Basic Linux Commands

- Resources:
  - Linux manpages
    - Absolute must to look it up for each command you study
  - “The Linux Command Line: A Complete Introduction” by William Shotts, 2nd edition
    - Chapters 1-6, relevant parts for the commands you need to study
    - Introduction before the chapters start is also nice to read for motivation
  - “Your Unix/Linux: The Ultimate Guide”, by Sumitabha Das (Chapters 2-4)
  - “A Practical Guide to Linux Commands, Editors, and Shell Programming, “ by Mark Sobell, 3rd edition (Chapters 2-4)

Bit of history first

- The operating system Unix was developed in 1970's by the AT&T Bell Labs.
  - Multi-user multi-tasking OS since near the beginning
- Many variants/implementations, both commercial and open-source
  - FreeBSD, Solaris, AIX, HP-UX, IRIX. MacOS, Linux…
- Linus Torvalds developed Linux in 1991
- Richard Stallman and Linus Torvalds developed the free and open-source GNU-Linux.
- Many different current implementations
  - Ubuntu, Debian, Fedora, Redhat, SUSE, CentOS, Linux Mint, …
- Looks different, but they are all based on the same "kernel" (core functionalities of OS)
- Android is also based on Linux!

Why Linux?

- Very popular and widespread in academia and industry.
  - Around 62% of servers run Linux
  - Around 96% of top webservers run Linux
  - Top 500 supercomputers of the world all run Linux!
  - Very popular with developers
  - Commonplace in many end-user desktops also
- Open source, so easy to get and install
- You will have to deal with Linux quite often
  - In several courses in your BTech
  - In the real world when you go out, especially if you deal with systems

Linux Shell

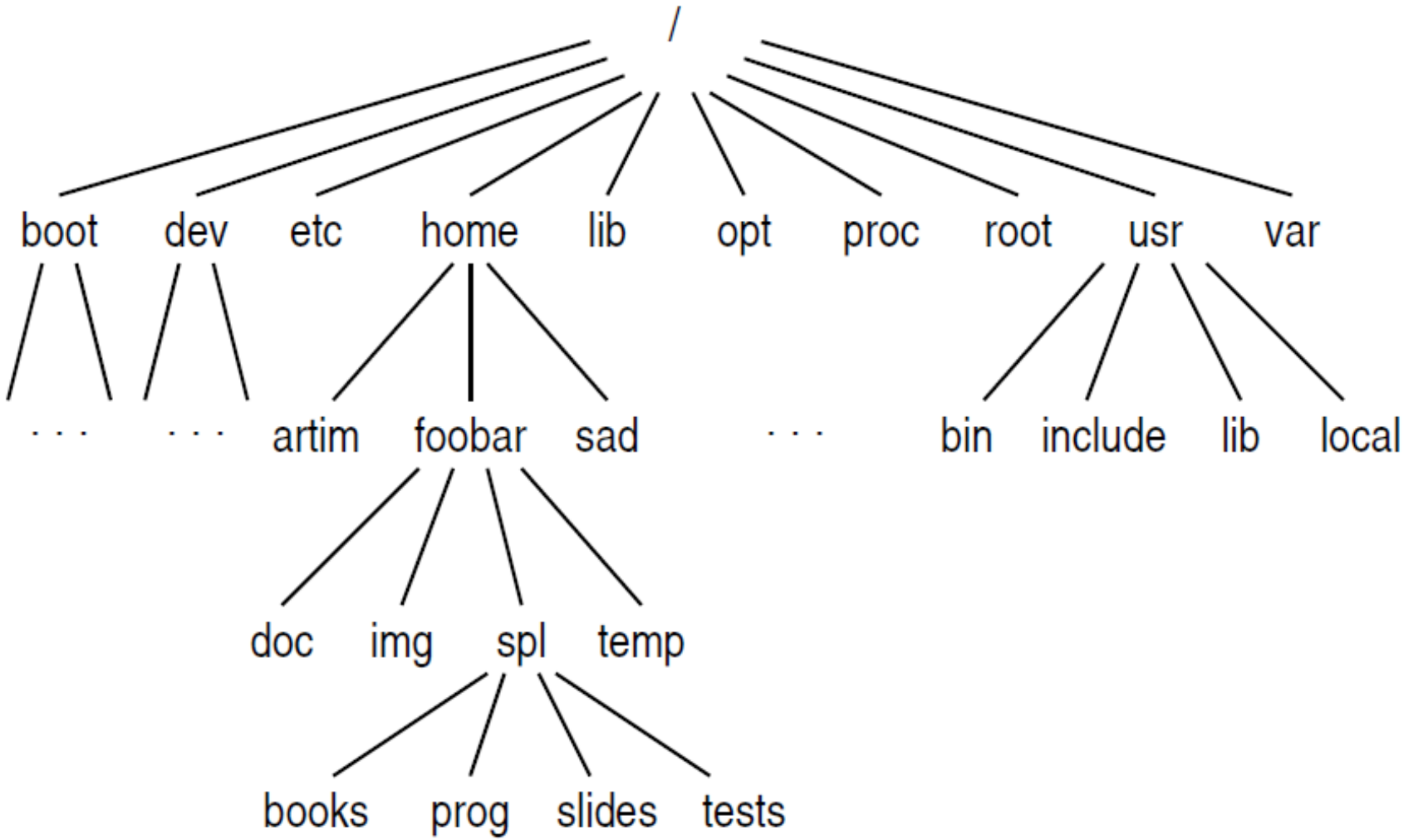
- A command interpreter that waits for user to type in commands from the keyboard for OS to execute, and shows the results back to the user
  - Just another program written on top of the OS
- Different shells exist differing somewhat in their features
  - Bourne shell (sh): original shell developed by Steve Bourne
  - GNU Bourne-again shell (bash): sh upgraded with more features
  - C shell (csh)
  - Korn shell (ksh)
  - …
- We will do only bash shell
  - The $ prompt you see in the terminal windows is the prompt from bash waiting for you to enter a command

