How Linux Saved My Files and My Job
Date: Tuesday, May 25, 2004
Topic: Other Software

Rory Winston

Next time your NTFS-based drive decides to take a sudden trip down south, give BG-Rescue Linux a try.

This is a story of how Linux saved my life. Well, actually, not my life but perhaps my job. This is the story of how Linux helped me to recover some important data that I had almost given up on ever getting back, saving my employer and me a whole lot of time, effort and frustration in the process.

Picture the scene: you're sitting at your desk, writing some code for a client on your aging but reliable ThinkPad. Things all seem to be coming together nicely, and you're putting the finishing touches on the last unit test in the current package. You make a modification, press Ctrl and F11, and Eclipse fires up the JUnit. As you wait for the unit test progress bar to reach 100%, you notice that Eclipse literally has begun to crawl. It's true that Eclipse can be a memory hog at the best of times, but this is unusually severe. You turn away for a second, drum your fingers on the desk,and when you turn back, there's a big ugly blue screen of death (BSOD) staring you in the face.

Back in the days of Win95/98, BSODs were common, and NT had its fair share as well--I even had a joke BSOD screensaver at one point. But, Win2k is better in this regard, and a blue screen usually means something is seriously amiss. The message on the screen reads: `KERNEL_DATA_INPAGE_ERROR`. The only driver visible in the stack trace is atapi.sys, the IDE driver. You scribble down some details. A few minutes of judicious Googling may be able to shed some light on the problem--when the machine restarts. The Win2k memory dump takes an inordinately long time to complete. As you reboot after the dump is complete, you make a mental note to make a backup. It's been a long time since you backed up anything.

The Win2k progress bar gets about 70% across; it's still noticeably slower than normal now and then it bluescreens again. This time it reads `INACCESSIBLE_BOOT_DEVICE`. The cold hand of fear begins to tighten around you; this is not good. From bitter experience, you know that spontaneous errors related to boot devices are the worst. A few reboots later and nothing has changed. Powering off and then back on produces the same result, and safe mode is no better. It's at this point that you remember the code you were working on hasn't been checked into CVS yet. It was going to be refactored heavily before you added it to the source repository. You could try and place the blame on CVS and its propensity for making difficult the large-scale refactoring of a project structure--and this would be true--but the reality is you should have checked in the code earlier, if only to guard against such a situation. This really is going to mess up the deadline for this portion of the project deliverables. So what to do? Taking a deep breath, you weigh up the options and decide to try and find a way to recover the data.

In my case, at first glance, it seemed as though the hard drive or possibly even the drive controller was at fault. So, I grabbed an identical ThinkPad model and swapped the drives across. The other ThinkPad drive booted up fine, but the drive still was, to use the technical vernacular, knackered. I couldn't simply whip out the drive and attach it as a slave drive on another computer's IDE controller, this being a ThinkPad drive, and all.

My next port of call was the Windows 2000 install CD. I put it in and fired up the recovery console. I recently used the recovery console to restore a corrupted NT boot loader, so I figured it might do the trick here. At the least, I hoped I might be able to get a command-prompt view into the hard drive so I could run chkdsk. When I finally got to the DOS prompt, chkdsk didn't want to know about it. `Drive C contains unrecoverable errors` was all it wanted to say on the matter. So that option was out.

The next few hours were tedious and frustrating, to say the least. I alternated between three well-known disk recovery and repair software programs, formatting floppies, installing the programs onto the floppies and then attempting to run the programs against the damaged disk. The experiences varied, but the end result was always the same. One of the programs couldn't run without utilizing EMS (extended memory) on DOS, but its extended memory manager (EMM368.EXE) always crapped out, complaining that it couldn't find an unused 64K frame to use as extended memory. A look at the help for EMM386.EXE and a quick look at CONFIG.SYS and AUTOEXEC.BAT (thought I'd seen the last of those guys a few years ago) revealed that there were some settings to alter the locations at which EMM368.EXE probed for unused memory. However, I really didn't know what memory ranges I safely could specify, and the

prospect of finding out by way of trial-and-error wasn't appealing. To make matters worse, any reference material I found on EMM386 was for the MS-DOS version; I was using Caldera DR-DOS.

I got similar results from the next program. It looked promising at first--it actually started enumerating through the filesystem on the disk. About 30% of the way through, however, it crashed with an ugly DOS page fault error.

