



Introduction to UNIX and High Performance Computing

Let's hack the mainframe!

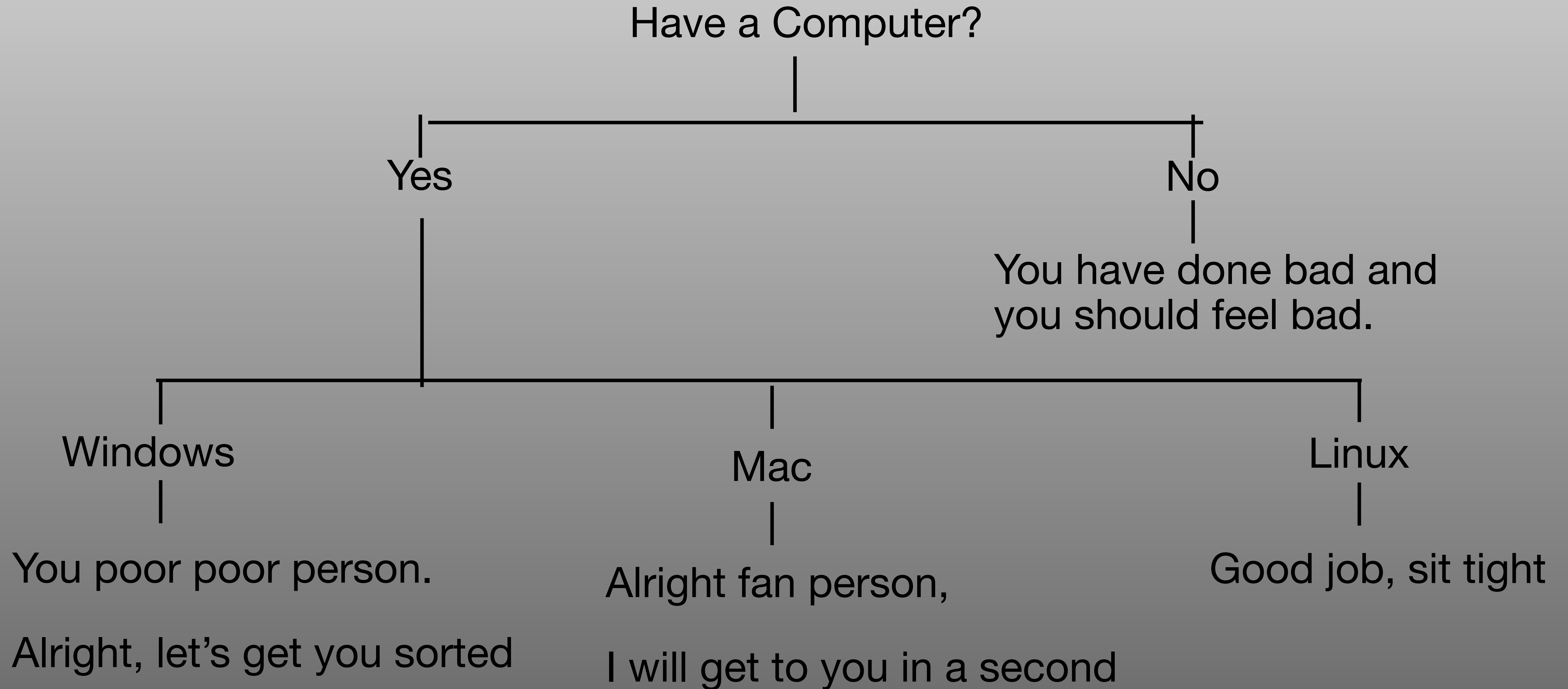
Thomas McElroy

Goal of lecture

2

- Get you to a point where you are comfortable logging onto the computer that you will use for research
- Confident in basic file navigation commands
- Make you aware of a number of functions and programs to make your life easier
- Give you knowledge as to how to work on a cluster.
- We will not have time to go into great depth on any subject but I will as much as possible point you to good resources for further information.

Step One



Windows

For almost all of you it will be most advantageous to install Ubuntu within the Windows Subsystem for Linux.

Hopefully you are a keener and have looked at the slides ahead of time and followed this link to get it setup on your own

<https://docs.microsoft.com/en-us/windows/wsl/install-win10>

and

<https://www.makeuseof.com/tag/linux-desktop-windows-subsystem/>

It will actually take too long to do it here soooo....

you're still on your own.

Windows

5

Compiling ROOT on WSL, follow the instructions here:

<https://medium.com/@blake.leverington/installing-cern-root-under-windows-10-with-subsystem-for-linux-beta-75295defc6d4>

With a few modifications:

-first, make sure you have the python libraries:

```
apt-get install -y python3 python3-dev python3-pip
```

-Get the source files from the GitHub repository so that you get the latest version (not wget):

```
git clone https://github.com/root-project/root.git
```

-Also build without XROOTD

```
cmake -Dxrootd=OFF -Dbuiltin_xrootd=OFF ../root
```

To make the build a little faster you can use more cores with

```
make -jN (replace N with number of cores to use)
```

Good job! You at least have an operating system with an underlying UNIX kernel.

Did you remember to install Xcode and command line tools so you can make use of it?

<https://osxdaily.com/2014/02/12/install-command-line-tools-mac-os-x/>

You will also want to make sure you have XQuartz installed for X11

<https://www.xquartz.org/>

Again, too slow to do with you but it works.

Docker

- A computer inside another computer.....
- <https://www.docker.com/> to get docker software.
- <https://hub.docker.com/repository/docker/thomasmcelroy/saporientation>
- This is a large file.... It contains a whole operating system.
- Contains all the good stuff.

Docker in Windows (must have Pro)

- Open a terminal window (Command Prompt or PowerShell)
- `> docker run thomasmcelroy/saporientation`
- Once inside docker container type `source env.sh` to get ROOT
- Look into how to mount folders from your computer to folder in the docker container as well as getting graphical interfaces forwarded.
- <http://somatorio.org/en/post/running-gui-apps-with-docker/> (graphics)
- <https://www.digitalocean.com/community/tutorials/how-to-share-data-between-the-docker-container-and-the-host> (volumes)

Docker on Mac

Ya, pretty much the same...

- Open a terminal window
- `$ docker run -it thomasmcelroy/saporientation`
- Once inside docker container type `source env.sh` to get ROOT
- Look into how to mount folders from your computer to folder in the docker container as well as getting graphical interfaces forwarded.
- <http://somatorio.org/en/post/running-gui-apps-with-docker/> (graphics)
- <https://www.digitalocean.com/community/tutorials/how-to-share-data-between-the-docker-container-and-the-host> (volumes)

Connecting using Putty on Windows

- Get Putty and Xming (or **VcXsrv**)
- <https://sourceforge.net/projects/xming/>
- <https://www.putty.org/>
- Start Xming
- In putty turn on X11 forwarding (Putty Configuration ->Connection->SSH->X11)
- SSH keys can be generated with PuttyGen
- <https://support.hostway.com/hc/en-us/articles/115001509884-How-To-Use-SSH-Keys-on-Windows-Clients-with-PuTTY->

Connecting to remote computer.

Since many of you will primarily be working on remote clusters, let's get connected first.

You need: a username, a password and the address of remote computer.

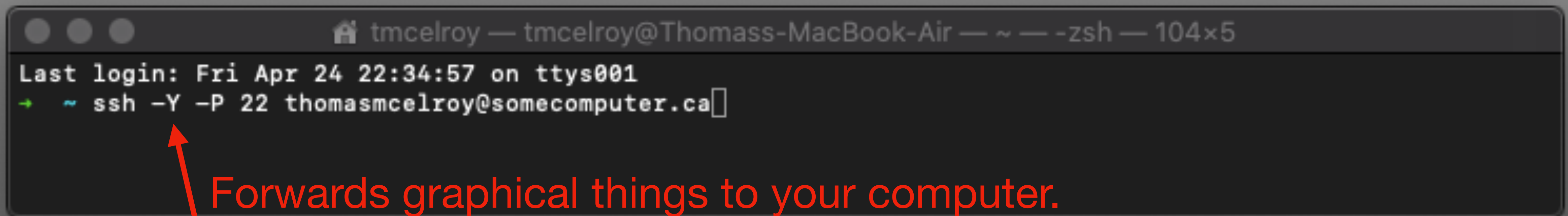
Example you ask?

