A1. BASIC TERMINAL USAGE

Accessing CRC supercomputer

- New user: https://crc.nd.edu/news-events/events/2020/08/13/new-user-training/

- Answers to basic questions:
  https://docs.crc.nd.edu/ (basic Linux guide is also here)
  Under Resources on Sakai, `Appendix_2_unix_tutorial.pdf`

- Log in to CRC super computer from your own computer
  - **Linux**
    - Use a linux terminal and type in:

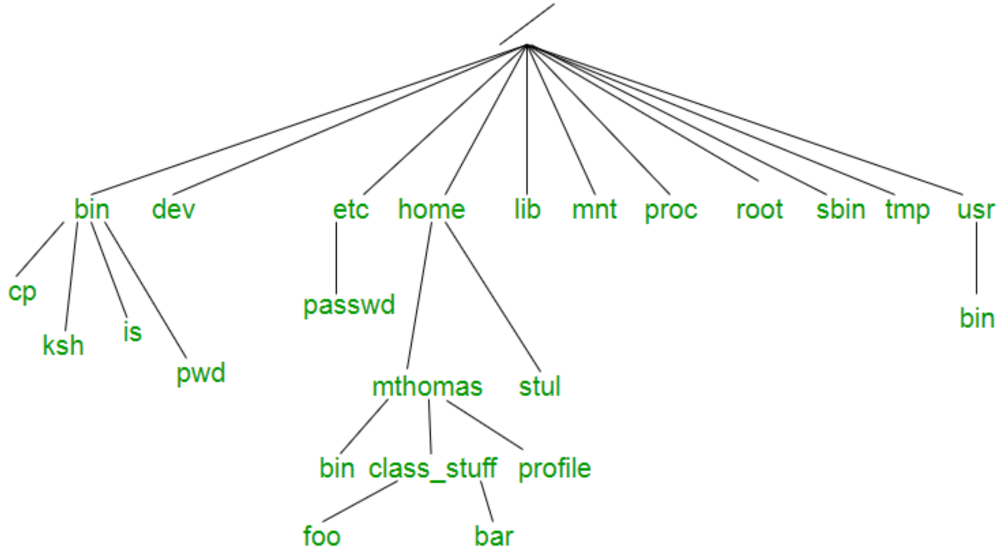
      ```
      ssh -X yourid@crcfe01.crc.nd.edu
      ```
  - **Windows**
    - Use putty https://www.chiark.greenend.org.uk/~sgtatham/putty/
    - Use MobaXterm https://wiki.crc.nd.edu/w/index.php/MobaXterm
  - **Mac**
    - Use built in X11 xterm or terminal.

      ```
      ssh -X yourid@crcfe01.crc.nd.edu
      ```

Unix file system

- https://www.geeksforgeeks.org/unix-file-system/

- Files in Unix (or unix-like) system are organized into multi-level hierarchy structure known as a directory tree. At the very top of the file system is a directory called “root” which is represented by a “/”. All other files are “descendants” of root.



Looking around inside terminal

- To see what is in a directory, look around using the command

```
ls
```

- It will tell you what files are in the current directory.
- Some files are hidden – those which start with periods.
  - To see them, use the 'all' option for ls

    ```
    ls -a
    ```

- To see file properties, use the 'long' flag

```
ls -l
```

Where am i?

- If you get lost, and wonder where you are,
- or, if you want to find out the path to a file in your current location,
- use 'print working directory'

```
pwd
```

To move around in folders

- To go into a directory , 'change directory'
  - `cd directoryname`
- You can go several steps at once
  - `cd first/second/third`
- You can go 'up'
  - `cd ..`
  - `cd ../`

Copy a file

- The name for the copy command is
  - `cp`
- It requires TWO arguments, source and target. The source is the file to be copied. The target is the name of the file to make. You can overwrite old files with this, and it doesn't ask permission...
  - `cp hello.cpp hellocopy.cpp`
- You can copy to other directories, too.
  - `cp hello.cpp ../other/hellocopy.cpp`

Delete a file

- the 'remove' command is
  - `rm`
- All files listed will be deleted.
- There is no trash can, and there is no confirmation dialog. This is final.
  - `rm file1 dir/file2.ext`

Look at a file

- You can 'preview' a file in several ways.
- `more` – display a file to the screen, up to how tall the screen is. press down or space to go forward. sorry, no backsies. Press `q` to stop.
- `less` – same as more, but you can go back. press `q` to stop.

- also, simply edit the file. `nano, emacs, vi, vim`, etc.

Tab completion

- If what you have typed into the terminal specifies a command or file name, 'tab' will complete for you
- Generally, it will complete as much as it can, up to the point where multiple commands or files differ.
- Type one more letter, and press tab to get it to complete more for you.
- You can quickly accomplish tasks by getting the shell to write your commands for you

Going back in time

- Your command history is tracked.
- Press up to go back one command.
- Down to go forward one command.
- Even garbage commands are kept, so watch out.
- Running a command twice will put it in the history twice. etc.

getting help

- options can be passed to many commands, via – or --.
- to find out what options there are, and get help, try
  - `man command`
  - `command --help`
  - `command –h`
- To get out of `man`, press `q`.

