CompSci 2SD3 Tutorial #1

TA: Jatin Chowdhary

DATE: January 25th, 2022

Introduction

- Welcome to the first 2SD3 tutorial
  - Tutorials have officially started
    - Last week's tutorial was a practice run (i.e. Dry run)
- Most of you already know me from 2GA3
  - And vice versa
- In case you're lost I can do a quick introduction
  - I doubt I need to
    - We are way past introductions

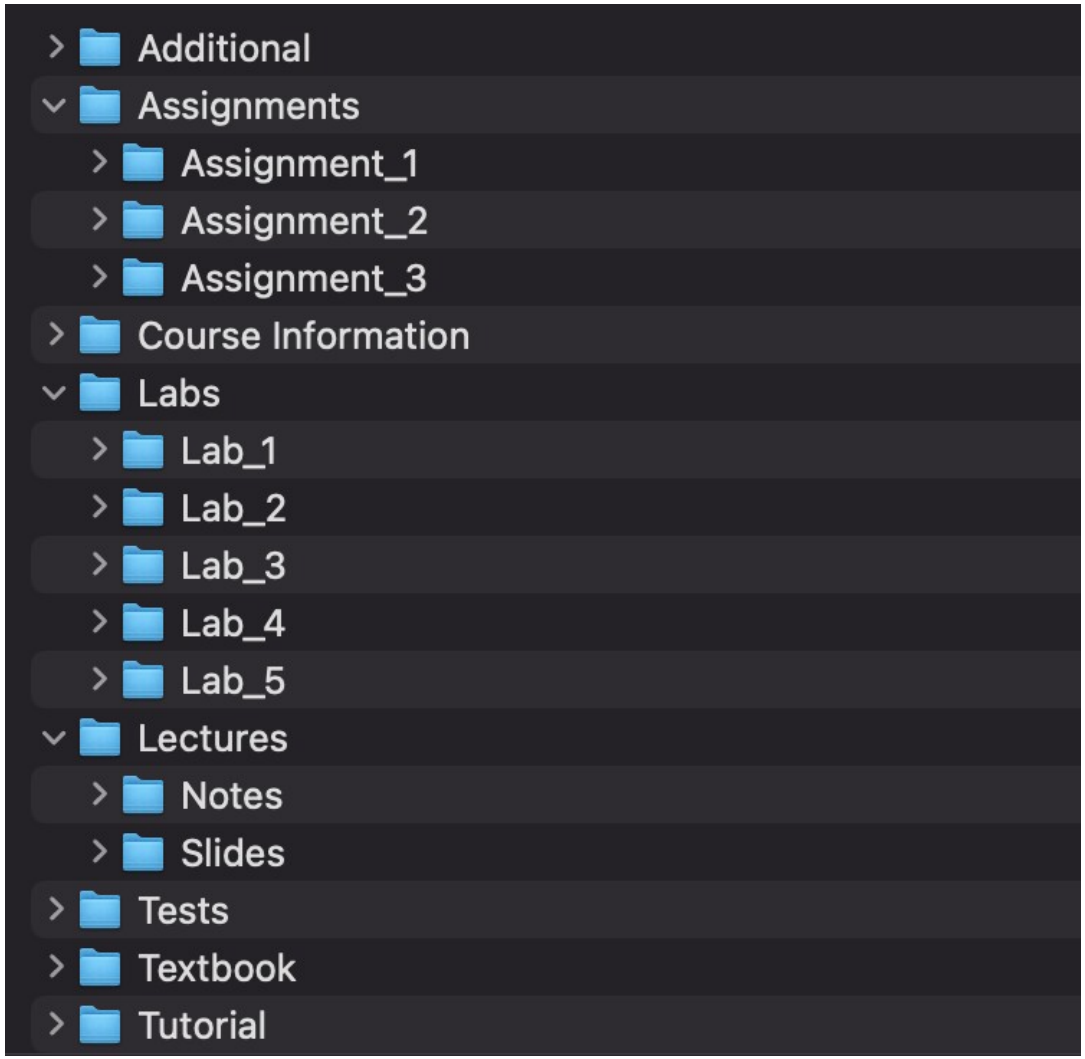
Outline

- Today, we will cover the following:
  - Lingering Questions
  - Lecture Stuff
  - Mills/Moore
  - Unix Commands
  - Compiling
  - Makefiles
  - Text Editors
  - *fork()*
  - Assignment 1

