Introduction to Unix

Tuomo Salmi

Spring 2020

Notes & exercises

Lecture material and exercises are available from:

https://github.com/thjsal/unix-intro

The original, almost identical, from Joonas Nättilä:

https://github.com/natj/unix-intro

Unix & Linux

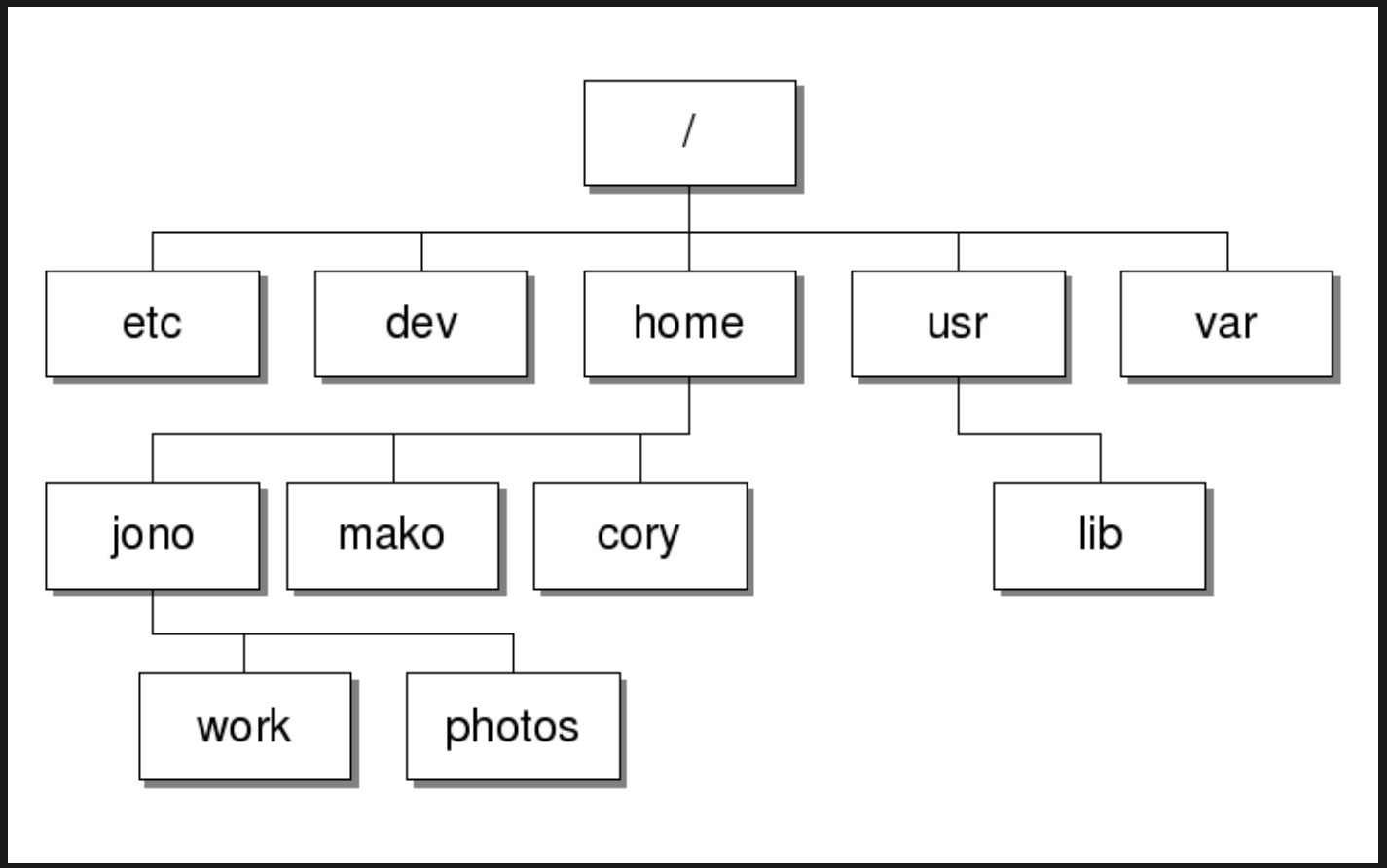
UNIX is a family of operating systems developed already in the 1960s, for which Unix-like systems, like Linux, are based on (developed by Finnish Linus Torvalds. By operating system we mean a suite of programs that make the computer work.

UNIX systems are characterized by being stable, multi-user and multi-tasking systems. Because of this, UNIX can be found everywhere from supercomputers to Apple laptops. For a researcher it is an invaluable tool as everything can be automated simplifying your life a whole bunch!

UNIX systems incorporate a modular design that is sometimes called the "Unix philosophy", meaning that the operating system provides a set of simple tools that each perform a limited, well-defined function. In addition, with the unified filesystem as the main mean of communication and a shell scripting to aid you, the system can perform complex workflows with ease.

Directory structure

- root /
- home ~
- absolute and relative paths
  - current dir .
  - previous dir ..



Shell

Linus Torvalds thinks mouse should not be used to operate computers. Keep the hands on the keyboard!