Username: thomasmcelroy

Password: ah nice try

Address: somecomputer.ca

Now you can open a secure shell (ssh):

A terminal window titled 'tmcelroy — tmcelroy@Thomass-MacBook-Air — ~ — -zsh — 104x5'. The window shows the output 'Last login: Fri Apr 24 22:34:57 on ttys001' and a prompt with the command 'ssh -Y -P 22 thomasmcelroy@somecomputer.ca'. A red arrow points from the '-Y' flag to the explanatory text below.

```
tmcelroy — tmcelroy@Thomass-MacBook-Air — ~ — -zsh — 104x5
Last login: Fri Apr 24 22:34:57 on ttys001
➔ ~ ssh -Y -P 22 thomasmcelroy@somecomputer.ca
```

Forwards graphical things to your computer.
Make sure you have X11 (XQuartz for Mac)
installed.

Connecting to remote computer.

Since many of you will primarily be working on remote clusters, let's get connected first.

You need: a username, a password and the address of remote computer.

Example you ask?

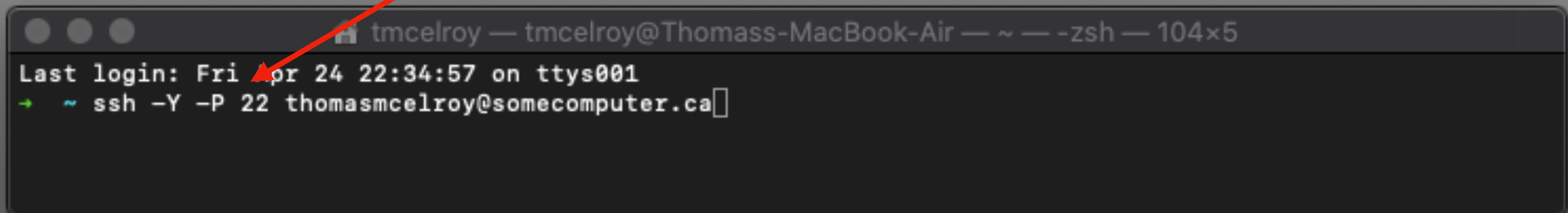
Username: thomasmcelroy

Password: ah nice try

Address: somecomputer.ca

This selects a port to access the computer through. This is not usually needed for connecting directly to a computer since port 22 is the default ssh port. It is needed if you are accessing a computer hidden on an internal network that is port forwarding you to the computer you want. If you need it you will be told.

Now you can open a secure shell (ssh):

A terminal window titled 'tmcelroy — tmcelroy@Thomass-MacBook-Air — ~ — -zsh — 104x5'. The window shows the output 'Last login: Fri Apr 24 22:34:57 on ttys001' and the command 'ssh -Y -P 22 thomasmcelroy@somecomputer.ca' being entered at the prompt. A red arrow points from the underlined address 'somecomputer.ca' in the text above to the 'somecomputer.ca' part of the command in the terminal.

```
tmcelroy — tmcelroy@Thomass-MacBook-Air — ~ — -zsh — 104x5
Last login: Fri Apr 24 22:34:57 on ttys001
➔ ~ ssh -Y -P 22 thomasmcelroy@somecomputer.ca
```

Connecting to remote computer.

13

```
[→ ~ ssh -Y -P 22 thomasmcelroy@somecomputer.ca  
thomasmcelroy@somecomputer.ca s password:  Input password  
Permission denied, please try again.  
thomasmcelroy@somecomputer.ca s password:  
Warning: No xauth data; using fake authentication data for X11 forwarding.  
Welcome to Ubuntu 18.04.1 LTS (GNU/Linux 4.15.0-74-generic x86_64)  
  
* Documentation:  https://help.ubuntu.com  
* Management:    https://landscape.canonical.com  
* Support:       https://ubuntu.com/advantage  
  
* Ubuntu 20.04 LTS is out, raising the bar on performance, security,  
  and optimisation for Intel, AMD, Nvidia, ARM64 and Z15 as well as  
  AWS, Azure and Google Cloud.  
  
  https://ubuntu.com/blog/ubuntu-20-04-lts-arrives  
  
* Canonical Livepatch is available for installation.  
  - Reduce system reboots and improve kernel security. Activate at:  
    https://ubuntu.com/livepatch  
  
464 packages can be updated.  
15 updates are security updates.  
  
*** System restart required ***  
Last login: Sat Apr 25 00:52:57 2020 from 000.000.000.000  
tmcelroy@EXO-Simulation:~$
```

Connecting to remote computer.

14

```
[→ ~ ssh -Y -P 22 thomasmcelroy@somecomputer.ca  
thomasmcelroy@somecomputer.ca$ password:  
Permission denied, please try again. ← Get it wrong  
thomasmcelroy@somecomputer.ca$ password:  
Warning: No xauth data; using fake authentication data for X11 forwarding.  
Welcome to Ubuntu 18.04.1 LTS (GNU/Linux 4.15.0-74-generic x86_64)  
  
* Documentation:  https://help.ubuntu.com  
* Management:    https://landscape.canonical.com  
* Support:       https://ubuntu.com/advantage  
  
* Ubuntu 20.04 LTS is out, raising the bar on performance, security,  
  and optimisation for Intel, AMD, Nvidia, ARM64 and Z15 as well as  
  AWS, Azure and Google Cloud.  
  
    https://ubuntu.com/blog/ubuntu-20-04-lts-arrives  
  
* Canonical Livepatch is available for installation.  
  - Reduce system reboots and improve kernel security. Activate at:  
    https://ubuntu.com/livepatch  
  
464 packages can be updated.  
15 updates are security updates.  
  
*** System restart required ***  
Last login: Sat Apr 25 00:52:57 2020 from 000.000.000.000  
tmcelroy@EXO-Simulation:~$
```


Connecting to remote computer.

15

```
[→ ~ ssh -Y -P 22 thomasmcelroy@somecomputer.ca  
thomasmcelroy@somecomputer.ca$ password:  
Permission denied, please try again.  
thomasmcelroy@somecomputer.ca$ password:  
Warning: No xauth data; using fake authentication data for X11 forwarding.  
Welcome to Ubuntu 18.04.1 LTS (GNU/Linux 4.15.0-74-generic x86_64)
```

```
* Documentation:  https://help.ubuntu.com  
* Management:    https://landscape.canonical.com  
* Support:       https://ubuntu.com/advantage  
  
* Ubuntu 20.04 LTS is out, raising the bar on performance, security,  
  and optimisation for Intel, AMD, Nvidia, ARM64 and Z15 as well as  
  AWS, Azure and Google Cloud.
```

```
https://ubuntu.com/blog/ubuntu-20-04-lts-arrives
```

```
* Canonical Livepatch is available for installation.  
  - Reduce system reboots and improve kernel security. Activate at:  
    https://ubuntu.com/livepatch
```

```
464 packages can be updated.  
15 updates are security updates.
```

```
*** System restart required ***  
Last login: Sat Apr 25 00:52:57 2020 from 000.000.000.000  
tmcelroy@EXO-Simulation:~$
```

Input correct password

Connecting to remote computer.

16

```
[~ ssh -Y -P 22 thomasmcelroy@somecomputer.ca]
thomasmcelroy@somecomputer.ca$ password:
Permission denied, please try again.
thomasmcelroy@somecomputer.ca$ password:
Warning: No xauth data; using fake authentication data for X11 forwarding.
Welcome to Ubuntu 18.04.1 LTS (GNU/Linux 4.15.0-74-generic x86_64)

* Documentation:  https://help.ubuntu.com
* Management:    https://landscape.canonical.com
* Support:       https://ubuntu.com/advantage

* Ubuntu 20.04 LTS is out, raising the bar on performance, security,
  and optimisation for Intel, AMD, Nvidia, ARM64 and Z15 as well as
  AWS, Azure and Google Cloud.

  https://ubuntu.com/blog/ubuntu-20-04-lts-arrives

* Canonical Livepatch is available for installation.
  - Reduce system reboots and improve kernel security. Activate at:
    https://ubuntu.com/livepatch

464 packages can be updated.
15 updates are security updates.

*** System restart required ***
Last login: Sat Apr 25 00:52:57 2020 from 000.000.000.000
tmcelroy@EXO-Simulation:~$
```

← Your computer saying hello,
some nag more than others.

