделов на диск и выйдите из программы.
```

Теперь скопируем содержимое исходного раздела на новый диск:

```
# dd if=/dev/hdx1 of=/dev/hdy1 bs=1k
```

Это может занять дительное время, в зависимости от размера раздела. Вы можете ускорить процесс, увеличив размер блока *bs*, но вам придется сделать его таким, чтобы раздел состоял из целого количества этих блоков.

С этого момента мы будем работать только с этой копией исходного раздела, чтобы мы всегда могли откатиться назад, если что-то пойдет не так.

---

4. Находим номера inode удаленных каталогов

Мы попытаемся выяснить номера inode удаленных каталогов:

```
# debugfs /dev/hdy1
```

Перейдите к тому месту, где были удаленные каталоги. Внутри *debugfs* вы можете обычным образом использовать команды *ls* и *cd*:

```
debugfs: ls -l
```

Эта команда выдаст на экран примерно следующее:

```
179289  20600      0      0       0 17-Feb-100 18:26 file-1
918209  40700    500    500    4096 16-Jan-100 15:18 file-2
160321  41777      0      0    4096  3-Jun-100 06:13 file-3
177275  60660      0      6       0  5-May-98  22:32 file-4
229380 100600    500    500   89891 19-Dec-99  15:40 file-5
213379 120777      0      0      17 16-Jan-100 14:24 file-6
```

Описание полей:

1. Номер inode.

2. Первые две (или одна) цифры означают вид inode:

   2 = Символьное устройство

   4 = Каталог

   6 = Блочное устройство

   10 = Обычный файл

   12 = Символьная ссылка

Следующие четыре цифры - это обычные права доступа к файлу в формате Unix.

3. Владелец файла (в числовой форме).

4. Группа файла (в числовой форме).

5. Размер в байтах.

6. Дата (Вы наверно уже заметили здесь "Проблему-2000" в действии =)).

7. Время.

8. Имя файла.

Теперь выгрузим родительский каталог на диск. Здесь и ниже *inode* - это соответствующий номер inode (не забудьте символы '<' и '>').

```
debugfs: dump <inode> debugfs-dump
```

Выйдите из *debugfs*:

```
debugfs: quit
```

---

5. Анализируем содержимое каталога

Просмотрите выгруженное содержимое каталога в читаемом формате:

```
# xxd debugfs-dump | less
```

Каждая запись состоит из пяти полей. Байты первых двух полей представлены в обратном порядке. Это значит, что первый байт - самый младший.

Описание полей:

1. 4 байта - номер inode.

2. 2 байта - длина этой записи.

3. 1 байт - длина имени файла (1-255).

4. 1 байт - тип файла (0-7).

0 = Неизвестный

1 = Обычный файл

2 = Каталог

3 = Символьное устройство

4 = Блочное устройство

5 = Поток FIFO

6 = Поток SOCK

7 = Символьная ссылка

5. Имя файла (1-255 символов).

Если запись удаляется из каталога, то размер предыдущей записи увеличивается на размер удаляемой записи (предыдущая запись как бы "съедает" следующую).

Если файл переименовывается в более короткое имя, то уменьшается значение третьего поля.

Первая запись, которую вы увидите - это сам каталог, представляемый одной точкой.

Предположим, что у нас есть следующая запись в каталоге:

