

Misc Linux Notes

Greg Baker – greg.baker@ifost.org.au

December 25, 2000

Contents

1	How to get extra information
2	Other Text Editors
3	Office Applications
4	Big Central Servers
5	Cron
6	Printing
7	Bash Shell Discussion
7.1	The Program [.
7.2	Loops and Conditions
7.3	Environment Variables
7.4	Backticks
8	Programs and Suggestions
9	Sites to Visit

1 How to get extra information

`man` pages are not the easiest things to read. They take a lot of getting used to. You may find yourself juggling between a HOWTO document, web pages, a book or to, and reading snippets of information of from the `man` pages. Eventually it does sort itself out, though. After a while you will be familiar with the kind of “Linux” way of doing things, and then things get quite comprehensible, really.

Anyway, even that doesn’t help you if don’t even know where to start. The first thing to do is to generate an index of all the keywords in all the `man` pages. Do this:

```
1 /usr/sbin/makewhatis
```

This will take some time to complete if you run it manually; fortunately it is run automatically once a week from a script in `/etc/cron.weekly`.

2 Now you can use `apropos`. You run it with one command line argument: something to look for. It will print out the names and descriptions of `man` pages which mention this word. If this give tons of output, try using `grep` to show you just some relevant lines. For example, try

```
4 apropos format
```

4 Try looking through this for something to do with formatting a floppy disk. Now try:

```
5 apropos format | grep floppy
```

```
6
```

2 Other Text Editors

6 Anything that can edit text that you like using is a good one. Here are some favourites:

- `emacs` – big and bloated, but great for programmers who like to have syntax highlighting and parenthesis matching
- `xemacs` – same as `emacs`, but nicer-looking
- `jed` – small and simple
- `pico` – your grandmother would cope with this
- `joe` – feels like the old classic WordStar, which is the same as the Borland development environment

- `vi` – available on every Unix; completely incomprehensible for the first hundred times you use it, and then becomes really quite natural.
- `kwrite` – advanced editor for the KDE environment; has some of the features of `emacs`
- `gedit` – default graphical editor for the Gnome environment.

3 Office Applications

Here is a quick grab bag of word processing / spreadsheet / database / etc. programs that do some or all of the functions of Office 2000.

Commercial (paid-for) options:

- ApplixWare – quite a big, full-featured office suite. Designed to run in Unix environments and heavily used in high security (e.g. defence and the military) environments. **More info:** www.applix.com
- Corel Office – the makers of WordPerfect and all that. It's meant to be a complete clone of Microsoft's office suite. **More info:** www.corel.com

Free options:

- Siag – A very extensible system which includes interpreters for Perl, Python, Tcl and Scheme. Despite the component names (Pathetic Writer, Scheme in a Grid) it is quite mature, but somewhat disorienting for non geeks. **More info:** siag.nu
- LaTeX – not for the faint-hearted, \LaTeX is the grand-daddy of text processing and publishing tools. It is in *very* wide use in the academic community and produces highly elegant output. There are friendly-ish interfaces to it called KLyX and LyX. It is installed with most Linux distributions. For a tutorial, run `xdvi 'locate lshort.dvi'`.
- AbiSuite – consisting of AbiWord and Gnumeric (just a word processor and spreadsheet). It is not complete yet, but does work on a wide very range of platforms – BeOS, MS-Windows

and most Unices (including Linux). Parts of it usually installed with most distributions.

- StarOffice – has become free since its authors (StarDivision in Germany) were bought out by Sun Microsystems. It also is available for a wide range of platforms and does a reasonable job of importing many Office formats. Download it from www.openoffice.org if it is not already installed.
- KOffice – not quite complete, but very elegant. It is part of the KDE2 suite (www.kde.org). The KDE2 environment is a very friendly environment for new users.

