Ext2fs 복구 미니-하우투

By Aaron Crane <*aaronc@pobox.com* <mailto:aaronc@pobox.com>>

1997년 1월 18일 토요일, v1.0

번역자 : 박은영 <*pey1@nownuri.net* <mailto:pey1@nownuri.net> >

번역일 : 1997년 5월 8일 목요일

번역주 : 우선 리눅스를 제대로 쓰지도 못하는 초보가 그것도 짧은 영어 실력으로 하우투 번역에 참가하는 점을 부끄럽게 생각하며, ext2 에서 삭제된 화일의 복구 방법을 설명한 이 미니 하우투를 두 부분으로 나누어 올립니다.(1편: 처음 8장 / 2편: 9장 끝 이렇게 둘로 나눔) 만약 오역이 되어 수정할 부분이 있다면 서슴치 마시고 위에 적힌 제 주소로 메일 주시면 즉시 고치겠습니다.

목 차

제 1 절 들어가면서

이 미니 하우투 문서는 ext2 화일 시스템에서 지워진 화일을 복구하는 방법을 알려 대비하게 하기 위해 작성 되었다. 이 문서는 또한 우선적으로 어떻게 하면 화일 삭제를 피하는지에 대한 논의들에 한정되어 있다.

나는 이 글을 'rm' 으로 인해 문제가 있는 사람들에게 확실한 도움을 주려는 의도로 작성했지만 - 언젠가 여기 있는 약간의 정보로 위험을 면할 수 있을지도 모르는 일이니 - 아무튼 여러 사람들이 읽어 보기를 바란다.

이 문서는 일반적인 유닉스 화일 시스템에 대한 약간의 배경지식을 가정해 작성되었지만 대부분의 리눅스 사용자들에게 쉽게 이해되길 바란다. 만약 아주 초보자라면 리눅스 체제 하에서 화일을 복구하는 것은 당분간 기술적 지식과 끈기 등등이 꼭 필요하다.

적어도 화일이 저장된 raw device에 read access 없이 ext2 화일 시스템에서 지워진 화일을 복구하는 것은 불가능할 것이다. 일반적으로 이것은 루트여야만 한다는 것을 의미한다. 또한 'e2fsprogs' 페키지의 'debugfs'가 필요한데 이는 구분해서 설치했을 것이다. 내가 이 글을 쓰는 이유는 내 자신이 주로 루트에서 어리석고 비참하게 'rm -r' 명령을 사용한 경험에서 나온 것이다. 난 내게 필요한 97개에 달하는 JPEG 화일을 지웠었는데 다른 소스로부터도 거의 복구가 불가능 했었다. 약간의 도움되는 정보와 굉장한 인내로 손상되지 않은 91개의 화일을 복구 했었고, 나머지 다섯 부분도 아무튼 되찾았다. 단지 하나가 보기 불가능 했었는데 이마저 1024 바이트에 지나지 않는 부분만 잃어 버렸다.

이에 나는 더 나가서 화일을 삭제했을때 기대할 만한 복구 방법을 설명 하고자 한다.

제 2 절 어떻게 하면 화일을 지우지 않을까

이것은 MS-DOG 보다 Linux 에서 치명적이다. Win95를 포함한 MS-DOG는 일반적으로 삭제된 화일이 상당히 완벽하게 복구된다 – 심지어 이 운영 체제는 복구과정 대부분이 자동화된 유틸리가 있다. Linux 에서 이런 경우는 있을 수 없다. 그러므로 당신이 기억해야 할 최고의 법칙이며 제일의 법칙은 : 어떤 일이 있어도

***** 백업을 하는 것이다 ***** 이게 말하기는 좋은 것인 줄 안다. 단지 나는 모든 리눅스 사용자들이 나가서 유용한 백업 장치를 구입하고 적절한 백업 계획을 세워 백업에 충실하라는 것을 권한다. 이에 대한 자세한 사항은 Frisch(1995)를 참고해라. 백업이 않되면 어떻게 할 것인가?

