Mastering Linux, part 18

Backups of all critical data, *Jarrod Spiga* explains, should be performed frequently, because — sooner or later — disaster is going to strike.

JARROD SPIGA

Jarrod Spiga is an infrastructure and network engineer who specialises in getting Linux and Windows systems happily coexisting with each other. He currently holds CCNA and numerous MCP certifications, and dabbles in Web application development in his spare time.

SKILL LEVEL

- Advanced

TIME TO COMPLETE

- Three hours

REQUIREMENTS

- An installation of Linux. (CentOS 4.2 and Red Hat Enterprise Linux ES 4.0 instances were used in the writing of this Workshop.)

IN THIS SERIES

- **Part 17 — April '06**
  Internet services.
- **Part 18 — May '06**
  Backups and troubleshooting.
- **Part 19 — June '06**
  Kernel compilation.

NEXT MONTH

- The final Mastering Linux episode steps you through one of the most daunting (and geekiest) things you can do to your system: compiling your own customised kernel.

PRESERVE YOUR DATA WITH TAR

If you've been working with Linux for the last year and a half, you've undoubtedly had to extract files from within a tar.gz file before. The .gz extension indicates that the data has been compressed using the GZip utility, while the .tar portion of the extension means that the files are stored in a tape archive format.

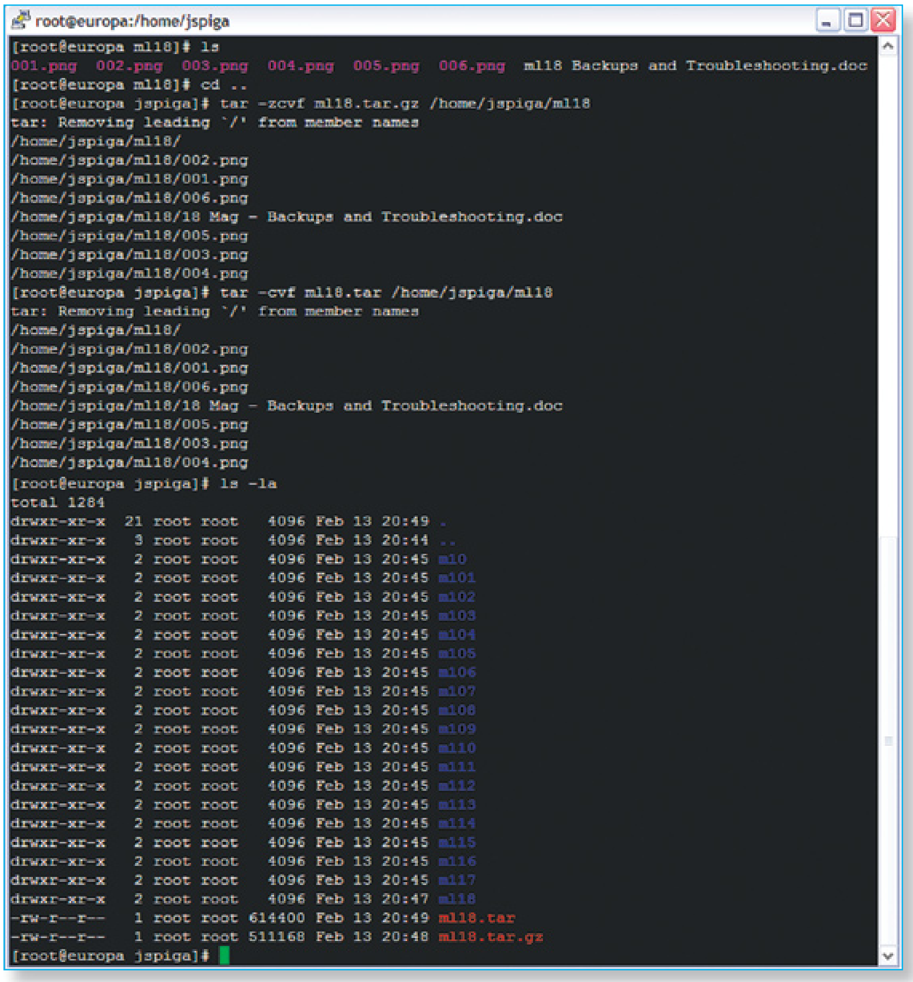
To put it another way, a tar contains a collection of files, archived in a contiguous fashion in one single file. If you compress the file using the GZip utility, you end up with the familiar .tar.gz file.

