4 * Daemons and processes

This is an assessed lab. You must write up and submit your answers to this lab for assessment as part of your logbook.

Before you begin this lab, you will need to have completed the * Linux systems and shell scripting lab **and have created the bob user account on the Slackware machine.** It is recommended that you also complete the remaining Linux Tutorials before beginning this lab: Further file handling; Redirection and Wildcards, filename conventions and getting help.

Learning objectives

The aims of this lab are to:

- Introduce the concept of processes, daemons and UNIX signals
- Explore displaying and manipulating running processes, and to gain experience in configuring daemon behaviour (in particular inetd)

By the end of this lab session, you should be able to:

- State the differences between daemons and processes
- Describe the purpose of the inetd super-server daemon
- Identify the directory where daemons and processes are located
- Start, stop and restart daemons and processes
- Modify daemon behaviour through editing configuration files
- Find the process ID number for a particular daemon or process
- List all active daemons and processes on the system

Log in to the Slackware machine as root. Execute finger bob to check that the bob user account exists in the system.

The internet service daemon: inetd

inetd (internet service daemon) is also known as the "super-server daemon" on many UNIX/Linux systems. It manages Internet services, runs at boot time and listens for connections on a number of ports. When a connection comes in, inetd decides which service should be started and starts the application required to handle the request. Having passed the request to another application, it can then becomes idle, waiting for the next request.

For example, when an ftp request comes in, it is received by inetd which starts the ftp daemon. The ftp daemon then takes over and handles the ftp session.

In UNIX/Linux, many Internet-related services are handled by inetd. The idea is that instead of starting every daemon that the system could possibly need, inetd listens for certain requests and starts services as appropriate, thus making better use of the system's valuable resources.

Note: Not all Internet services are started by inetd, for example nfsd is run by the system at boot time (if there is anything to export).

Like other daemons, inetd reads its configuration file (usually located at **/etc/inetd.conf**) as it starts up and "remembers" this configuration from then on. Therefore, if you edit the configuration file, you must restart the inetd daemon in order to ensure that the new changes will take effect.

Because there is only one instance of inetd running for a given UNIX/Linux system, we can simply restart the daemon with a HUP (hang up) signal:

```
kill -HUP process_pid
```

where `process_pid` is the PID (process identifier) number of the daemon to be restarted. For example, the command

```
ps ax | grep inetd
```

will return a line containing the process details for inetd:

```
1686   ?     Ss     0:00    /usr/sbin/inetd
```

so the command

```
kill -HUP 1686
```

would restart inetd. There is also a command called pidof that returns the PID for a process, for example

```
pidof inetd
```

which, in the above example, would return 1686.

Exploring processes and jobs

A process is an executable program identified by a unique PID. To see information about your processes, with their associated PID and status, execute

```
ps
```

A process may be in the foreground, in the background, or be suspended. In general the shell does not return the UNIX prompt until the current process has finished executing.

Some processes take a long time to run and hold up the terminal. "Background"ing a long process has the effect that the UNIX prompt is returned immediately, and other tasks can be carried out while the original process continues executing.

Listing all processes

To list all the processes currently running, execute

```
ps aux
```

A command which lists the most CPU-intensive processes is

```
top
```

which is dynamically updated (press [q] to quit).

Exercise 4.1 Exploring currently running processes

Using ps aux and top (and any other commands you feel would be useful), find out the total number of processes that are currently running. Identify the 10 most CPU-intensive processes and give a brief (one sentence) description of what each of them do. **Hint:** you can use ps aux | less to scroll through the list and man *command* to find out more about a command.

Listing suspended and background processes

When a process is running, suspended or in the background, it will be entered onto a list along with a job number. To examine this list, execute

```
jobs
```

An example of a job list could be:

1. Suspended sleep 100
2. Running netscape
3. Running nedit

To restart (or foreground) a suspended processes, execute

```
fg %job_number
```

For example, to restart sleep 100 in the above example, execute

```
fg %1
```

Executing fg with no job number foregrounds the last suspended process. To resume a suspended process in the background, the bg command is used instead.

Looking up processes

It is useful to be able to look up a particular process/daemon. Sometimes a port scan will do the job.

nmap is a network discovery utility useful for managing network services. It uses raw IP packets to determine the network hosts and the services (application name and version) they offer as well as the type of operating systems (and OS versions) that they are running.

Exercise 4.2 Exploring network processes

Execute the command nmap localhost. Write down all the processes returned and explain their purpose.

To look for a particular process, execute

```
ps aux | grep process_name
```

Starting processes

Processes are listed in **/etc/rc.d/** directory. To start a process/daemon simply lookup the name of the process/daemon and execute

```
/etc/rc.d/process_name start
```

Stopping a currently running process

A currently running process can be terminated using the command kill, which sends a signal to the selected process. There are several different types of signal: signal 9 (SIGKILL) will terminate the process immediately whilst signal 1 (SIGHUP) will wait until all other dependent child processes are terminated.

To stop a running process one must need to know the process ID. Once you know the process ID, then

kill -9 *process_id*   or   kill -SIGKILL *process_id*

will kill the process immediately.

Exercise 4.3 Exploring UNIX signals

Execute the command kill -l. Explain what this command does and use a table to summarise the results (signal number, signal name, short description). You only need to consider signal numbers 1 – 9, and five more of your choice between signal numbers 10 and 31. Please remember to cite any sources that you use to answer this exercise using a recognised academic referencing system (see http://libweb.anglia.ac.uk/referencing/referencing.htm).

Exercise 4.4 Processes and networking

You created the bob user account in Exercise 3.2 on page 22, check that it exists before completing this exercise. When completing this exercise, please remember to provide screenshots in your logbook to demonstrate your results.

1. Edit **/etc/inetd.conf** to enable ftp and telnet. Restart inetd and execute ftp localhost and telnet localhost and log in as bob to see if they are enabled. Make sure you know how to upload, download, delete and rename files. **Hint:** use the vsftpd daemon. Its configuration file is located at: **/etc/vsftpd.conf**.
2. Edit **/etc/inetd.conf** again to disable ftp. Restart inetd and test out your changes.
3. What is sftp and ssh? Why is the use of telnet discouraged in the "real world"?

Exercise 4.5 Optional exercises

1. How could you combine pidof with kill -HUP in order to restart inetd in **one** command? **Hint:** try using quotation marks (such as: 1, ", `).
2. Can you ftp as root? Is it a good idea to be able to ftp as root and why? If so, in what context?
3. From your Windows host machine, ftp and telnet to your Slackware virtual machine.