Mastering Linux, part 12

Jarrod Spiga **explains that the ability to properly administer a Linux system from the command line is a handy skill to have, even if you aren't a systems administrator.**

On the cover DVD

Bonus DVD software

PDFs of every instalment of the Mastering Linux series.

Skill level

Intermediate

Requirements

An installation of Linux (Fedora Core 4 and Red Hat Enterprise Linux ES3 were used in the writing of this article).

Time to complete

3 hours

In this series . . .

USER SWITCHING

A point that is constantly emphasised in this series is that you should avoid logging in as the root user wherever possible. If you're already logged in as a non-root user at the console, you can quickly bring up a temporary sub-shell by using the su (substitute user) command.

When used with no arguments, su will prompt you for the root password. If you can supply the correct password, the shell will behave exactly as it would if you were logged in as root — even down to the # as your prompt instead of a $.

Once you've finished entering commands as the root user, use exit to take you back to the shell that you started in.

Alternatively, if you only have to execute a single command as another user, consider using sudo. However, some setup is required in order to use this command effectively (mainly in relation to configuration of the /etc/sudoers file). The man page for sudo is easy to follow and will guide you through the process of how to use the command.

WHAT'S RUNNING?

On a Windows system, you can load up Task Manager to see what processes are running on your system. Under Linux, the ps (Process Status) command is used to provide a snapshot of the processes running at the current time. Executing the command with no arguments will return a list of all processes that have started from your current session.

This command shows some valuable information. For starters, the PID (Process Identifier) is required to kill or re-prioritise a process. TTY (Terminal Identifier) shows which system terminal the process was executed from. The TIME column displays the amount of processor time that has been assigned to the process. Lastly, the CMD (command) column displays the command entered to commence the process.

Like most commands, arguments can be used to alter the behaviour of the ps command. The table below shows the most frequently-used arguments.

Argument	Description
-e	List all processes, including processes that weren't launched from a terminal.
-f	Provide a full listing of information about the process including the parent PID, start time and process owner.
-u user	List the processes that are owned by user.
-g group	List the processes that are owned by users who belong to a group.
a	List all processes, excluding the ones that were not launched from a terminal.

RE-PRIORITISE!

Unix was one of the first true multi-tasking operating systems. One resource that must be shared is the CPU. Each and every process needs to use the CPU for short amounts of time in order to get work done.

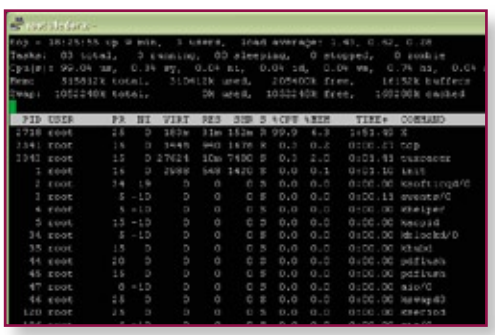
Some applications need more CPU resources than others. For instance, if a process is performing real-time encoding of video on your Linux system, it requires enough CPU time to encode each frame of video, otherwise frames will be dropped. Due to the time-sensitive requirements of this application, you could give it a higher priority than a spreadsheet process.

Every process in Linux is given a nice value — a politically correct name for priority between -20 (the highest priority) and +19 (the lowest priority). However, it makes little sense to assign a value of -20 to all tasks — this won't magically make all tasks run any faster since all processes still have the same priority.

The top command can be used to show what processes are using the most system resources at any given time. You can also change how nice a process is by using the renice command as follows:

```
renice new_value PID
```

where new_value refers to the new nice value that you want to assign to the process and PID is the process identifier of the process you want to adjust



The top command continually refreshes, showing you the process consuming the most resources.

(you can get both the current nice value and the PID ps -from the previously mentioned top command).

THE PROCESS OF KILLING

You may occasionally need to kill a process — to forcibly stop it from continuing to execute. For instance, if one process is stuck in a loop or has crashed, you'd generally want to kill it off using the following command:

```
kill PID
```

where PID is the process identifier of the process you want to terminate. When used with no argument, as shown above, the shell will send a kill signal to the process, requesting that it closes any files the process has open and then terminates. This is the equivalent of hitting the End Task button in Windows Task Manager if an application is still responding.

