Hands-on: Basic Linux

Marcus Lundberg

- Log in to UPPMAX system
- Navigate the filesystem
- Edit files
- Miscellaneous useful skills

Connect to UPPMAX

- You should have done this already at least once before
- Linux and MacOS:
  - start Terminal
  - `$ ssh -X username@rackham.uppmax.uu.se`
- Windows:
  - Download and use an SSH program such as MobaXTerm

Everyone connected?

- Take time now to connect
- Consider connecting a second time with a new window
- Organise your windows so you can watch Zoom and work in your terminal. If you have screen space, keep the presentation open locally and an eye on the Zoom chat.

File system basics

- Just like in Windows, files are organised in a hierarchy of “folders” or “directories”
- The top of the hierarchy is the **root directory**, denoted by ‘`/`’
- File locations are given by a **path**, either **relative** to where you are right now or **absolute**, starting from the root directory.

`pwd` — where are you now

- When you log in, you will be in your home directory, `~/`, `$HOME`, and `/home/username/`
- `$ pwd`
- `$ pwd -P` gives you the physical path (ignores how you got there)

`ls` — contents of a directory

- Type `ls` to display the contents of the current directory.

- `$ ls -a` also shows hidden files and directories
- `$ ls -l` gives you detailed information
- `$ ls -lt` sorts things by time modified

`cd` — moving around

- To **c**hange **d**irectory, use `cd <target>`
- `$ cd /proj/g2020018`
- `$ pwd`
- `$ ls`
- `$ cd labs`
- `$ pwd`

cd

- Experiment with `cd`. Try adding spaces or extra `/` in various places
- Use **tab completion** to avoid typos and typing “ls” a lot.
- Figure out the use of the following:
- `$ cd -`
- `$ cd ..`
- `$ cd`
- `$ cd ~`

`mkdir` — create a new directory

- Make sure you're in your home directory
- `$ mkdir uppmax-intro`
- Go in there:
- `$ cd ~/uppmax-intro/`

`cp` — copy files

- Copy files with: `cp <source> <target>`
- `$ cp /proj/g2020018/labs/linux_tutorial/ .`
- Well, that didn’t work. What does the error say?
- `$ cp **-r** /proj/g2020018/labs/linux_tutorial/ .`

`cp` — copy files

- Move to `linux_tutorial/`
- Make a copy the file “newfile” in the same directory:
- `$ cp newfile copyfile`

`scp` — copying remote files

- Linux/MacOS: To copy data to/from Rackham, you can use `scp` from your local machine:
- `[bob@macbook]$ scp myinput bob@rackham.uppmax.uu.se:~/copyofmyinput`
- `[bob@macbook]$ scp bob@rackham.uppmax.uu.se:~/mydata copyofmydata`
- Windows: Drag and drop files from MobaXTerm window.
- *This was just a preview, you will try this out later.*

`mv` — move/rename file

- Moving files works just like copying files:

  ```
  mv <source> <target>
  ```

- Move the copy you just made to another place:
- `$ mv copyfile ../`
- Rename it.
- `$ mv ../copyfile ../renamedfile`

`rm` — delete file

- Deleting files works just like copying or moving them:
  - `rm <target>`
- Try it out:
- `$ rm ../renamedfile`

Caution!!

- Some words of warning:
  - There is no **undo** for `cp`, `mv`, and `rm`.
  - Beware of overwriting (clobbering) files and deleting the wrong ones.
- If you do destroy your data, email UPPMAX support, we may be able to help.

`tar` — archiving and compression

- We’re going to need more files.
- `$ tar -vxzf files.tar.gz`
- The flags mean:
  - **V**erbosely
  - E**x**tract
  - **Z**ipped
  - **F**ile
- You should see a list of files being extracted

`rmdir` — delete an empty directory

- `$ rm this_is_empty`
- Need another command to delete directories
- `$ rmdir this_is_empty`
- `$ rmdir this_has_a_file`
- Is there a way to use `rm` to delete directories?

`rm` more

- Recursive commands are applied to directories and their contents
- `$ rm -r this_has_a_file`
- Compare:
  - `$ ls ..`
  - `$ ls -R ..`

`man` — look up the right flags

- Nobody can remember whether it's `-R` or `-r` for **r**ecursive, or if `-f` lets you choose a **f**ile or **f**orces an action.
- `$ man ls` shows you how to use ls and all its options
- Type '/keyword' to search for "keyword", use 'n' and 'N' to scan through hits.
- Type 'q' to **q**uit.
- Spend some time now to browse the man pages for the commands you've just learned

Review exercise

- Now try this:
  - Create a new directory inside your home directory
  - Cd into it
  - Copy any file into the directory
  - Rename the file to something else
  - Delete the directory and its contents

Wildcards

- `$ ls many_files`
- `$ ls many_files/*.txt`
- `$ ls many_files/file_1*1.docx`
- Want to clean out temporary files ending in .tmp in all the subdirectories?
  - `$ rm */*.tmp`
  - (could be wise to do `ls -a */*.tmp` first to see what will be deleted...)
- Exercise:
  - Create a new directory and move all .txt files in `many_files` to it

Reading files

- In Linux, you can display files without being able to change them
- `$ cd old_project`
- `$ ls`
- Hmm, which of these files are useful?

