LINUX VOICE TUTORIAL

DDRESCUE: SALVAGE DATA FROM DAMAGED DISKS

MARK CRUTCH

How a GNU and a penguin rescued a bear from a broken hard drive and the clutches of the evil empire.

WHY DO THIS?

- Earn respect and admiration by recovering 'lost' data.
- Learn about disk images.
- Remind yourself that it's time to make another backup!

LV PRO TIP

Some distros' repositories contain both **dd_rescue** and **GNU ddrescue**, so double-check which one you're installing. Debian derivatives use **gddrescue** as the package name for the GNU version and **ddrescue** for Kurt Garloff's **dd_rescue** command.

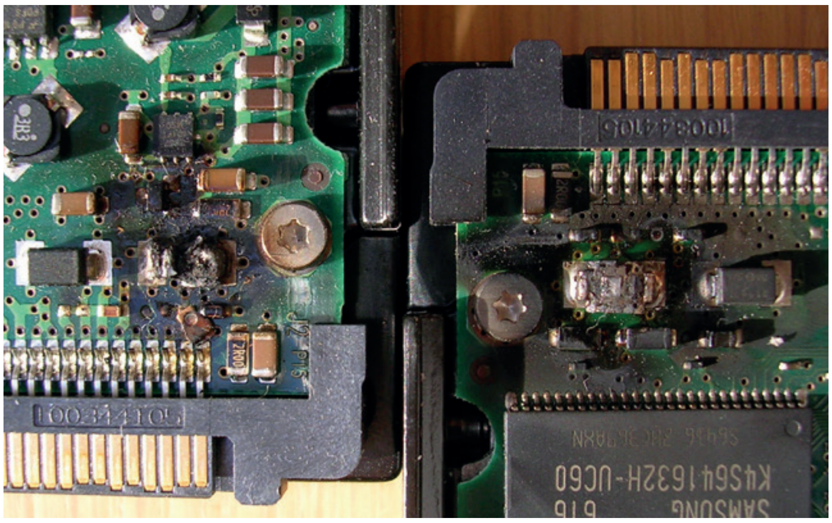
Come with us, gentle reader, on an exciting adventure into the world of data recovery. There will be loss and sadness, a hostage-taking mega corporation, a triumphant recovery, and lessons learned. Plus a comic book bear with an eye-patch. Welcome to the first draft of *Bertie Bear and the Disk of Corruption*...

Chapter 1: An artist's grief

It seemed like any other day for independent artist Andy Clift, as he booted his Apple Mac to continue work on the latest instalment of his comic book series, "Bertie Bear and the Dagger of a Thousand Souls". But within seconds of hearing the familiar start-up chime, his Apple turned sour. "Drive Error", it reported, refusing to proceed any further. Andy looked on in dismay, his mind churning with frantic thoughts; he knew he had backups of most of his files, but the latest drawings of Bertie's adventures had yet to make their way off the reluctant drive. He packed his iMac into a box and headed off to the nearest Apple store's "Genius Bar".

Too late he found that the "Genius Bar" is something of a misnomer. The staff there are better trained than most sales assistants, but it transpires that there's not even an IQ test required, let alone membership of Mensa, before a keen employee can be promoted to the position. Suffice to say that the "genius" failed to demonstrate any advanced skills beyond the ability to send the machine to Apple's service centre for a new drive to be fitted. Resigned to never seeing his latest creations again, Andy made his way to one of the mockingly intact Macs in Apple's store, and proceeded to post to his blog.

The scorch marks show where a single bad PSU simultaneously destroyed the electronics of both these drives.



Chapter 2: A drive held hostage

Meanwhile, at Linux Voice HQ, a Raspberry Pi was grepping its way through the internet. It paused to parse Andy's message, before triggering the launch of a foam dart and the careering dance of an ice-cream tub with wheels. This was our signal to leap tigerish into action.

"Somebody on the internet's got a hard drive problem!" cried our illustrious leader, his fingers already walking over to a pile of Linux CDs as he contemplated his rescue plan.

"But it's only a Mac user," replied a cynical voice from the corner, rising on a column of solder fumes and flux.

"We're better than that!" came the response. "We should be prepared to help our fellow man wherever we can. Besides, it's a good opportunity to see how effective Linux's HFS+ support is."