중요 화일의 permissions을 440이나 그보다 적게 둬라: 만약 자신이 access하기 거부한다면 'rm'은 지우기 전에 확실한 확인이 필요하다. (그러나 나는 'rm -r'로 디렉토리 삭제시 우선 프로그램을 중단하고 확인한 다음 'rm -rf' 명령을 재실행 할 것을 발견했다)

선택된 화일을 위한 좋은 방법은 숨은 디렉토리 안에 하드 링크를 만드는 것이다. 언젠가 나는 /etc/passwd 를 사고로 인해 반복적으로 지우고 그것 때문에 시스템의 반을 날린 sysadmin 에 대한 이야기를 들은 적이 있다. 이를 고치는 것은 루트에서 다음과 같이 하면 된다.

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

완벽히 접근해서 화일을 지우는 것은 사실상 많은 노력이 필요하다. 만약 다음과 같이 하고

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

그리고 나서

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

이렇게 하면 위에서 지웠던 것이 복구될 것이다. 물론 이것은 화일에 덮어쓰기를 했을때는 소용이 없다.그러므로 아무튼 백업을 해야 한다. 몇몇 사람들은 일명 쉘이나 함수에서 'rm'을 'rm -i'로 만들것을 주장한다.('rm -i'는 모든 삭제할 화일 확인해 묻는 것이다) 개인적으로 나는 소프트웨어는 부주의 하게 운영하지 않을 것이다 라는 입장에 있지 않아서 그렇게 하지 않는다.

또한 머지 않아서나 나중이나, 당신이 한사람만의 모드를 운영하건, 다른 쉘을 사용하건 혹은 다른 기계를 사용하건 어디서나 당신의 'rm' 함수는 존재하지 않는다는 것이 문제점이다. 만약 화일 삭제시 일일히 확인해 물어주기를 바란다면은 삭제 화일이 너무 많아서 지정할 수 없고 어디에 있는지 잊기 쉽다.

'rm' 명령의 어떠한 변형 없이 사용하는 것을 제외한 좀더 나은 해결책은 재생 가능한 삭제 페키지를 사용하기 시작하는 것이다. 이에 대해 자세한 것은 Peek, et al(1993)을 참고하라.

제 3 절 얼마 만큼의 복구를 기대할 수 있을까?

그건 나름대로이다. Linux와 같은 High-quality, Multi-tasking, Mult-user 에서 화일 복구시 문제는 언제 누가 디스크에 기록하기를 원하는지 모르는 것이라는데 있다. 그래서 운영체제에서 화일 삭제를 명령시, 이는 화일이 새화일의 장소 할당을 원할때 공평하게끔 블럭이 사용 된다고 가정한다. 일반적으로 컴퓨터 사용자가 많아질수록 성공적으로 화일을 복구할 수 있는 가능성은 적어진다.

또한 디스크 파손은 화일 복구를 용이하게 하는 효과를 가져올 수 있다. 만약 파티션을 포함해 화일이 지워진 거라면 너무 파손된 것이여서 모든 화일을 읽는 것은 거의 불가능하다.

만일 컴퓨터가 자신의 것과 같이 실제로 sing-user workstation 이라면 치명적인 삭제가 아닐시 상당히 높은 비율의 복구를 기대한다. 나는 손상되지 않은 바이너리 화일의 거의 94%를 회복 시킨다. 당신이 80%나 그 이상을 가지고 있다면 자신에게 꽤 기쁨을 느낄 것이다.

제 4 절 그럼 화일을 어떻게 복구 시키나?

대체적으로 화일 복구 절차는 raw partition device 에서 데이타를 발견해 그것을 다시 볼수 있게 해서 운영체제로 가져가는 것을 의미한다. 이것을 하는데에는 기본적으로 두가지 방법이 있다 :

하나는 삭제된 inodes가 가지고 있는 'deleted' flag 를 존재하는 화일 시스템으로 변경해 그 자료들이 기적적으로 그곳에 오길 바라는 것이다. 속도는 늦지만 안전한 다른 방법은 파티션에 있는 자료들을 계획을 세워 새로운 화일에 옮겨 쓰는 것이다.

4장 6장에 걸쳐 자료 복구를 시작하기 전에 이해해야 할 단계들을 상세히 설명했으며 7장 10장에 걸쳐서는 화일 복구법을 설명하였다.

제 5 절 Unmounting the Filesystem

어떠한 방법을 선택하는지에 관계 없이 복구의 첫째 단계는 삭제된 화일을 포함한 화일 시스템을 unmount 하는 것이다.

mount된 화일 시스템에서 아마도 실수할 것이 틀림 없다는 어떤 주장도 강하게 부인한다. 이 단계는 화일이 삭제된걸 안 후에 가능한한 빨리 수행되어야 한다.

