7

Services and Regular Jobs

If you're running a server, you need to run daemon processes to provide services as they're requested. This module describes the startup and shutdown of daemons, and also how you can run regular jobs on your server to perform housekeeping and other tasks.

7.1 Provision of services via daemons

On Linux systems, services are provided by daemon processes.

Daemons are started when the computer is booted up, in accordance with a number of scripts and directories, and are left running until manually stopped or until the computer is halted.

Some daemons provide local services, such as the printer daemon which allows you to send jobs to the printer and have them wait in a queue while you get on with other work. Some are vital parts of the operating systems, and others are awaiting connections on the network to provide services. For example, the Apache webserver is a daemon process called httpd.

There's a special daemon called xinetd which is a sort of "daemon of daemons". You hope that on a web server there's going to be sufficient traffic to justify a daemon running all the time, or even several copies running so that you don't end up with long queues of users, but other services may be called up rarely. For example, you may decide to offer a time service so that other computers in your organisation can set their system clocks from yours. Great, but how often will people actually call up the service? Pretty rarely, so there's a great case for a service to be provided by xinetd that can start up the time server just as its needed.

7.2 Controlling the startup and shutdown of Daemons

Scripts that start up and shut down daemons are located in the */etc/init.d* directory. They're run when you start your Linux system, and when you halt it. You can also run them manually to control individual daemons on a live system if you wish.

Let's try that out with the web server on our test machine with telnet on port 80:

```
earth-wind-and-fire:/Library/WebServer/live_html grahamellis$ telnet saturday 80
Trying 192.168.200.146...
Connected to saturday.
Escape character is '^]'.
GET
<body bgcolor=white text=red>
The sky is blue<br>
The sea is pink<br>
We chose the wrong developer<br>
for our photos.<br><br>
<a href=right.html>Put this right</a>
</br>
<p>
<a href="form1.html>Link to test form</a>
Connection closed by foreign host.
earth-wind-and-fire:/Library/WebServer/live_html grahamellis$
```

OK, the service is live. Let's stop the service

```
[root@saturday root]# /etc/init.d/httpd stop
Stopping httpd:                                            [  OK  ]
[root@saturday root]#
```

And if we try to reach the service now:

```
earth-wind-and-fire:/Library/WebServer/live_html grahamellis$ telnet saturday 80
Trying 192.168.200.146...
telnet: connect to address 192.168.200.146: Connection refused
telnet: Unable to connect to remote host
earth-wind-and-fire:/Library/WebServer/live_html grahamellis$
```

```
Let's restart it:

[root@saturday root]# /etc/init.d/httpd start
Starting httpd:                                            [  OK  ]
[root@saturday root]#

And test again:

earth-wind-and-fire:/Library/WebServer/live_html grahamellis$ telnet saturday 80
Trying 192.168.200.146...
Connected to saturday.
Escape character is '^]'.
GET
<body bgcolor=white text=red>
The sky is blue<br>
The sea is pink<br>
We chose the wrong developer<br>
for our photos.<br><br>
<a href=right.html>Put this right</a>
</br>
<p>
<a href="form1.html>Link to test form</a>
Connection closed by foreign host.
earth-wind-and-fire:/Library/WebServer/live_html grahamellis$
```

System shutdown

When shutting down, you are encouraged to use the **shutdown** command, as it generates warning messages for any users logged in via services capable of handling such messages, and it also takes a time parameter to specify when the shutdown is to occur and a directive as to what the system's to do when that time's elapsed.

-h	shutdown and halt
-r	shutdown and reboot
-k	just generate the messages (just **k**idding)

time parameter examples:

now	straight away
+10	in 10 minutes
15:00	at 15:00

It also sets up a file to advise certain daemons not to allow logins in the period before the shutdown.