If you get desperate, you could try running a MS-Windows application in wine (www.winehq.com). It probably won't work fully, but you never know your luck. Failing that, try VMWare, which will let you run a whole x86 PC under Linux – so you could run a couple of copies of Windows 95 and a few WinNT systems all *inside* your Linux system. Or its free alternative Plex86 (www.plex86.org). Inside them, you could run anything you want.

As a last option, there is Win4Lin, which is a commercial product that lets you install MS-Windows 98 inside Linux, and run it as a client program.

4 Big Central Servers

A common configuration that is very easy to manage and very cheap to run is to buy one¹ big computer to run Linux on, and then use some throw-away computers as dumb graphical terminals. It doesn't really matter how old the throw-aways are, since all they have to do is just fill the screen in ways dictated to it by the big server. This extends the life of old PCs.

Since every X-windows program can either run locally (i.e. the processing is being done where the displaying is being done) or remotely (i.e. the processing is done on one machine, but it is displayed elsewhere), it is *completely* transparent to users that there mouse movements and keyboard events are actually being sent across a network.

¹Or better still, buy two servers – then one of them can be a stand-by.

On your main server system, you will need to configure the graphical login program to respond to requests from other computers. This is quite simple: edit `/etc/X11/gdm/gdm.conf` – there is a section marked `[xdmcp]`, in which there is a variable `Enable` which will be set to 0. Set it to 1.

You might vaguely want to consider looking through the `[security]` section and consider whether or not to `AllowRoot` and how much of a `RetryDelay` there is in the event of an incorrect username or password.

You will now need to restart `gdm`. It gets respawned by `/sbin/init` because it is configured that way in `/etc/inittab`. Thus, if you kill it, it will be restarted. Use `ps -ef | grep gdm` to find its process id, and then kill it.

Now on the scummy second hand machines – do as minimal a Linux install as you like – just make sure you include X11! Edit `/etc/inittab` and replace the line

```
x:5:respawn:/etc/X11/prefdm -nodaemon
```

with

```
x:5:respawn:/usr/X11R6/bin/X -query big
```

(Obviously, replace *big* with the real name of the computer!) Reboot the desktop machine, and you should get a graphical login to *bigserver*.

Some rules of thumb:

- You only need about 12Mb RAM on the desktop machines – having more will serve no purpose, since no significant programs will be running there.
- Anything much slower than a 486 at 50MhZ gets to be a bit of drag, as it often can't keep up with the drawing tasks it gets given!
- If in ordinary usage a user needs 64Mb RAM on a MS-Windows machine, they will probably only 48Mb or so on Linux. With Linux running on a shared big server, however, there are further advantages – any binaries (programs) that are being run by two people have their base memory shared, which can often save 10-15 MB per user.

- A single-user system only spends about 10-20 per cent of its time actually doing anything – the remaining 80-90 per cent it is just waiting for something to happen.
- Each user will need around 100kB/s of bandwidth. This should vaguely indicate that 40 users on a 10Base-T network would be just OK. If you can use 100Mb ethernet everywhere, all the better as it will help minimise latency.
- **Summary:** A dual-processor 800MhZ computer with 768 Mb RAM and a Fast Ethernet card should be able to support 20 users quite happily with little noticeable slowness. Of course, your mileage may vary, and it depends very much on the kinds of users that you have.

5 Cron

You can schedule any job to run by either

- Putting the program into one of `/etc/cron.hourly`, `/etc/cron.daily`, `/etc/cron.weekly`, or `/etc/cron.monthly`. This is only an option for the `root` user (no one else will have permissions), and it is specific to the RedHat and SuSE distributions.
- Running `crontab -e` to edit your personal scheduled jobs.

The crontab format basically consists of 6 fields, the first five of which specify when the job should be run –

1. What the minute hand of the clock should be reading
2. What the hour hand of the clock should be reading
3. What the date (i.e. day of the month) should be
4. What month it should be
5. What day of the week it should be
6. The command to run.