가장 간단한 방법은 다음과 같이 하는 것이다: 지워진 화일이 /usr partition 에 있다 가정하고

```
# umount /usr
```

그러나 /usr 에 이용할 수 있는 것을 두길 원할수도 있으므로 이것을 read-only로 remount 한다:

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

만약 root 에서 화일을 지웠다면 /etc/mtab 에 쓰여지는 것으로부터 mount를 막기위해 '-n' option 을 추가해야 한다.

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

이 모든 것을 개의치 않은 filesystem을 사용한 다른 과정이 가능하다. (이는 'Resource busy' 같은 에러 메세지로 unmount 실패를 초래한다) 이것은 주워진 화일이나 mount point를 사용한 어떤 과정에게라도 신호를 보내는 프로그램이다: 'fuser'.

/usr partition 에 다음과 같이 해보자.

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

이것은 그 과정에 관련된 것을 기록한다. 그것들 모두 중요하지 않다고 가정하고 다음과 같이 명령해 SIGKILL 각각의 과정을 보낸다.

```
    # fuser -k -v -m /usr
 혹은 예를 들어
    # fuser -k -TERM -v -m /usr
 각각 SIGTERM 을 준다.
```

제 6 절 직접적으로 inodes 를 바꿀 준비를 한다.

나의 충고는 이 방법을 쓰지 말는 것이다.. 나는 정말 이 일을 하기에 수준이 낮은 화일 시스템에서 작동하는 것을 현명하다고 생각하지 않는다. 또한 이는 미덥게도 단지 각 화일의 첫번째 12 blocks만 복구할 수 있는 문제점이 있다. 그러므로 긴 화일을 복구 할려면 아무튼 다른 방법을 찾아야 할 것이다. (11장을 봐도 부가적인 정보이다.)

여러분이 이 방법으로 꼭 해야만 한다고 느낀다면 raw 파티션 자료를 다른 파티션에 복사한 다음 이것을 loopback 를 사용해 mount 해라.

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

그러나 이것은 'mount'의 최신 버젼이 필요하다. 화일 시스템을 완전히 파괴했을때 loopback 을 사용한다는 의미는 모두 raw 파티션을 복사해서 다시 시작해야만 한다는 것이다.

제 7 절 다른곳에 자료를 쓸(write) 준비를 한다.

여러분은 어딘가에 있는 구출된 파티션을 확실하게 만들 필요가 있다. 잘만되면 당신 시스템은 아마 root, /usr, /home 에 몇개의 파티션을 가지고 있다. 이들중 고르면 아무 문제도 없을 것이다: 단지 이중 하나에 새로운 디렉토리를 생성한다.

오직 root 만 가지고 있고 root에다가 모든것을 저장 했다면 좀더 다루기 힘들 것이다. 당신은 MS-DOG 나 Windows 파티션을 가지고 있고 그것을 사용하는가? 혹은 커널에 모듈처럼 ramdisk 드라이버를 가지고 있는가? 커널이 1.3.48 이상이면 ramdisk 를 사용해서 다음과 같이 해라.

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

이것은 2MB 의 ramdisk volume 을 만들고 이를 /mnt에 mount 하는 것이다. 경고: 만약 'kerneld'를 automatically load and unload kernel modules로 사용한다면 이것으로부터 어떤 화일도 복사하기 전까지는 비

휘발성 기억장치 위에 ramdisk 를 unmount 하지 말아라. 한번 이것을 unmount 하면 'knrneld'는 이것은 모듈을 unload 할수 있다고 가정하며, 이일이 일어나면 메모리는 커널의 다른 부분에서 다시 사용되고, 자료를 복구 하느라 걸린 모든 힘겨운 시간들을 잃어버리게 되는 것이다.

만약 어떤 새로운 superfioopy removable device 를 가지고 있다면 그 방법들은 파티션을 복구하는데 좋은 선택이다. 그렇지 않으면 당신은 플로피에 끝까지 충실해야 한다.

