

Exercise 9: Getting more advanced with the command line

Hanne Munkholm <hm@itu.dk>
IT University of Copenhagen

January 24, 2005

1 File ownership and access control

- A user on a Linux system belong to one or more groups. On a Fedora system, each user has a group for himself, with the same name as his user name. Users can be organized in more meaningful groups by the system administrator.

A user's files are protected from the other users, and the system files are protected from all the ordinary users. The system administrator on a machine has a special user account called "root", from which he has access to all the files on the system.

- All files on a Linux system are owned by a user and a group. Create a new file with the "touch" command, and look at it with "ls -l":

```
$ touch linuxptest.txt
$ ls -l linuxptest.txt
-rw-rw-r-- 1 hm hm 0 Jan 28 15:46 linuxptest.txt
```

Starting from the *right*, the fields are: File name, time and date of creation, file size (here it is zero because we just created it with the "touch" command), group and user owning the file (both "hm" in this case), link count, and file permissions.

- Let us have a closer look at the file permissions. It consists of four parts:

```
- rwx rwx rwx
  user group other
```

The first part says something about the file type. In the example, it is a minus sign, which means an ordinary file. It can also be a "d" for directory, an "l" for symbolic link, a "b" or a "c" for different kind of devices, etc.

The next three fields are equivalent, and show the permissions for the file's owner (**u**ser), **g**roup and all **o**thers, respectively. They consist of each three places (bits), which can contain a letter or a minus sign. The three bits represent the permission to: r (read), w (write) and x (execute) the file. If a bit is set (if there is a letter), the given action is allowed for the given user(s). If there is a minus sign, the action is not allowed.

- In the example before, the permissions were `rw-rw-r--`. The owner of the file is allowed to read and write the file. Members of his group are also allowed to read and write, while everyone else is only allowed to read it.
- Try looking at the ownership and permissions of some more files with `ls -l`.
- You can change the permissions on a file with `chmod` (change mode). Syntax:
`chmod <permission mode> <file name>`
 The permission mode describes the permission changes you want to make, in the form `[ugoa][[+|=][rwx]`:

Who	Permission bit	Action
u=user (owner)	r=read	"+" add permission
g=group	w=write	"-" remove permission
o=others	x=execute	"=" set permissions equal to
a=all		

- Try the following examples and run `ls -l` after each one, to see what happened:

```
$ chmod a+x linuxtest.txt (allow all to execute the file.)
$ chmod go-w linuxtest.txt (remove write permissions from group and others.)
$ chmod 644 linuxtest.txt (set permissions to rw-r--r--.)
```

Can you figure out the last one? The file permissions are represented by an octal number. `r=4`, `w=2`, `x=1`. See `man chmod` for further explanation.

2 Finding files

which, whereis, locate: Find out where a file is on the system

`Which`, `whereis` and `locate` are useful for fast searching. Common for them is that they don't search the entire system, but only a specific part.

Try the following examples and look at the output:

```
$ which chmod
$ whereis chmod
$ locate chmod
```

which will search your path for an executable called `"chmod"`.

whereis will tell you where to find the binary file and the manual page for the `"chmod"` command.

locate will search for any file name on the system containing `"chmod"`. It is fast because it does not search the actual file system. Instead, it uses a database, which is recreated once a day. This means that `locate` will often fail to find a recently created file.

See `man which`, `man whereis` and `man locate` for further information.

find: Search for file

”Find” is used to search your system for files matching a certain criteria. ”Find” does not use a database created in advance, like ”locate”, and can therefore take some time. ”Find” is a complex command, and we will only look at a few examples here. For further information, see ”man find”.

```
$ find ./ -name linuxtest.txt (find files named "linuxtest.txt" under current directory.)
```

```
$ find /var -cmin -30 2>/dev/null (find files under /var, for which status has changed within 30 minutes.)
```

file: Determine file type

Try the following commands:

```
$ file /bin/ls
$ file /etc/printcap
$ file linuxtest.txt
$ file `which firefox`
```

3 Job Control

ctrl+c Break (terminate) running command

Start Firefox with the command: `$ firefox`. Click the terminal window to focus. You will notice that you haven't got your command prompt back. Firefox is running "on top" of your command prompt. Press `ctrl+c`. Firefox will be closed and you get your command prompt back.

ctrl+z Freeze running job.