Here's an example of shutdown:

```
[root@saturday root]# shutdown -h 14:20 "Down at 14:20 for approx 10 minutes - moving computer to
different mains outlet"

Broadcast message from root (pts/0) (Mon May 31 14:11:43 2004):

Down at 14:20 for approx 10 minutes - moving computer to different mains outlet
The system is going DOWN for system halt in 9 minutes!

Broadcast message from root (pts/0) (Mon May 31 14:12:43 2004):

[etc.]

Down at 14:20 for approx 10 minutes - moving computer to different mains outlet
The system is going DOWN for system halt in 1 minute!
```

```
Broadcast message from root (pts/0) (Mon May 31 14:20:43 2004):

Down at 14:20 for approx 10 minutes - moving computer to different mains outlet
The system is going down for system halt NOW!
[root@saturday root]#
```

And here's the effect on a telnet session:

```
[trainee@saturday trainee]$
Broadcast message from root (pts/0) (Mon May 31 14:11:43 2004):

Down at 14:20 for approx 10 minutes - moving computer to different mains outlet
The system is going DOWN for system halt in 9 minutes!

[trainee@saturday trainee]$ logout
```

Run states and init

Linux systems have a "run state – a number which defines the current operational state, and which can be used as a parameter to the **init** command to request a change. For example

init 0	halt
init 1	switch to single user mode
init 3	full multi-user mode
init 6	reboot

This run level is also used at startup via the file */etc/inittab* and the script **/etc/rc.d/rc** to control which actions are taken. The */etc.init.d* directory that you saw a few minutes ago is used to provide each of the scripts to run at startup, and they're run in asciibetic order. That's why the startup scripts have strange numbers in front of them.

```
[root@saturday rc3.d]# ls S*
S00microcode_ctl  S13portmap   S56rawdevices    S90crond
S05kudzu          S14nfslock   S56xinetd        S90cups
S08ip6tables      S17keytable  S59hpoj          S90FreeWnn
S08ipchains       S20random    S60vsftpd        S90xfs
S08iptables       S24pcmcia    S80sendmail      S95anacron
S09isdn           S25netfs     S80spamassassin  S95atd
S10network        S26apmd      S85gpm           S97rhnsd
S12syslog         S28autofs    S85httpd         S99local
S13irqbalance     S55sshd      S90canna         S99mdmonitor
[root@saturday rc3.d]#
```

Automating daemon controls via chkconfig

The **chkconfig** utility lets you control which daemons are (and are not) started at system reboot. Syntax summary:

```
usage:   chkconfig --list [name]
         chkconfig --add <name>
         chkconfig --del <name>
         chkconfig [--level <levels>] <name> <on|off|reset>
```

So:

```
[root@saturday rc3.d]# chkconfig --list
microcode_ctl   0:off   1:off   2:on    3:on    4:on    5:on    6:off
kudzu           0:off   1:off   2:off   3:on    4:on    5:on    6:off
syslog          0:off   1:off   2:on    3:on    4:on    5:on    6:off
netfs           0:off   1:off   2:off   3:on    4:on    5:on    6:off
network         0:off   1:off   2:on    3:on    4:on    5:on    6:off
random          0:off   1:off   2:on    3:on    4:on    5:on    6:off
rawdevices      0:off   1:off   2:off   3:on    4:on    5:on    6:off
pcmcia          0:off   1:off   2:on    3:on    4:on    5:on    6:off
saslauthd       0:off   1:off   2:off   3:off   4:off   5:off   6:off
keytable        0:off   1:on    2:on    3:on    4:on    5:on    6:off
apmd            0:off   1:off   2:on    3:on    4:on    5:on    6:off
atd             0:off   1:off   2:off   3:on    4:on    5:on    6:off
gpm             0:off   1:off   2:on    3:on    4:on    5:on    6:off
autofs          0:off   1:off   2:off   3:on    4:on    5:on    6:off
iptables        0:off   1:off   2:on    3:on    4:on    5:on    6:off
irda            0:off   1:off   2:off   3:off   4:off   5:off   6:off
isdn            0:off   1:off   2:on    3:on    4:on    5:on    6:off
sshd            0:off   1:off   2:on    3:on    4:on    5:on    6:off
portmap         0:off   1:off   2:off   3:on    4:on    5:on    6:off
nfs             0:off   1:off   2:off   3:off   4:off   5:off   6:off
nfslock         0:off   1:off   2:off   3:on    4:on    5:on    6:off
sendmail        0:off   1:off   2:on    3:on    4:on    5:on    6:off
rhnsd           0:off   1:off   2:off   3:on    4:on    5:on    6:off
crond           0:off   1:off   2:on    3:on    4:on    5:on    6:off
anacron         0:off   1:off   2:on    3:on    4:on    5:on    6:off
httpd           0:off   1:off   2:on    3:on    4:on    5:on    6:off
[etc.] ...
xinetd          0:off   1:off   2:off   3:on    4:on    5:on    6:off
[etc.] ...
xinetd based services:
        chargen-udp:    off
        rsync:  off
        chargen:        off
        daytime-udp:    off
        daytime:        off
[etc.] ...
        rsh:    off
        swat:   off
        ntalk:  off
        talk:   off
        telnet: on
        tftp:   off
```

