

## Recovering Deleted Files within the Linux Ext2 Filesystem

Contributed by LE Webmaster  
Wednesday, 26 January 2005

Lost those files accidentally? Those horror stories of logging in as root and losing things you need coming true? Then, read this article. You'll learn how-to undelete a file, or files, in a sense. One morning, I meant to clean out my webalizer/ directory, which was located in my website's directory. I could've sworn I was in the right directory, but, nope I wasn't. I was actually one step up from the webalizer directory, the directory with my whole web-page. Uh-oh, you're probably thinking, and yes, it was a disaster. I did a, `rm \*` and yep, erased my whole webpage. I was bummed after all the work I put into it. After all, my backups were not as current as I hoped them to be. Then, I had an idea.

{mosgoogle right}Oh-no! You lost that important document by accident! NO! You're probably thinking, I'll have to re-do it from scratch! Never fear, Linux comes with a utility called, `debugfs` With this, you can find a load of information, including the lost inodes and lost files of your hard drive. Time to go exploring! Straight of the Linux man pages, SYNOPSIS debugfs [-b blocksize] [-s superblock] [-f cmd\_file] [-R request] [-V] [[-w] [-c] [-i] [device]] DESCRIPTION The debugfs program is a file system debugger. It can be used to examine and change the state of an ext2 file sys- tem. device is the special file corresponding to the device containing the ext2 file system (e.g /dev/hdXX).

```
[root@darkstar]# /sbin/debugfs /dev/hda1 debugfs 1.22, 22-Jun-2001 for EXT2 FS 0.5b, 95/08/09 debugfs: So, you're at
that debugfs: prompt, what now? Type in, '?' or "help" for commands, or just open up man debugfs. debugfs: lsdel [...]
32614 1000 100664 503 1/ 1 Sun Jun 2 09:06:42 2002 32615 1000 100664 118 1/ 1 Sun Jun 2 09:06:42 2002 32616
1000 100664 119 1/ 1 Sun Jun 2 09:06:42 2002 32617 1000 100664 117 1/ 1 Sun Jun 2 09:06:42 2002 32618 1000
100664 795 1/ 1 Sun Jun 2 09:06:42 2002 32619 1000 100664 5250 2/ 2 Sun Jun 2 09:06:42 2002 32620 1000 100664
1415 1/ 1 Sun Jun 2 09:06:42 2002 32621 1000 100664 4013 1/ 1 Sun Jun 2 09:06:42 2002 32622 1000 100664 118 1/
1 Sun Jun 2 09:06:42 2002 32623 1000 100664 887 1/ 1 Sun Jun 2 09:06:42 2002 32624 1000 100664 641 1/ 1 Sun Jun
2 09:06:42 2002 32625 1000 100664 961 1/ 1 Sun Jun 2 09:06:42 2002 450581 0 100600 504 1/ 1 Sun Jun 2 10:05:56
2002 64450 0 100600 951 1/ 1 Sun Jun 2 10:10:04 2002 675842 0 100600 6 1/ 1 Sun Jun 2 10:10:42 2002 675870 1001
100600 2451 1/ 1 Sun Jun 2 10:10:42 2002 32572 0 100600 1094 1/ 1 Sun Jun 2 11:10:48 2002 64329 0 100600 1094
1/ 1 Sun Jun 2 11:10:48 2002 64330 0 100644 5 1/ 1 Sun Jun 2 11:11:01 2002 450275 1000 100600 1112 1/ 1 Sun Jun
2 11:48:26 2002 740012 0 100600 1753 1/ 1 Sun Jun 2 16:09:26 2002 740010 0 100600 78 1/ 1 Sun Jun 2 16:09:57
2002 740011 0 100600 1481 1/ 1 Sun Jun 2 16:09:57 2002 740013 0 100600 1753 1/ 1 Sun Jun 2 16:09:57 2002
450273 0 100660 1024 1/ 1 Sun Jun 2 16:18:19 2002 450274 0 100660 488 1/ 1 Sun Jun 2 16:18:19 2002 debugfs:
Hmmm, looks cool, doesn't it? `lsdel` list's all the deleted files that are not _that_ old, or that have not been overwritten
yet. It should pipe the information to your favorite pager ($PAGER environment variable). The first column is the inode
number, the second column is the userid of the person who owns the file, the third is the mode of the file, fourth is the
size of the file, fifth is the amount of blocks the file is on, and the last one, the time deleted. 3/ Dumping Now that you've
gotten some inodes, it's going to be pretty hard to tell which file is your's, but, if you deleted it only a couple minutes ago,