Connecting to remote computer.

17

```
[~ ssh -Y -P 22 thomasmcelroy@somecomputer.ca]
thomasmcelroy@somecomputer.ca$ password:
Permission denied, please try again.
thomasmcelroy@somecomputer.ca$ password:
Warning: No xauth data; using fake authentication data for X11 forwarding.
Welcome to Ubuntu 18.04.1 LTS (GNU/Linux 4.15.0-74-generic x86_64)

* Documentation:  https://help.ubuntu.com
* Management:    https://landscape.canonical.com
* Support:       https://ubuntu.com/advantage

* Ubuntu 20.04 LTS is out, raising the bar on performance, security,
  and optimisation for Intel, AMD, Nvidia, ARM64 and Z15 as well as
  AWS, Azure and Google Cloud.

  https://ubuntu.com/blog/ubuntu-20-04-lts-arrives

* Canonical Livepatch is available for installation.
  - Reduce system reboots and improve kernel security. Activate at:
    https://ubuntu.com/livepatch

464 packages can be updated.
15 updates are security updates.

*** System restart required ***
Last login: Sat Apr 25 00:52:17 2020 from 000.000.000.000
tmcelroy@EXO-Simulation:~$
```

Pay attention here if it tells you it couldn't create some autogenerated file. This usually means you have overfilled your home directory. Delete some stuff and relogin. Also, use your scratch disk space (will explain later).

Connecting to remote computer.

18

```
[~ ssh -Y -P 22 thomasmcelroy@somecomputer.ca]
thomasmcelroy@somecomputer.ca$ password:
Permission denied, please try again.
thomasmcelroy@somecomputer.ca$ password:
Warning: No xauth data; using fake authentication data for X11 forwarding.
Welcome to Ubuntu 18.04.1 LTS (GNU/Linux 4.15.0-74-generic x86_64)

* Documentation:  https://help.ubuntu.com
* Management:    https://landscape.canonical.com
* Support:       https://ubuntu.com/advantage

* Ubuntu 20.04 LTS is out, raising the bar on performance, security,
  and optimisation for Intel, AMD, Nvidia, ARM64 and Z15 as well as
  AWS, Azure and Google Cloud.

  https://ubuntu.com/blog/ubuntu-20-04-lts-arrives

* Canonical Livepatch is available for installation.
  - Reduce system reboots and improve kernel security. Activate at:
    https://ubuntu.com/livepatch

464 packages can be updated.
15 updates are security updates.

*** System restart required ***
Last login: Sat Apr 25 00:52:57 2020 from 000.000.000.000
tmcelroy@EXO-Simulation:~$
```

We are in! We hacked the mainframe!

But seriously, who uses passwords anymore?

Backup. Let's get out of here.

Generating an ssh key

19

```
[tmcelroy@EX0-Simulation:~$ exit
logout
Connection to 1000.000.000.000
[tmcelroy@EX0-Simulation:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/Users/tmcelroy/.ssh/id_rsa):
/Users/tmcelroy/.ssh/id_rsa already exists.
Overwrite (y/n)? y
[Enter passphrase (empty for no passphrase):
[Enter same passphrase again:
Your identification has been saved in /Users/tmcelroy/.ssh/id_rsa.
Your public key has been saved in /Users/tmcelroy/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:R6cpwjdbwFv12630w6NSeQdbVHidRdIfgZEFqvpXCE4 tmcelroy@Thomass-MacBook-Air.local
The key's randomart image is:
+---[RSA 2048]---+
|                .oB*X|
|      .   o.o.+==|
|      o o.o  oo|
|      .   E.+ o..o|
|      o S.* o ++.|
|      o.B . =o+.|
|      ..  o o+o|
|      .  o  . o|
|      .. ..|
+---[SHA256]---+
[tmcelroy@EX0-Simulation:~$
```

close ssh, also closed
when the terminal window
is closed

Generating an ssh key

20

```
[tmcelroy@EX0-Simulation:~$ exit
logout
Connection to 000.000.000.000
[tmcelroy@EX0-Simulation:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/Users/tmcelroy/.ssh/id_rsa):
/Users/tmcelroy/.ssh/id_rsa already exists.
Overwrite (y/n)? y
[Enter passphrase (empty for no passphrase):
[Enter same passphrase again:
Your identification has been saved in /Users/tmcelroy/.ssh/id_rsa.
Your public key has been saved in /Users/tmcelroy/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:R6cpwjdbwFv12630w6NSeQdbVHidRdIfgZEFqvpXCE4 tmcelroy@Thomass-MacBook-Air.local
The key's randomart image is:
+---[RSA 2048]-----+
|          .oB*X|
|      .  o.o.+|=|
|      o o.o  oo|
|      .  E.+ o..o|
|      o S.* o ++.|
|      o.B . =o+.|
|      ..  o o+o|
|      .  o . o|
|      .. ..|
+---[SHA256]-----+
[tmcelroy@EX0-Simulation:~$ ]
```

Creates an ssh key
unique to your computer

Generating an ssh key

21

```
[tmcelroy@EX0-Simulation:~$ exit
logout
Connection to 000.000.000.000
[tmcelroy@EX0-Simulation:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/Users/tmcelroy/.ssh/id_rsa):
/Users/tmcelroy/.ssh/id_rsa already exists.
Overwrite (y/n)? y
[Enter passphrase (empty for no passphrase):
[Enter same passphrase again:
Your identification has been saved in /Users/tmcelroy/.ssh/id_rsa.
Your public key has been saved in /Users/tmcelroy/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:R6cpwjdbwFv12630w6NSeQdbVHidRdIfgZEFqvpXCE4 tmcelroy@Thomass-MacBook-Air.local
The key's randomart image is:
+---[RSA 2048]---+
|                .oB*X|
|      .    o.o.+|=|
|      o o.o  oo|
|      .    E.+ o..o|
|      o S.* o ++.|
|      o.B . =o+.|
|      ..   o o+o|
|      .   o . o|
|      .. ..|
+---[SHA256]---+
[tmcelroy@EX0-Simulation:~$ ]
```

Hit enter for default

Generating an ssh key

22

```
[tmcelroy@EX0-Simulation:~$ exit
logout
Connection to 000.000.000.000
[tmcelroy@~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/Users/tmcelroy/.ssh/id_rsa):
/Users/tmcelroy/.ssh/id_rsa already exists.
Overwrite (y/n)? y
[Enter passphrase (empty for no passphrase):
[Enter same passphrase again:
Your identification has been saved in /Users/tmcelroy/.ssh/id_rsa.
Your public key has been saved in /Users/tmcelroy/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:R6cpwjdbwFv12630w6NSeQdbVHidRdIfgZEFqvpXCE4 tmcelroy@Thomass-MacBook-Air.local
The key's randomart image is:
+---[RSA 2048]---+
|                .oB*X|
|      .    o.o.+|=|
|      o o.o  oo|
|      .    E.+ o..o|
|      o S.* o ++.|
|      o.B . =o+.|
|      ..   o o+o|
|      .   o . o|
|      .. ..|
+---[SHA256]---+
[tmcelroy@~$ ]
```

If you are asked this then
you already have one.....

Generating an ssh key

23

```
[tmcelroy@EX0-Simulation:~$ exit
logout
Connection to 000.000.000.000
[tmcelroy@~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/Users/tmcelroy/.ssh/id_rsa):
/Users/tmcelroy/.ssh/id_rsa already exists.
Overwrite (y/n)? y
[Enter passphrase (empty for no passphrase):
[Enter same passphrase again:
Your identification has been saved in /Users/tmcelroy/.ssh/id_rsa.
Your public key has been saved in /Users/tmcelroy/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:R6cpwjdbwFv12630w6NSeQdbVHidRdIfgZEFqvpXCE4 tmcelroy@Thomass-MacBook-Air.local
The key's randomart image is:
+---[RSA 2048]-----+
|          .oB*X|
|      .  o.o.+|=|
|      o o.o  oo|
|      .  E.+ o..o|
|      o S.* o ++.|
|      o.B . =o+.|
|      ..  o o+o|
|      .  o . o|
|      .. ..|
+---[SHA256]-----+
[tmcelroy@~$ ]
```

For extra security, I have not used it.... Don't hack me.

