

# UNDELETING FILES ON THE LINUX ext2 FILESYSTEM

## WITH debugfs & e2undel

Oliver Diedrich

This text describes the steps needed to recover the data of a file that was recently deleted. It is neither a complete usage instruction for the e2undel and debugfs tools nor does it explain the backgrounds of file undeletion on the ext2 filesystem.

**Attention: This will not work with ext3!**

### The first steps

First of all: Immediately **unmount** the file system the deleted file was located on. This minimizes the risk that the data of the deleted file are overwritten while taking steps to recover them. All data written to the file system containing the deleted file - either by you or by any other process running on your machine - might overwrite some of the data you want to recover!

If you **can't unmount** the file system, e.g. because the deleted file was located on your root file system, you should consider to shut down the computer, remove the hard drive and put it into another Linux machine. Probably, you will have to compile and install some software, such possibly destroying the data you want to recover.

I strongly suggest this option if either the deleted data are so important that you don't want to take any unnecessary risk to lose them, or if the file system has very few unused space available (less than 20 percent or less than 100 MByte), or if the file is larger than 48 kByte: If few free space is available on the file system, ext2 might choose to store new data in areas that were recently used by deleted files; and if the file is larger than 48 kBytes, its data may be distributed across the file system, such enhancing the risk of overwriting parts of the deleted data with new files. It is your decision; but you have been warned.

If you are in the very unpleasant situation that you neither can use your system without the file system containing the deleted file nor have another Linux machine available, I strongly suggest to create a **RAM disk** to build and install the necessary software. With kernel 2.4, it is quite easy: Just do a `mount tmpfs /mnt -t tmpfs`. This will create a dynamically growing RAM disk on directory /mnt.

### Using debugfs

Especially if you can't unmount the file system containing the deleted data, **debugfs** is a less comfortable, but usable alternative if it is already installed on your system. (If you have to install it, you can use the more comfortable e2undel as well.) Just try a

```
/sbin/debugfs device
```

Replace *device* by your file system, e.g. `/dev/hda1` for the first partition on your first IDE drive. At the "debugfs:" prompt, enter the command

```
lsdel
```

After some time, you will be presented a list of deleted files. You must identify the file you want to recover by its owner (2nd column), size (4th column), and deletion date. When found, you can write the data of the file via

# UNDELETING FILES ON THE LINUX ext2 FILESYSTEM WITH debugfs & e2undel

Oliver Diedrich

```
dump <inode_number> filename
```

The inode\_number is printed in the 1st column of the "lsdel" command. The file filename should reside on a different file system than the one you opened with debugfs. This might be another partition, a RAM disk or even a floppy disk.

Repeat the "dump" command for all files that you want to recover; then quit debugfs by entering "q".

## Using e2undel

e2undel works in a similar way, using the same deleted file detection routines, but provides a somewhat friendlier user interface and some additional help to identify the files you want to recover.

## Building e2undel

**Download** the sources from [sourceforge.net/projects/e2undel](http://sourceforge.net/projects/e2undel). Don't save the tgz file found there on the file system with the deleted file! (You should have unmounted it, do you remember?) Use a different file system or a RAM disk.

Now **build** e2undel: Untar the tgz file (`tar xzf e2undel-nnn.tgz`), change to the resulting e2undel directory, and do a simple make in that directory. If everything works fine, this will result in an executable e2undel binary. Again, even if it annoys you, this old warning about file systems containing deleted files that are not unmounted (just skip to the next paragraph if you are not concerned): Building the program will create some object files in the e2undel directory and some temporary compiler files in /tmp. Each of these files could overwrite the data you want to recover...

If the **compiler complains** about missing header files or unknown variable types, you probably lack the e2fsprogs include files. They usually are included in your Linux distribution, hidden in a package named e2fsprogs-devel, ext2fs-devel or something similar. Simply install this package from your distribution, or from the web. Attention: This rpm will install a lot of files in your /usr file system.