여러분이 필요한 다른 한가지는 파티션 device의 중앙으로 부터 필요한 자료를 읽을 수 있는 프로그램이다. 곤경에 처했을때 'dd'는 그 일을 할 것이지만, 600MB 파티션을 800MB 파티션으로 읽어들여 명령 하고, 'dd'는 읽기를 고집하지만 처음 600MB는 무시한다.이는 시간이 많이 걸린다. 방법은 파티션 중앙을 찾을 프로그램을 기록하는 것이다. 이를 'fsgrab'라 부르는데 uuencoded, gizpped tar file 에서 사용법, 설치법, 저작권 메세지와 함께 구할 수 있다. 다음 명령으로 이를 검색해 보자 :

```
    $ uuencoded < Ext2fs-Undeldtion-mini-HOWTO
    $ gzip -dc fsgrab.tar.gz | tar xf -
begin 644 fsgrab.tar.gz
,
end
```

이 프로그램이 완벽하게 작동되더라도 어떻게 이를 수행하는지는 책임질 수 없다는 점을 경고한다. warranty 에서 부족한 부분의 상세한 것은 GNU General Public Licence를 포함한 'COPYING' 화일의 'No warranty' 섹션을 봐라.

제 8 절 삭제된 inodes 를 찾는다.

다음 단계는 최근 inodes가 자유롭게 된 화일 시스템을 묻는 것이다. 이 일은 'debugfs'와 같이 해야 하는 일이다. 화일 시스템이 저장된 device에 이름을 주고 'debugfs'를 시작하자:

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

inodes를 직접적으로 변경하려면 기록 가능한 화일 시스템에 '-w' 옵션을 추가한다:

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

삭제된 inodes 를 찾는 'debugfs' 명령은 'lsdel' 이다. 자, 다음 명령을 프롬프트에 써보자.

```
     debugfs:  lsdel
```

디스크가 돌아가는 소리가 난 후 길다란 목록이 원하는 곳으로 운반된다. 그러면 이것을 어딘가에 복사해 저장하기를 원할 것이다. 'less' 옵션을 가지고 있다면 '-o' 옵션을 출력 화일 이름에 붙힐 수 있다. 그렇지 않으면 그 출력을 보낼 곳을 정해야 한다.

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

자, 삭제된 시간, 크기, 형태, 퍼미션 숫자, 소유자 등등에 근거를 두고 원하는 지워진 inodes 를 생각해 내야만 한다. 운좋으면 삭제된 inode 가 커다란 무리이고 5분전에 삭제된 것이므로 발견할 수 있을 것이다. 그렇지 않으면 세세한 목록을 통해 trawl 한다. 가능하다면 나는 복구하기를 원하는 inodes 를 프린트 하길 제안한다. 이는 위의 것을 더욱 쉽게 만들 것이다.

제 9 절 inodes 의 세부 과정을 얻는다.

Debugfs 에는 inode의 세부사항을 출력하는 'stat'명령이 있다. 이는 복구 목록에 있는 각 inode 를 발행한다. 예를들어 자주 사용하는 inode 번호가 148003 이라면 다음과 같이 해보자.

```
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
```

복구할 화일이 많다면 위의 것이 자동적으로 되길 바랄 것이다. 복구할 inodes 'lsdel' 목록이 'list'에 있다 가정하고 다음과 같이 해보자.

cut -c1-6 list | grep "[0-9]" | tr -d " " > inodes 이 화일의 'inodes'를 나중에 편리하게 하기위해 복구할 inodes의 번호를 줄마다 하나씩 포함해 저장했었다. 그리고 나서 다음과 같이 해보자.

sed 's/^.*$/stat <\0>/' inodes | debugfs /dev/hda5 > stats 여기서 'stats'는 모든 'stat' 명령의 출력을 포함한다.

제 10 절 데이터 블럭을 복구한다.

복구할 화일이 12 블럭보다 길지 않다면 이 모든 자료의 블럭 번호는 inode 에 저장되어 있다. 이는 'stat' 명령으로 즉시 inode 를 출력해 읽을 수 있지만, 'fsgrab'를 사용하면 더 쉽게 할수 있다. 8장에서 봤던 예를 다시 보자.

```
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
```

이 화일은 6개 블럭을 가지고 있다. 'recoverde.001' 이라는 새화일에 정확히 기록하기 위해 다음과 같이 해보자.

```
# fsgrab -c 2 -s 594810 /dev/hda5 > recovered.001 # fsgrab -c 4 -s 594817 /dev/hda5 >> recovered.001
```

끝부분의 'recovered.001'는 필요 없을 수도 있지만 별로 중요치 않다. 물론 하나 또는 그 이상의 관련된 블럭들이 덮여 쓰였을 가능성도 있다. 만약 그렇다면 운이 없는 것이니 그 블럭에 대해선 잊어 버려라.