it should be at the bottom, also, the fifth column is your friend. :) So, you've gotten the inode of you're file (or you're just
picking one's outta the blue...) and you want to see the contents of it. The cat and dump commands work for a file of [...]
debugfs: Whoa! That was my file! Is there a way to have the contents go to a file _on another partition_? Yes, of course.
debugfs: dump /home/mikecc/recovered_32611 debugfs: NOTE: You _need_ the greater than, and less then signs when
cat'ing OR dump'ing the inode. There, you did it! You save your file! Now look in, /home/mikecc/recovered_32611
debugfs: quit To quit. `man debugfs` for the man page of debugfs. Lots more stuff you can do. For files larger than 12
blocks, it pays to know a little about how UNIX filesystems are structured. A "block" is a storage place for a file's data.
These blocks may be numbered sequentially. A file also has an, "inode", which is the place where information is kept.
Such information is, owner, permissions, and type. Liek blocks, inodes are numbered sequentially also, but, they may
have a different sequence. A directory consists of a name of the file, and an inode. Even with this, it's still impossible for
the kernel to find the data to corresponding directory. So, the inode also stores the location of a file's data blocks, as
follows: 1.) The first 12 block numbers are stored directly in the inode; these are sometimes referred to as the "direct
blocks." 2.) The inode contains a block number of what's called an, "indirect-block." An indirect-block has the block
numbers of 256 additional data blocks. Block's do not stop at indirect-block, they can go on for a "doubly indirect-block,"
"triply indirect-block," and so on.. Once you're past 12 blocks, there is no guarantee that you can all the block numbers
you need, let alone their contents. Assuming the file is not fragmented, these are some of the layouts for blocks. Block
numbers, 0-12 are stored directly into the inode. Block numbers, 13-268 count for one indirect block. and so on... 4/
Modifying Inode's Directly For each inode you want to recover, you must set the usage count (link count) to one, and the
deletion time to 0. You can do this with the `mi` command to debugfs. Here is some sample output. Also, you must
open up debugfs with the '-w' option so you can write to the filesystem, instead of it's default, read-only. /sbin/debugfs -w
/dev/hda1 debugfs 1.22, 22-Jun-2001 for EXT2 FS 0.5b, 95/08/09 debugfs: mi Mode [0100664] User ID [1000] Group ID
[102] Size [1515] Creation time [1023034002] Modification time [1016411327] Access time [1023033844] Deletion time
[1023034002] 0 Link count [0] 1 Block count [8] File flags [0x0] Generation [0x5239d7] File acl [0] High 32bits of size [0]
Fragment address [0] Fragment number [0] Fragment size [0] Direct Block #0 [34814] Direct Block #1 [0] Direct Block #2
[0] Direct Block #3 [0] Direct Block #4 [0] Direct Block #5 [0] 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] debugfs: I pressed enter on almost all of these to get the defaults, except on "Deletion time [1023034002]"
where I set it to 0 and "Link count [0]" where I set it to 1. To complete your modified inode, [root@darkstar]# e2fsck
/dev/hda1 And you're done! Another fun way of recovering a file. 5/ Other Tools Yes, there are tools out there to undelete
stuff for you, but, this is way more fun in my opinon. Some of the tools that I know of are, e2undel
```

(<http://e2undel.sourceforge.net>) LDE (<http://lde.sourceforge.net>) e2undel requires you to un-mount the disk you lost files on, which was very inconvenient for me at the time. LDE is pretty good if you know the inode numbers, but the UI takes a bit of getting used to. Other than those, I don't know of any others.