Generating an ssh key

```
[tmcelroy@EXO-Simulation:~$ cd .ssh  
[tmcelroy@EXO-Simulation:~/.ssh$ ls  
authorized_keys  id_rsa  id_rsa.pub  known_hosts  
[tmcelroy@EXO-Simulation:~/.ssh$ vi authorized_keys
```

This folder will also be created.
Note: the “.” as a file’s or folder’s first character makes it hidden to standard file browsers including the default ls command (need ls -a).

Generating an ssh key

25

```
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQCA12gM0QQvUqYTQub4h/9fUGMSny3Qk/7umYwQwMBink7nbtVp/AftUkj1d7KFB1xxDGLgwCZRSgVW0aNMjVsFJLUfDZW5oRtqkxTxw  
+inVtmnHzfcIfqFo67Vw5I0hzwmFd0ILNe0BtJkbY1W851/Wp0XNCLC7nDVQ/1Z8m4yIz4LS38QM+AV1/VRfJv41tBVKqrjSiLf5cK2lnwQQ0u00putVW9XjgeI+p+0ueP5klhe9gN1l1  
1jcdXZHNbmBqx+q5lKCKoMjVLyPA1SoLerGcEv0LQyX7G2Xn2M+B8mnmBXWsUC3gk1DV7fqk3BvH2GeGMQB+c+CfikDBkNt8Sd tmcelroy@Thomass-MacBook-Air.local  
~  
"id_rsa.pub" 1L, 416C
```

Copy this, then exit

[esc] :q

Generating an ssh key

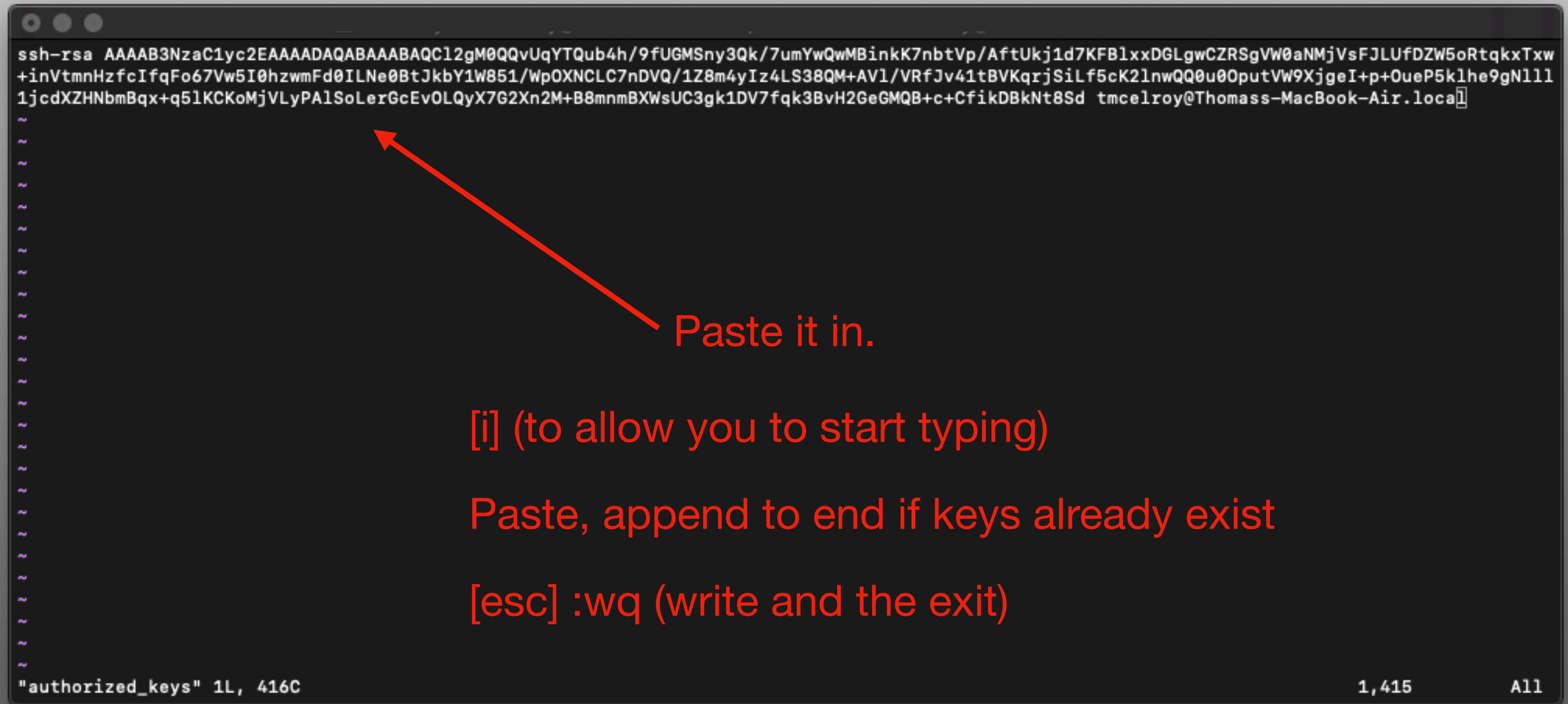
26

```
[tmcelroy@EXO-Simulation:~$ cd .ssh  
[tmcelroy@EXO-Simulation:~/.ssh$ ls  
authorized_keys  id_rsa  id_rsa.pub  known_hosts  
[tmcelroy@EXO-Simulation:~/.ssh$ vi authorized_keys ]
```

Reconnect to the remote computer and create another ssh key for that computer if one doesn't already exist. Enter the .ssh folder and open or create the file `authorized_keys`.

Generating an ssh key

27



The image shows a terminal window with a dark background. At the top, a long SSH key is pasted into the terminal. A red arrow points from the text "Paste it in." to the key. Below the key, there are several lines of red text providing instructions: "[i] (to allow you to start typing)", "Paste, append to end if keys already exist", and "[esc] :wq (write and the exit)". At the bottom of the terminal, the status bar shows "authorized_keys" 1L, 416C, 1,415, and All.

```
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQACl2gM0QQvUqYTQub4h/9fUGMSny3Qk/7umYwQwMBinkK7nbtVp/AftUkj1d7KFB1xxDGLgwCZRSgVW0aNMjVsFJLUfDZW5oRtqkxTxw
+inVtmnHzfcIfqFo67Vw5I0hzwmFd0ILNe0BtJkbY1W851/Wp0XNCLC7nDVQ/1Z8m4yIz4LS38QM+AV1/VRfJv41tBVKqrjSiLf5cK2lnwQQ0u00putVW9XjgeI+p+OueP5klhe9gN111
1jcdXZHNbmBqx+q5lKCKoMjVLyPA1SoLerGcEvOLQyX7G2Xn2M+B8mnmBXWsUC3gk1DV7fqk3BvH2GeGMQB+c+CfikDBkNt8Sd tmcelroy@Thomass-MacBook-Air.local
```

Paste it in.

[i] (to allow you to start typing)

Paste, append to end if keys already exist

[esc] :wq (write and the exit)

"authorized_keys" 1L, 416C 1,415 All

You shouldn't be asked for a password on future connections.

Round 1 of useful UNIX commands

- “.” Stands for current directory, “..” Stands for one directory back. “~” Reference your home directory
- cd - change directory
 - If “/” is first character in directory name then it looks at the start of the file system, otherwise it starts in current directory
- mkdir - makes a new directory (does not enter directory! Must use cd after.)
- cp - copy one file to a new location while keeping the original.
- mv - move a file to a new location, removing the previous file (be careful!)
- rm - remove file, also be careful. Will discuss safety with this command later.