문제는 화일이 12 블럭보다 길때 나타난다. 이때는 유닉스 화일시스템에 대해 약간은 알고 있어야 한다. 화일의 자료는 'block' 이라 불리는 집합체에 저장된다. 그리고 이 블럭들은 순차적으로 번호가 매겨져 있다. 또한 화일에는 소유자, 퍼미션, 종류등의 정보가 보관되어 있는 'inode' 라는 것이 있다. 이러한 블럭과 inode는 순차적으로 번호가 매겨져 있고 다른 순서를 갖는다. 그리고 디렉토리는 화일 이름과 inode 번호로 구성 된다.

퍼미션 정보 등등과 마찬가지로 inode 또한 화일 데이터의 위치를 보유한다. 처음 12 데이터 블럭은 inode 그 자체에 저장되는데 이를 direct blocks라 한다. 그러고 나면 inode 는 'indirect blocks'을 포함 하는데 여기서 블럭은 부가적인 direct blocks 번호를 포함한다. 또한 inode는 indirect blocks 번호 목록을 포함한 두배의 indirect blocks 번호를 가지고, 또 이를 포함한 목록을 가진 세배의 indirect blocks을 가진다.

다시 읽어보자 : 이는 복잡하지만 중요하다. 자, 불행하게도 최근의 커널 수행은 화일 deletion의 모든 indirect blocks 이 0이다.그래서 12블럭 보다 긴 화일이라면 필요한 모든 블럭의 번호를 발견할 수 있다는것 조차 보증할 수 없다. 그러한 것을 알아낼 유일한 방법은 그 화일들이 조각나지 않았다고 가정 하는 것이다. 만약 산산 조각이 났다면 문제가 있다. 화일에 indirect block 이 필요하다면 조각나지 않은 것들은 다음과 같이 나타난다.

```
    #Blocks   Contents
    12        Data (direct blocks)
    1         Indirect block
    256       Data (blocks in the indirect block)
    1         Doubly indirect block
   [1         Indirect block of the doubly indirect block
    256       Data
   ] (256 times -- one for each indirect block in the doubly indirect)
    1         Triply indirect block
  [[1         Doubly indirect block
    256       Data
   ] (256 times -- one for each indirect block in the doubly indirect
      block of the triply indirect block)
  ] (256 times -- one for each doubly indirect block in the triply
     indirect block)
```

위의 것이 명확하기를 바란다. 화일이 단지 268 블럭 뿐이라면 이는 전적으로 가능하다. 이들은 12블럭 데이터이고 (이 번호는 inode 자체에 나타난다) indirect block 을 따르며 다른 256 제이터 블럭을 따른다. 만약 화일이 위의 것보다 길다면 두배의 indirect block 을 추가한다. 그 이후 각각의 부가적인 256 데이터 블럭에다 하나의 indirect block 과 256 데이터 블럭을 추가하는데 최대가 256 이다. 이보다 화일이 더 길다면 조각나 있지 않을 가능성은 거의 없지만 더 긴 화일에 그 조직이 이어져 나타날 수도 있다.

여러분의 블럭 크기는 모두 1024 bytes 이고 이는 표준값이다. 블럭이 이보다 크다면 약간의 번호들이 바뀌어 나타난다. 특별히: 각 블럭 번호가 4 bytes 딜다면 blocksize/4 는 각각의 indirect block 에 저장되는 블럭 번호이다. 그래서 매번 목차에 나타나는 256 이라는 번호를 blocksize/4 로 바꾸는 것이다.

제 11 절 inodes 를 직접 변경한다.

이 방법은 겉보기에 더 쉬워 보이지만 12 블럭 보다 길때와 맞먹을 수 없다. 복구를 원하는 각각의 inode 에 반드시 사용법을 하나로 해야하고 deletion 시간을 0으로 해야한다. 이 작업은 ' degugfs' 명령중 'mi' 에 의해 이루어진다. inode 14800 3으로부터 약간의 예를 출력해 보자.

```
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]
```

위의것은 deletion 시간을 0으로 하고 링크를 1로 맞춰 각각 다른 곳에서 압축해 되돌린 것이다. 설령 이것이 어리석게 보일 지라도 많은 화일을 복구해야 한다면 이 방법으로 대처할 수 있다.