But if an application has completely crashed, the default kill may not always work — you'll need to send the process a more destructive signal. Adding the -9 argument after the kill command will forcefully stop the process before it even has the chance to close any files or pipes that it's currently working on. This argument should only be used when the more graceful kill doesn't work.

Another argument to the kill command is the -HUP (hang up) argument. Many server processes (such as the Apache Web server) understand this argument and interpret it as a request to reload the configuration files and restart itself. However, not all processes interpret this argument in this manner, and many will simply terminate as if you hadn't passed the argument at all.

STARTING AND STOPPING SERVICES

When you boot up to Linux, a number of system services automatically start. For instance, the network service allows your system to communicate over the network, and X provides the framework for starting X-Windows. The table below lists some other services frequently found on a Linux system:

Service	Name	Purpose
httpd	Apache Web server	The most popular Web server on Linux and Unix systems.
sendmail	Sendmail SMTP server	An SMTP (email) server.
smb	Samba	A Windows File and Print Sharing server.
sshd	Secure shell	A secure shell protocol server.
telnet	Telnet	Another shell protocol server, although much less secure than SSH.
named	The BIND DNS server	A DNS server.

You can control whether a service is running or not with the service command located in the /sbin directory. For instance, to start the Apache Web server (assuming that you have it installed), enter:

```
sbin/service httpd start
```

And as you'd expect, replacing start with stop will stop the service. Most commands that are present in the /sbin directory will require root permissions to run.

RUN LEVELS

The chkconfig command, also located in the /sbin directory, is used to display and modify the run level statuses where system services are started. A run level can be thought of as a state at which your Linux system is currently running in. There are seven run levels, as described in the table below:

Level	Description
0	System Halt. The system has either shut down, or is in the process of shutting down.
1	Single-user. A single virtual console is running a single command line at the console. This is generally used when recovering from hardware failures — akin to the recovery console in Windows.
2	Multi-user, file services only. This run level is used to start most non-networking-based services on a system. Multiple users can have access to the file system from the local console.
3	Multi-user, all services. The common run level used when a system is not running X-Windows (i.e., when only the command line is used at the console). Full network and file system functionality is present.
4	User-defined.
5	Multi-user, all services with X-Windows. This run level is active when your system is booted and you're logged in the X-Windows environment. Full network and file system functionality is present.
6	Reboot. The system is in the process of being rebooted.

To view a list of which services run at which run levels, execute:

```
sbin/chkconfig --list
```

The vncserver service is only operational at run levels 4 and 5. This means that the service will generally only run when X-Windows is running. If you booted to a command line and didn't start X-Windows, this service wouldn't run.

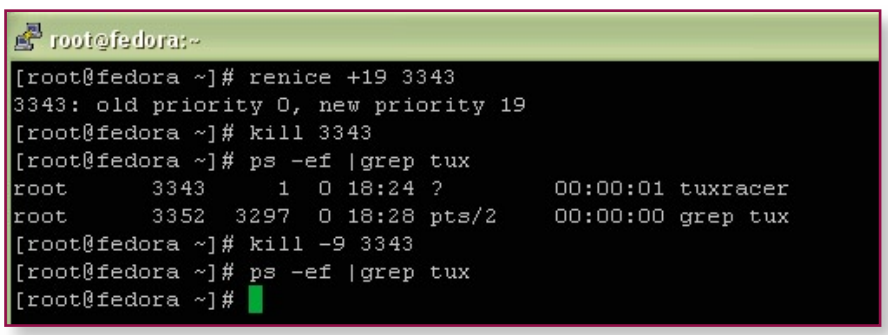
To change whether a service runs when the system is at a particular run level, use:

```
sbin/chkconfig --level <level numbers>
<service> [on|off]
```

where <level numbers> is a list of the run level numbers that you wish to modify and <service> is the name of the service that you're modifying.

For example, if you use your Linux PC predominantly for Web development and run X-Windows while using your development apps, the Apache Web server (with service name of httpd) should be running. However, in order to preserve resources on your system, you don't want httpd to run when X-Windows isn't running. The below commands would ensure this:

```
sbin/chkconfig --level 01236 httpd off
/sbin/chkconfig --level 5 httpd on
```



No time for games: using the renice command to forcibly kill Tux Racer.