So we contacted Andy to offer our meagre skills in trying to recover his work from the drive, just as soon as he received it back from Apple.

You might expect that, having paid Apple over £200 to replace the hard drive, the old disk would be returned with the refurbished machine. Instead Apple demanded a ransom (our term, not theirs) of an extra £90 for its return! At first Andy was reluctant to increase his spend to almost £300, but when we pointed out that his credit card details, usernames and passwords would soon be in the hands of some third party salvage company, he decided to pay the ransom and reclaim the drive. Any data we could get off it would be a bonus.

When the disk arrived our first step was a visual inspection. We've seen several drives rendered useless by bad power supplies, but with no obvious burn marks or charred components on the visible side of the drive's circuit board, we connected it to our recovery machine. Attaching it straight to the motherboard would give us the fastest transfer speeds, but unseen electrical problems would be more likely to damage the host machine. We chose, instead, to place it into an external drive caddy, to provide a little extra electrical insulation, at the expense of limiting the data transfer to USB2 speeds.

We plugged in the USB connector and to our delight the drive was picked up instantly, our Mate desktop promptly opening a window showing the drive's contents. This told us that the device was basically working, and we listened for the tell-tale sounds of mechanical issues emanating from the disk's moving components. We also took the opportunity to examine the structure of the drive using Mate's 'Disks' program and found that there were three partitions: 'boot', 'data' and 'recovery'.

With everything appearing – and sounding – as a good drive should, we guessed that the problem was down to a few bad sectors that had been enough to annoy OS X. With no idea where on the disk those bad sectors might lie, it would be foolish to simply copy the data using the normal desktop tools, so we opted to create a full disk image to work with. Now we ask you, honoured reader, to imagine the strumming of a harp and a wobbly fade effect, as we take a break in our narrative to switch to the flashback section...

Chapter 3: A tale of four DDs

As the soft sound of the harp dwindles to nothing, we find ourselves at the dawn of time. Okay, a little after the dawn of time, but still pretty early in the annals of history. And, of course, we mean shortly after the dawn of "Unix time" – 1 January 1970.

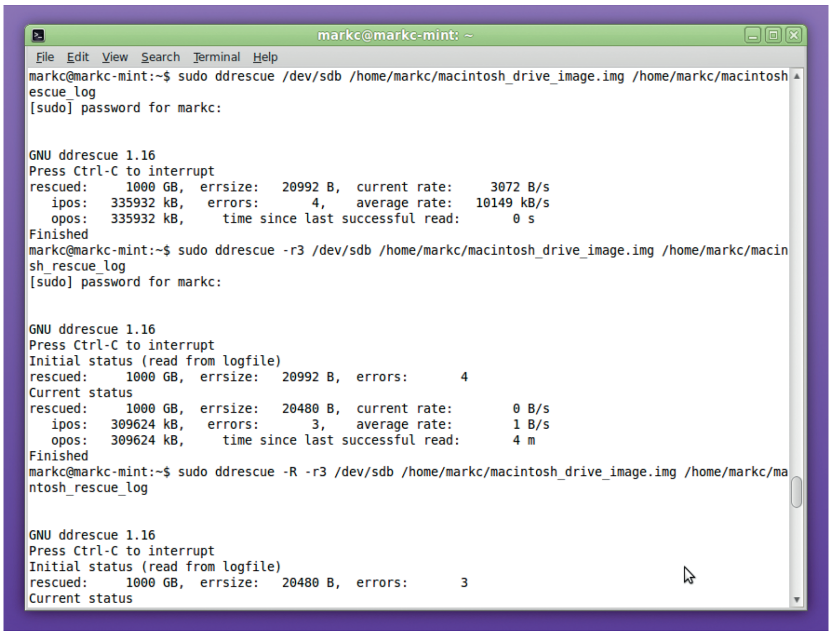
Back in those early days of Unix the **dd** command was created as a means of copying blocks of data between devices. You can read more about it in Linux Voice #08, but it falls into our story because it's a classic method for cloning from a drive to an image file. Because **dd** deals with blocks of data directly, it can be used to create a sector-for-sector clone of a drive even if it has foreign partitions. Unfortunately the way **dd** works makes it less than ideal for recovery tasks: it aborts on read errors, for example, which is definitely not what you want when dealing with a suspect drive.