```
c1 02 0e 00 40 00 05 01 'u' 't' 'i' 'l' 's'
```

В ней номер inode будет "e02c1" (в шестнадцатиричной форме) или 918209 (в десятичной). Следующая запись находится через 64 байте (шестнадцатиричное 40). Мы также видим, что имя файла состоит из 5 байт ("utils") и что тип файла (01) соответствует обычному файлу.

Теперь пересчитаем номера inode подкаталогов в десятичную форму.

Если вы не любите производить такие операции вручную, то я для вас написал небольшую программу на C. Программа берет содержимое каталога (созданное *debugfs*, как описано ранее в разделе Разд. 4). На стандартном выводе вы получаете список имен файлов и номеров inode.

Перед запуском этой программы вам надо загрузить записанное содержимое каталога в двоичный редактор и изменить поле *"длина записи каталога"* в записи, предшествующей восстанавливаемой. Это просто: если

мы обозначим длину предшествующей записи как *x*, а длину записи, которую вы хотите восстановить как *y*, то вам надо заменить поле, содержащее *x* на *x*-*y*.

Программа называется *e2dirana* (ext2fs directory analyse), и ее можно найти по адресу http://www.matematik.su.se/~tomase/ext2fs-undeletion/

---

6. Находим удаленные inode

Получаем список всех удаленных inode.

```
# echo lsdel | debugfs /dev/hdy1 > lsdel.out
```

Одна проблема состоит в том, что *debugfs* не выдаст номера inode файлов, у которых была нулевая длина (у вас могли быть такие файлы, например, в каталоге /etc). Я опишу решение этой проблемы в разделах Разд. 9 и Разд. 11.

Загрузите "lsdel.out" в текстовый редактор. Список inode должен быть отсортирован по времени удаления. Попробуйте точно вспомнить время, когда вы дали команду *rm -rf*. Скорее всего, это была последняя команда, и удаленные inode будут находиться в конце списка. Удалите все не интересующие вас строки. Запишите этот файл как "lsdel.out-selected".

Теперь мы удалим из этого файла все, кроме номеров inode:

```
# cut -b 1-8 lsdel.out-selected | tr -d " " > inodes
```

Для полной уверенности проверьте, что удаленные каталоги, номера которых мы нашли ранее, находятся в этом списке.

```
# grep ^inode$ inodes
```

, где *inode* - это соответствующий номер inode.

---

7. Активизируем inode

Теперь пришло время изменить некоторые флаги удаленных inode.

Скопируйте следующие 6 строк в файл, назвав его "make-debugfs-input".

```
#!/bin/sh
awk '{ print "mi <" $1 ">\n"\
             "\n\n\n\n\n\n\n"\
             "0\n"\
             "1\n"\
             "\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n" }'
```

Этим мы имитируем ручное исправление inode. Мы устанавливаем *время удаления* в 0 и *количество ссылок* в 1.

> Я использую *debugfs* версии 1.18, и, если у вас другая версия, то вам, возможно, придется изменить количество "нажатий" клавиши Enter в вышеприведенном скрипте.

Теперь изменяем inode:

```
# ./make-debugfs-input < inodes | debugfs -w /dev/hdy1 | tail -c 40
```

Если все пройдет хорошо, то последнее сообщение должно быть таким: "Triple Indirect Block [0] debugfs:".

---

8. Добавляем записи в каталоги

Запустите *debugfs* в режиме чтения-записи.

```
# debugfs -w /dev/hdy1
```

Теперь вым надо добавить удаленные каталоги в каталог, где они ранее находились:

```
debugfs: link <inode> directoryname
```

Здесь *inode* - это номер inode, а *directoryname* - это номер каталога.

После того, как вы добавите ссылки, вы заметите, что удаленные каталоги появились в текущем каталоге. Вы можете теперь просмотреть его содержимое (при помощи *debugfs*).

Правда, размер каждого каталога равен 0, и это надо исправить, иначе они будут выглядеть пустыми в команде *ls*.

Выйдите из *debugfs*:

```
debugfs: quit
```

---

9. Пересчет

Теперь наступило время вызвать *e2fsck* для пересчета размеров и контрольных сумм.

> Я использую *e2fsck* версии 1.18. Если у вас другая версия, то, возможно, ее параметры или сама работа с программой могли измениться.

Если вы точно знаете, что у вас *НЕ* было файлов с нулевой длиной, то вы можете сделать следующее: (см. ниже); и пропустить все остальное (Вы, конечно, можете не использовать параметр *y*, но вам придется вручную отвечать на все вопросы - это может занять длительное время.).

