Ext2fs Undeletion of Directory Structures mini-HOWTO

Tomas Ericsson

tomase (at) matematik.su.se

도운이

정지용

ways (at) gon.kaist.ac.kr

이 문서는 Aaron Crane에 의해 쓰여진 **Ext2fs-Undeletion mini-HOWTO**를 보충하기 위한 것이다. 나는 이 문서를 읽기 전에 그 문서를 먼저 자세히 읽기를 강력히 추천한다.

여기서 나는 실수로 실행시킨 **rm -rf**명령에 의해 지워진 파일 단위가 아닌 전체 디렉토리 구조를 복구하는 직접적인 방법을 제시할 것이다.

고친 과정

고침 v0.1.114 November 2000고친이 T.E.

Initial release.

고침 v0.1.129 November 2001고친이 정 지용

최초 번역

1. 소개

1.1. 책임의 한계

본 저자는 이 문서의 내용이 야기할 수 있는 어떠한 결과에 대해서도 책임을 지지 않습니다. 모든 일은 당신의 책임 하에서 수행되어야 합니다.

1.2. 저작권

1.3. 피드백

어떤 형태의 피드백이든 환영한다. 이 문서의 오류 수정은 매우 중요하다. 만약, 누군가가 이 문서를 유용하게 사용하였다면, 나는 그것에 대해 듣는 것을 좋아한다.

1.4. 이 문서의 새 버전

이 문서의 최신 버전은 http://www.matematik.su.se/~tomase/ext2fs-undeletion/ (http://www.matematik.su.se/~tomase/ext2fs-undeletion/) 에서 얻을 수 있다.

1.5. 감사

오류를 알려준 다음의 사람들에게 감사한다. (알파벳 순서)

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

1.6. 배경

이 문서는 내가 얼마 전 겪었던 복구 문제를 해결하는 도중에 쓰여졌다. 나는 몇개의 디렉토리를 하나의 디스크로부터 다른 디스크로 옮기려 하고 있었다. 그러나 문제는 옮기려 했던 디스크가 이동 직후, 모종의 이유로 인해 오류가 발생하였다.

그래서 나는 이동시킨 디렉토리를 원본 디스크로 옮기고자 했다. 그러나 약 40000개의 복구할 파일로 인하여 그들을 일일이 손으로 찾아서 복구하기가 어려워졌다.

나는 디렉토리의 전체 구조를 되돌리고 싶었다. 똑같은 상황은 만약 내가 그 디렉토리들을 대상으로 **rm -rf** 를 실행했어도 일어났을 것이다.

2. 전제조건

무엇보다 영향받은 파티션을 최대한 빨리 아무것도 하지 않은채로 언마운트 시키는 것이 매우 중요하다. 만약 당신이 사고후에 이 파티션에서 파일을 카피하거나 했다면, 이 방법이 성공할 가능성은 매우 낮아진다.

또한, 꽤 최근의 커널 버전을 가져야만 한다. 왜냐하면 2.0.x 나 그 이하의 버전 커널에서는 이 과정이 12블록 이상의 데이터를 가진 파일에 대해서는 작동하지 않을 것이기 때문이다.

나는 복구하는 한 방법을 설명할 것이나, 대부분의 에러에 대한 설명은 생략할 것이다. 만약 다음에 설명된 과정이 잘 작동하지 않는 것 같으면 거기서 멈추고 더이상 하지 않기를 추천한다.

3. 준비

지워진 파일이 있는 파티션을 언마운트한다. 이 파티션을 /dev/hdx1이라 하겠다.

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

/dev/hdx1의 크기를 블록단위로 체크한다.

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

이제 안전을 기하기 위하여 /dev/hdx1과 같은 크기의 또다른 파티션이 필요하다. 당신이 /dev/hdy에 비어있는 하드드라이브를 가지고 있다고 가정하자.

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

/dev/hdx1과 같은 사이즈의 파티션을 새로 만든다. 여기서 사이즈는 블록 단위(각 블록은 1024kB)로 나타낸 /dev/hdx1의 사이즈이다.