Years later, building on the name and basic premise of **dd**, Kurt Garloff created **dd_rescue**, a tool specifically designed to recover data from failing drives. It has error handling that enables it to keep going where **dd** would fail. But that also means that it can take a very long time to image a drive with lots of read errors. To speed up this process, Valentin Lab created a *Bash* script called **dd_rhelp**, which optimises the way in which **dd_rescue** performs its job. When **dd_rescue** finds errors **dd_rhelp** makes it re-start at a later sector, hoping to find another good section of the drive, while keeping a log of the recovered parts of the drive so that it can work out which ones still need to be revisited. In this way it aims to recover the readable parts of the drive as quickly as possible, before going back to areas that may not yield any useful results.

It seemed that the combination of **dd_rescue** and **dd_rhelp** is just what we needed, but there was one more contender to consider: the confusingly named **ddrescue** (without the underscore).

Officially known as "GNU ddrescue", **ddrescue** aims to do the job of the **dd_rescue**/**dd_rhelp** combination, but in a single application. It keeps a log file of its progress, and attempts to speed through the readable data on a drive as quickly as possible, coming back to bad areas later. It can be stopped and then resumed at another time, and you can run it repeatedly without affecting previously recovered blocks. On our Linux Mint box, **sudo apt-get install gddrescue** was the right invocation to install it. With our weapon of choice in place, it's time to return to our adventure. Cue harp and wobbly fade as we head back to the present day...

"dd can be used to create a sector-for-sector clone of a drive even if it has foreign partitions."



```
markc@markc-mint:~$ sudo ddrescue /dev/sdb /home/markc/macintosh_drive_image.img /home/markc/macintosh
escue_log
[sudo] password for markc:


GNU ddrescue 1.16
Press Ctrl-C to interrupt
rescued:     1000 GB,  errsize:   20992 B,  current rate:     3072 B/s
   ipos:   335932 kB,   errors:       4,    average rate:   10149 kB/s
   opos:   335932 kB,     time since last successful read:       0 s
Finished
markc@markc-mint:~$ sudo ddrescue -r3 /dev/sdb /home/markc/macintosh_drive_image.img /home/markc/macin
sh_rescue_log
[sudo] password for markc:


GNU ddrescue 1.16
Press Ctrl-C to interrupt
Initial status (read from logfile)
rescued:     1000 GB,  errsize:   20992 B,  errors:       4
Current status
rescued:     1000 GB,  errsize:   20480 B,  current rate:        0 B/s
   ipos:   309624 kB,   errors:       3,    average rate:        1 B/s
   opos:   309624 kB,     time since last successful read:       4 m
Finished
markc@markc-mint:~$ sudo ddrescue -R -r3 /dev/sdb /home/markc/macintosh_drive_image.img /home/markc/ma
ntosh_rescue_log


GNU ddrescue 1.16
Press Ctrl-C to interrupt
Initial status (read from logfile)
rescued:     1000 GB,  errsize:   20480 B,  errors:       3
Current status
```

The first pass took 28 hours. Thankfully the subsequent runs only took minutes.

Alternative tools

During this little adventure we used the Mate *Disks* application. This was formerly known as *Gnome Disk Utility* or *Palimpsest*, and is available in most Gnome-derived desktop environments. If you're using a different environment you can still do everything in the article, but you'll need an alternative set of tools.

For simply looking at the partition structure of a disk or mounted image, the venerable **fdisk** command line tool is present on almost all distributions, as is the GNU **parted** application. If you prefer a GUI then *GParted* is a great GTK front-end to **parted**, or you might prefer the *KDE Partition Manager*, which also uses GNU's **libparted** library under the hood.

Mounting a partition within a full disk image is a little more tricky, with a general lack of GUI tools. It is possible to manually calculate the start position of your partition, then use it as an offset to the **mount** command. A less masochistic option is the **kpartx** utility, which can list partitions in a disk image and mount them all for you, without the need for maths.