VIM

- Easily mistaken for plain vi (vim is often alias to vi)
- Takes a bit more learning to master but once you get the hang of it, it becomes a very useful light weight editor that is usually installed everywhere.
- Good resources for command (<https://michaelgoerz.net/refcards/vimqrc.pdf>)
- There are two main modes, command mode where you are telling VIM how to handle the document (write, exit, goto line....) and editing (insert or replace).
- Writing more is entered by hitting i or r and returned to command mode with [esc].

VIM

- VIM can be tailored to your preferences using a .vimrc file.
- Good resource for setting this up (<https://dougblack.io/words/a-good-vimrc.html>)
- Good first start is provided in course unix course materials (we will pull it from the git repository soon)

VIM - the world I never knew existed

* don't dive into this until you have a good grasp of basic vim

- VIM is a big deal and an odd flex for programmers
- I asked my software engineer brother for a nice .vimrc file to share and I was quickly thrown into a world of plugins that I had never heard of...
- I have posted the “simple” .vimrc that he provided but it will take a few steps to use
- Need to install vundle (<https://github.com/VundleVim/Vundle.vim>)
- And then run: vim +PluginInstall +qall
- Based on what is installed on your system you might need to remove some plugins (YouCompleteMe didn't work with my python3)

Emacs

- This editor requires the ability to view GUI interfaces.
- Acts more like a standard text editor.
- Has a command bar for saving and exiting.
- Can also customize with an initialization file. (https://www.gnu.org/software/emacs/manual/html_node/emacs/Init-File.html)
- Emacs is not default on a lot of distributions and might not be on every cluster.

Nano

- Happy medium between vim and emacs.
- Most useful command always displayed at bottom of editor
- Installed by default on most distributions.
- Good tutorial (<https://www.howtogeek.com/howto/42980/the-beginners-guide-to-nano-the-linux-command-line-text-editor/>)
- Customized by creating a .nanorc ([https://linuxhint.com/configure nano text editor nanorc/](https://linuxhint.com/configure_nano_text_editor_nanorc/))

Non command line editors

- There are a number of good editors depending on operating system
- My preferred editor is sublime because it is good and works on linux, Mac and Windows.
- There is also a new git client version, I have not used but looks good.
- Free version works well but will often ask you to buy (can just ignore)
- Again, remember to configure your tab key.
- If on windows, also make sure that the end of line character is properly set or files will not work properly on unix.

Other useful text commands

- cat - good for quickly viewing simple text files. Can also combine files.
 - More here: <https://www.geeksforgeeks.org/cat-command-in-linux-with-examples/>
- diff- looks through two files and displays the difference.
 - More here: <https://www.geeksforgeeks.org/diff-command-linux-examples/>
- grep - searches files for matches in text.
 - More here: <https://www.geeksforgeeks.org/grep-command-in-unixlinux/>

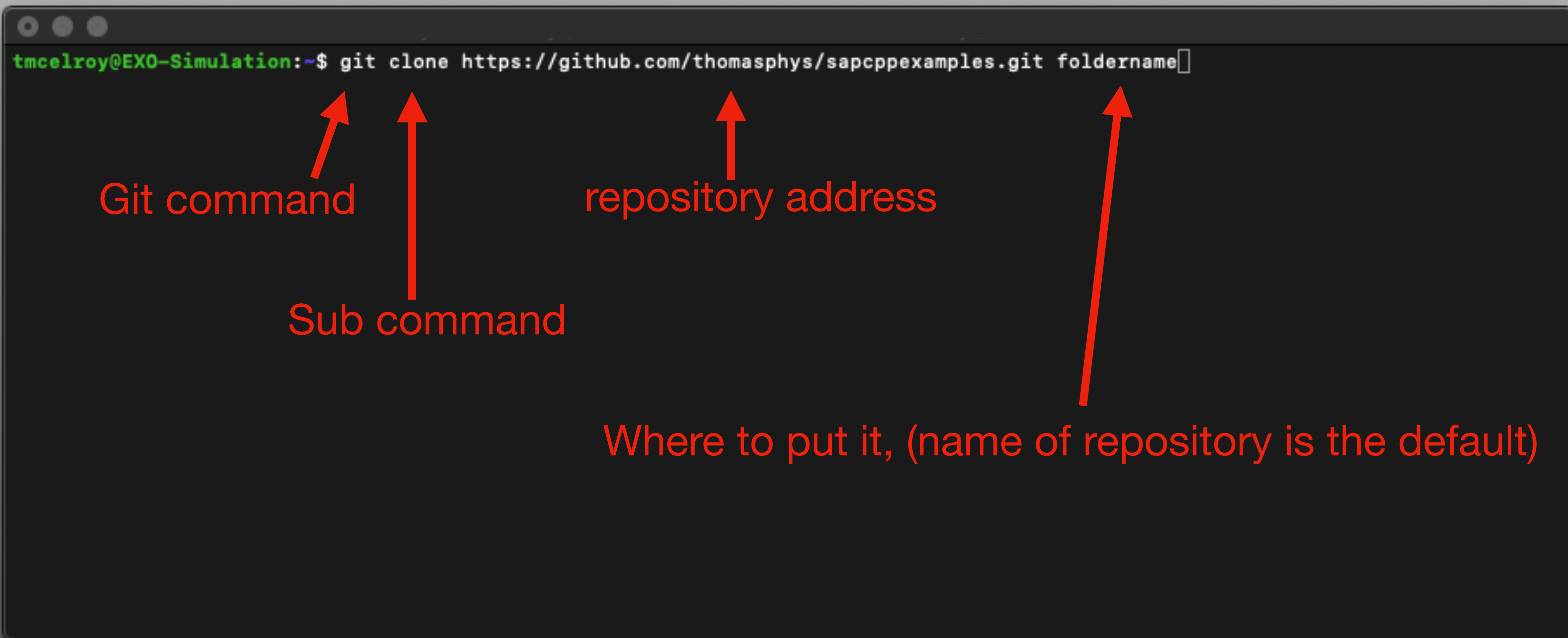
Git

Let's get some stuff and play with it.

- Git is a version control system.
- Keeps track of changes to code and manages combining files being worked on by different people.
- Almost all physics projects will use this to manage code. (really old projects might be using svn, but we won't get into that)
- Git allows you to create your own “branch” of the code to develop and then merge it back into the master code when you have everything functioning.
- There are also a number of git repositories on the internet have loads of useful code (github, gitlab, bitbucket)

Git

Lets start by grabbing a repository



```
tmcelroy@EX0-Simulation:~$ git clone https://github.com/thomasphys/sapcppexamples.git foldername
```

The image shows a terminal window with a dark background and light green text. The command being entered is `git clone https://github.com/thomasphys/sapcppexamples.git foldername`. Four red arrows point from text labels to parts of the command: one from 'Git command' to 'git', one from 'Sub command' to 'clone', one from 'repository address' to the URL, and one from 'Where to put it, (name of repository is the default)' to 'foldername'.

Git command

Sub command

repository address

Where to put it, (name of repository is the default)

Git

Lets make a branch to track our own work

```
* Canonical Livepatch is available for installation.
- Reduce system reboots and improve kernel security. Activate at:
  https://ubuntu.com/livepatch

467 packages can be updated.
15 updates are security updates.

*** System restart required ***
Last login: Sun May  3 19:28:27 2020 from 000.000.000.000
[tmcelroy@EXO-Simulation:~$ git clone https://github.com/thomasphys/sapcppexamples.git
Cloning into 'sapcppexamples'...
remote: Enumerating objects: 25, done.
remote: Counting objects: 100% (25/25), done.
remote: Compressing objects: 100% (17/17), done.
remote: Total 25 (delta 2), reused 25 (delta 2), pack-reused 0
Unpacking objects: 100% (25/25), done.
[tmcelroy@EXO-Simulation:~$ cd sapcppexamples/
[tmcelroy@EXO-Simulation:~/sapcppexamples$ git branch mybranch
[tmcelroy@EXO-Simulation:~/sapcppexamples$ git branch
* master
  mybranch
[tmcelroy@EXO-Simulation:~/sapcppexamples$ git checkout mybranch
Switched to branch 'mybranch'
tmcelroy@EXO-Simulation:~/sapcppexamples$
```