> 참고**:** 나는 **fdisk** 2.10f 버전을 사용한다. 만약 당신이 다른 버전의 **fdisk** 를 사용한다면, 아래의 내용은 약간 다를 수 있다.

```
fdisk: n       <- 새로운 파티션을 만든다.
fdisk: p       <- Primary 파티션.
fdisk:         <- 초기치인 첫째 실린더를 선택하기 위해 그냥 엔터를 누른다.
fdisk: +sizeK <- /dev/hdx1과 같은 사이즈의 파티션을 만든다.
fdisk: w       <- 테이블을 디스크에 기록하고 종료한다.
```

이제 원본 파티션의 내용을 새로운 파티션으로 복사한다.

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

이 과정은 파티션의 크기에 따라 꽤 오래걸릴 수도 있다. 만약 블록사이즈 **bs**를 늘린다면, 더 빠르게 할 수 있겠지만, 그럴 경우에는 파티션의 크기가 **bs** 로 나누어 떨어져야 한다.

이제부터 우리는 잘못되는 것을 방지하기 위해 원본 파티션의 복사본만을 가지고 작업할 것이다.

4. 지워진 디렉토리의 inode를 찾기

우리는 지워진 디렉토리의 inode 번호들을 찾을 것이다.

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

지워진 디렉토리가 원래 위치했던 곳으로 이동한다. **debugfs** 안에서는 **ls** 나 **cd** 로 이동할 수 있다.

```
debugfs: ls -l
```

위의 명령에 대한 output으로 다음과 같은 예시를 들 수 있다.

```
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
```

위 내용의 각 필드들을 설명하면,

1. Inode 번호.
2. 맨 앞 두(혹은 첫) 숫자는 inode의 종류를 의미한다.

   2 = 문자 디바이스

   4 = 디렉토리

   6 = 블록 디바이스

   10 = 일반 파일

   12 = 심볼릭 링크

   남은 네 숫자는 Unix에서 정해지는 것이다.
3. 숫자로 표현된 소유주.
4. 숫자로 표현된 소유 그룹.
5. 바이트로 나타낸 크기.

6. 날짜 (여기서 우리는 Y2K 버그를 볼 수 있다 =)).
7. 시간.
8. 파일이름.

이제 상위 디렉토리를 디스크에 덤프하자. 여기서 **inode** 는 그에 해당하는 inode 번호를 가리킨다 ('<' 와 '>'를 빼먹지 않도록 해야한다.).

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

debugfs에서 빠져나온다.

```
debugfs: quit
```

5. 덤프된 디렉토리의 분석

읽을 수 있는 형태로 덤프된 inode를 보자.

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

모든 엔트리는 다섯개의 필드로 구성된다. 첫 두 필드는 역순으로 바이트가 배열되어있다. 이는 첫번째 바이트가 제일 낮은 자리의 수라는 것을 의미한다.

각 필드의 설명.

1. 4 바이트 - Inode 번호.
2. 2 바이트 - 디렉토리 엔트리 길이.
3. 1 바이트 - 파일이름 길이 (1-255).
4. 1 바이트 - 파일의 종류.

   0 = 알수없음

   1 = 일반 파일

   2 = 디렉토리

   3 = 문자 디바이스

   4 = 블럭 디바이스

   5 = FIFO

   6 = SOCK

   7 = 심볼릭 링크
5. 파일이름 (1-255자).

만약 디렉토리의 어떤 엔트리가 지워져야 한다면, 지워져야 할 엔트리 바로 앞에 있는 엔트리의 두번째 필드가 지워져야 할 엔트리의 두번째 필드 값만큼 증가된다.

만약, 파일이름이 더 짧은 것으로 바뀌면, 세 번째 필드값이 줄어든다.

맨 첫 엔트리는 '.'으로 표현되는 그 디렉토리 자신이다.

우리가 다음과 같은 디렉토리 엔트리들을 가지고 있다고 하자.

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

그러면 inode는 16진수로 e02c1이 될 것이고, 이는 10진수로 918209이다. 다음 엔트리는 64바이트(16진수로 40) 뒤에 위치하게 될 것이고, 우리는 파일 이름이 5자("utils")로 구성된 것을 알 수 있다. 그리고 파일의 종류는 일반 파일임을 알 수 있다.

이제 디렉토리의 inode 번호를 10진수로 다시 계산하자.