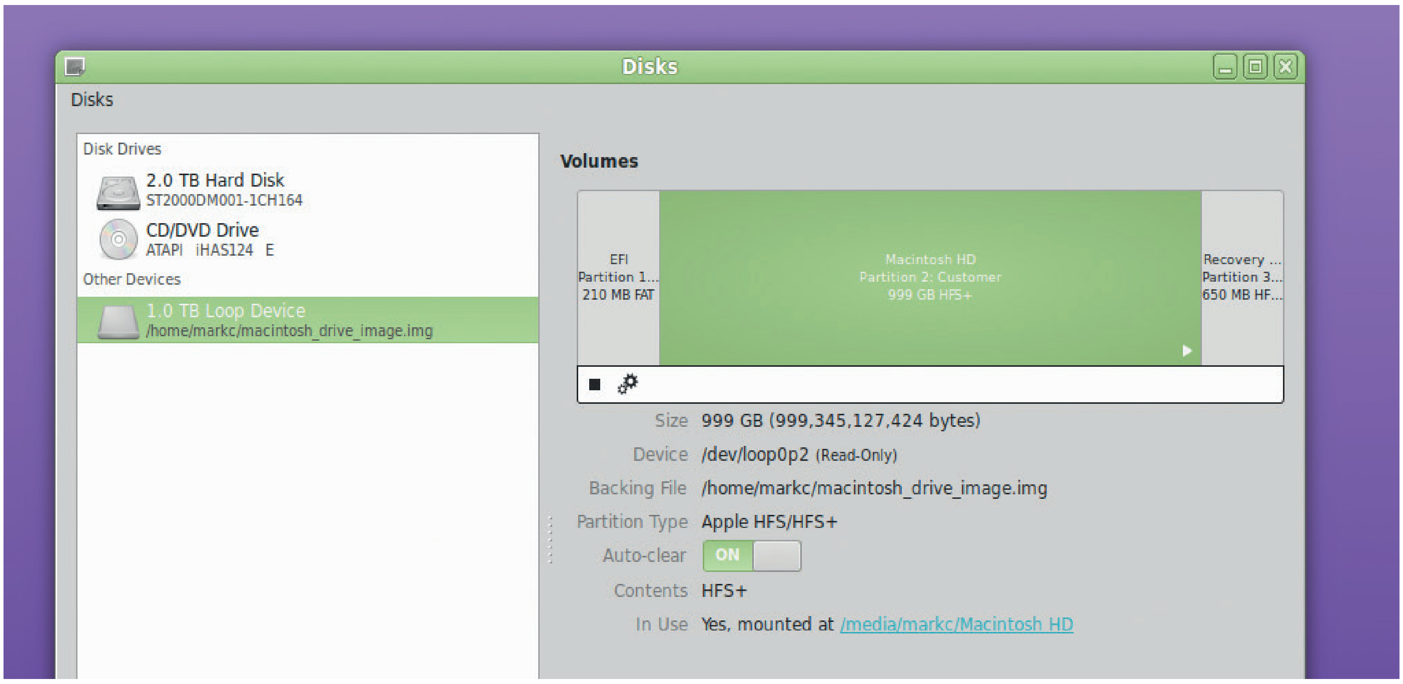
```
kpartx -av hard_drive.img
```

This will mount all the partitions to devices under **/dev/mapper** in the filesystem – see the **kpartx** man page for more details. Once you're done remember to delete the partition mapping and unmount the image:

```
kpartx -d hard_drive.img
```

LV PRO TIP

Use Ctrl+C to stop **ddrescue**. Provided you use a log file you can just start it again later and it will resume from where it left off. This even works if your machine crashes mid-recovery!


Disks
Disks
Disk Drives
2.0 TB Hard Disk
ST2000DM001-1CH164
CD/DVD Drive
ATAPI iHAS124 E
Other Devices
1.0 TB Loop Device
/home/markc/macintosh_drive_image.img
Volumes
EFI Partition 1… 210 MB FAT
Macintosh HD Partition 2: Customer 999 GB HFS+
Recovery … Partition 3… 650 MB HF…

Size 999 GB (999,345,127,424 bytes)
Device /dev/loop0p2 (Read-Only)
Backing File /home/markc/macintosh_drive_image.img
Partition Type Apple HFS/HFS+
Auto-clear ON
Contents HFS+
In Use Yes, mounted at /media/markc/Macintosh HD

Yes, that bit of text in the corner that says "Disks" is actually a menu.

LV PRO TIP

Sometimes a damaged drive is more cooperative when it's had a chance to cool down. Stop **ddrescue**, detach the drive for a while, then reattach and resume a little later.