Start Firefox again with the command `$ firefox`, and click the terminal window to focus. Press "ctrl+z". You will see the following output:

```
[1]+ Stopped      firefox
```

The Firefox window will still be there, but it will be dead - frozen.

jobs Display job numbers and current state of jobs

If you run the "jobs" command, you will get the same information: Job number 1 is firefox, and it is stopped.

fg Put job in the foreground

Type `fg %1`. Now the Firefox window is active again.

bg Put job in the background

Freeze Firefox again with "ctrl+z". This time, put it in the background with the command `bg %1`. Now firefox is running in the background, just as if we had started it with `firefox &`.

ctrl+d Exit current shell, or finish typing interactive command

Press "ctrl+d". Did your terminal window close? That is what ctrl+d does - unless you are typing an interactive command. Open a new shell, and type: `$ cat`
The "cat" program will be waiting for input, and will repeat anything you type. To finish it, press "ctrl+d".

4 Process control

Job control is nice for the jobs started from the current shell, but to control all our processes on the system, we need something more. (Only root can control all the processes on the system, you can control only those owned by you.)

ps Report process status

```
$ ps -aux
```

list all processes on the system. Now we want to find our Firefox from before:

```
$ ps -aux |grep firefox
hm 795  7.3 11.5 23484 14584 pts/6 S  17:52 0:01  \
                                     /usr/lib/firefox-1.0/firefox-bin
```

To see what all the fields mean, type `$ ps -aux |head` or see "man ps". We will only look at a few fields here:

```
USER PID ...  COMMAND
hm 795 ...  /usr/lib/firefox-1.0/firefox-bin
```

kill Terminate a process

Kill the Firefox process by typing "kill", followed by its PID (process id), which you found in the "ps" output. In the example it is 795 - yours will be a different number!

```
$ kill 795
```

Normally the kill command kindly asks the program to shut down. But sometimes a program has gotten itself locked up and cannot be shut down gracefully. In that case, you can usually kill it by force with the command "kill -9 PID".

5 System resources

df Disk free

```
Run $ df -lh
```

if you want to see how much free space there is on the hard disk of the machine. If you run `df` with no arguments, you will get all the network drives as well. The "-h" option makes "df" display MB and GB instead of 1k-blocks.

du Disk usage

Go to your home directory with the command "cd", and see how much space your home directory takes up, with the command

```
$ du -sh ./
```

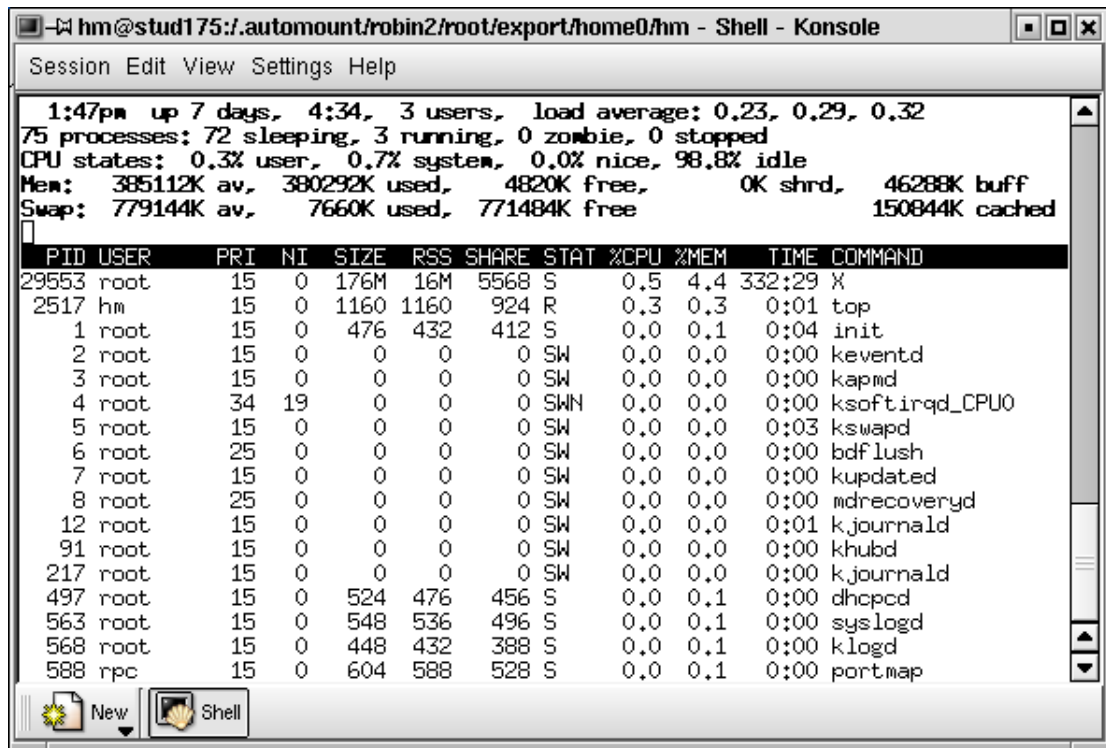
See "man du" for further information.