`cat`

- `cat` dumps the contents of files to the terminal as text
- `$ cat the_best`
- Yummy!
- `$ cat a`
- ???
- Con**cat**enate files with this wizardry:
  - `$ cat a the_best > combinedfiles.txt`

`head` — display the top of a file

- `$ head a`
- You can choose how many lines to display (default 10)
  - `$ head -n 4 a`
- `Tail` is the same as head, but for the other end
  - `$ tail -n 5 a`
  - Handy to look at log files

`less` — read a whole file

- Cat doesn’t really work for long files
- `$ less a`
- Search with ‘/keyword’ and ‘n’/’N’
- Hit ‘q’ to quit.

Editing files

- File editors :
  - `nano` (keyboard shortcuts shown on-screen)
  - `gedit` (graphical, needs X11)
  - `vim` (fast and powerful, once you learn it)
  - `emacs` (fast and powerful, once you learn it)
- Try them out and pick one.

A bit of a side-line:

X11-forwarding: graphics from the command line

- Graphics can be sent through the SSH connection you're using to connect
  - Use `ssh -Y` or `ssh -X`
- **MacOS users will need to install XQuartz**.
- When starting a graphical program, a new window will open, but your terminal will be "locked".
  - Run e.g. "`gedit &`" to send the gedit process to the background
  - Alternatively, use `ctrl-z` to put gedit to sleep and type `bg %1` to make process number one run in background

File permissions

```
$ ls -l
drwxrwxr-x 2 marcusl marcusl  4096 Sep 19  2012 external_hdd
-rwxr-xr-x 1 marcusl marcusl 17198 Jul 16 14:12 files.tar.gz
```

- Leading symbol:
  - `d` directory
  - `-` regular file
  - `l` symbolic link (more on this tomorrow)
  - Others exist, but you can ignore them

File permissions

```
$ ls -l
drwxrwxr-x 2 marcusl marcusl  4096 Sep 19  2012 external_hdd
-rwxr-xr-x 1 marcusl marcusl 17198 Jul 16 14:12 files.tar.gz
```

- Three sets of “rwx” permissions
  - rwx: read, write, execute
  - User: the user account that owns the file (usually the one that created it)
  - Group: the group that owns the file (usually the project group in /proj/xyz or the user’s group elsewhere)
  - Others: everyone else on the system (literally a thousand strangers)

File permissions

- `r` – read
  - Files: Read the contents of the file
  - Directories: List the files in the directory
- `w` – write
  - Files: Modify the file
  - Directories: Add, rename, or delete files in the directory
- `x` – execute
  - Files: Run the file as a program
  - Directories: Traverse the directory (e.g. with “cd”)

More on permissions

- `$ ls /proj/g2020018`
- Huh, `rwxrw`***`s`***`r-x`?
- ‘s’ in the group means ‘x’ but with gid bit set.
- ‘S’ means ‘-’ with gid bit set (rarely seen).
- Among other things, this makes the default group for new files/subdirectories the g2020018 group.

`chmod` — changing permissions

- Files with `w` can be modified and destroyed by accident. Protect your input data!
- If you want to share data or scripts with a person not in your project (e.g. support staff like me), you can!
- If you want to keep non-members from even seeing which files you have, you can!

`chmod` — changing permissions

- **Ch**ange file **mod**e: `chmod <mode> <files>`
- <mode> can be e.g.
  - `u+x` (let you run a script you just wrote)
  - `-w` (no write permissions for anyone)
  - `g+rw` (let group members read and edit this file)
  - `g=xw` (let group members go into your directory and put files there, but not see which files are there)
- Chmod takes flags as usual, e.g. -R for recursive

`chmod` – numerical permissions

- Online, you will come across e.g. “chmod 755”, what does this mean?
- It’s a “bit mask”:
  - `7 = 4 + 2 + 1 = r + w + x`
  - `5 = 4 + 0 + 1 = r + x`
- What number would `r + w` be?

`chmod` – Hands-on

- In the `linux_tutorial` directory, find important files and old saved data that you wouldn't want to lose.
  - Directories: `important_results/`, `old_project/`
  - File: `last_years_data`
- Use `chmod` to remove write permission from those files and directories (use the `-R` flag to also do the files in the directories).
- Take a moment to play around with chmod and explore the effects of permissions on files and directories.

Summarising exercises (1)

- Find and delete the files named temp_file-1 and temp_file-2.
  - Can you do it with one command, standing in `linux_tutorial/`?
  - You may have to give yourself permission.

Summarising exercises (2)

- Create a directory named “text_files”
- Move all the `.txt` files in subdirectories of `linux_tutorial` into this directory
  - Use the “verbose” flag to get a report of which files were moved.

Summarising exercises (3)

- Transfer files to and from Rackham. Use `scp` or FileZilla or MobaXterm or whatever you like.
- Read up on the `rsync` tool for moving files.