A tar archive stores all files in a sequential manner and preserves directory structures and file system permissions. The format was originally designed to store data appropriately on tape backup devices, where all data must be stored in a sequential fashion (you can't copy data to tape devices using the cp command).

The GZip utility only has the ability to compress a single file into a compressed archive. Using both tar and GZip allows you to compress a large number of files and directories into a single archive, while preserving filesystem permissions and the directory structure. The two applications go hand-in-hand in regards to archiving data.

CREATING A TAPE ARCHIVE

In general, the tar command can only be used at the command line. The following command string can be used to create basic Tape Archives:



root@europa:/home/jspiga

```
[root@europa ml18]# ls
001.png  002.png  003.png  004.png  005.png  006.png  ml18 Backups and Troubleshooting.doc
[root@europa ml18]# cd ..
[root@europa jspiga]# tar -zcvf ml18.tar.gz /home/jspiga/ml18
tar: Removing leading `/' from member names
/home/jspiga/ml18/
/home/jspiga/ml18/002.png
/home/jspiga/ml18/001.png
/home/jspiga/ml18/006.png
/home/jspiga/ml18/18 Mag - Backups and Troubleshooting.doc
/home/jspiga/ml18/005.png
/home/jspiga/ml18/003.png
/home/jspiga/ml18/004.png
[root@europa jspiga]# tar -cvf ml18.tar /home/jspiga/ml18
tar: Removing leading `/' from member names
/home/jspiga/ml18/
/home/jspiga/ml18/002.png
/home/jspiga/ml18/001.png
/home/jspiga/ml18/006.png
/home/jspiga/ml18/18 Mag - Backups and Troubleshooting.doc
/home/jspiga/ml18/005.png
/home/jspiga/ml18/003.png
/home/jspiga/ml18/004.png
[root@europa jspiga]# ls -la
total 1284
drwxr-xr-x  21 root root   4096 Feb 13 20:49 .
drwxr-xr-x   3 root root   4096 Feb 13 20:44 ..
drwxr-xr-x   2 root root   4096 Feb 13 20:45 ml0
drwxr-xr-x   2 root root   4096 Feb 13 20:45 ml01
drwxr-xr-x   2 root root   4096 Feb 13 20:45 ml02
drwxr-xr-x   2 root root   4096 Feb 13 20:45 ml03
drwxr-xr-x   2 root root   4096 Feb 13 20:45 ml04
drwxr-xr-x   2 root root   4096 Feb 13 20:45 ml05
drwxr-xr-x   2 root root   4096 Feb 13 20:45 ml06
drwxr-xr-x   2 root root   4096 Feb 13 20:45 ml07
drwxr-xr-x   2 root root   4096 Feb 13 20:45 ml08
drwxr-xr-x   2 root root   4096 Feb 13 20:45 ml09
drwxr-xr-x   2 root root   4096 Feb 13 20:45 ml10
drwxr-xr-x   2 root root   4096 Feb 13 20:45 ml11
drwxr-xr-x   2 root root   4096 Feb 13 20:45 ml12
drwxr-xr-x   2 root root   4096 Feb 13 20:45 ml13
drwxr-xr-x   2 root root   4096 Feb 13 20:45 ml14
drwxr-xr-x   2 root root   4096 Feb 13 20:45 ml15
drwxr-xr-x   2 root root   4096 Feb 13 20:45 ml16
drwxr-xr-x   2 root root   4096 Feb 13 20:45 ml17
drwxr-xr-x   2 root root   4096 Feb 13 20:47 ml18
-rw-r--r--   1 root root 614400 Feb 13 20:49 ml18.tar
-rw-r--r--   1 root root 511168 Feb 13 20:48 ml18.tar.gz
[root@europa jspiga]#
```

Compression test: compress your tar if you're sending data to someone else. Don't compress if you're backing up!

```
tar –cvf dest target
```

where dest is the destination filename (or device if backing up to a tape drive) of the archive and target refers to the names of the directories and/or files that you wish to include in the archive. More than one target can be specified, allowing you to store data from different areas of the filesystem in the one archive.

