Exercise 6: Get started with the command line

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

January 24, 2005

1 The command line

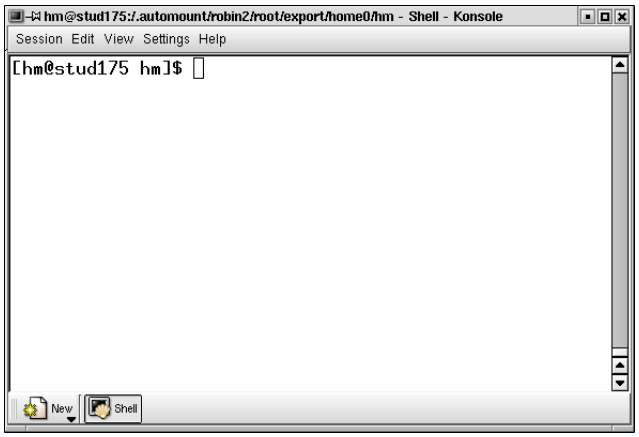
The command line was the primary user interface to computers, before the window-based point and click graphical desktops existed. You interfaced with the computer by typing commands on a text terminal, that way telling the computer which programs to run, and how to behave.

Why do we still use the command line today? Because it is much more efficient for certain things than a point and click interface is. But it also takes a bit more effort to get started - you have to learn some commands. However, when you have learned a certain amount of basic commands, and you are getting the hang of it, you will get a control and understanding of your computer that you usually don't get in the point and click world. It's your choice. There will always be more to learn...

2 Starting a terminal window

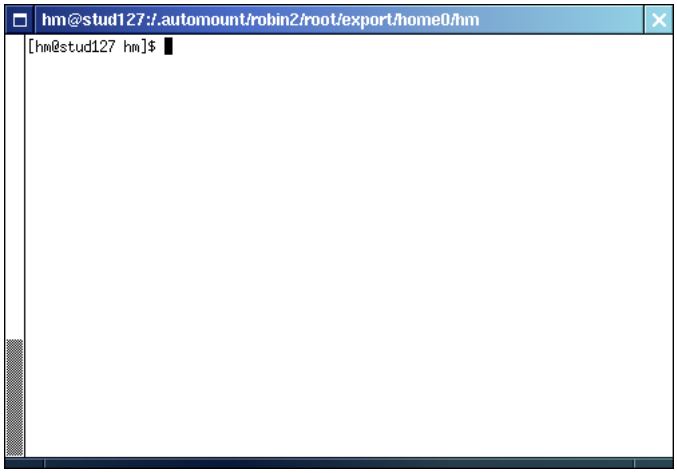
- There are many different kinds of terminal windows on a Linux system. They might look slightly different, but one is as good as the other.

  If you are using KDE, you can work with KDE's "konsole". program. Open it from the KDE-menu "→ System Tools → Terminal". The terminal window will look like this:



  In GNOME, use the GNOME Terminal: "GNOME-menu → System Tools → Terminal". It looks almost the same as "konsole".

If you are using Window Maker, start the ”xterm” program by double clicking the  icon in the Window Maker dock. The terminal window will look like this:



- The command prompt looks like the following (except ”hm” and ”stud127” is replaced with your user name and machine name):

  ```
  [hm@stud127 hm]$
  ```

  In the rest of this exercise, we will refer to the command prompt by a single $ sign, followed by the command you should execute.

  You type a command, then hit ”enter” to execute it. When you get the command prompt back, you can type the next command.

3 The shell

The program that is displaying the command prompt and will interpret the commands you type, is called a shell. Many different shells exist for UNIX/Linux, and they have some differences. We will work with ”Bash”, the ”Bourne again shell” (based on the older ”Bourne Shell”). Bash is the default shell in Linux. For more information: `man bash` (press q to quit).

4 Your first commands

Try the commands below, see what they do, and make sure you understand.

ls List directory contents

The ”ls” command will show a list of the files and directories in the current directory:

```
$ ls
```

pwd Print working directory

The ”pwd” command will print the path to the current directory:

```
$ pwd
/.automount/robin2/root/export/home0/hm
```

Your current directory will always be your home directory when you start the terminal window.

man Display on-line manual pages

The ”man” command will show you the on-line manual page for any command, provided that the manual page exist on the system. If the manual page is more that one page long (it usually is), you move down one line at a time with ”enter”, one page at a time with ”space”, and you exit the manual page with ”q”. Try typing

```
$ man ls
```

An ”option” to a command is a letter or a word you type after the command, to alter it’s behavior. An option is usually preceded by a dash.

