Create, Monitor and Kill Processes

Information

These notes were originally written in the year 2000 as part of a set of LPI Exam 101 training materials. The LPI training course at Bromley College was subsequently discontinued and some of the sections of the notes modified and incorporated into our one-day System Administration Courses. The remainder of the notes have now been made publicly available on the linuxtraining.org.uk website.

If you are a beginner please do not be put off of training courses by these notes, as they are rather technical. On the other hand if you are a more experienced Linux user we hope you find the coverage of this topic refreshingly clear.

For full details of our current Linux training please visit the site:
http://ce.bromley.ac.uk/linux

If you have reached this page from a search engine and wish to see the full contents list for the published notes please visit the site:
http://www.linuxtraining.org.uk

We hope you find these notes useful, but please remember that they apply to the 2.2 kernel. I will update them when I have the time.

Clive Gould - 21st December 2004

Create, Monitor and Kill Processes

Objective 5

Create, Monitor, and Kill Processes: Includes running jobs in the foreground and background, bringing a job from the background to the foreground and vice versa, monitoring active processes, sending signals to processes, and killing processes. Includes using commands ps, top, kill, bg, fg, and jobs.

Report Process Status - ps

You can view all the processes that Linux is running using the **ps** command.

The syntax for ps is shown below:

```
ps option(s)
```

Some common ps options are:

Option	*Explanation*
e	Shows the environment
f	Formats the output as a tree, showing parent and child processes
a	Displays all processes
l	Provides more detail including nice values
u	Shows who each process belongs to
x	Shows processes without an associated terminal

Note that with ps the - is not needed before the option.

An example of using ps is illustrated below:

```
[clive@ext7144 /etc]$ ps aux ¦ less
```

Create, Monitor and Kill Processes

A sample of the first six lines resulting from running the above command is illustrated below:

```
USER  PID %CPU %MEM VSZ RSS TTY STAT START TIME COMMAND
root  1   0.0  0.2  1104 72 ?   S    08:47 0:03 init
root  2   0.0  0.0    0  0  ?   SW   08:47 0:00 [kflushd]
root  3   0.0  0.0    0  0  ?   SW   08:47 0:00 [kupdate]
root  4   0.0  0.0    0  0  ?   SW   08:47 0:00 [kpiod]
root  5   0.0  0.0    0  0  ?   SW   08:47 0:02 [kswapd]
```

Display Top CPU Processes - top

The top command provides an ongoing look at processor activity in real time. It displays a listing of the most CPU intensive tasks on the system. Unlike ps, top continually updates its display.

You can enter instructions to top as it is running. The following information is provided by typing ? on the top command line.

```
Proc-Top Revision 1.2
Secure mode off; cumulative mode off; noidle mode off
Interactive commands are:

space     Update display
^L        Redraw the screen
fF        add and remove fields
oO        Change order of displayed fields
h or ?    Print this list
S         Toggle cumulative mode
i         Toggle display of idle proceses
c         Toggle display of command name/line
l         Toggle display of load average
m         Toggle display of memory information
t         Toggle display of summary information
k         Kill a task (with any signal)
r         Renice a task
P         Sort by CPU usage
M         Sort by resident memory usage
T         Sort by time / cumulative time
u         Show only a specific user
```

```
n or #    Set the number of process to show
s         Set the delay in seconds between updates
W         Write configuration file ~/.toprc
q         Quit
```

The syntax for top is shown below:

```
[clive@ext7144 /etc]$ top
```

A sample of the first thirteen lines resulting from running the above command is illustrated below:

```
61 processes: 60 sleeping, 1 running, 0 zombie, 0 stopped
CPU states:  1.5% user,  0.7% system,  0.0% nice, 97.6% idle
Mem:  30652K av,  29836K used,    816K free,  15968K shrd,   656K buff
Swap: 72252K av,  24848K used,  47404K free   15760K cached

PID  USER  PRI  NI  SIZE  RSS SHARE STAT  LIB %CPU %MEM   TIME COMMAND
1133 clive 15   0   1032 1032   824    R    0  1.5  3.3   0:00 top
591 root    9   0   8476 5196  1440    S    0  0.9 16.9   1:47 X
1 root      0   0    108   52    36    S    0  0.0  0.1   0:03 init
2 root      0   0      0    0     0   SW    0  0.0  0.0   0:00 kflushd
3 root      0   0      0    0     0   SW    0  0.0  0.0   0:00 kupdate
4 root      0   0      0    0     0   SW    0  0.0  0.0   0:00 kpiod
5 root      0   0      0    0     0   SW    0  0.0  0.0   0:02 kswapd
```

You can see that the top command is using most CPU time, whereas X is using the most memory.

Below are illustrated the processes running when Netscape starts up:

```
11:54am  up  3:07,  2 users,  load average: 0.16, 0.05, 0.01
63 processes: 61 sleeping, 2 running, 0 zombie, 0 stopped
CPU states:  6.2% user, 11.4% system,  0.0% nice, 82.2% idle
Mem:  30652K av, 30116K used,    536K free,  18660K shrd,    656K buff
Swap: 72252K av, 25272K used,  46980K free   15380K cached

PID  USER  PRI  NI  SIZE  RSS SHARE STAT  LIB %CPU %MEM   TIME COMMAND
1150 clive  12   0  2764 2660  2176 R       0 10.2  8.6   0:00 netscape-commun
1168 clive  16   0  3636 3636  3080 S       0  2.8 11.8   0:00 netscape-commun
```

You can see that Netscape is relatively processor and memory hungry.