Here is the example (partly from the man page):

```
# run five minutes after midnight, every day
5 0 * * * $HOME/bin/daily.job >> $HOME/tmp/out
# run at 2:15pm on the first of every month
15 14 1 * * $HOME/bin/monthly
5 4 * * 0 echo "run at 5 after 4 every sunday"
0,20,40 * * * * /usr/local/bin/diskspace_check
```

```
if not(config.has_key('port')):
    sys.exit('port not found')
printer_ip = config['printer_ip']
port = string.atoi(config['port'])
s = socket.socket(socket.AF_INET,
                  socket.SOCK_STREAM)
s.connect(printer_ip,port)
s.send(sys.stdin.read())
s.close()
```

6 Printing

Printing set up on RedHat is really quite simple. Either by searching through menus, or by typing it on the command line, run `control-panel`. There is a printer icon in the list – it runs `printtool`.

There is a major bug in RedHat 6.2 for direct-to-port printing (such as you would do for LaserJets). The file

`/usr/lib/rhs/rhs-printfilters/directprint` was left out. Here is my quick re-implementation of it:

```
#!/usr/bin/python
import string
import socket
import sys
import os

accounting_file=sys.argv[-1]
spool_dir=os.path.dirname(accounting_file)
config_file=os.path.join(spool_dir, '.config')
config = {}

for line in open(config_file).readlines():
    if len(line)==0:
        continue
    if line[0] == '#':
        continue
    parts = string.split(line,',',2)
    if (len(parts)) != 2:
        continue
    lhs = string.strip(parts[0])
    rhs = string.strip(parts[1])
    if len(rhs)==0:
        continue
    if rhs[0]=='"' or rhs[0]=="'":
        rhs = rhs[1:-1]
    config[lhs] = rhs

if not(config.has_key('printer_ip')):
    sys.exit('printer_ip not found')
```

There may be a patch for this by the time you read this. If not, just use something similar to the above, and make sure that it is executable.

Printer	Use
Parallel port attached	Local
Attached to Unix print spooler	Remote/LPD
Attached to WinNT print spooler	SMB
Attached to NetWare print spooler	NCP
HP LaserJet on a network	Direct ²
Kyocera, Xerox, Canon, Epson on a network	Remote/LPD
LPD-capable HP LaserJet (optionally)	LPD

The May 2000 issue of *Linux Journal* has an article on how the print filtering system works, if the other sources of documentation (such as the `printcap` man page) are insufficient..

If you want a fun challenge exercise, write a print filter that uses `ps2pdf` to take postscript files and turn them into PDF files. For an extra challenge, mail these as attachments to the user whose print job it is.

For very large networks, or ones with very complicated spooling mechanisms, some people like to install LPRng (from www.astart.com).

7 Bash Shell Discussion

7.1 The Program [

The [can take all sorts of unusual arguments. Here are some examples:

- [-f filename] – is filename a file?

- [`-d filename`] – is filename a directory?
- [`-e filename`] – does filename exist?
- [`10 -lt 12`] – is 10 less than 12?
- [`$X -ge 12`] – is the variable `$X` greater than or equal to 12?
- [`"$answer" = "y"`] – is the variable `$answer` the same string as the string “y” ?

If [completes successfully, then it won't print anything, but the shell variable `$?` will be set to 0. If there were any problems (i.e. if the statement was false, or there was a syntax error), `$?` will be set to something other than 0.

Actually, this works for any kind of program. `grep` for example,

- If it finds a match, it completes successfully (i.e. `$?` is set to 0)
- If it doesn't find a match, it completes with an error code of 1 (i.e `$?` is set to 1)
- If one of the file name arguments given to `grep` didn't exist, then it completes with error code 2. (`$?` is set to 2).

7.2 Loops and Conditions

if's first part consists of a program to run – if it completes with error code 0 the `then` part will get run, otherwise it will try `elif` and `else` parts respectively.