만약 이를 손으로 계산하고 싶지 않다면, 내가 C로 만든 작은 프로그램을 사용해라. 이 프로그램은 디렉토리 덤프(4절에서 설명된 **debugfs** 으로 만들어진다)를 입력으로 받고, 각 엔트리의 inode번호와 파일이름을 stdout에 출력한다.

프로그램을 실행시키기 전에, 덤프를 hex에디터로 불러들여 되살리고자 하는 디렉토리 엔트리 바로 앞 엔트리의 디렉토리 엔트리 길이필드를 조정할 필요가 있다. 그러나, 이는 간단하다. 만약 앞쪽 엔트리의 필드 값을 **x**라 하고, 되살리고자 하는 엔트리의 필드 값을 **y**라 하면, **x**를 **x - y**로 바꾸면 된다.

http://www.matematik.su.se/~tomase/ext2fs-undeletion/ (http://www.matematik.su.se/~tomase/ext2fs-undeletion/) 에서 **e2dirana** (ext2fs directory analyse)란 프로그램을 구할 수 있다.

6. 지워진 inode들을 원위치 시키기

지워진 inode들의 목록을 얻는다.

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

문제는 여기서 **debugfs**가 크기가 0인 파일(당신은 아마도 /etc디렉토리에서 이러한 것들을 볼 수 있을 것이다)의 inode 번호를 주지 않는다는 것이다. 이 문제에 대한 해결책은 9절 과 11절에서 제시하도록 하겠다.

"lsdel.out"을 문서 편집기로 불러온다. inode들의 목록은 시간 순서대로 정렬 되어있을 것이다. 당신이 **rm -rf**를 언제 했는지를 기억해 보아라. 아마도 그것은 당신이 마지막으로 지운 것이었을 것이고 목록이 시간순으로 정렬되어있기 때문에 그것들은 목록의 맨 마지막에 있을 것이다. 필요하지 않은 것들을 모두 지우고, 이를 "lsdel.out-selected"로 저장한다.

이제 inode를 제외한 모든 정보를 지운다.

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

확실히 하기 위해서, 위에서 찾은 지워진 디렉토리들의 inode가 목록에 있는지 확인하자.

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

여기서 **inode** 는 그에 해당하는 inode번호이다.

7. inode들을 활성화시키기

이제 지워진 inode들의 플래그들을 조정해야한다.

"make-debugfs-input"이란 파일을 만들고, 다음 6줄을 넣어라.

```
#!/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\n\n" }'
```

이는 inode를 직접 수정할 때, 사람의 입력을 시뮬레이트 할 것이다. 우리는 지워진 시간을 0으로 하고, 링크 숫자 를 1로 할 것이다.

> 참고**:** 나는 **debugfs** 1.18 버전을 사용하고 있다. 만약 다른 버전을 사용하고 있다면, 당신은 위 스크립트의 엔터 갯수를 적절하게 조절해야 할 것이다.

이제 inode를 수정하자.

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

만약, 모든 것이 잘 수행되었다면, 위 명령은 "Triple Indirect Block [0] debugfs:"라고 하면서 끝나야 한다.

8. 디렉토리 엔트리를 더하기

debugfs을 읽기-쓰기 모드로 시작한다.

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

이제 당신은 지워진 디렉토리가 위치했던 곳에 디렉토리를 새로 넣어야 한다.

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

여기서 **inode** 는 inode 번호이고, **directoryname** 은 디렉토리 이름이다.

링크를 추가한다음, 디렉토리가 현재 디렉토리에 추가된 것을 확인할 수 있을 것이다. 당신은 이제 그들의 내용을 확인 할 수 있다 (**debugfs**에서).

그러나 각 디렉토리의 크기가 0이기 때문에 이는 고쳐져야한다. 그렇지 않으면 쉘에서 **ls** 명령으로 보았을 때, 비어있는 것으로 보일 것이다.

debugfs에서 빠져나온다.

```
debugfs: quit
```

9. 재계산

이제 크기와 체크섬을 다시 계산하기 위해 **e2fsck** 을 실행시킬 차례이다.

> 참고**:** 나는 **e2fsck** 1.18 버전을 쓰고 있다. 만약 다른 버전을 사용하고 있다면, 파라미터나 입/출력이 바뀌지 않았는지 확인해야 할 것이다.