free Display memory usage

Displays the total amount of free and used memory on the system, in KB. See "man free" for more information.

top Display the processes using most CPU

"Top" is an interactive program that displays the top 10 CPU using processes. "Top" also displays the systems uptime, the CPU load, and the RAM and swap usage, and the state of the processes. Press "q" to quit.



6 Text file manipulation

A few useful commands for text file manipulation:

head Display first 10 lines of a file

We already tried this in `ps -aux | head`. Can you make it display only the first line? See "man head".

tail Display last 10 lines of a file

`tail -f` is neat if you want to keep looking at the last few lines of a log file that keeps changing. See "man tail".

sed Stream editor

”Sed” is a complex and powerful command. You can edit a file ”on the fly” from the command line or from a script, replacing one word or expression/pattern with another. For details, see [3]. A simple usage of sed:

Make a file, linuxtest2.txt, containing some text.

```
$ echo "A cow in the stable in the summer" > linuxtest2.txt
$ echo "in the middle of June when all the" >> linuxtest2.txt
$ echo "other cows are out in the field." >> linuxtest2.txt
```

Use ”sed” to replace the word ”cow” with ”computer”, and save as linuxtest3.txt.

```
$ cat linuxtest2.txt | sed 's/cow/computer/' > linuxtest3.txt
```

diff Find differences between two files

Finding the difference between the two files from the previous example:

```
$ diff linuxtest2.txt linuxtest3.txt
```

Diff is useful when you are writing books or code and want to see which changes was made between two versions.

cut Cut out section of each line of input

”/etc/group” is a colon separated file:

```
$ tail /etc/group
```

Use ”cut” to see only the group names (the first field):

```
$ cut -d: -f1 /etc/group |tail
```

The ”-d:” means that you use the ”:” character to limit the fields, and ”-f1” means you want field number one. See ”man cut” for more information.

7 Hard links and symbolic links

In Linux you can make a link to a file, either a ”hard link” or a ”symbolic link”. You make a ”hard link” like this:

```
$ ln linuxtest.txt link2.txt
```

You make a symlink with the ”-s” option to ln:

```
$ ln -s linuxtest.txt link3.txt
```

In section 1 on page 1, we mentioned ”link count” when we were looking at the output from ”ls -l”. Look at the files you have just created, with ”ls -l”, and find out what ”link count” is.

8 Shell scripts

All the commands you have learned can be run from the command line. But you can also put commands together in a file, making a shell script.

- Using your favorite editor or the "echo" command, make a file called `myspace.sh`, with the following content:

```
#!/bin/sh
# My first shell script
# Calculates how much disk space my home directory use

echo "Calculating disk space"
echo " "

cd
SPACE=`du -sh ./ 2>/dev/null | cut -f1`
echo "My homedir is $SPACE".
echo " "
```

- Make the shell script executable:

```
$ chmod u+x myspace.sh
```

- Run the shell script:

```
$ ./myspace.sh
```

References

- [1] *The on-line manual pages for each command.*
- [2] Peter Toft *Linux - Friheden til at lære Unix.*
<http://www.linuxbog.dk/unix/bog/index.html>
- [3] Arnold Robbins *UNIX in a Nutshell.*