The c switch is supplied to the tar command to instruct it to create a new archive. The v switch refers to verbose, which means that the command will display the name of each file that is added to the archive. Lastly, the f switch instructs tar to archive all data to a single file.

Another commonly appended switch is z — which instructs the tar command to filter the archive through GZip. For instance:

```
tar –cvfz ml18.tar.gz /home/jspiga/ml18
```

This argument should be used appropriately though. It's perfectly fine to use when you're packaging data to send to another person, but it should not be used for mission-critical backups. The reason for this is that if there is an error in compression, all of the data within the archive is not able to be retrieved. Mission-critical backups are best not parsed through GZip and left in uncompressed .tar files (and most tape drives handle compression in a native and recoverable manner anyway).

Whenever you've created a new archive, it's prudent to test the archive to ensure that tar can recover the data stored within it. To do so, use the t (test) switch as follows:

```
tar –tf archive
```

WHAT NEEDS BACKING UP?

One of the most difficult aspects of backing up is knowing what data to include. On a Windows XP system, most would back up as much of the obvious user data as possible (eg. most of the data under the Documents and Settings directory of the system drive). But after backing this up, there's always other data to be found in other areas of the file system (eg. C:\InetPub on an IIS-enabled system, and some user data that could be located under Program Files).

Knowing what to back up on a Linux system is just as much of a science. In general, you don't want to back up what you can easily restore from your installation media or from secure online sources (via yum). This is especially important to remember in the context of data integrity — if a rootkit has made its way onto your Linux system, you don't want to be restoring the malicious software on a fresh system from your backup.

In all cases, you will want to backup: /home (contains all user home directories), /root (the root user's home directory), /etc (contains all service configuration information) and /usr/local (contains local executable data installed by the users of the system).

It's also worthwhile perusing through the /var directory, as this area of the file system can also contain user data (especially the /var/www directory — which usually contains web data served by the Apache Web Server).

Lastly, it is good practice to write down your partitioning scheme on a piece of paper — that way, you can keep the same partition sizes if you have a catastrophic system failure. The following command will display the list of partitions on the primary master IDE drive (and should be run for all hard disk devices installed on your system):

```
fdisk –l /dev/hda
```

RESTORING FROM A TAR BACKUP

The tar command is also used to restore from your Tape Archive backup:

```
tar –xvf source
```

The only difference between the command used to create the archive and to restore it is that the c switch has been replaced by x (for extract). If the archive has been compressed with GZip, simply add the z switch in exactly the same manner as you did when creating the archive.

Care should be taken when extracting files from a tape archive. The data contained within the archive will overwrite any files and directories that already exist on your file system. You should always check the contents of your current working directory before extracting data from any archive.

If you only want to extract a subset of the data contained within an archive, a third argument can be passed to the command. For example, if you had a single archive backing up all of the data listed in the previous section and you only wanted to extract the data that was in the /root directory, the following command would be used:

```
tar –xvf backup.tar "/root/*"
```

It may take a while to restore a small subsection of data from a tape archive because of how the data is stored in the archive file. This is because each file is stored in a sequential manner within the file, and tar must read every file in the order it was archived.

AUTOMATING BACKUPS

One of the main advantages to using a CLI tool such as tar to perform backups of your system is that you can automate the process using cron — this way, you never have to remember to manually initiate a backup.