만약 당신이 복구하고자 하는 파일 중 크기가 0인 파일이 하나도 없다면 아래의 내용을 한 뒤, 이 글의 나머지 부분은 넘어가도 된다. (물론 당신은 인자 **y**를 쓰지 않을 수도 있겠지만, 그럴 경우 모든 질문에 일일이 손으로 대답해야 하므로, 많은 시간이 걸릴 것이다.

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

만약 당신이 크기가 0인 파일들을 되살리고 싶다면, 엔트리를 지우겠냐는 질문에 **no** 라고 대답해야하고, 지워도 되는 엔트리 에 대해서는 **yes** 라고 대답하면 된다.

아래 7줄의 내용을 "e2fsck-wrapper"란 파일에 넣는다.

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

위 스크립트를 실행시킨다.

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

e2fsck에서 어떻게 파티션을 처리했는지 보기위해서는, e2fsck.out의 내용을 보아라.

10. 만약 /lost+found 디렉토리가 비어있지 않다면

몇몇 디렉토리나 파일이 정확한 위치에 나타나지 않을 수도 있다. 대신에 그 파일들은 /lost+found파일에 그들의 inode를 파일이름으로 하여 나타났을 것이다.

이 경우는, ".." 디렉토리 엔트리에 대한 포인터가 증가되어 그 다음 디렉토리를 가리키고 있게된다. (이런 현상이 일어나는 이유는 알 수 없다. 아마 파일시스템의 버그인 것 같다)

디렉토리의 연결성이 검사되는 "e2fsck.out"의 **pass 3** 을 조사해보아라. 아마도 거기서 영향받은 디렉토리를 찾을 수 있을 것이다. 4절에 나온대로 디스크를 덤프하여라.

e2dirana (http://www.matematik.su.se/~tomase/ext2fs-undeletion/) 를 **p** 인자를 주고, 실행시키고, 인자 없이 또 실행시켜라.(이는 ".." 디렉토리 엔트리의 포인터를 변화시킬 것이다). 여기서 **dump**는 덤프된 디렉토리이다.

```
# e2dirana dump > dump1
# e2dirana -p dump > dump2
```

두 가지 출력된 것을 비교해 보아라.

```
# diff dump1 dump2
```

만약 두 가지 출력이 같지 않다면 디렉토리의 어떤 파일인가가 없어진 것이다. 그러면 해당하는 파일을 /lost+found 에서 찾아서 올바른 위치로 옮겨주어라. 여기서 **dest** 는 목적 디렉토리에 대한 심볼릭 링크이다. 출력을 스크립트에 넣고, 당신이 동의한다면 스크립트를 실행시켜라.

```
# 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'
```

이 과정을 /lost+found가 비게 될 때까지 반복한다.

11. 마지막 손질

만약 9절 에서 크기가 0인 파일들을 되살리기로 결정 했으면 문제가 하나 남아있다. 왜냐하면 이 파일들은 0이 아닌 삭제 시간과 0의 링크 숫자를 가지고 있기 때문이다. 이는 **e2fsck**가 실행될 때마다 이 파일들을 삭제하라고 물어보게 될 것이기 때문이다.

이 문제를 해결하는 가장 쉬운 방법은 전체 디렉토리 구조를 다른 곳(같은 파티션 상에 있어도 된다)으로 복사 하고, 원본 파일들을 지운 후, 다시 원래 위치로 복사해오는 것이다. 이렇게 하지 않으면, 일일이 inode를 알아내서 **debugfs**로 바꿔주어야 한다.

이제 모든 것이 잘 되었다면, 모든 것들이 삭제되기 전의 상태가 되어야 한다. 최소한 내가 이 글을 쓰면서 해본 테스트에서는 그랬다. 기억할 것은 반드시 2절에 나온 전제조건을 만족해야 한다는 것이다.

12. 참고문헌

Linux Ext2fs Undeletion mini-HOWTO, v1.3

- Aaron Crane

Design and Implementation of the Second Extended Filesystem, http://e2fsprogs.sourceforge.net/ext2intro.html (http://e2fsprogs.sourceforge.net/ext2intro.html)

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

Kernel Source for Linux 2.2.16

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