Lingering Questions (1)

- **Question:** Do I have a portfolio?
  - *I can't remember who asked this*
- **Answer:** Soon™
  - I'm working on it – should be done by summer.
  - You should copy what I'm doing
    - *Next slide*

Lingering Questions (2)



- All the work you have done for every course is neatly organized into their respective folders
  - Anyone can easily see what you have

Lingering Questions (3)

- **Question:** Why structure it this way?
  - *I can't remember who asked this*
- **Answer:** Because it is effective and…
  - Easy to link everything to your website, LinkedIn, Resume, etc.
    - Make a website similar to this
  - Recruiters and employers can easily see the work you have done over the course of your undergraduate career
  - You can copy the website (above)
    - *It is all on Github*

Lecture Questions

- Post your questions about lecture content in the Teams channel and someone will get back to you
- Avoid asking (i.e. Typing) questions in lecture, because:
  - Typing is time consuming and distracting
    - *Instead, focus on the lecture*
  - Hard to convey information over text
    - ‘‘A picture means a thousand words’’
      - *And I can’t type a thousand words in 2 mins, nor can I show code*
  - Breakfast
    - I need to eat

Any Questions
(So Far)

Mills & Moore (1)

- **IMPORTANT:** Stick to using *Mills*
  - Do not use *Moore*, because it does not have all of the software/tools that *Mills* has (i.e. GO Programming Language)
- Credentials
  - Username: *MACID@Mills.McMaster.ca*
  - Password: *Whatever you use for Avenue, Mosaic, etc.*
- RSA Key Authentication:
  - If it is your *first* time connecting to Mills/Moore/Anything, just say (i.e. Type) ‘‘yes’’ when prompted about RSA
    - In a nutshell, this is the secure part of *ssh*, the tool we will be using to connect remotely to Mills

Mills & Moore (2)

- How to connect to Mills?
  - **S**ecure **Sh**ell (**ssh**)
- **S**ecure **S**hell (**ssh**) is a remote login program
  - Used to access systems, remotely
    - *i.e. I can be in Canada, and remotely control a computer in Australia.*
- Secure → Encrypted
  - No one else can eavesdrop on your connection
    - *i.e. Communication is protected; freedom to say whatever you want*
- Shell → Terminal/Command-line
  - This is what you will be working out of
    - Note: Technologically, there is no difference between GUI and CLI

Mills & Moore (3)

- MacOS/Linux/Unix
  - Use the in-built terminal
- Windows
  - Download Putty, MobaXTerm, WSL, etc.
    - *Note: WSL = Windows Subsystem For Linux*
  - *Windows is trash, get a *nix machine*
- How to use ***ssh***?
  - **ssh** MACID@Mills.McMaster.ca
    - Note: MACID is case-sensitive
      - *i.e. "bill" is NOT the same as "BILL"*

Mills & Moore (4)

- How to use ***ssh***?
  - **ssh** macid@mills.mcmaster.ca
    - Note: macid is case-sensitive
      - *i.e. "bill" is NOT the same as "BILL"*
- "I'm typing my password, but it's not working"
  - It is working, but the terminal does not show keyboard input in any manner or form
    - This is a security feature; it prevents people from knowing the length of your password.
      - *It's easier to crack a password if you know how long it is. And if you have some information about the characters, it's exponentially easier.*

Any Questions *(About SSH)*

Linux/Unix Time

- Now we're going to work exclusively out of the terminal
  - I recommend pulling up a terminal and following along
    - You should type the commands yourself and see what happens
      - *Seeing is believing*
- I'm going to work on the left side, and you will work on the right side
  - In other words, the video will be on the left, and your terminal will be on the right.
    - *If you're left-handed, it's time to start being right-handed*

Start The Terminal && Login To Mills

Unix Commands (1)