In the man page, you can see that there are a lot of ”options” to the ls command. What do you think `ls -l` does? Try it.

Also try `$ man pwd` and `$ man man`

more Pager

You have already seen ”more” being used: ”More” is the program used by ”man” to let you read the manual pages one page at a time. You can use ”more” to see the output of a command, one page at a time. Try

```
$ more /usr/share/doc/emacs-common-21.3/README
```

less Pager

Less is used just like ”more”, but is a bit more flexible. You can move backward in the file as well as forward. Try

```
$ less /usr/share/doc/emacs-common-21.3/README
```

Remember, you can use ”q” to quit.

5 Managing files and directories

mkdir Make directory

Create a directory called ”linuxintro”:

```
$ mkdir linuxintro
```

cd Change directory

Go to your newly made directory:

```
$ cd linuxintro
```

You might want to check that you got there, with `pwd`.

cp Copy file

Copy the file ”/usr/share/pixmaps/firefox.png” to the ”linuxintro” directory:

```
$ cp /usr/share/pixmaps/firefox.png .
```

The dot means ”current directory”. Now copy it to a new file name, so you have two files:

```
$ cp firefox.png testfile
```

mv Move or rename file

Rename firefox.png to testfile2

```
$ mv firefox.png testfile2
```

rm Remove file

You can delete a file with the ”rm” command:

```
$ rm testfile2
```

rmdir Remove directory

You can delete an empty directory with the command ”rmdir”. Now create one, so you can delete it afterward:

```
$ mkdir linuxsubdir
```

Type `ls` to see it is really there. Then type

```
$ rmdir linuxsubdir
```

to remove it.

6 A few useful commands

cat Concatenate file

Show the content of a file on the screen. Try

```
$ cat testfile
```

echo Echo a line of text
”Echo” will repeat the text you type, to the screen:

```
$ echo "Hello Linux"
```

sort Sort lines in file alphabetically

```
$ sort testfile
```

grep Show lines matching a pattern
With this command, you can pick lines containing a specific word /phrase from of a file. Grep is a very complex command but we will use it in its simplest form:

```
$ grep "world" testfile
```

This command will display all the lines in testfile containing the word ”world”.

7 Pipes and redirection

Pipe In Linux/UNIX, there are many small programs that can do one simple thing. List the files in a directory, show one page of something at the time, etc. The real usefulness of these programs are revealed, when they are put together, one after the other. Try the following:

```
$ ls /usr/lib | less
```

The vertical symbol between the two commands is called ”pipe”, and it takes the output of the first command and use it as input for the second command.

On a Danish keyboard, the | key is found as the third function on the ' key, to the left of the backspace key.

$>$ Redirect stdout
Stdout (standard output) from a program usually goes to the terminal. Example:

```
$ echo "Hello Linux"
```

The output ”Hello Linux is displayed on the screen. But you can redirect stdout to a file instead of the terminal:

```
$ echo "Hello Linux" > testfile3
```

If the file exist it will be overwritten. If it does not exist, it will be created. Test it with `cat testfile3`.

$>>$ Append stdout to file
Append something to a file, not overwriting previous entires:

```
$ echo "12345678" >> testfile3
```

Check it with `cat testfile3`.

< Redirect stdin

Usually, stdin (standard input) is what you type on your keyboard, or what comes from the previous program, if in a pipe line. But you can get stdin from a file instead:

```
$ cat < testfile3
```

Now, try setting some programs together in a row:

```
$ cat /etc/printcap | sort > testfile4

$ cat /proc/pci | grep Ethernet

$ grep -r "inode"  /usr/src/linux-2.4 | less
```

This becomes much more useful when you know more commands.

8 Starting a program from the command line

- You can start a graphical application from the command line by typing its name. Try

  ```
  $ firefox
  ```

- You didn't get your command prompt back - you can't type new commands. That is because you are running Firefox "in the foreground". Exit Firefox, either the ordinary way from the Firefox menu, or by pressing ctrl+c in the terminal window you started it from.

- Start Firefox from the command line in "background mode"

  ```
  $ firefox &
  ```

  This time, you get your command prompt back so you can type new commands, while Firefox keeps running independently.

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*.