[그런데 : 'mi' 출력이 inode에 있는 'Creation time'에 관련되어 있다는 건 거짓말이다. 사실 화일이 만들어질 때는 유닉스 화일 시스템에 영향을 미칠 수 없다. 'st_ctime' 'struct stat'구성은 'inode change time'에 관련되어 있다. 이는 inode 세부사항이 바뀌는 마지막 시점에서이다.] 변경된 inodes가 있다면 'debugfs' 에서 벗어나 명령할 수 있다.

e2fsck -f /dev/hda5 여러분은 도움되는 출력과 약간의 질문을 받을 것이다. 'summary information' 과 변경한 inodes에 관한 모든 잘문에 'Yes'라고 해라. 모든 질문에 'Yes'라 답하는 것은 좋은 방법이다. 'e2fsck'가 끝나면 화일 시스템을 remount 할 수 있다.

삭제된 화일은 화일 시스템의 /lost+found 디렉토리에 자리하고 있다. (파티션이 /usr 에 mount 됐다면 /usr/lost+found 를 봐라) 그것들은 inode 번호에 따라 이름이 불려진다. 여러분이 계속 해야 할 것은 그 화일들의 내용에 알맞게 이름을 생각해 내는 것이며, 그것들을 화일 시스템 구조에 알맞는 곳으로 되돌리는 일이다.

사실 저것들은 ’e2fsck’로 화일을 /lost+found 로 보내는 것중 하나이다. 여러분은 inode 에다 화일 시스템에서 링크를 만드는데 ’debugfs’를 사용 할 수 있다. ’debugfs’ ’link’ 명령은 inode 를 변경한 후 사용해라.

debugfs: link <148003> foo.txt

위의 예는 ’debugfs’ 가 최근의 디렉토리라 생각하는 곳에다가 ’foo.txt’ 라는 화일을 만드는 것이며 ’foo.txt’ 는 당신의 화일이 될 것이다. ’e2fsck’ 명령은 ’summary information’ 등등을 고칠때까지 계속 사용할 필요성이 있다.

제 12 절 이런것은 나중에는 더 쉽게 얻을 수 있을까?

그렇다. 사실 난 이것이 벌써 있다고 믿는다. 2.1.X 시리즈에 있는 최근 커널은 indirect blocks 을 0으로 하지 않는다고 들었다. indirect blocks 이 삭제시 피해를 입지 않고 남아 있는 또 다른 2.0.X 커널 제품을 제작한다는 얘기가 나오기 시작한 것이 1996년 12월이다. 리누스와 다른 커널 해커들은 이 한계를 극복했고, 손으로 inode 를 변경하는 나의 방법에는 많은 결함이 나타날 것이다.

제 13 절 이 과정이 자동적으로 되는 도구가 있는 곳은?

이 일이 일어난다면 바로 그곳에 있다. 불행히 그곳에도 inode 변경 기술 메뉴얼과 같은 문제점이 있으리라 생각한다 : indirect blocks 은 복구가 불가능하다. 그러나 이는 머지않아 더이상 문제가 되지 않을 것이며, 지금 이러한 프로그램이 나올 가능성이 있다고 본다.

누군가가 net 에서 Scott D. Heavner 의 ’lde’ 를 언급했었다. 솔직히 나는 이것을 자동적으로 화일이 복구되는 도구로 보지 않는다. 이는 ’debugfs’ 로 화면을 가득 채우기는 하지만, 확실한 화일의 종류를 정밀 검사하는 능력과 같은 특징을 가지고 있다. 또한 이것은 xia와 minix 화일 시스템과 함께 작업을 하는데 이러한 이유들로 당시에 이것을 잘 팔리게 한것 같다. URL : *sunsite.unc.edu/pub/Linux/system/Filesystems/lde-2.2.tar.gz* <ftp://sunsite.unc.edu/pub/Linux/system/Filesystems/lde-2.2.tar.gz> 실제로 작업은 GNU Midnight Commander,‘mc’ 처럼 들린다. 이는 full-screen 화일 조정 도구이며, ’NC’라 알려져 있는 MS-DOG full-screen 화일 조정 도구인 AFAIK에 근거를 둔 것이다. ’mc’는 리눅스 콘솔과 xterm 에서 마우스를 지원하며 tarfile에 ’cd’ing 하는 것과 마찬가지로 실질적인 화일 시스템을 제공한다. 이들 실질적 화일 시스템은 ext2 undeletion 이다. 이것은 매우 번거롭게 들리고 내 자신이 프로그램을 사용할 수 없다 여겨 난 오래된 쉘 명령을 오히려 좋아한다. 분명히 하나는 ‘–with-ext2undel’ 옵션으로 프로그램을 배열한다. 또한 라이브러리와 ’e2fsprogs’ 패키지와 오는 화일을 포함한 발전이 필요할 것이다. 나는 일단 ‘cd undel:/dev/hda5’ 라 부를수 있는 만들어진 프로그램을 수집하고, 삭제된 화일의 ‘directory listing’ 을 취했다.