Shell is a program that provides the traditional text-only user interface for Linux (and other Unix systems)

A command is an instruction given by an user to the computer to do something

```
command -options [input]
```

Navigation

- `cd` - change directory
- `ls` - list dir content
- `cat` - print file
- wildcard `*` is “whatever character”
- `pwd` - print working directory
- tabulator (your best friend) - fill in partial words
- `man` pages
  - enter - scroll
  - space - next page
  - Q - quit
- Keyboard commands
  - ctrl+c
  - Q (in man pages)

TODO Exercises

1. Run the following commands step by step and check with pwd (and ls) where you are
   - `cd /`
   - `cd ~`
   - `cd /`
   - `cd`
2. Check using the man command what is the option (-XXX) for ls to display all of the files in the current dir (so we can also see the hidden dot-files)

File management

- `touch` - create an empty file
  - touch foo.txt
- `mkdir` - make directory
  - mkdir mydir
- `rm` - remove
  - ‘rm‘ file
  - `rm -r` - recursive
  - `rm -f` - force
  - `rmdir` - for directories
- `cp` - copy
  - cp origfile copyofthat
- `mv` - move
  - mv origfile new/loc/origfile
- `df` - disk usage
- `du` - dir usage
  - -h - human readable sizes

File permissions & ownership

- `ls -la foo.txt`
  - l - list format
  - a - show all files (also hidden)

```
-rw-r--r-- 1 userid groupid 0 Jan 29 11:04 foo.txt
```

- `chmod` +/- (give or take)
  - read / write / execute
  - all / user / group
- `chmod a-r foo.txt`

```
--w------- 1 userid groupid 0 Jan 29 11:05 foo.txt
```

TODO Exercises

1. Create an empty file called “foo.txt”
2. Copy the file “foo.txt” into “foo.txt.copy”
3. Rename the file “foo.txt.copy” to “foo2.txt”
4. Create an empty directory called “data”
5. Move the file “foo2.txt” into “data” directory
6. Create a new subdirectory inside “data” called “new”
7. Copy the file “foo2.txt” in the “data” directory into “new” directory; rename it at the same time to “foo3.txt”
8. Move the file “foo3.txt” in the “new” directory back to home directory; at the same time, change the name into “foo_old.txt”
9. Delete the file “foo_old.txt”
10. Remove the “new” subdirectory inside the “data” directory
11. Check the file permissions (with ls -l) for the “foo.txt”
12. Make the file read-only for all; check the permissions
13. Add write permission to yourself
14. How big is the “foo.txt”?
15. What is the current disk space quota?

Process management

Demo:

- `top` : list currently running commands
  - pid : command id
- `kill` pid

TODO Exercises

1. Check what is the heaviest process somebody (who?) is running using the top command
2. What is the PID number of this process?

Searching

- `grep` - search patterns/words
  - `grep mysearchword mysearchlocation`
- `find` - search filesystem
  - `find myfindlocation -name filename`

More advanced stuff

I/O and chaining

- | piping, output of first command is fed to the second
  - `cat file.txt | grep word`
- > insert, output is directed to where the arrow points
  - `ls -la > mydirs.txt`
- >> append, similar to insert but stuff is appended, not overwritten

SSH

- `ssh user@host`
  - `ssh user@linux.utu.fi`
  - PuttY
- `scp user@host:~/path/to/file file_name`

Compression

- `tar -caf file.tar.gz files`
  - c - create
  - a - automatic format detection (.gz)
  - f - file
- `tar -xf file.tar.gz`
  - x - decompress

Misc

- `echo`
- `date`
- `cal` (calendar)
- `uptime`
- `whoami`
- `w` (who is online)
- `finger user`
- `cat /proc/cpuinfo`
- `cat /proc/meminfo`
- `ping host`

TODO Exercises

1. Compress the “data” dir into “data.tar.gz”
2. Remove the “data” dir
3. List the contents of the data.tar.gz (HINT: Don’t forget the -f option at the end so that tar reads your file)
4. Uncompress the data.tar.gz
5. List the content of the data.tar.gz with the additional verbose command BUT redirect the output to go into a file “data_content.txt”

BONUS:

6. SSH to linux.utu.fi with your username (if not already inside)
7. Check how many users there are online (using e.g. irssi)
   - use w to list users
   - use grep to filter for specific words (don’t forget piping)
   - check grep manual for counting the hits

Editors

Nano

- No specialities - simple, easy and often too basic
- ctrl+x (to quit)

Emacs

- Good basic workhorse with relatively easy interface
- ctrl+x+s (save)
- ctrl+x+c (exit)
- More commands:
  - ctrl+k (kill = cut)
  - ctrl+f (open file)
  - ctrl+s (search for word)
  - ctrl+n/p (next/previous page)
  - ctrl+f/b (forward/backward in text)
  - ctrl+XXX+g (abort)

vi/vim

- More advanced editor with some tricks and quirks that one should know about before using
- Good to know how to exit:
- :q!

TODO Exercises

1. Edit the file foo.txt to include “Hello from NANO!”
2. Do the same using the Emacs
3. Finally open the file using vi and try to exit

- BONUS: add text using vi also (HINT: google vi tutorial for basic usage)

Scripting

Script is a text file that is like a Linux spell - contains commands that are executed

Where is our bash-script interpreter located

- `which bash`
- -> /bin/bash

So now we know how to start bash interpreter - start your .sh file with #!location

Basic script

```
#!/bin/bash
MSG="Hello"
ME=`whoami`
echo $MSG $ME
```

Notice that there are no spaces around “=”.

Backticks are used when calling commands (not ’ or “)

Usage

Now we must make it executable (for safety reasons)

- chmod +x xxx.sh
- ./xxx.sh

TODO Exercises

1. Write a script myscript.sh that:

- Welcomes the current user and
- displays the current date

2. Run the script, is it working? What is the output?
3. BONUS: Extend the previous script to work as a basic backup script:

- Make it so that when run, it compresses the previously made "data" dir into a file called backup_XXX.tar.gz where
- XXX is the current date (HINT: use TIME=`date +%Y_%m_%d` to get the date in a format where underscores are used)