Repairing Ext2/3 Filesystems On-the-Fly

Last Gasp Data Rescue

You can feel your blood run cold when you suddenly discover that a file is no longer accessible, especially if it happens to be stored on a remote access root server. But unfortunately, modern hard disks are no longer made for eternity, so it can happen to everyone. Read on for tips on how to survive such a crisis.

BY PATRICIA JUNG

Even hardened SCSI fans who tended to invest a few dollars more for SCSI hard disks just a few years ago may be tempted to opt for IDE storage today. IDE disks not only provide a far cheaper approach to solving that mass storage problem, they are also a lot quieter.

This leads to IDE being chosen for smaller server solutions. IDE disks are not built for eternity, even IDE disks from reputable manufacturers have been known to fail after just a years service.

Failure is often a gradual process: a bad sector here, and another one there … if you don't save your data regularly, pure luck will determine whether you notice this gradual decay. If you create a backup **cronjob**, such a job will issue a timely warning – as long as you read the messages cron sends.

If you use *rsync* [3] as your backup tool, the bad news will be something like:

```
building file list ... ➥
/home/pjung/article/LinMag0308:➥
 Input/output error
done
IO error encountered - ➥
skipping file deletion
```

An Input/Output error like the one shown for the directory in our example, has far-reaching consequences. No matter whether you attempt to list the directory content using *ls -al* or a similar command, or attempt to move the directory with the *mv* command, or try to display information on the directory using *stat*, or even launch a desperate attempt to remove the directory with the ***rm -rf*** syntax – the result will be the same old I/O error.

Deleting the directory and restoring from backups is not an option. There is no way to access the files in the directory, as a directory is in fact nothing more than a file itself, albeit a file that contains the names and **inodes** of the files in the current directory. If this information is missing, since your disk is

GLOSSARY

Cronjob: *The cron daemon [1,2] inspects a special job table, called a crontab, every minute, checking for user-defined jobs that indicate the programs and scripts the user wants to launch at a specific time. These tasks are referred to as cronjobs. By default, after completing a job, the cron daemon will notify the user responsible for the job by mail.*

rm -rf: *This is one of the most dangerous commands on a UNIX system. It will delete the directory supplied as an argument with any subdirectories below it, assuming the user calling the command owns the directory or has appropriate write privileges. As root can delete anything, this command can actually destroy a whole system.*

Inode: *For each file stored on a Linux filesystem, there is a special data structure containing information on the owner, the group, and privileges for the file, and most importantly a pointer to the physical positions on the hard disk where the file data is stored. The operating system uses this structure to manage the filesystem tree. Inodes themselves need to be stored – obviously on the hard disk. If the hard disk location where an inode physically resides is damaged, the operating system cannot access the file it references.*

Single user mode: *Runlevel 1 specifies a rescue mode without a GUI, network connections, or indeed the ability to log on as a non-privileged user. This mode is used by root to repair the system directly at the console.*

damaged at the point where the directory is physically store, the operating system will not know how to access the files, although they may not be physically damaged on your disk.

In other words, the filesystem is our patient. If everything turns out OK, a file system check should be able to restore our files to a point where we can access them temporarily, although this will not stop the gradual decay of your hard disk.

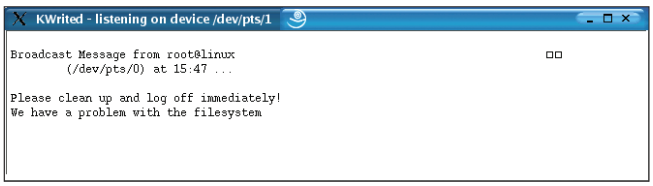

Figure 1: KDE opens up a window for the wall message

Rescue Mode

If you are sitting right in front of the computer, you will probably decide to boot to **single user mode**, by typing *single* at the boot prompt while the kernel is loading. You are then prompted for the superuser password:

```
Give root password to login:
```

If you are not sure which hard disk partition the damaged directory is on, you can now enter *cat /etc/fstab* to find out. Then pass this directory as an argument to the file system check utility, *fsck*. After completing the repair task, you will need to restart your computer, which hopefully will be back to normal again.

No console access

If the affected computer happens to be a root server, located at a provider's data center, you cannot simply boot to single user mode – direct access to the keyboard is not an option, so that leaves remote access via *ssh*. Provided the bad sectors have not affected partitions mounted below */* and */usr*, where system critical programs are stored, you still have a chance.

After logging on as *root* via *ssh*, the *mount* command without any options will indicate which hard disk partitions are mounted at which position of the filesystem. The affected files from our example are located in the */home* partition on */dev/hda6* for example.

```
/dev/hda6 on /home type ext3 ➥
(rw)
```

If you now launch *fsck* as *root*, a warning is issued – and you will want to take it seriously, see Listing 1. Your best option is to hit [n][Enter] to go back.

The answer is not as simple as unmounting the partition; after all, other non-privileged users may be logged on and currently accessing their home directories. If other users are working on the machine that needs repairing, *root* can use the *wall* command to display a message on the users' screens (Figure 1):

```
echo -e "Please clean up and ➥
log off immediately! ➥
\nWe have a problem with the ➥
filesystem" | wall
```

Users who choose to ignore this friendly warning will have to be removed forcibly. You can now enter the following command from a safe location in her own home directory below */root*:

```
fuser -mk /dev/hda6
```

The *-k* ("*k*ill") option kills any processes currently accessing files on the partition */dev/hda6*, which is currently mounted (*-m*). If *root* has used *su* to assume administrative privileges for a non-privileged account, this command will actually kill itself, as the parent process of *su* will access the non-privileged user's home directory.

The next task is to unmount the damaged partition using the *umount /dev/hda6* syntax, launch *fsck /dev/hda6* and hope for the best. Mere mortal system administrators will tend to type [y] when prompted by the tool – in fact, this is your only option.

After completing file system repairs, you can enter *mount /home* to remount and check the damaged location. If your luck holds, you will now discover that *fsck* has restored the damaged directory. If not, the only option left to you is to restore a backup.

Grave Robbing

If the backup is missing, or not up-to-date, *root* has one final option: using a text editor to go grave robbing in the */home/lost+found* directory. Files or parts of them that *fsck* was unable to reassign land in this directory, although the original name unfortunately is lost in the process. As long as you remember the original filename any "lost and found" files can be renamed manually and restored to their original location.

With luck, *lost+found* may contain files that can be repaired and restored using *fsck*. But not even *fsck* is capable of restoring data from a defective disk.

Once again, the answer is to be prepared, make backups, and perform file system checks periodically. For mission critical systems you might like to consider replacing hard disks "sooner" rather than "later". If a hard disk with mission critical data does start to fail, you can expect the disk to continue to deteriorate rather than improve. ■

Listing 1: Mounted partition repair

```
linux:/home/pjung # fsck /dev/hda6
fsck 1.28 (31-Aug-2002)
e2fsck 1.28 (31-Aug-2002)
/dev/hda6 is mounted.

WARNING!!!  Running e2fsck on a mounted filesystem may cause
SEVERE filesystem damage.

Do you really want to continue (y/n)?  no

check aborted.
```

INFO

[1] Patricia Jung: "Task creator at your service", Linux Magazine, Issue 6, March 2001, p 108ff

[2] Jürgen Jentsch: "Count down", Linux Magazine, Issue 22, July/August 2002, p 68ff

[3] Heike Jurzik: "Keeping in sync", Linux Magazine, Issue 28, March 2003, p 66ff