If this does not work for you or your system is not rpm based, you have to download the source code of Ted Ts'o's Ext2 Filesystem Utilities, untar the tgz file, build and install at least the ext2fs library contained within, and change the path to the ext2 related header files in e2undel's e2undel.h file. Attention: Just the tgz file contains about 1000 files, summing up to 6 MByte (without even having started the compiler). If you have to install and build the ext2fs library on the file system containing your deleted file: This is the point you seriously should think about doing all the build stuff on another machine.

## Starting e2undel

The synopsis of e2undel is

```
e2undel -d device -s path [-a] [-t]
```

# UNDELETING FILES ON THE LINUX ext2 FILESYSTEM

## WITH debugfs & e2undel

Oliver Dierich

-d: file system where to look for deleted files (e.g., /dev/hdb1 for the 1st partition on the 2nd IDE drive) -s: directory where to save undeleted files -a: work on all files, not only on those listed in undel log file -t: try to determine type of deleted files w/o names, works only with -a

The -a option always must be given in your situation, and the -t option will help you a lot in finding the files you want to recover.

### An example

Suppose, your system uses two partitions, one (let's say, /dev/hdb5) is mounted on /, the other one (let's say, /dev/hdb6) is mounted on /home. You as user foo just deleted a file in your home directory and noticed your error. You have logged out, logged in as root and unmounted the /home partition.

```
e2undel -d /dev/hdb6 -s /tmp -a -t
```

will scan your home partition for deleted files, and present you a table like this:

```
user name | 1 <12 h | 2 <48 h | 3 <7 d | 4 <30 d | 5 <1 y | 6 older
-----+-----+-----+-----+-----+-----foo | 10 | 72 | 33 | 28 | 384 | 0 bar | 0 | 83 | 48 |
          35 | 114 | 0
```

Select user name from table or press enter to exit:

Enter your user name ("foo"), then choose the time interval when you deleted the file (1: within the last 12 hours; 2: more then 12, but less then 48 hours ago; 3: more than 2, but less than 7 days ago; etc). You will get a list similar to this:

inode size deleted at name

|        |         |     |  |
|--------|---------|-----|--|
| 184219 | 7240    | Dec | 9 18:14 2001 * data                                      |
| 184231 | 2192    | Dec | 9 18:14 2001 * ASCII C program text9 18:28 2001 *        |
| 184233 | 42024   | Dec | application/x-gzip9 18:28 2001 * text/html9 18:28 2001 * |
| 184235 | 1520    | Dec | ASCII text   |
| 184237 | 1780    | Dec |  |
| 184239 | 0 30032 | Dec | 9 16:18 2001 * empty9 16:18 2001 * data                  |
| 184246 |         | Dec |  |
| 202733 | 1344    | Dec |  |
| 217164 | 7404    | Dec | 9 12:51 2001 * ASCII English text9 17:58 2001 *          |
| 217165 | 13998   | Dec | image/jpeg9 17:58 2001 * ASCII text                      |

Select an inode listed above or press enter to go back:

Choose the file you're interested in by size, deletion data, and file type (displayed in the "name" column). After entering the inode number, the file is automatically named after its inode number and the file type

# **UNDELETING FILES ON THE LINUX ext2 FILESYSTEM**

## **WITH debugfs & e2undel**

**Oliver Diedrich**

and stored in the directory you entered with the "-s" option when starting e2undel (e.g., /tmp/inode-217165-ASCII\_text). Repeat for all files you want to recover, then press Enter two times to go back to the table and then exit e2undel.

The file type is determined by the mechanism the file(1) program uses. "data" means that no specific file type has been found: Either, it is a file type not known by the program, or the file contains some random binary data.

The data of files printed in red are partly overwritten. e2undel will give its best, but expect the resulting files to be corrupted. However, a large text file might be of use even if it partly contains garbage.