LV PRO TIP

dd, **dd_rescue** and **ddrescue** all use different command line options and parameters, so make sure you know which one you're using, as a mistake could be disastrous.

Chapter 4: An ursine rescue

With Andy's drive showing up as **/dev/sdb** on our penguin-powered machine it was time to send **ddrescue** stampeding in to flush out the easily recoverable data. We just had to specify the source drive followed by the names of the image and log files. Note that the command is **ddrescue**, even though the package we installed was **gddrescue**.

```
sudo ddrescue /dev/sdb hard_drive.img rescue_log
```

To our dismay the first errors arrived quickly. The errsize figure in the output grew rapidly. 40,000 bytes… 50,000… then at just over 60kB the figure stopped increasing. Could we really have been so lucky? Was the damaged data confined solely to the boot partition, meaning that Andy's personal files were all intact? We wouldn't know the answer for some time – over 28 hours at USB2 speed – when **ddrescue** finished its first pass.

The errsize still stood at 60kB!

We didn't want anything from the boot partition, so there was no real need to continue. But we were curious to discover just how much **ddrescue** might be able to recover from those damaged sectors. We let it run on, continuing through its remaining phases, and quickly the errsize dropped to about 20kB.

20kB of bad data after just a single run was certainly impressive. But stubborn sectors can sometimes be persuaded to give up their data if you just ask them often enough, so we ran **ddrescue** a second time, instructing it to retry each bad sector up to three times.

```
sudo ddrescue -r3 /dev/sdb hard_drive.img rescue_log
```

That recovered another half a kilobyte of data. Perhaps we could surprise the drive into responding by sneaking up on it from the other direction? Adding **-R** to the command told it to read the sectors in reverse order, working back into the damaged areas.

```
sudo ddrescue -R -r3 /dev/sdb hard_drive.img rescue_log
```

Almost 15kB was recovered by that approach, leaving us with only 5,120 bytes of unreadable data. We tried a few more passes, but no additional data was forthcoming. Still, 5kB of bad data seemed pretty good to us – and as it was all on the boot partition we were confident that Bertie Bear would live to fight another day. Had the errors been on the data partition then we might have persevered a little more. With any suspect drive, however, there's always a danger that you'll speed up the degradation of the device, so the rule should usually be to get as much data as possible, as quickly as you can, and only spend extra time on stubborn sectors if you really need to.

Chapter 5: A bear in a gilded cage

Although we now had an image of the drive to work with, in some respects we'd actually taken a step backwards. Whereas we had previously been able to access the files on the drive directly from the desktop, now the crown jewels we sought were trapped inside a partition which was in turn inside a disk image.

We were only really interested in one of the three partitions. Had we imaged each one individually we would be able to mount it directly using Linux's loopback interface. But we'd imaged a whole drive, with partitions inside it. We needed a way to tell Linux to mount the drive, then mount the partitions within it, before we could gain access to the files themselves.

It turns out that Mate's *Disks* application has a secret ability. It does such a god job of looking like a simple, single dialog application that few people notice it has a menu bar, hiding in plain sight. Clicking on the lone menu reveals that it holds an entry that reads "Attach Disk Image".

Using that option to attach our disk image immediately placed it into the list of drives alongside

When recovery gets tough

We were lucky with this recovery job because the damaged sectors were all in an unneeded partition. But what would we have done if the damage had been to a partition we wanted? In any data recovery situation you should always make a disk image first, rather than working directly on the suspect drive, so the **ddrescue** step would be similar – but we would probably have been a little more careful in our subsequent runs to recover as much as possible. The most likely result would be a readable disk image with some missing data. In that case it's simply time to keep your fingers crossed that the rogue bytes aren't in any files you actually want.

If the partition information itself is unrecoverable it's time to install *TestDisk* and *PhotoRec*, a pair of applications written by Christophe Grenier. These are often bundled together: installing them both on a Debian-based system just requires a single **sudo apt-get install testdisk** command.

TestDisk (though the executable name is **testdisk**) can be used to recover lost and damaged partitions by analysing the disk structure and recognising a number of partition types. If *TestDisk* is unable to recover the partition, *PhotoRec* can often recover files at an even lower level. Despite its name, *PhotoRec* can find more than just photos: it understands an extensive list of file types, and it's possible to add your own file signatures should you need something more esoteric. It works by reading blocks from the disk or image directly, so can recover files even if the partition format is unknown. *PhotoRec* only works reliably on unfragmented data, though, so don't expect miracles when dealing with a well-used drive that's full to bursting.