7.3 Regular jobs via crontab

You'll probably have bookkeeping jobs you want to perform on a regular basis, and automatically. For example, you may want to rotate your application's log files on a daily basis, backup a dynamic directory every two hours during the day in the working week, and run a script to change an image at the start of most months.

The cron daemon runs all the time, and controls batch and times jobs; the batch system is of much more use on heavy computational systems than webservers, but timed jobs are "worth their wait in gold"

Each user (unless denied the facility) has a single crontab – a table of jobs that are scheduled when the system clock matches the settings given in the file. Each line of the file is a series of time parameters followed by a single line command. Use

crontab -l to list out your current settings, and **crontab -e** to edit it. Do NOT type in the word "crontab" on its own unless you want to delete all current regular jobs!

Let's see what regular jobs our trainee user has set up:

Important to note: Do NOT type in the word "crontab" on its own unless you want to delete all current regular jobs

```
[trainee@saturday etc]$ crontab -l
# DO NOT EDIT THIS FILE - edit the master and reinstall.
# (/tmp/crontab.7423 installed on Fri May  7 14:33:14 2004)
# (Cron version -- $Id: crontab.c,v 2.13 1994/01/17 03:20:37 vixie Exp $)
34 14 * * * /usr/local/mysql/bin/mysqldump -hsaturday -uwebuser -pabc123 www > /home/www/backup/
mysqlbackup.txt
[trainee@saturday etc]$
```

At 34 minutes after the 14th hour, every day of the month, every month of the year, every day number of the week run the **mysqldump** command (which takes a database backup). The backup is to be taken off the www database and stored in a backup directory.

Let's add in the other regular jobs we wanted:

```
[trainee@saturday etc]$ crontab -e
crontab: installing new crontab
[trainee@saturday etc]$

[trainee@saturday etc]$ crontab -l
# DO NOT EDIT THIS FILE - edit the master and reinstall.
# (/tmp/crontab.2842 installed on Mon May 31 15:24:38 2004)
# (Cron version -- $Id: crontab.c,v 2.13 1994/01/17 03:20:37 vixie Exp $)
34 14 * * * /usr/local/mysql/bin/mysqldump -hsaturday -uwebuser -pabc123 www > /home/www/backup/
mysqlbackup.txt
5 0 * * * mv /home/www/private/logs/access.log /home/www/private/logs/l_`date`; touch
 /home/www/private/logs/access.log
27 8-18/2 * * 1-5 tar czf /home/www/private/backups/`date`.tgz /home/www/data
7 6 * 1,3-7,9-12 * /home/www/programs/switchimage

[trainee@saturday etc]$
```

Notes:

- A single figure for one specific minute/hour/day etc.
- Hyphenated ranges, comma-separated lists allowed
- The **/2** says "every 2" thus **8-18/2** is every 2 hours from 8am to 6pm
- It's typical to specify full paths
- More complex commands can be run as scripts, for example our **switchimage**
- You can't be sure when a job will run/complete; cron schedules it at the given time, but if there are already other regular jobs running it may get queued
- Any output is emailed back to you. If there is no output to do, NOT get an empty email