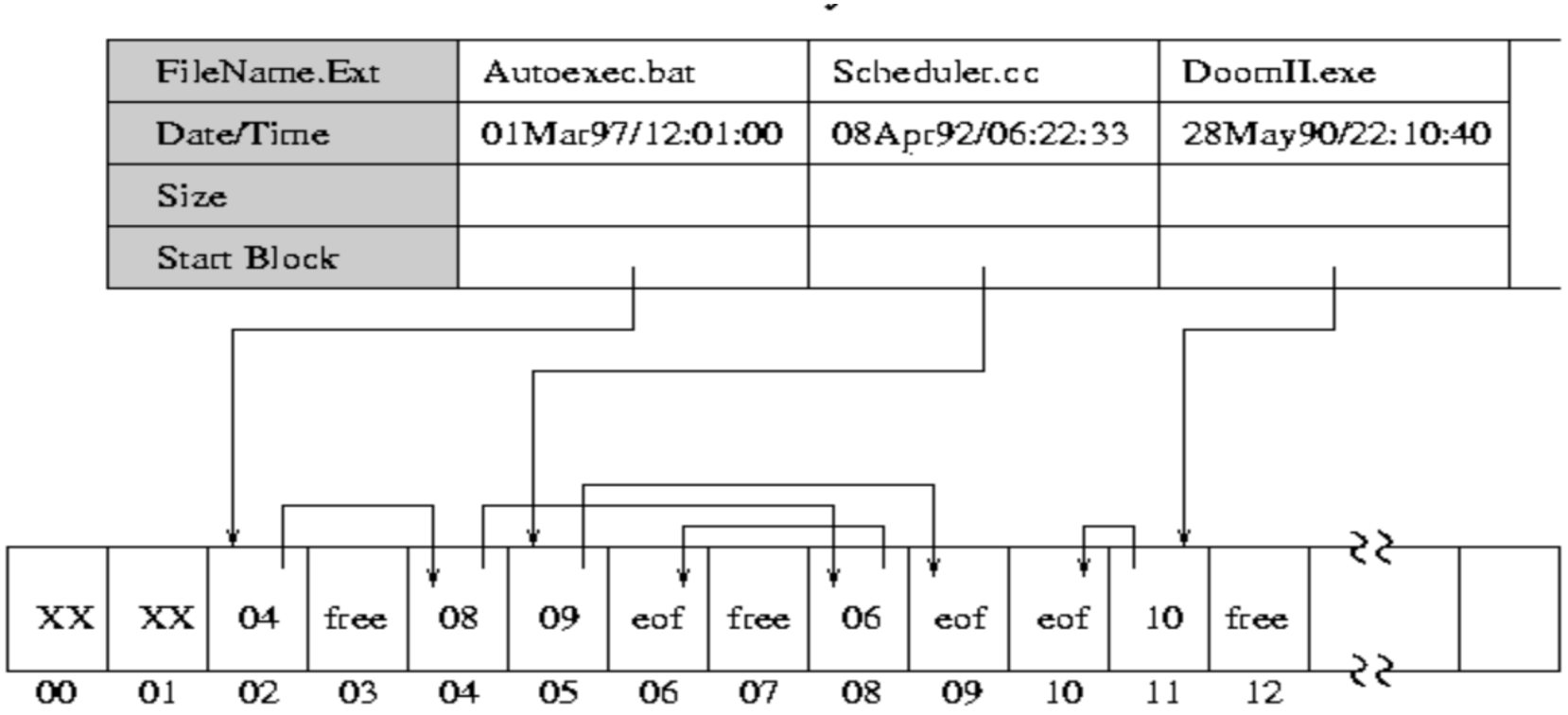
- **ls**
  - Show files and folders in current directory/folder
    - To list files/folders in a different directory, specify it after *ls*
      - *i.e.* ***ls folder***
  - Stands for **l**i**s**t
- **cd**
  - Change to a different directory/folder
    - i.e. **cd directory**
    - i.e. **cd path/of/directory**
  - Stands for **c**hange **d**irectory
- **cp**
  - Copy a file
    - i.e **cp file sameFile**
  - Copy a folder
    - i.e. **cp -r folder sameFolder**
      - **-r** stands for *recursive*
  - Stands for **c**o**p**y
  - **Note:** The name of the new file/folder has to be different
    - BUT, the extension has to be the same – *an image (PNG) cannot become a spreadsheet (XLSX)*

Unix Commands (2)

- **mv**
  - Move a file/folder to a different directory/folder
    - i.e. **mv file path/to/directory**
    - i.e. **mv folder path/to/directory/folder**
  - Stands for **m**o**v**e
  - Interestingly, ***mv*** is also used for renaming
    - Why? Because of how file/directory tables work
      - *Next slide*

Unix Commands (3)

- **mv**
  - Interestingly, ***mv*** is also used for renaming
    - Why? Because of how the File Allocation Table (FAT) works