Here's an example:

```
echo "How old are you?"
read age
if [ $age -lt 18 ]
then
    echo "You're a youngster"
elif [ $age -gt 120 ]
then
    echo "I think you're lying"
else
    echo "Hello there $age year-old"
fi
```

`while` works similarly – as long as the statement is true, we keep on going. Here's an example

```
X=0
while [ $X -lt 10 ]
do
    echo "This is message $X"
    let X=X+1
done
```

`for` works through its arguments one by one and does the part between `do` and `done`. This is often used in conjunction with *globbing*. “Globbing” is a name used for the use of wildcard patterns, most of which can be guessed just by looking at them:

- `*.html` – any `.html` file
- `backup*.dat` – any file starting with `backup` and ending with `.dat`
- `file0??.txt` – the `?` means any single character.

Here's an example:

```
for f in *.c
do
    if [ -f $f ]
    then
        cp $f $f.bak
    else
        echo "$f is a directory?"
    fi
done
```

7.3 Environment Variables

Environment variables are a magical extra attribute associated with each process. When one program starts another one, the second one gets a complete copy of the first one's environment variables. This second program could modify it or leave it intact.

You can examine your environment by running `env`. Note that if one process decides to change its environment, it *can not* affect any other process that is already running. And when it exits, that environment change also disappears. If you want to make it

“permanent”, you need to fake the change by writing it to a file somewhere that gets read on the next startup. (For example, `bash` reads `.bash_profile` at login, so you can do environment set up there).

Here is a brief summary of environment access methods in various languages:

Lang.	Examine	Modify
C/C++	<code>getenv()</code>	<code>setenv()</code>
Perl	<code>\$ENV{...}</code>	<code>\$ENV{...}</code>
Python	<code>os.environ</code>	<code>os.environ[...]</code>
Shell	Same as var	Same as var

For the shell to know that from now on you want a particular variable name to be a handle for accessing the equivalent environment definition, you need to `export` it.

Here is a sample for the `PATH` environment variable:

```
{export PATH=/bin:/usr/bin:/usr/X11R6/bin}
```

Here it is again, done a little differently:

```
PATH=/bin:/usr/bin:/usr/X11R6/bin
export PATH
```

7.4 Backticks

Many programs print to standard output. The shell can capture this and use it as if you had typed it in. This is done with backward quotes (```), or with the construction `$(...)`.

Here’s a quick example that uses `rcp` to synchronise a file between several machines, whose names are in a file `/tmp/computers`

```
#!/bin/bash

cd /www
for computer in `cat /tmp/computers`
do
  rcp index.html $computer:/www
done
```

8 Programs and Suggestions

- `lastb` – check the list of failed bad logins. **Suggestion:** schedule a job to mail this to you every day.
- `df` – check the amount of disk space free on every filesystem. **Suggestion:** schedule a

job every 10 minutes to check this and alert you when something starts getting full.

- `mrtg` – lets you graph utilisation on router links. It is not part of the base Red Hat install. Actually, it can graph almost anything, so if you have an SNMP-enabled device (or you have installed the `ucd-snmp` tools, you can graph all sorts of things. **Suggestion:** make a pretty graph of your internet link utilisation.

9 Sites to Visit

- Every Thursday afternoon, `lwn.net` gets update with the latest “Linux Weekly News”. This is incredibly informative.
- Regularly visit `www.securityfocus.com` and `www.redhat.com/errata.html` to get the latest on security fixes and bug reports. If you don’t, hackers will, and will happily try to exploit them.
- New software is announced in `freshmeat.net`. There can be anywhere up to 50 announcements per day. Many of these are bug fixes and security announcements, but every now and again you can find something extraordinarily useful that can save months of work. Most items include a link for downloading RPM files.
- If you are using an RPM-based distribution, such as RedHat, SuSE, or Mandrake, you can download almost any RPM from `rufus.w3.org`.
- And for recreational insults about Microsoft, or for the latest buzz in new ideas and new developments in all things geeky, visit `slashdot.org`.