You can find out more about *TestDisk* and *PhotoRec*, including worked examples, at Christophe's website: **www.cgsecurity.org**.

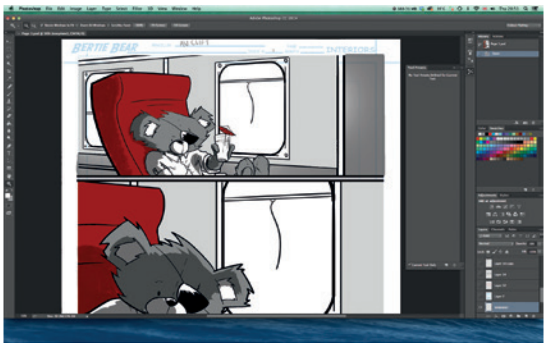
our other, physical devices. Selecting it populated the rest of the window with the same overview of the partitions as we had previously seen with the real drive. Then we selected the data partition and clicked the "mount" button. A link appeared, proclaiming the path to the mount point. With some scepticism we clicked the link, paused for a second, then released a sigh of relief as a window opened before us, displaying the contents of Andy's drive in all its Mac-based glory.

You would be forgiven for thinking that the rest was easy. But this is the tale of data recovery across disparate operating systems, and for all the hubris of Silicon Valley the truth is that computers rarely make things that straightforward. Quickly we were stymied by permissions issues preventing us accessing all the files we wanted.

The problem is that Linux's HFS+ support is a little too good. As OS X is a Unix system at heart, so its filesystem carries with it all the finer details of ownership and access rights that you might expect from a Linux-native format. On the Linux box our user ID was 1000. Andy's Mac had given him an ID of 500 – and that ID was gladly honoured by Linux, denying us access to many of the files. In a pique of laziness we used **sudo** to launch Mate's *Caja* file manager, elevating ourselves to a position of computer godhood, so that trivialities like file permissions would no longer impede us. But ask yourself, dear reader, who among you would not have taken the same approach, so long as you thought that nobody was looking?

```
sudo caja --no-desktop
```

The external drive that Andy had sent us was formatted using Microsoft's NTFS filesystem – which doesn't preserve Unix permissions. Knowing that OS X is quite capable of reading from such a drive, we just selected everything in Andy's home directory and dragged it straight to the external drive, assured that the pesky user ID wouldn't be preserved, so wouldn't cause Andy a problem later. Finally Bertie Bear was freed from captivity.


At last Bertie could relax with a drink, now he was back at home in *Photoshop* on Andy's Mac.

Epilogue

A few days later we received news that Andy was able to read the files from the backup drive. Andy's files were intact, and he had learned a vital lesson about making backups. We had discovered a little more about **ddrescue** and how to recover data from inside a partition in a disk image. And we had a rip-roaring adventure to write up for Linux Voice.

Thanks to our efforts, *Bertie Bear and the Dagger of a Thousand Souls, Volume 3* was released on schedule. Interested readers can find this, and the previous two instalments at **http://bertiebear.bigcartel.com**.

But alas! as is so often the case in such tales, the antagonist of our story still lives on and continues with their evil ways. Who knows how many drives are being held hostage by Apple and their ilk? Despite our slight dramatisation it really wasn't too hard to get Andy's personal data from his drive. Imagine what that means for all the "dead" drives that Apple has sent for salvage, or for those that grace the listings of Ebay, that fill the shelves of pawn shops, or that reside in the carcasses of abandoned computers at rubbish dumps across the land. Remember this tale the next time you're tempted to let an old hard drive out of your hands. Oh, and one final thing: go and make a backup. Now. We can't always be there to save you.

Mark Crutch has been helping the world through Linux for a while, but more importantly, he's one half of the team that creates the Elvie cartoon in our letters pages: peppertop.com.

LV PRO TIP

You can use GNU ddrescue as a replacement for dd in a lot of cases – such as writing an OS to an SD card for use in a Raspberry Pi. They will do the same job, but ddrescue provides more feedback as it progresses.