가장 마지막 버전은 아마 3.2.11 이다. 커널 자체와 마찬가지로 개발버젼은 해커가 아닌 사람에게 권하지 않는다. 이것은 URLs로 부터 이용할 수 있다. *sunsite.unc.edu/pub/Linux/utils/file/managers/* <ftp://sunsite.unc.edu/pub/Linux/utils/file/managers/> (may be an older version) *ftp.nuclecu.unam.mx/linux/local/* <ftp://ftp.nuclecu.unam.mx/linux/local/> *ftp.nuclecu.unam.mx/linux/local/devel/* <ftp://ftp.nuclecu.unam.mx/linux/local/devel/> (unsupported, developmental, alpha versions) *ftp.cvut.cz/pub/mc/* <ftp://ftp.cvut.cz/pub/mc/> (European mirror of public and alpha releases) 웹 사이트는 다음과 같다 *stekt.oulu.fi/~jtklehto/mc/* <http://stekt.oulu.fi/~jtklehto/mc/>

제 14 절 끝

```
 A.  Colophon
============
I intend to produce regular updates to this document as long as I have both
time and something interesting to say.  This means that I am eager to hear
comments from readers.  Could my writing be clearer?.  Can you think of
```

```
something that would make matters easier?  Is there some new tool that does
it all automatically?  Who *did* kill JFK?
Whatever.  If you have something to say, about this document or any other
subject, drop me a line on <<url url="mailto:aaronc@pobox.com" name="aaronc@pobox.com">>.

B.  Credits
===========
    "If I have seen farther than others, it is because I was standing
    on the shoulders of giants."
                    -- Isaac Newton
Much of this mini-Howto was derived from a posting in comp.os.linux.misc by
Robin Glover <<url url="swrglovr@met.rdg.ac.uk" name="swrglovr@met.rdg.ac.uk">>.  I would like to thank
graciously allowing me to rework his ideas into this mini-Howto.

C.  Bibliography
================
Frisch, ?een (1995), 'Essential System Administration', second edition,
    O'Reilly and Associates, Inc., ISBN: 1-56592-127-5.
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.

D.  Legalities
==============
MS-DOG and Windoze are not trademarks.  Any resemblance these names may
bear to those of any commercial software products is entirely coincidental.
Unix is not a trademark, and has not been for quite some time now.  Wake up
and smell the flowers!
The trademark status of the name 'Linux' is currently being contested by
lawyers, as a certain Walter R. Della Croce has made an apparently false
trademark registration for the term.  I however refuse to acknowledge the
name as anything other than public property of all net.citizens.  If Linus
had wanted a trademark on the name, he'd have taken one out himself.
This document is Copyright ?1997 Aaron Crane <<url url="aaronc@pobox.com" name="aaronc@pobox.com">>.  It
be freely redistributed in its entirety, including the whole of this
copyright notice, but may not changed without permission from the author or
from the Linux Documentation Project Co-ordinator.  Dispensation is granted
for copying small verbatim portions for the purposes of reviews or for
quoting: in these circumstances, sections may be reproduced in the presence
of an appropriate citation but without this copyright notice.
The author requests but does not require that parties intending to sell
copies of this document, whether on computer-readable media or on paper,
inform him or the Linux HOWTO Co-ordinator of their intentions.
The Linux HOWTO Co-ordinator is Greg Hankins <<url url="gregh@sunsite.unc.edu" name="gregh@sunsite.unc.
```

샤론 박(pey1)