```
# e2fsck -f -y /dev/hdy1 > e2fsck.out 2>&1
```

Если же вы хотите восстановить файлы с нулевой длиной, то вам надо ответить *n* на все вопросы об удалении записей и *y* на все остальные.

Скопируйте следующие 7 строк в файл "e2fsck-wrapper".

```
#!/usr/bin/expect -f
set timeout -1
spawn /sbin/e2fsck -f $argv
expect {
    "Clear<y>? " { send "n" ; exp_continue }
    "<y>? "      { send "y" ; exp_continue }
}
```

Запустите скрипт.

```
# ./e2fsck-wrapper /dev/hdy1 > e2fsck.out 2>&1
```

Просмотрите файл "e2fsck.out", чтобы узнать, что сообщил *e2fsck* о вашем разделе.

---

10. Если каталог /lost+found не пуст

Некоторые из ваших каталогов или файлов могут не появиться в обычных местах. Вместо этого они могут появиться в каталоге /lost+found под именами, состоящими из их номеров inode и старых имен.

В этом случае указатель в элементе ".." каталога скорее всего изменился, и указывает на один из последних файлов каталога (я не знаю, почему это происходит - возможно, это ошибка в драйвере файловой системы).

Изучите *3-ю фазу* файла "e2fsck.out" (в ней проверяется связность каталогов). Там вы увидите названия каталогов, которые были затронуты e2fsck. Запишите их на диск (как было описано в главе Разд. 4).

Запустите *e2dirana* как с флагом *p*, так и без него (так вы измените указатель на ".."). Здесь и ниже *dump* - это записанное на диск содержимое каталога.

```
# ext2fs-directory-analyse dump > dump1
# ext2fs-directory-analyse -p dump > dump2
```

Сравните результат работы программ

```
# diff dump1 dump2
```

Если эти файлы не равны, значит в этом каталоге есть пропавшие файлы. Переместите данные файлы из каталога /lost+found в правильное место. Здесь *dest* - это символьная ссылка на каталог-приемник. Поместите результат работы этого мини-скрипта в файл, и запустите его, если там все правильно.

```
# diff dump1 dump2 |\
  tail -n $[`diff dump1 dump2 | wc -l`-1] | cut -b 3- |\
  sed -e 's/^\([^ ]*\) \(.*\)$/mv lost+found\/#\1 dest\/"\2"/' |\
  sed -e 's/!/"\\\!"/g'
```

Повторяйте эти действия до тех пор, пока каталог /lost+found не будет пуст.

---

11. Последние коррективы

Если в разделе Разд. 9 вы решили, что хотите восстановить файлы с нулевой длиной, то тут у вас возникла проблема - дело в том, что у этих файлов не очищено время удаления, а количество ссылок установлено в 0. Это означает, что *e2fsck* при каждом запуске будет предлагать удалить (очистить) данные файлы.

Самое простое - скопировать всю структуру каталогов в другое место (возможно, и на этот же раздел), и затем удалить старую структуру каталогов. В противном случае вам придется отыскать эти inode и исправить их при помощи *debugfs*.

Теперь, если все прошло хорошо, все должно было прийти в исходное состояние (которое было до удаления файлов). Так, по-крайней мере, случилось у меня, и подтвердилось в процессе испытания этих советов при написании данного документа. Убедитесь в том, что вы полностью выполнили все условия, описанные в разделе Разд. 2.

---

12. Ссылки

Мини-HOWTO: Восстановление файлов в Ext2fs, версия 1.3

- Aaron Crane

Дизайн и Реализация файловой системы Second Extended, http://e2fsprogs.sourceforge.net/ext2intro.html

- RИmy Card, Laboratoire MASI--Institut Blaise Pascal
- Theodore Ts'o, Massachussets Institute of Technology
- Stephen Tweedie, University of Edinburgh

Исходные тексты ядра версии 2.2.16

- linux/include/linux/ext2_fs.h
- linux/fs/ext2/namei.c