Each process has a unique identification number (PID) which characterises the process. The command top allows you to kill processes using the **k** interactive command and entering the PID of the relevant process.
To leave top to just press the **q** key.

Create, Monitor and Kill Processes

Job Control

Job control refers to the ability to selectively stop (suspend) the execution of processes and continue (resume) their execution at a later point.

A job is one or more processes started from a single command line. By default, only one job can be run in the foreground. This means that when a job is being executed in the foreground the command line is unavailable. When the job has finished executing the command prompt is reissued.

It is also possible to suspend jobs and/or run multiple jobs in the background, in which case the command line is still available in the foreground, although any output from running background jobs will still be displayed at the terminal.

You can see the jobs currently running or stopped in the background using the **jobs** command. The syntax for the jobs command is shown below:

```
jobs option(s)
```

Common jobs options are:

Option	*Explanation*
-l	Shows the job number, its PID, its status, and its name
-p	Shows just the PID of running jobs

Issuing the jobs command without any options will show a list of all running, stopped and suspended background jobs.

An example of using the job command is illustrated below:

```
[root@redhat /root]# jobs -l
[1]-  1229 Running        tail -n5 -f /var/log/secure
[2]+  1230 Stopped        joe fred
```

In the above example there are two jobs in the background, one running and one stopped.

Create, Monitor and Kill Processes

The table below shows commands specifically associated with job control:

Command	***Description***
COMMAND &	Will cause the specified *COMMAND* to run in the background
[CTRL]+[Z}	Will stop a foreground job and put it into the background
% fg fg %	If there is a single job in the background this will bring it to the foreground and restart it if necessary.
fg *%N* fg *%NAME*	If there are multiple jobs in the background this will bring the job with job number *N* or with *NAME* to the foreground, restarting it if necessary.
bg	If there is a single job stopped in the background this will restart the job in the background
bg *%N* bg *%NAME*	If there are multiple jobs stopped in the background this will restart the job with job number *N* or with *NAME* in the background

An example of job control is illustrated below:

```
[root@redhat /root]# jobs -l
[1]-  1229 Running       tail -n5 -f /var/log/secure
[2]+  1230 Stopped       joe fred
[root@redhat /root]# fg %1
tail -n5 -f /var/log/secure

[root@redhat /root]# jobs
[2]+  Stopped                joe fred
```

The running job is brought to the foreground and then terminated using [CTRL]+[C]. Only the editing job remains stopped in the background.

Create, Monitor and Kill Processes

Another example is illustrated below:

```
[clive@redhat clive]$ sleep 400 &
[1] 1303
[clive@redhat clive]$ sleep 300

[2]+  Stopped                 sleep 300
[clive@redhat clive]$ jobs
[1]-  Running                 sleep 400 &
[2]+  Stopped                 sleep 300
[clive@redhat clive]$ bg %2
[2]+ sleep 300 &
[clive@redhat clive]$ jobs
[1]-  Running                 sleep 400 &
[2]+  Done                    sleep 300
[clive@redhat clive]$
```

In the above example a job, **sleep 400**, is started in the background, whereas a job, **sleep 300** is started in the foreground and then stopped and moved into the background using [CTRL]+[Z]. The sleep 300 job is then restarted and terminates.

Note that Bash will not always warn you that you have background jobs stopped or still running when you try to close a shell. The jobs will just be lost.

Create, Monitor and Kill Processes

Terminate a Process - kill

The **kill** command can be used to control running processes and send a variety of signals to processes.

The syntax for the kill command is shown below:

```
kill -signal pid
```

Signals passed to the kill command can either be symbolic or numeric. You must either own a process, or be root, to signal it.

The table below shows some of the common signals used with the kill command:

Symbolic	*Numeric*	*Description*
HUP	1	Causes the process to re-read its configuration file
KILL	**9**	Forcibly kills the process (nuke)
TERM	**15**	Causes the process to terminate gracefully (close). This is the default action for kill if no signal is passed

Processes are best shut down using the close signal, which gives them time to shut down gracefully. Nuking, which kills a process instantaneously, is normally used as a last resort, when a process does not respond to the close signal.

The use of the kill command to shut down the SAMBA processes smbd and nmbd is illustrated below:

```
root 1207  0.0  3.0 2404 928 ?     S 12:13   0:00 smbd -D
root 1218  0.0  3.2 1984 988 ?     S 12:13   0:00 nmbd -D
root 1226  22.0 3.1 2680 968 pts/0 R 12:15 0:00 ps -aux
[root@ext7144 /root]# kill -TERM 1207
[root@ext7144 /root]# kill -9 1218
```

Processes may also be killed using the **killall** command, which with the **-v** option notify you when the process has been killed and with the **-i** option will ask for confirmation before killing a process.