Unix Commands (4)

- **mkdir**
  - Make a new (empty) folder
    - i.e. **mkdir folder**
  - Make a bunch of new (empty) folders
    - i.e. **mkdir folder1 folder2 folder3**
  - Make new (empty) folders inside each other
    - i.e. **mkdir -p folder/anotherFolder/anotherOne/andAnotherOne**
  - Stands for **m**a**k**e **dir**ectory
- **touch**
  - Can be used to make a new (empty) file
    - i.e. **touch file.txt**
    - i.e. **touch hello.c**
    - i.e. **touch program.java**
  - In reality, touch is used to change file access and modification times
    - *Don't worry about this for now*

Unix Commands (5)

- Commands are case sensitive on Mills
  - **mkdir** is the same as **mkDIR**
  - **touch** is the same as **ToUcH**
- Manual (man) pages
  - Great resource for finding more information about a command (or almost anything else)
    - If you want to know more about the **mkdir** command, then you type: **man mkdir**
    - If you want to know more about the **touch** command, then you type: **man touch**

Unix Commands (6)

- Flags
  - Used to specify alternate options
    - List **all** (hidden) files in a folder/directory

      $ **ls -a**
    - List all files in a **table**-like format

      $ **ls -l**
    - List **all** hidden files in a **table** format

      $ **ls -la**
    - Copy a folder and all of its contents

      $ **cp -r folder sameFolder**
    - Move/Rename a folder
      - $ **mv -r folder sameFolder**
    - Make empty folders inside each other
      - $ **mkdir -p folder1/folder2/folder3**

- Use the **man** pages to figure out what flags are supported by a command, and how to use it.
  - *Note: The **man** pages were not written for undergraduate students. So you will find it cumbersome to use. However, once you get the hang of it, you'll appreciate it more.*
    - *Feel free to Google how to do `X` command, and then supplement it with the **man** pages.*

Unix Commands (7)

- More commands:
  - **locate**
    - Find filenames quickly
      - i.e. **locate nameOfFile**
      - i.e. **locate stdio.h**
      - i.e. **locate jiffies.h**
  - **grep**
    - Looks at the contents of files and returns/prints lines that match a pattern
      - i.e. **grep "pattern" file.txt**
      - i.e. **grep -irn "pattern" .**
        - *Useful for online tests*
    - *Stands for **g**lobal **r**egular **ex**pression*

- More commands:
  - **find**
    - Recursively traverse a directory; good for listing all files in a directory
      - Also good for piping (i.e. Providing input to another command)
    - i.e. **find folder**
    - i.e. **find folder1 folder2**
  - **head / tail**
    - List the first / last `X` number of lines in a file
      - i.e. **head -10 file.txt**
      - i.e. **tail -10 file.txt**

Unix Commands (8)

- Piping commands/input
  - Connects two or more commands, where one command provides input to the other command
    - Expands the functionality of basic commands
  - Examples:
    - Only show the first 10 entries of **ls**

      $ **ls | head -10**
    - Only show the last 10 entries of **find**

      $ **find folder | tail -10**
    - Only **list** files that were made/modified in January

      $ **ls -la | grep -i ‘‘Jan’’**

      # **-i** means to ignore case
    - Search the **find** *man pages* for a ‘‘pattern’’

      $ **man find | grep ‘‘recurs’’**

      # This is pattern matching

Any Questions *(About *nix)*

Transferring Files (1)

- **S**ecure **C**opy
  - **scp** [Host machine → Remote server]
    - Copy a file from host machine *(you)* **to** remote server *(mills)*:
      - *General format:* **scp** file user@remote.server.ca**:~**
        - *i.e.* **scp** assignment1.c chowdhaj@mills.mcmaster.ca**:~**
    - Copy a folder (and all of its contents) from host machine *(you)* **to** remote server *(mills)*:
      - *General format:* **scp** **-r** folder user@remote.server.ca**:~**
        - *i.e.* **scp** **-r** final_project chowdhaj@mills.mcmaster.ca**:~**
  - *Note: These commands need to be executed on your machine, and not the remote server. Regardless of which way files are being sent, type the commands on your machine's terminal, and not the remote server*

Transferring Files (3)