Enter repository folder

Git

Lets make a branch to track our own work

```
* Canonical Livepatch is available for installation.
- Reduce system reboots and improve kernel security. Activate at:
  https://ubuntu.com/livepatch

467 packages can be updated.
15 updates are security updates.

*** System restart required ***
Last login: Sun May  3 19:28:27 2020 from 000.000.000.000
[tmcelroy@EXO-Simulation:~$ git clone https://github.com/thomasphys/sapcppexamples.git
Cloning into 'sapcppexamples'...
remote: Enumerating objects: 25, done.
remote: Counting objects: 100% (25/25), done.
remote: Compressing objects: 100% (17/17), done.
remote: Total 25 (delta 2), reused 25 (delta 2), pack-reused 0
Unpacking objects: 100% (25/25), done.
[tmcelroy@EXO-Simulation:~$ cd sapcppexamples/
[tmcelroy@EXO-Simulation:~/sapcppexamples$ git branch mybranch
[tmcelroy@EXO-Simulation:~/sapcppexamples$ git branch
* master
  mybranch
[tmcelroy@EXO-Simulation:~/sapcppexamples$ git checkout mybranch
Switched to branch 'mybranch'
tmcelroy@EXO-Simulation:~/sapcppexamples$
```

Create new Branch

Git

Lets make a branch to track our own work

```
* Canonical Livepatch is available for installation.
- Reduce system reboots and improve kernel security. Activate at:
  https://ubuntu.com/livepatch

467 packages can be updated.
15 updates are security updates.

*** System restart required ***
Last login: Sun May  3 19:28:27 2020 from 100.000.000.000
[tmcelroy@EXO-Simulation:~$ git clone https://github.com/thomasphys/sapcppexamples.git
Cloning into 'sapcppexamples'...
remote: Enumerating objects: 25, done.
remote: Counting objects: 100% (25/25), done.
remote: Compressing objects: 100% (17/17), done.
remote: Total 25 (delta 2), reused 25 (delta 2), pack-reused 0
Unpacking objects: 100% (25/25), done.
[tmcelroy@EXO-Simulation:~$ cd sapcppexamples/
[tmcelroy@EXO-Simulation:~/sapcppexamples$ git branch mybranch
[tmcelroy@EXO-Simulation:~/sapcppexamples$ git branch
* master
  mybranch
[tmcelroy@EXO-Simulation:~/sapcppexamples$ git checkout mybranch
Switched to branch 'mybranch'
tmcelroy@EXO-Simulation:~/sapcppexamples$
```

← List branches

Git

Lets make a branch to track our own work

```
* Canonical Livepatch is available for installation.
- Reduce system reboots and improve kernel security. Activate at:
  https://ubuntu.com/livepatch

467 packages can be updated.
15 updates are security updates.

*** System restart required ***
Last login: Sun May  3 19:28:27 2020 from 000.000.000.000
[tmcelroy@EXO-Simulation:~$ git clone https://github.com/thomasphys/sapcppexamples.git
Cloning into 'sapcppexamples'...
remote: Enumerating objects: 25, done.
remote: Counting objects: 100% (25/25), done.
remote: Compressing objects: 100% (17/17), done.
remote: Total 25 (delta 2), reused 25 (delta 2), pack-reused 0
Unpacking objects: 100% (25/25), done.
[tmcelroy@EXO-Simulation:~$ cd sapcppexamples/
[tmcelroy@EXO-Simulation:~/sapcppexamples$ git branch mybranch
[tmcelroy@EXO-Simulation:~/sapcppexamples$ git branch
* master
  mybranch
[tmcelroy@EXO-Simulation:~/sapcppexamples$ git checkout mybranch
Switched to branch 'mybranch'
tmcelroy@EXO-Simulation:~/sapcppexamples$
```

Switch to your branch

Git

Make a change and commit your changes locally

```
[tmcelroy@EX0-Simulation:~/sapcppexamples$ git checkout mybranch
Switched to branch 'mybranch'
[tmcelroy@EX0-Simulation:~/sapcppexamples$ ls
CMakeHelloWorld  GNUMakeHelloWorld  hellocsv  helloworld  README.md
[tmcelroy@EX0-Simulation:~/sapcppexamples$ vi README.md
[tmcelroy@EX0-Simulation:~/sapcppexamples$ git commit --all
Aborting commit due to empty commit message.
[tmcelroy@EX0-Simulation:~/sapcppexamples$ git commit --all -m "first commit to my branch"
[mybranch 664b7ec] first commit to my branch
Committer: Thomas McElroy <tmcelroy@physics.mcgill.ca>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly:

    git config --global user.name "Your Name"
    git config --global user.email you@example.com

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author

1 file changed, 2 insertions(+)
```

Make changes

Git

Make a change and commit your changes locally

```
[tmcelroy@EX0-Simulation:~/sapcppexamples$ git checkout mybranch
Switched to branch 'mybranch'
[tmcelroy@EX0-Simulation:~/sapcppexamples$ ls
CMakeHelloWorld  GNUMakeHelloWorld  hellocsv  helloworld  README.md
[tmcelroy@EX0-Simulation:~/sapcppexamples$ vi README.md
[tmcelroy@EX0-Simulation:~/sapcppexamples$ git commit --all
Aborting commit due to empty commit message.
[tmcelroy@EX0-Simulation:~/sapcppexamples$ git commit --all -m "first commit to my branch"
[mybranch 664b7ec] first commit to my branch
Committer: Thomas McElroy <tmcelroy@physics.mcgill.ca>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly:

    git config --global user.name "Your Name"
    git config --global user.email you@example.com

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author

1 file changed, 2 insertions(+)
```

Commit the changes locally

Git

Make a change and commit your changes locally

```
[tmcelroy@EX0-Simulation:~/sapcppexamples$ git checkout mybranch
Switched to branch 'mybranch'
[tmcelroy@EX0-Simulation:~/sapcppexamples$ ls
CMakeHelloWorld  GNUMakeHelloWorld  hellocsv  helloworld  README.md
[tmcelroy@EX0-Simulation:~/sapcppexamples$ vi README.md
[tmcelroy@EX0-Simulation:~/sapcppexamples$ git commit --all
Aborting commit due to empty commit message.
[tmcelroy@EX0-Simulation:~/sapcppexamples$ git commit --all -m "first commit to my branch"
[mybranch 664b7ec] first commit to my branch
Committer: Thomas McElroy <tmcelroy@physics.mcgill.ca>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly:

    git config --global user.name "Your Name"
    git config --global user.email you@example.com

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author

1 file changed, 2 insertions(+)
```

Must give a message, git will open the default text editor for you to write the comment, if you don't it won't commit. You can also use -m "message" for short comments.

Git

Now to make use of file backup, regularly (daily) push your changes to the remote repository

```
1 file changed, 2 insertions(+)
[tmcelroy@EXO-Simulation:~/sapcppexamples$ git push
fatal: The current branch mybranch has no upstream branch.
To push the current branch and set the remote as upstream, use

    git push --set-upstream origin mybranch

[tmcelroy@EXO-Simulation:~/sapcppexamples$ git push --set-upstream origin mybranch
[Username for 'https://github.com': thomasphys
[Password for 'https://thomasphys@github.com':
Counting objects: 3, done.
Delta compression using up to 16 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 342 bytes | 342.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote:
remote: Create a pull request for 'mybranch' on GitHub by visiting:
remote:   https://github.com/thomasphys/sapcppexamples/pull/new/mybranch
remote:
To https://github.com/thomasphys/sapcppexamples.git
 * [new branch]      mybranch -> mybranch
Branch 'mybranch' set up to track remote branch 'mybranch' from 'origin'.
tmcelroy@EXO-Simulation:~/sapcppexamples$
```