A2. EDITING REMOTE FILES

A2.1. In terminal

- Use provided editors in the terminal. Many are faster to use, with practice, than GUI point-and-click editors.

- `emacs`
- `vi`, `vim`
- `nano`
- many more.

- no clicky, hard to copy-paste stuff unless you practice.
- `emacs` versus `vi` war is long standing.

vi / vim

- vi/vim is a modal editor meaning that different things happen when you type, depending on which mode you are in. There are two modes, **Command mode** and **Typing mode**.

  - Use **vi *filename*** to open a file. It opens in **Command mode**.
  - Use **vi** (without a filename).
  - Use **arrows** to go left and right, up and down in either mode.
  - Use **o** in **Command mode** to go to **Typing mode** and open a new line.
  - Use **"ESC"** to go back to **Command mode** from **Typing mode**.
  - Use **:w** to save your file from the **Command mode** (remain in Command mode).
  - Use **:q** from the **Command mode** to quit (**:wq** saves and then quits with one command).
  - Use **x** in the **Command mode** to delete (remain in Command mode).
  - Use **i** in the **Command mode** to go into **Typing mode.**
  - Similar to **x**, there is also **dd** for deleting an entire line in Command mode (remain in Command mode); **5 dd** deletes the next 5 lines.
  - Similar to **i**, there **a** for append (change to Typing mode).
  - If the cursor is over a brace, bracket or parenthesis in Command mode, the **%** key will jump it to the matching brace, bracket or parenthesis (the utility of this will become apparent as the course develops).

Additional vi/vim references

- https://www.cs.colostate.edu/helpdocs/vi.html
- http://heather.cs.ucdavis.edu/~matloff/UnixAndC/Editors/ViIntro.html

A2.2. Enable X11 forwarding

Taxing on the network, so not as responsive as you may wish... But here it is:

  - `ssh -X xxx@crcfe01.crc.nd.edu`

- That X is capital. It matters. This option turns on X11 forwarding.
- Then, you can open the program `gedit` from the command line, and get a GUI text editor remotely.
- Consider opening the editor with an ampersand “&” after its name so you can continue to issue commands without closing the editor.
- `emacs` also works like this.
- Windows users, “`Xming`” is needed.
- Mac users will likely have to install an X11 env, is easy ;)
- Consider turning on compression for `ssh` to improve performance and reduce lag.

A2.3. SFTP + local editor

- Use a program such as `Cyberduck` (kinda free) to SFTP into the CRC front end, and use `Cyberduck` to open your remote file.
- Can use whatever editor you like.
- When you save file opened through `Cyberduck`, automatically uploaded to CRC.

- Recommended editors:
    - Sublime (kinda free),
    - Atom
    - Xcode
    - Notepad++

Pros and cons

- In-terminal
  - All changes saved without risk of loss of connection
  - Good for slower connection
  - Gets you better at using terminal.
  - Boasting rights of knowing emacs or vi
- X forwarded
  - familiar GUI, with clicking
  - slow connections will suffer
  - taxes network and front end for other users.
  - Can only use provided editors
- SFTP + local
  - Arbitrary editor
  - Good for mediocre and fast connections
  - Complete file transfer when save may waste bandwith

A3. BASIC COMPILATION

A3.1. Creating a program

1. Use a text editor to write a program and save it in a file (or multiple files) -> source code
2. Compile the source code (compiler is a program that translates the source code to machine language) -> object code
3. Link the object code with additional code (libraries) -> executable code

Remark: step 2 & 3 sometimes can be combined.