Maybe if I could get a boot disk, I could get a DOS command prompt, I thought. So I dutifully looked around and found a great site called www.bootdisk.com. I downloaded the DOS 6.22 and Win98 SE boot disks. I decided to go with the Win98 SE boot disk, figuring it would have better support for long filenames and so on. I then dimly remembered a site I hadn't visited for a while, a Windows kernel development site called www.sysinternals.com. This site is a gem; it's full of hardcode techie info, plus loads of useful, tricky little utilities, including a lot of useful source code. The utility I was after was called NTFSDOS, and it basically consists of a driver (NTFSDOS.VXD) that allows read-only (for the free version) and read-write (for the pay version) access to NTFS from DOS. I downloaded the utility, stuck it on a floppy and booted into DOS. I ran NTFSDOS.EXE and, lo and behold, there was my drive--the first positive development all day.

I quickly browsed around the file structure and verified that things seemed to be intact. I then cataloged the important files I needed to rescue. The Trouble was my only media was a few floppies. This might seem okay in theory, but the more I browsed through the disk, the more stuff I found that I wanted to keep. Also, the source tree alone that I wanted to retrieve was pretty large. There also were things such as directories with 12 PowerPoint presentations at about 2MB apiece. I also had other essential items, .MAME for one. Clearly, this was not going to be easy. I searched around for a DOS NIC driver for my specific MiniPCI Ethernet/56k combo, but I couldn't find one anywhere.

At this point, things looked pretty bleak. Without a working NIC under DOS, I could spend about a week with a handful of floppies and a copy of PkZIP--not an ideal solution. It was at this point that I remembered using Linux once to recover some data from an NT server I ran in university. I also recently remembered a friend recounting to me how he had lost the administrator password to his Windows box and had reset it using a Linux-based boot floppy that altered the SAM. I shrugged and thought I might as well give it a try. No other practical options were left at this point.

I've been a Linux user since about 1997 and a FreeBSD user for the last couple of years, but I had never tried any of the floppy-based distros. I looked around and quickly came across a floppy distro called BG-Rescue Linux. This seemed to be a pretty capable little distro, specifically aimed at disaster recovery. The current version of BG-Rescue Linux is 0.3.1, which is compiled with kernel version 2.4.24, and it supported a host of Ethernet devices--it even had USB and PCMCIA network device support. A host of command-line utilities are provided by BusyBox, and BG-Rescue Linux uses the uClibC C library. What really made my eyes light up was the inclusion of NTFS support. This uses the LinuxNTFS driver, which is a complete and heavily improved rewrite of the older NTFS driver I had used. BG-Rescue Linux comes complete with the ntfsprogs package, which contains, among other things, tools to non-destructively resize an NTFS partition under Linux.

I downloaded the two floppy disk install images and wrote them onto two blank floppies using `dd if=imagename.img of=/dev/fd0`. I then inserted the first floppy into the stricken laptop and rebooted. The familiar Linux init sequence rolled up onto the screen, accompanied by a colorful Tux image. I was prompted for a preferred display resolution, followed by a prompt to insert the second floppy disk. It then unzipped its filesystem images into RAM, and a minute or so later I was sitting at a root shell prompt.

First of all, I checked to see if it had picked up the network card. A quick scan of the output of dmesg confirmed that it had found something, all right, and assigned it a driver. I set the IP address and netmask using ifconfig (`ifconfig eth0 xx.xx.xx.xx. netmask yy.yy.yy.yy`), and then attempted to ping another machine on the same network. It worked. A quick moment of elation ensued, and then it was on to the hard work--accessing and backing up the filesystem. Truth be told, it wasn't that hard at all. In fact, it was downright simple. My next step was to create the directory /mnt/win2k and mount my NTFS partition under that by running `mount -t ntfs /dev/hda1 /mnt/win2k`. Two seconds later, I had access to the entire NTFS directory tree. Things were finally starting to look positive.

The next step was to perform the actual file backups. I had a couple of options; BG-Rescue Linux comes with SMB and NFS support and also has cmdftp built in. Deciding to go for the quick-and-easy SMB share route, I hopped over to our new Dell server (running Red Hat 9) and set up a Samba share called, appropriately enough, disaster. Going back to the laptop, I accessed the share with smbclient, like so:

```
smbclient //xx.xx.xx.xx/disaster <password>
```

where xx.xx.xx.xx was the Red Hat server's IP address. This dropped me into an FTP-like environment; it

even had a similar command set (prompt, mget, mput). I was able to back up all of the necessary files and directory trees quickly and easily by turning on smbclient's recurse option to copy entire directory trees with one command. In fact, it would have been even easier if the smbfs filesystem was supported and smbmount was included in the command set for the distro--a suggestion for the next release of BG Rescue, perhaps?

So, now I'm writing this article on a backup laptop. My new one currently is on order. Both work and personal data has been recovered and dutifully backed up, the project manager was placated and I'm overall safe in the knowledge that I have come out pretty well from this potentially disastrous situation, thanks to Linux. More specifically, thanks to the maintainer of BG-Rescue Linux, an invaluable distro for this type of situation.