

Advanced Systems and Network Course
@ UCC

Linux commands

12th Sept. 07

By Fabian

Overview

- UNIX
- Linux
- Simple Linux commands
- Linux text editors

UNIX

- UNIX is an operating system, originally written at Bell Labs and popularized by the University of California - Berkeley
- It includes all of the features you'd expect from an OS and a few more
- It is a lot more flexible and configurable than Windows
- It's also harder to learn for beginners

LINUX

- “Linux is a free Unix-type operating system originally created by Linus Torvalds with the assistance of developers around the world.
- Developed under the GNU General Public License, the source code for Linux is freely available to everyone.”

– *Taken from Linux Online!*

<http://www.linux.org>

From Windows to Linux

- Linux is an operating system – its purpose is the same as all operating systems
- All Linux operations can be done on the command line
- This is more similar to DOS than Windows
- Most Linux boxes have a windowing system running on them
- We'll look at how you'd do most of the general commands you'd expect to use in Linux

Linux essentials

- Two most important things to know in Linux are:
 1. *Linux is case sensitive*
 2. *To find out more about any command in Linux, type **man <command>***
- Every command has the same basic structure:
 - *<command> <options> <arguments>*
- All commands end with either a newline or a ;

Listing files

- To list the files in a directory, type: **ls**
- This prints the listing to stdout (the standard output stream – usually the screen)
- For a more detailed listing, type: **ls -l**
 - *-l is an option for the ls command*
- Other options to try are: **-C** and **-F**
- Options can be grouped together: **ls -CF**
- Arguments can be applied to it: **ls bob***
 - ** means all files*
 - *bob* means all files that start with the characters bob*

Example ls -l

File permissions	Owner	Group	File size	Time modified	Filename
-rw-----	1 pbiggs	faculty	127	Apr 12 2000	test2.cc
-rw-----	1 pbiggs	faculty	63	Jan 11 2000	test2.cpp
-rw-----	1 pbiggs	faculty	198	Feb 15 13:41	test2.dat
-rw-----	1 pbiggs	faculty	559	Aug 1 2000	test3.cpp
-rw-r--r--	1 pbiggs	faculty	34	Feb 22 14:08	test3.dat
-rw-----	1 pbiggs	faculty	133	Aug 7 2000	test4.cpp
-rw-----	1 pbiggs	faculty	165	Sep 12 2000	test5.cpp
-rw-----	1 pbiggs	faculty	258	Oct 31 2000	test6.cpp
-rw-----	1 pbiggs	faculty	348	Nov 9 2000	test7.cpp
-rw-----	1 pbiggs	faculty	430	Jan 18 11:48	test8.cpp
-rw-----	1 pbiggs	faculty	112	Jan 23 11:00	test9.cpp
-rw-----	1 pbiggs	faculty	116	Mar 13 2000	testa.cc
-rw-----	1 pbiggs	faculty	1169	Feb 28 21:36	testa.cpp
-rwx--x--x	1 pbiggs	faculty	118963	Sep 7 2000	testbank
-rw-----	1 pbiggs	faculty	1365	Sep 7 2000	testbank.cc
-rw-----	1 pbiggs	faculty	78924	Sep 7 2000	testbank.o
-rw-----	1 pbiggs	faculty	462	Jul 31 2000	test.cc
-rw-----	1 pbiggs	faculty	528	Apr 5 2000	testclass.cpp
-rwx--x--x	1 pbiggs	faculty	182252	Jan 24 2000	testexe
drwx-----	2 pbiggs	faculty	4096	Dec 5 2000	testinh
drwx-----	3 pbiggs	faculty	4096	Aug 15 2000	tod
drwx-----	2 pbiggs	faculty	4096	Aug 20 2000	url
drwxr-x---	3 pbiggs	faculty	4096	May 15 2000	van
drwx-----	2 pbiggs	faculty	4096	Jul 29 2000	vector

Using stdout

- The output of the command `ls` goes to stdout, which is usually the screen
- However, it can be redirected to other places using stream redirection >
 - **`ls -l > myfile`**
 - this sends the command's output to the file `myfile`, creating it if needed, or overwriting it if it exists
- You can append to the file rather than overwriting it using >>
 - **`ls -l >> myfile`**

more

- **more** is a command that shows its input (either a file or stdin) one screen at a time
 - ***more myfile***
- **more** commands:
 - ***<enter>*** - move down one line
 - ***<space>*** - move to next screen
 - ***b*** - move back one screen
 - ***/ word*** - search for a word
 - ***/ <enter>*** - repeat last search
 - ***q** or **Q*** - exit

stdin, pipes and more

- stdin (standard input, which is usually the keyboard) is the IO stream where the input for commands comes from
- **more** can also accept its input from stdin
- A pipe | can be used to send the output of one program to the input of another, for example:
 - ***ls -l | more***
- This allows you to view this long listing through **more** i.e. one screenful at a time

Files and directories

- To move/rename a file: **mv** <name1> <name2>
- To remove a file: **rm** <filename>
- Create a directory: **mkdir** <dirname>
- Remove a dir: **rmdir** <dirname>
- Change directories: **cd** <dirname>
- Special characters:
 - *. is the current directory*
 - *.. is the parent of the current directory*
- **cd .** means move to the current dir (no change)
- **cd ..** means move up one directory

Linux directory structure

- Linux has a tree structure for directories much like Windows
- Each directory may contain files and subdirectories
- The directory tree starts at the root directory /
- There is no concept of “disk drive letters” in Linux – all disk space is treated equally
- A Linux path:
 - */usr/local/lib/filename.bob.txt.a*
- The dot in a filename has no special meaning in Linux
- To find where you currently are in the directory structure, type: **pwd**

File permissions

- You must have permission to access each file
- File permissions are: read, write and execute
- These are designated: r w x
- **ls -l** will show you the permissions for each file as follows:
 - *drwxrwxrwx*
- The d is for directory
- The first rwx is for the user (you), the second is for the group that you've been assigned to (all students), the third is for everyone else
- These are designated: u (user), g (group), o (others)

Changing permissions

- You can change the permissions on any file that you own
- This is done with the **chmod** command
- For example:
 - **chmod u+r *** - *add read permission for the user for all files*
 - **chmod o-r *** - *remove read access from others for all files*
- Any file with executable permission may be executed (even if it doesn't make sense to do so)
- Directories must always be executable

Useful Linux tips

- To repeat the last command: **!!**
- To repeat the last command you run that started with a g: **!g**
- To look at the commands you've run recently: **history**
- To run a numbered command from the history list: **!**25****
- To run a command in the background: **<command> &**
- To correct a mistake in the last command:
^<mistake>^<correction>
- If the Linux station locks up, try freeing it with the Intellimouse™ roller – never switch off a Linux box

Changing the path

- Linux has a set of environment variables that allow you to configure your Linux environment
- The `PATH` variable tells Linux which directories to look in when a program is run
- It is a good idea to add the current directory to the `PATH`
- This can be done by typing:

```
setenv PATH ".:$PATH"
```

- If you add this line to the end of the file `.login`, this will automatically be done for you every time you log in

Linux text editors

- Most Linux boxes you use will have a windowing system with its own text editor (similar to Notepad in Windows)
- At the command line, some text editors you could use are:
 - *vi* – most common and basic – tough to learn
 - *emacs* – more powerful and useful
 - *pico* – easiest to learn
 - *kwrite* - GUI editor on Linux (aka Advanced Editor)
- Play with them – use whichever editor you prefer

Conclusion, questions