A3.2. our compiler – gcc 8.3/intel 19.0

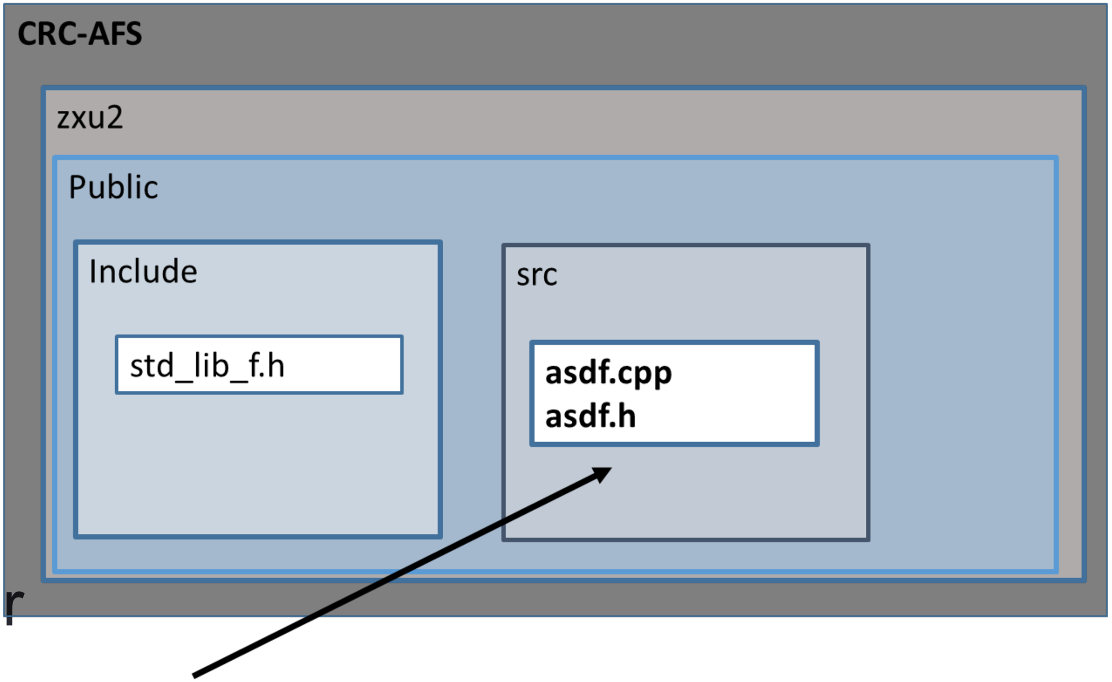
- This semester, we will use intel 19.0 or gcc 8.3
- GCC – GNU Compiler Collection

- Particularly, we use `icpc` (or `icc`) intel C++ compiler. Or we use `g++` `(`or `gcc)`, the GNU C++ (or C) compiler

- you invoke it on the command line to compile.

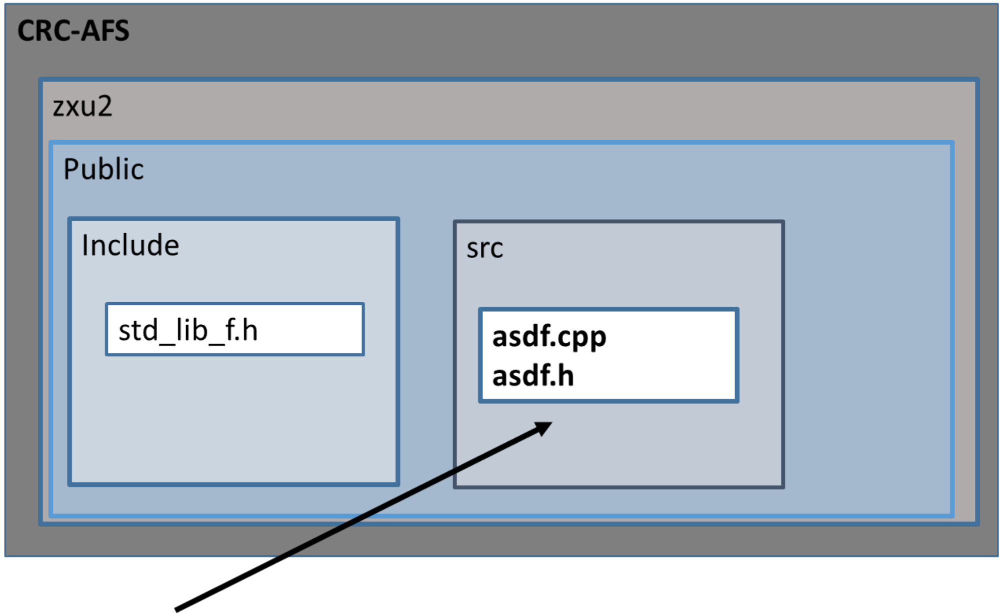
A3.3. How to use compiler -- basic

- Suppose you have one file, `asdf.cpp`
- You also want to compile the program and have it named `zxcv`
- You need to include files located up and over, in `include` folder
- suppose you are working in folder Public/prog/src



Make a program called zxcv

g++ -o zxcv -I../include asdf.cpp



Make a program called zxcv

1. Compile under src directory.

```
cd  ~zxu2/Public/src
```

2. Type

```
$ g++ -o zxcv -I../include asdf.cpp
```

```
g++ -o zxcv -I../include asdf.cpp

icpc -o zxcv -I../include asdf.cpp
```

1. `g++` (`icpc`) -- the name of the compiler, the command to run
2. `-o` -- argument to `g++`, followed by `zxcv`, tells `g++` to name the output `zxcv`
3. `-I../include` -- `-I` tells g++ to search for files to include in the folder "`../include`". hence, we named the folder "`include`" to make it obvious. If the specified directory is a standard system include directory, the option is ignored to ensure that the default search order for system directories and the special treatment of system headers are not defeated .
4. `../` -- `../` is up one level. in contrast, `./` is here, wherever `pwd` tell where you are.
5. `asdf.cpp` -- the name of the source file to compile. Can list several, but many-source programs require a build system to not become cumbersome.