

# When Disaster Strikes: Hard Drive Crashes

Kyle Rankin

(Reprinted from Linux Journal)

The following is the beginning of a series of columns on Linux disasters and how to recover from them, inspired in part by a Halloween *Linux Journal* Live episode titled “Horror Stories”. You can watch the original episode at [www.linuxjournal.com/video/linux-journal-live-horror-stories](http://www.linuxjournal.com/video/linux-journal-live-horror-stories).

Nothing teaches you about Linux like a good disaster. Whether it's a hard drive crash, a wayward `rm -rf` command or `fdisk` mistakes, there are any number of ways your normal day as a Linux user can turn into a nightmare. Now, with that nightmare comes great opportunity: I've learned more about how Linux works by accidentally breaking it and then having to fix it again, than I ever have learned when everything was running smoothly. Believe me when I say that the following series of articles on system recovery is hard-earned knowledge.

Treated well, computer equipment is pretty reliable. Although I've experienced failures in just about every major computer part over the years, the fact is, I've had more computers outlast their usefulness than not. That being said, there's one computer component you can almost count on to fail at some point—the hard drive. You can blame it on the fast-moving parts, the vibration and heat inside a computer system or even a mistake on a forklift at the factory, but when your hard drive fails prematurely, no five-year warranty is going to make you feel better about all that lost data you forgot to back up.

The most important thing you can do to protect yourself from a hard drive crash (or really most Linux disasters) is back up your data. Back up your data! Not even a good RAID system can protect you from all hard drive failures (plus RAID doesn't protect you if you delete a file accidentally), so if the data is important, be sure to back it up. Testing your backups is just as important as backing up in the first place. You have not truly backed up anything if you haven't tested restoring the backup. The methods I list below for recovering data from a crashed hard drive are much more time consuming than restoring from a backup, so if at all possible, back up your data.

Now that I'm done with my lecture, let's assume that for some reason, one of your hard drives crashed and you did not have a backup. All is not necessarily lost. There are many different kinds of hard drive failure. Now, in a true hard drive crash, the head of the hard drive actually will crash into the platter as it spins at high speed. I've seen platters after a head crash that are translucent in sections as the head scraped off all of the magnetic coating. If this has happened to you, no command I list here will help you. Your only recourse will be one of the forensics firms out there that specialize in hard drive recovery. When most people say their hard drive has crashed, they are talking about a less extreme failure. Often, what has happened is that the hard drive has developed a number of bad blocks—so many that you cannot mount the filesystem—or in other cases, there is some different failure that results in I/O errors when you try to read from the hard drive. In many of these circumstances, you can recover at least some, if not most, of the data. I've been able to recover data from drives that sounded *horrible* and other people had completely written off, and it took only a few commands and a little patience.

## Create a Recovery Image

Hard drive recovery works on the assumption that not all of the data on the drive is bad. Generally speaking, if you have bad blocks on a hard drive, they often are clustered together. The rest of the data on the drive could be fine if you could only access it. When hard drives start to die, they often do

# When Disaster Strikes: Hard Drive Crashes

Kyle Rankin

(Reprinted from Linux Journal)

it in phases, so you want to recover as much data as quickly as possible. If a hard drive has I/O errors, you sometimes can damage the data further if you run filesystem checks or other repairs on the device itself. Instead, what you want to do is create a complete image of the drive, stored on good media, and then work with that image.

A number of imaging tools are available for Linux—from the classic `dd` program to advanced GUI tools—but the problem with most of them is that they are designed to image healthy drives. The problem with unhealthy drives is that when you attempt to read from a bad block, you will get an I/O error, and most standard imaging tools will fail in some way when they get an error. Although you can tell `dd` to ignore errors, it happily will skip to the next block and write nothing for the block it can't read, so you can end up with an image that's smaller than your drive. When you image an unhealthy drive, you want a tool designed for the job. For Linux, that tool is `ddrescue`.

## **ddrescue or dd\_rescue**

To make things a little confusing, there are two similar tools with almost identical names. `dd_rescue` (with an underscore) is an older rescue tool that still does the job, but it works in a fairly basic manner. It starts at the beginning of the drive, and when it encounters errors, it retries a number of times and then moves to the next block. Eventually (usually after a few days), it reaches the end of the drive. Often bad blocks are clustered together, and in the case when all of the bad blocks are near the beginning of the drive, you could waste a lot of time trying to read them instead of recovering all of the good blocks.

The `ddrescue` tool (no underscore) is part of the GNU Project and takes the basic algorithm of `dd_rescue` further. `ddrescue` tries to recover all of the good data from the device first and then divides and conquers the remaining bad blocks until it has tried to recover the entire drive. Another added feature of `ddrescue` is that it optionally can maintain a log file of what it already has recovered, so you can stop the program and then resume later right where you left off. This is useful when you believe `ddrescue` has recovered the bulk of the good data. You can stop the program and make a copy of the mostly complete image, so you can attempt to repair it, and then start `ddrescue` again to complete the image.

## **Prepare to Image**

The first thing you will need when creating an image of your failed drive is another drive of equal or greater size to store the image. If you plan to use the second drive as a replacement, you probably will want to image directly from one device to the next. However, if you just want to mount the image and recover particular files, or want to store the image on an already-formatted partition or want to recover from another computer, you likely will create the image as a file. If you do want to image to a file, your job will be simpler if you image one partition from the drive at a time. That way, it will be easier to mount and `fsck` the image later.