- **S**ecure **C**opy
  - **scp** [Remote server → Host machine]
    - Copy a file from a remote server *(mills)* **to** your machine *(you)*:
      - *General format:* **scp** user@remote.server.ca**:**/path/file.txt /directory/on/host
        - *i.e.* **scp** chowdhaj@mills.mcmaster.ca**:**~/code/assignment1.c ~/Desktop
    - Copy a folder (and all of its contents) from a remote server *(mills)* **to** your machine (you):
      - *General format:* **scp** **-r** user@remote.server.ca**:**~/path/folder /directory/on/host/machine
        - *i.e.* **scp** **-r** chowdhaj@mills.mcmaster.ca**:**~/code/final_project ~/Desktop/2SD3
  - *Note: These commands need to be executed on your machine, and not the remote server. Regardless of which way files are being sent, type the commands on your machine's terminal, and not the remote server*

Transferring Files (3)

- **S**ecure **F**ile **T**ransfer **P**rotocol
  - **sftp**
    - General format: **sftp** user@remote.server.ca
      - *i.e.* ***sftp*** *chowdhaj@mills.mcmaster.ca*
  - 0. Can only be used to transfer files/folders from a remote server to your machine
  - 1. Once the connection has been established, use commands like **ls** and **cd** to list and change directories, respectively
  - 2. Then, use ***get*** to retrieve files/folders:
    - Copy file from remote server to host machine:
      - *i.e.* **get** someFile
    - Copy folder (and its contents) from remote server to host machine:
      - i.e. **get -r** someFolder

Alias

- Constantly typing the entire destination address (or remote server) is cumbersome
  - Typing long commands with flags and arguments is also tedious
- Use **aliases** to shorten or abbreviate long commands
  - *i.e.* alias mills-ssh="ssh chowdhaj@mills.mcmaster.ca"
  - *i.e.* function mills-scp() {scp -r $1 chowdhaj@mills.mcmaster.ca:~ ;}
- Creating aliases is more-or-less the same for every shell
  - However, depending on what shell is being used, a different file needs to be edited
    - *i.e.* On Z-Shell *(zsh)*, look for *.zprofile* in the *home* directory
      - Navigate to the *home* directory via: cd ~
      - Use **ls -la** to list all (hidden) files
  - *You will need to do some research to figure out which shell you are using, and subsequently, which file needs to be modified*
- Note: Don't forget to *source* the file after making changes
  - *i.e.* source .zprofile

Text Editors

- In order to edit (text) files on *Mills*, you will need to use a text editor
  - For example:
    - nano nameOfFile.txt
    - vi/vim fileName.c
    - emacs someFile.java
- Which one to use?
  - nano is very simple and easy to use
  - vi/vim is more complicated has a moderate learning curve
  - emacs is extremely complicated and has a steep learning
- The steeper the learning curve, the more features it has
- *For beginners, stick to nano – it is simple, quick, and easy*

C Programming (1)

- Basic boilerplate for C programs
  - ***main()*** function
    - Return type
      - void
    - Parameters
      - int argc, int * argv[]
      - void
  - The ***main()*** function is very special because it is:
    1. The entry point. In other words, it is the first function that is executed
    2. Quite versatile. The return type and parameters do not have to be strictly followed
- Examples:
  - *boilerplate_minimal.c*
  - *boilerplate_void.c*
  - *boilerplate_parameters.c*

C Programming (2)

- How to compile C programs?
  - Use `***gcc***` to compile C source files
- `***gcc***` stands for **G**NU **C** **C**ompiler
  - What does **G**NU stand for?
    - It's a lame nerd joke about Unix and recursion
- To compile your C source files to executables, type the following into the terminal:
  - **gcc** program.c **-o** program
    - **gcc** → Compiler
    - program.c → Name of the C source file
    - **-o** → Output flag
    - program → Name of the executable

Spawning A Process

- Processes can use `*fork()*` to spawn a child process
  - A child process is nearly identical to its parent process
    - For all intents and purposes, the child is a copy of the parent process – it is like mitosis/meiosis in biology
  - The process that calls `*fork()*` is the parent process, and the newly created process is the child process
  - `fork()` returns a value of 0 to the child process, and to the parent process it returns the PID of the child process
  - Note: `*fork()*` is a system call
- Learn how `*fork()*` works via the man pages
  - Type the following into the terminal:
    - **man** -s2 fork
  - **man** pages are the best resource for *nix machines
- Refer to the following:
  - *fork.c*
    - *How to compile it:* ***gcc fork.c -o fork***

THE END