Push current branch to usual place online

Git

Now to make use of file backup, regularly (>daily) push your changes to the remote repository

```
1 file changed, 2 insertions(+)
[tmcelroy@EXO-Simulation:~/sapcppexamples$ git push
fatal: The current branch mybranch has no upstream branch.
To push the current branch and set the remote as upstream, use

    git push --set-upstream origin mybranch

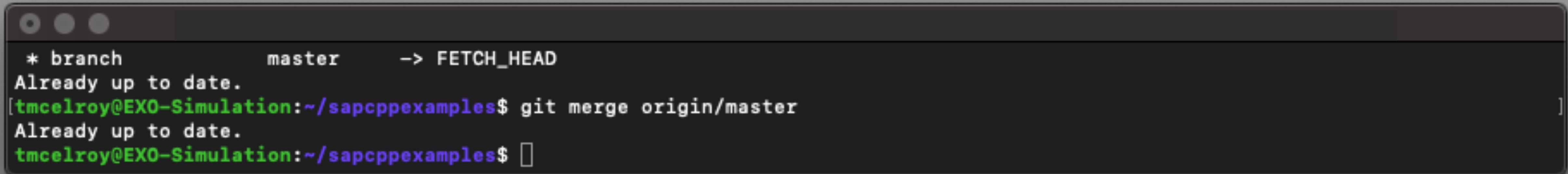
[tmcelroy@EXO-Simulation:~/sapcppexamples$ git push --set-upstream origin mybranch
Username for 'https://github.com': thomasphys
Password for 'https://thomasphys@github.com':
Counting objects: 3, done.
Delta compression using up to 16 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 342 bytes | 342.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote:
remote: Create a pull request for 'mybranch' on GitHub by visiting:
remote:   https://github.com/thomasphys/sapcppexamples/pull/new/mybranch
remote:
To https://github.com/thomasphys/sapcppexamples.git
 * [new branch]      mybranch -> mybranch
Branch 'mybranch' set up to track remote branch 'mybranch' from 'origin'.
tmcelroy@EXO-Simulation:~/sapcppexamples$
```

First push of this branch so we need to first set up branch on remote repository. Git is usually pretty good with telling you that you messed up and what to do.

Git

Once you have finished your changes to the code you will want to merge it into the master branch.

If you have been working for a long time and master branch has changed since you branched from it, you should always pull the master branch changes into your branch and double check that things are all working.

A terminal window with a dark background and light text. It shows the output of a git merge command. The first line indicates the current branch is 'master' and it is up to date. The second line shows the command 'git merge origin/master' being executed. The third line shows the output 'Already up to date.' and the prompt returns to the user.

```
* branch          master    -> FETCH_HEAD
Already up to date.
[tmcelroy@EX0-Simulation:~/sapcppexamples$ git merge origin/master
Already up to date.
tmcelroy@EX0-Simulation:~/sapcppexamples$ ]
```

If there were changes in the master branch you would be informed if there were non-mergeable changes that you would need to deal with. Once all the changes have been made you recommit and push. The merge request is best done from the repository website.

Git

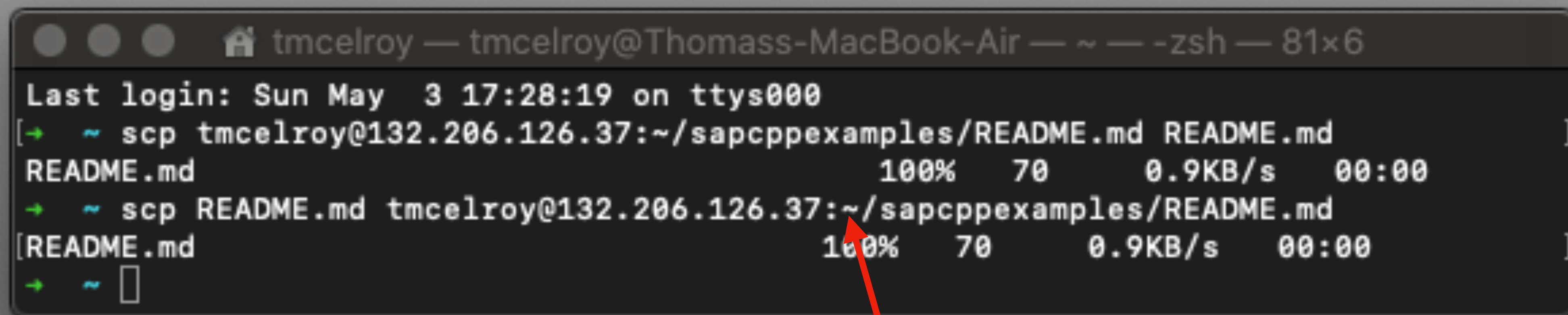
Some extra things

- git fetch - this will get information on remote branches but will not pull the changes.
- git pull - this will pull remote changes of current branch
- git pull origin/<branch> will pull changes of <branch>
- Git stash - saves the changes to the branch locally. Use if you want to temporarily go back to a previous version.
- Git reset - reset the code to the last version online (can also go back the specific version) user git reset —hard to force override any changes without stashing.

Online documentation: <https://git-scm.com/>

Other file download / transfer

- Wget, commonly used to retrieve code that you are not expected to change.
- <https://www.webhostface.com/kb/knowledgebase/examples-using-wget/>
- SCP (secure copy) like ssh but just transfers files.



```
tmcelroy — tmcelroy@Thomass-MacBook-Air — ~ — -zsh — 81x6
Last login: Sun May  3 17:28:19 on ttys000
[→ ~ scp tmcelroy@132.206.126.37:~/sapcppexamples/README.md README.md
README.md                                100% 70    0.9KB/s   00:00
[→ ~ scp README.md tmcelroy@132.206.126.37:~/sapcppexamples/README.md
[README.md                                100% 70    0.9KB/s   00:00
[→ ~ ]
```

retrieve

send

~ reference home directory of user

More info: <https://linuxize.com/post/how-to-use-scp-command-to-securely-transfer-files/>

Other file download / transfer

- You can also use SSHFS to mount a remote disk for more direct access.
- On linux you install sshfs, on Mac osxfuse and sshfs and Windows WinFsp and SSHFS-Win.
- A folder is created to mount the remote folder to
- Sshfs uses ssh to communicate and transfer files.
- `sshfs tmcelroy@000.000.000.000:/home/tmcelroy /Users/tmcelroy/office/`
- More information here: <https://linuxize.com/post/how-to-use-sshfs-to-mount-remote-directories-over-ssh/>

Installing programs and libraries

- Let's consider a simple library and program, let's borrow the one in the cppexamples that we just pulled from GitHub.
- We can either build the program and keep it as a local build (typical for building things on clusters) or install the program into the standard location.
- This is also true for libraries.

Lets quickly build the library

- Details will be given on cake in the c++ lecture, for now just copy me.

```
[tmcelroy@EXO-Simulation:~/sapcppexamples$ ls
CMakeHelloWorld  GNUMakeHelloWorld  hellocsv  hellolibrary  helloworld  README.md
[tmcelroy@EXO-Simulation:~/sapcppexamples$ cd hellolibrary/
[tmcelroy@EXO-Simulation:~/sapcppexamples/hellolibrary$ mkdir build
[tmcelroy@EXO-Simulation:~/sapcppexamples/hellolibrary$ cd build
[tmcelroy@EXO-Simulation:~/sapcppexamples/hellolibrary/build$ cmake ../
-- The C compiler identification is GNU 7.5.0
-- The CXX compiler identification is GNU 7.5.0
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Configuring done
-- Generating done
-- Build files have been written to: /home/tmcelroy/sapcppexamples/hellolibrary/build
[tmcelroy@EXO-Simulation:~/sapcppexamples/hellolibrary/build$ cmake --build ./
Scanning dependencies of target hellolib
[ 50%] Building CXX object CMakeFiles/hellolib.dir/src/helloclass.cpp.o
[100%] Linking CXX shared library lib/libhellolib.so
[100%] Built target hellolib
tmcelroy@EXO-Simulation:~/sapcppexamples/hellolibrary/build$
```

Lets quickly build the library

- We now have a locally built library.
- Compilers do not know that it exist, if we want to use it we have to tell them where to find it.
- We have two options to automate this.
 - 1. Add the library location to our library search path
 - 2. Install the library into the standard search path.