The following script (backup.sh) can be used to initiate automated daily and weekly backups of different sections of the file system:

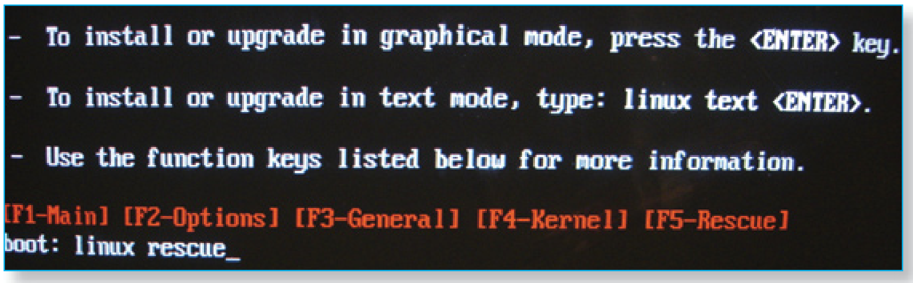
```
#!/bin/sh
if [ "$1" = daily ] ; then
tar –cf /daily.tar /home /root
tar –tf /daily.tar 2> /var/log/daily_backup.log
fi
if [ "$1" = weekly] ; then
tar –cf /weekly.tar /etc /usr/local /var/www
tar –tf /weekly.tar 2> /var/log/weekly_backup.
log
fi
```

After you've created the script and saved it (as the root user), you'll need to schedule it so that it runs with appropriate arguments. Adding the following lines to the /etc/crontab file should do the trick:

```
0 5 * * * /root/backup.sh daily
30 5 * * 0 /root/backup.sh weekly
```

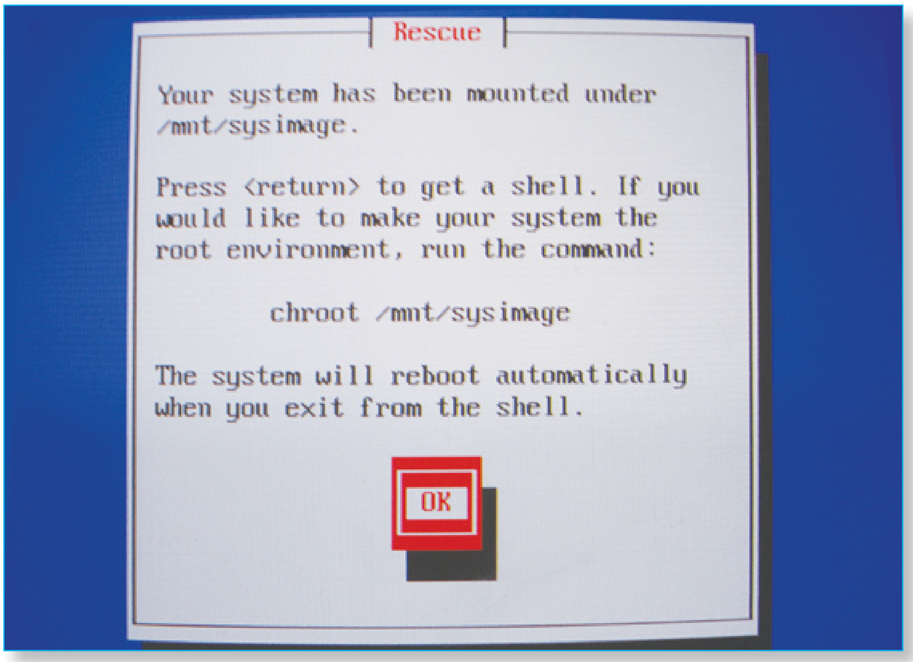
SYSTEM RESCUE

"Given a long enough timeframe, the survival rate for everything drops to zero" is a quote from one of my favourite movies, and it certainly applies to computer systems — sooner or later, every system fails. Even though Linux is a relatively stable and robust operating system, it's still handy to know how to recover when things do go awry.



```
-  To install or upgrade in graphical mode, press the <ENTER> key.
-  To install or upgrade in text mode, type: linux text <ENTER>.
-  Use the function keys listed below for more information.
[F1-Main] [F2-Options] [F3-General] [F4-Kernel] [F5-Rescue]
boot: linux rescue_
```

Rescue me: to boot to rescue mode, insert your installation CD, boot to it and enter linux rescue at the boot prompt.



```
Rescue

Your system has been mounted under
/mnt/sysimage.

Press <return> to get a shell. If you
would like to make your system the
root environment, run the command:

        chroot /mnt/sysimage

The system will reboot automatically
when you exit from the shell.

              OK
```

Mounted patrol: after your Linux file system has been detected and mounted, you can then start the rescue process.