The `ddrescue` program is available as a package (`ddrescue` in Debian and Ubuntu), or you can download and install it from the project page. Note that if you are trying to recover the main disk of a system, you clearly will need to recover either using a second system or find a rescue disk that has `ddrescue` or can install it live (Knoppix fits the bill, for instance).

# When Disaster Strikes: Hard Drive Crashes

Kyle Rankin

(Reprinted from Linux Journal)

## Run ddrescue

Once ddrescue is installed, it is relatively simple to run. The first argument is the device you want to image. The second argument is the device or file to which you want to image. The optional third argument is the path to a log file ddrescue can maintain so that it can resume. For our example, let's say I have a failing hard drive at /dev/sda and have mounted a large partition to store the image at /mnt/recovery/. I would run the following command to rescue the first partition on /dev/sda:

```
$ sudo ddrescue /dev/sda1 /mnt/recovery/sda1_image.img
/mnt/recovery/logfile
Press Ctrl-C to interrupt
Initial status (read from logfile)
rescued:      0 B,  errsize:  0 B,  errors:      0
Current status
rescued:  349372 kB,  errsize:  0 B,  current rate: 19398 kB/s
  ipos:   349372 kB,  errors:   0,    average rate: 16162 kB/s
  opos:   349372 kB
```

Note that you need to run ddrescue with root privileges. Also notice that I specified /dev/sda1 as the source device, as I wanted to image to a file. If I were going to output to another hard drive device (like /dev/sdb), I would have specified /dev/sda instead. If there were more than one partition on this drive that I wanted to recover, I would repeat this command for each partition and save each as its own image.

As you can see, a great thing about ddrescue is that it gives you constantly updating output, so you can gauge your progress as you rescue the partition. In fact, in some circumstances, I prefer using ddrescue over dd for regular imaging as well, just for the progress output. Having constant progress output additionally is useful when considering how long it can take to rescue a failing drive. In some circumstances, it even can take a few days, depending on the size of the drive, so it's good to know how far along you are.

## Repair the Image Filesystem

Once you have a complete image of your drive or partition, the next step is to repair the filesystem. Presumably, there were bad blocks and areas that ddrescue could not recover, so the goal here is to attempt to repair enough of the filesystem so you at least can mount it. Now, if you had imaged to another hard drive, you would run the fsck against individual partitions on the drive. In my case, I created an image file, so I can run fsck directly against the file:

```
$ sudo fsck -y /mnt/recovery/sda1_image.img
```

I'm assuming I will encounter errors on the filesystem, so I added the -y option, which will make fsck go ahead and attempt to repair all of the errors without prompting me.

## Mount the Image

Once the fsck has completed, I can attempt to mount the filesystem and recover my important files. If you imaged to a complete hard drive and want to try to boot from it, after you fsck each partition, you would try to mount them individually and see whether you can read from them, and then swap the

# When Disaster Strikes: Hard Drive Crashes

Kyle Rankin

(Reprinted from Linux Journal)

drive into your original computer and try to boot from it. In my example here, I just want to try to recover some important files from this image, so I would mount the image file loopback:

```
$ sudo mount -o loop /mnt/recovery/sda1_image.img /mnt/image
```

Now I can browse through /mnt/image and hope that my important files weren't among the corrupted blocks.

## Method of Last Resort

Unfortunately in some cases, a hard drive has far too many errors for fsck to correct. In these situations, you might not even be able to mount the filesystem at all. If this happens, you aren't necessarily completely out of luck. Depending on what type of files you want to recover, you may be able to pull the information you need directly from the image. If, for instance, you have a critical term paper or other document you need to retrieve from the machine, simply run the strings command on the image and output to a second file:

```
$ sudo strings /mnt/recovery/sda1_image.img > /mnt/recovery/sda1_strings.txt
```

The sda1\_strings.txt file will contain all of the text from the image (which might turn out to be a lot of data) from man page entries to config files to output within program binaries. It's a lot of data to sift through, but if you know a keyword in your term paper, you can open up this text file in less, and then press the / key and type your keyword in to see whether it can be found. Alternatively, you can grep through the strings file for your keyword and the surrounding lines. For instance, if you were writing a term paper on dolphins, you could run:

```
$ sudo grep -C 1000 dolphin /mnt/recovery/sda1_strings.txt > /mnt/recovery/dolphin_paper.txt
```

This would not only pull out any lines containing the word dolphin, it also would pull out the surrounding 1,000 lines. Then, you can just browse through the dolphin\_paper.txt file and remove lines that aren't part of your paper. You might need to tweak the -C argument in grep so that it grabs even more lines.

In conclusion, when your hard drive starts to make funny noises and won't mount, it isn't necessarily the end of the world. Although ddrescue is no replacement for a good, tested backup, it still can save the day when disaster strikes your hard drive. Also note that ddrescue will work on just about any device, so you can use it to attempt recovery on those scratched CD-ROM discs too.

Kyle Rankin is a Senior Systems Administrator in the San Francisco Bay Area and the author of a number of books, including *Knoppix Hacks* and *Ubuntu Hacks* for O'Reilly Media. He is currently the president of the North Bay Linux Users' Group.