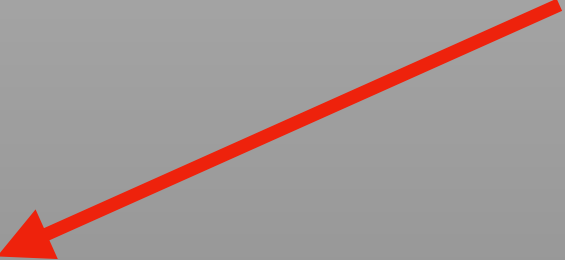
Lets quickly build the library

Adding library to search path

- This is very simple. Just need to append the LD_LIBRARY_PATH variable.
- In your home directory, open or create .bashrc. (might be named diff on some linux/UNIX)
- Create or edit the line:
 - `export LD_LIBRARY_PATH=/path/to/library/:$LD_LIBRARY_PATH`
- `source .bashrc` (or just open new terminal window)
- You will sill need to tell your programs where to find the header files located in the include folder.
- If you are installing things on a cluster this is often the only option.
- If you have access to your experiments preinstalled libraries, they can be added to you system by adding the path

Lets quickly build the library

Adding library to search path

- This is very simple. Just need to append the LD_LIBRARY_PATH variable.
 - In your home directory, open or create .bashrc.
 - Create or edit the line:
 - `export LD_LIBRARY_PATH=/path/to/library/:$LD_LIBRARY_PATH`
 - `source .bashrc` (or just open new terminal window)
 - You will sill need to tell your programs where to find the header files located in the include folder.
 - If you are installing things on a cluster this is often the only option.
 - If you have access to your experiments preinstalled libraries, they can be added to you system by adding the path
- By putting path first, if duplicate library names exist it takes this library first.
- 

Lets quickly build the library

Adding library to search path

- This is very simple. Just need to append the LD_LIBRARY_PATH variable.
- In your home directory, open or create .bashrc.
- Create or edit the line:
 - `export LD_LIBRARY_PATH=/path/to/library/:$LD_LIBRARY_PATH`
- `source .bashrc` (or just open new terminal window)
- You will sill need to tell your programs where to find the header files located in the include folder.
- If you are installing things on a cluster this is often the only option.
- If you have access to your experiments preinstalled libraries, they can be added to you system by adding the path

Still want the original search paths.



Lets quickly build the library

Installing library to standard search path

- This can be done by moving the library and include files manually but that is not preferred.
- Should be done by the install function built into the compile scripts.
- This will require ROOT permission which you will only have if you are working on your own computer.
- Typical install location on linux is /usr/local/ but should double check for your specific linux distribution.

Lets quickly build the library

Installing library to standard search path

- This can be done by moving the library and include files manually but that is not preferred.
- Should be done by the install function built into the compile scripts.
- This will require ROOT permission which you will only have if you are working on your own computer.
- Typical install location on linux is /usr/local/ but should double check for your specific linux distribution.

```
[tmcelroy@EXO-Simulation:~/sapcppexamples/hellolibrary/build$ sudo cmake --build ./ --target install ]  
[[sudo] password for tmcelroy:  
tmcelroy is not in the sudoers file. This incident will be reported.  
tmcelroy@EXO-Simulation:~/sapcppexamples/hellolibrary/build$ ]
```

Need to be on account
with root permissions

Installing programs

- Same with libraries, these can be installed locally or into the standard location.
- Program can be run in local directory with `./programname`
- From anywhere by putting full path `/home/user/folder/programname`
- Or the path to the program can be added to the `PATH` variable same as we did with the libraries.

Sometimes they do this all for us

- For many packages that we download and build, environment files are created that all you need to do is source that file and all the required variable will be set.
- This should be mentioned in the build instructions, sometimes you must run the install command, for local installs give it your own install folder.
- To automate this again, we can add “source /path/to/envfile.sh” to the .bashrc

Additional environmental setup

- I suggest you do some work to fine tune your working environment for you.
- This means setting things in your `.bashrc` files to speedup standard task.
- The `alias` command can be quite useful for doing this.

Useful Aliases

Be careful to not overwrite commands that you don't mean to!

- For safety alias rm to have some safety: `alias rm="rm\ -I"`
- For VIM not VI if not already done: `alias vi="vim"`
- I alias my ssh calls: `alias office="ssh -Y user@address.ca"`
- Alias Docker setup
- Common Make commands: `alias cbuild="cmake --build ./"`
- Common work directories: `alias scratch="cd /scratch/user/"`
- Clear terminal window: `alias c='clear'`
- more: <https://www.cyberciti.biz/tips/bash-aliases-mac-centos-linux-unix.html>

Bash Scripts

- More advanced automation can be created using a bash
- To first order, they are just a list of commands but can be much more.
- Can be run my just typing path the script. `$./script.sh`
- Too much to go through here, if you feel like you want to automate something read this <https://linuxconfig.org/bash-scripting-tutorial-for-beginners>
- You must have executable permission of the file to run it.

File Permission

- Files have different permission on computers both for privacy and for safety in the operating system.
- You have the ability to change the permission on files that you yourself have permission on.
- If you have root access then you can do anything but should only take advantage of this if you know what your are doing!
- If you are on an account with root access, you may still need to use the sudo command to access to perform tasks that require root access.
- Permissions for files can be changed using chmod

chmod

- Two ways to specify the permissions, alpha and numeric
- `chmod u=rwx,g=rx,o=r myfile` (u=user, g=group, o=other, r=read, w=write, x=execute)
- `chmod 754 myfile` (0=no permission, 1=execute, 2=write, 4=read, sum together) digits in ugo order.
- More info: <https://www.computerhope.com/unix/uchmod.htm>

Working on a Cluster

- There are a number clusters across Canada that students are using.
- While practically all of these clusters are linux based, there are a few differences that students should look at the documentation of the cluster to get specific commands.
- We will cover information useful for Cedar and Graham, similar protocols exist for other clusters but details should be obtained from that clusters documentation.
- Compute Canada Doc: https://docs.computecanada.ca/wiki/Getting_started
- Extra Slurm Stuff: <https://www.ch.cam.ac.uk/computing/slurm-usage>

Cluster Etiquet

- Login Nodes are for light work! Do not run large scripts/programs (salloc, srun)
- This includes compiling large codes (check with your group, large packages are often compiled for the group and you can just link to them)
- When you run a lot of jobs and create a lot of small files, if possible compile them together into fewer but still manageable sized files (see hadd for root files)
There is often a max number of files that can be stored and that limit is hit by many experiments.
- Try to always put output files into the /scratch area first and then transfer what files need to be kept long term to your project directory or if you must, your home directory.

Setting up the environment

- Clusters often have multiple version of libraries available and non standard versions can be fetched by loading appropriate modules.
- When called these modules change your LD_LIBRARY_PATH and PATH to point to the correct version of the library or program.
- Modules are loaded with:
 - module <name of module>
- This can be added to your .bashrc to automate every time you login.
- More information: https://docs.computecanada.ca/wiki/Utiliser_des_modules/en

Running Interactive Session

- If we need to perform an interactive task on a cluster (compiling large pieces of code, debugging, long root sessions making plots.....) you should enter an interactive session.
- Two ways to do this:
 - `salloc - -time=h:m:s - -ntasks=2 - -account=def-someuser` (add `- -x11` for graphical applications)
 - `srun - -account=def-someuser - -time=h:m:s --mem=(in mb) executable`

Running Interactive Session

salloc

- This will open an interactive session where you can work and run code.
- The session is over when you run out of time or you type exit.
- This is useful if you are making root plots and will be stopping and starting different applications within the interactive session.

Running Interactive Session

srun

- Srun is used when you just need to run a single execution of a command.
- Useful for simple tasks like moving large files between /scratch and project directories.
- If programs crash during an srun session it will kill the session (unless -W 0 flag is given)
- Session can also be killed if job finishes and terminal is left idle for too long.

Submitting Jobs

sbatch

- Most of the computing on clusters is done by submitting jobs to the queue.
- Jobs are passed using a job script.
- simple_job.sh

Job submission:

```
$ sbatch simple_job.sh
```

```
#!/bin/bash
```

```
#SBATCH - -time=hh:mm:ss
```

```
#SBATCH - -account=fed-someuser
```

```
#SBATCH - -mem= in mb
```

```
command_to_execute
```

```
maybe_another
```

Job Management

- `$ squeue -u <username>` : list your jobs in the queue
- `$ srun --jobid 123456 --pty <command>` : connect to node with job and run a command, good for checking things. I never knew this was possible... we all learn.
- `$ scancel <jobid>` - cancel job: please do this if you know something is from with code. Clean up the queue early for others.