You can confirm this by running:

```
/sbin/chkconfig --list | grep httpd
```

ACCOUNT MANAGEMENT

If you're running a multi-user Linux system, the ability to manage user accounts and groups is critically important. The concept of Users and Groups was covered in an earlier instalment of this series (*APC* February, page 103) when filesystem permissions were explained.

Adding a user account is normally a two-step task that needs to be performed as the root user. The useradd command adds the account, while passwd assigns a password to the account. To create an account with your username and to assign that account a password, run:

```
usr/sbin/useradd yourname
passwd yourname
```

After you've run the first command, a home directory should have been created for the user under the /home directory. Other default settings are also applied to the account, such as the default shell that you will be greeted with after logging on.

If you also inspect the contents of the /etc/passwd file, you'll be able to see a list of all the accounts set up on your system as well as a unique user identifier for each account.

To remove an account, use the userdel command like so:

```
usr/sbin/userdel jspiga
```

But take care when removing accounts — those with a user identifier lower than 500 are usually special accounts which have been set up to manage services. Some services may no longer behave as intended if you remove one of these accounts.

GROUP MANAGEMENT

Looking after individual users is only half of the task, since groups need administering too. As you'd expect, the groupadd and groupdel commands can be used in the same fashion as shown above to create and remove groups on your system.

A group is a collection of individual users with some common aspect. In order to be effective from a permissions context, you need to be able to manage group memberships. To add a user to a group, enter:

```
/usr/bin/gpasswd -a user group
```

where user is the user you're adding to the group named group. Using the gpasswd command with the -d switch will remove the specified user from the specified group.

Administrators can't be everywhere at once, and it's often useful to delegate the responsibility of managing certain groups to other users.

To delegate control of a group, use the -A (note the capital A) switch with the gpasswd command shown above.

SCHEDULING TASKS

The cron service within Linux is used to manage scheduled tasks.

There are two types of scheduled tasks: system-wide tasks including backups, system monitoring and log rotation; and user-defined tasks.

The cron daemon's task is to execute shell scripts that perform defined tasks at given times.

System-wide cron jobs are usually called from one of the cron directories under /etc. Generally, only the root user has the access to copy shell scripts to cron directories.

The following table summarises when these system-wide jobs are run:

Directory	Execution frequency
/etc/cron.hourly	Shell scripts placed in this directory are executed at one minute past every hour.
/etc/cron.daily	Shell scripts placed in this directory are executed at 04:02 every day.
/etc/cron.weekly	Shell scripts placed in this directory are executed on Sundays at 04:22.
/etc/cron.monthly	Shell scripts placed in this directory are executed at 04:42 on the first day of every month.

If your system happens to be switched off when cron is meant to execute something, the anacron service will identify which tasks were skipped and then execute them. This ensures that maintenance tasks are run on your system, even if you occasionally shut it down.

Individual users can also add scheduled tasks by running:

```
crontab -e
```

When this command is executed, a vi editor will appear, allowing you to define your task. Each task should appear on its own line, using the following convention:

```
min hour date month day command
```

The following table summarises what these fields should be:

Field	Description
min	The minute (0-60) at which the command should be executed.
hour	The hour (0-24) at which the command should be executed.
date	The date (1-31) at which the command should be executed.
month	The month (1-12) at which the command should be executed.
day	The day of the week (0 = Sunday, 1 Monday and so forth until 7 = Sunday) at which the command should be executed.
command	The command that is to be executed at the times defined by the above fields.

You can also use special characters within any of the non-command related fields to allow more flexible execution times. For instance, using an asterisk in the hour field will execute the command every hour in line with the other time-based fields. Multiple values can be listed separated by commas, while a range of values can be separated by hyphens. Lastly, the forward-slash can be used as a divider. Take the following cron entry, for example:

```
*/5 9-13,14-18 * * 1-5 fetchmail
```

This line will execute fetchmail (which downloads email from a POP3 server) every five minutes, but only during working hours — specifically between 9am and 1pm, 2pm and 6pm on any date of the month and in any month as long as the day is a weekday. The tasks will not be run on weekends, lunch breaks, or days off.

Once you've finished adding or editing your cron entries, always be sure to save your changes (:s!) and then exit from the vi editor (:q).

Next month . . .

The next instalment of Mastering Linux will cover file system management. It will then head back into X-Windows to show how to extend shell scripts so you can interact with them while they're running.