If your Linux system won't boot, the first thing to check is your hardware. Is your system getting past the POST? Is your hard disk spinning up and being detected by BIOS properly?

If your hardware seems to be working properly, pop your first Fedora Core installation CD into your CD/DVD drive and boot to that. At the prompt, type in:

```
linux rescue
```

Skip the media test if you prefer and select the language that you wish to use. The rescue utility should then advise you that it will attempt to detect your Linux installation and mount it under /mnt/sysimage. In general, it's better to mount the installation as read-only — that way, you can't do more damage to the file system than what has already been done.

The rescue utility should confirm that it has detected your installation. If it does not find any data, there are not a lot of options available to you — recovery from these scenarios is possible but extremely difficult (and outside the scope of this Workshop).

ONCE MOUNTED…

If the rescue utility is able to detect your Linux installation successfully, a thorough file system check is usually all that is required in order to fix any storage subsystem errors. The e2fsck command should only ever be on unmounted file systems. You can obtain a list of mounted file systems associated with the detected Linux installation as follows:

```
mount | grep sysimage
```

Once the list is displayed, proceed to unmount all listed file systems:

```
umount /mnt/sysimage/proc
umount /mnt/sysimage/dev/pts
umount /mnt/sysimage/boot
umount /mnt/sysimage
```

Once all file systems are no longer mounted, run the e2fsck command with the –f switch (to force a full disk scan) on each hard disk in your system:

```
e2fsck –f /dev/hda1
```

The output from e2fsck will depend on what problems it encounters. Generally, it's safe to select the suggested options that the command gives you. Once the command has finished running (and it can take quite a while), type exit to get out of rescue mode and then reboot your system. With any luck, your system now boots.

IF ALL ELSE FAILS

If the file system check does not resolve your problems, it's probably a good idea to jump back into rescue mode and start salvaging your data using the backup techniques listed earlier in this Workshop. While it's possible that you could still recover from this level of system failure, you'll probably end up spending more time trying to fix things than it would take to rebuild your system.

PREDICTING HARD DISK FAILURES

Because many Linux systems are built on older hardware, you should always be on the lookout for symptoms that your hard disk drive is failing. The telltale sign of hard disk failure is file system corruption, which has the following symptoms:

- Files which you know should contain legitimate data are replaced with random characters and nonsense.
- Filenames and directory names start having random characters in them, or files and directories start randomly appearing in locations.
- System crashes occur when you access a certain file.
- Files and directories disappear all of a sudden.

If any of these symptoms occur, it's worthwhile booting to rescue mode and running the e2fsck command as shown earlier.

However, you should heed these warning signs; even if running e2fsck initially fixes these issues, your hard disk is on the way out and will need replacing sooner rather than later. At this time, it's prudent to ensure that your backups are occurring more frequently and you are storing the backups on another drive or system.

DISCOVERING MALICIOUS NETWORK ACTIVITY

The last major alarm bells that ring from time to time are security problems related to rootkits, viruses and malware. To think that these nasty pieces of code are solely in the realm of Windows is naïve, hence the reason why even Linux systems should have security measures such as firewalls implemented.

If you've taken enough preventative measures, you should have very little to worry about. However, you should always remain alert for the following symptoms:

- System logs reporting that users are logging on remotely using accounts that you're not aware of.
- Unexplained heavy network traffic.
- The appearance of new accounts in the /etc/passwd file, or the appearance of new groups in the /etc/groups file.
- Newly appearing SUID or SGID files (which usually indicates that someone is trying to, or already has, root access on the system — a sure sign that a rootkit may be installed).

If these symptoms start appearing, it's always a good idea to back up your data, format your system and re-install Linux — especially if you suspect a rootkit is installed. It's generally not worth the effort to try cleaning up such an infection — because you can't guarantee that you will close off every hole that a malicious user has exploited. The secure thing to do is to start again with a secure re-install. apc