Linux commands

- Can have three parts
  - The actual **command**
  - Command **option(s)** (optional)
  - **Argument(s)** (optional for some commands, mandatory for others)
- Some examples:
  - **gcc myfile.c** (command and argument, no option)
  - **gcc -Wall myfile.c** (all three present)
  - Cannot have only **gcc**
  - **ls** (only command, no option or argument)
  - **ls -l** (command and option, no argument)
  - **ls -l myfile.c** (all three present)
  - **ls -lar** (command with three options (**-l**, **-a**, and **-r**) and no arguments)

- Can create your own commands also from the existing ones, will do later
- The executable files you ran from the $ prompt in 1st year are also like commands
- For now, we will study only commands inbuilt in bash
- Even there we will study only a small subset of commands that we may need
  - You will learn more commands as you go on

Linux directory structure



Linux directory structure

- All files and directories are organized in a tree structure with
  - Directory
  - Subdirectory
  - Files
    - Can be of different types (text, binary)
- A directory can contain
  - Subdirectories
    - Subdirectories can contain other subdirectories and files, same as a directory
  - Files
- Every directory contains two subdirectories: . (pointing to itself), . . (pointing to the parent directory in the tree)
- Files/directories with names starting with . are called hidden files/directories.
- When you log in, you enter your home directory (like /home/foobar)

How do you identify a file/directory?

- Absolute names
  - You specify the exact path starting from the root /. Examples:
    - /usr/local/lib/
    - /usr/local/lib/libstaque.so
    - /home/foobar/spl/prog/assignments/A1/src/
    - /home/foobar/spl/prog/assignments/A1/src/Makefile
- Relative names
  - Relative to the current directory. Examples (assume that you are in /home/foobar):
    - spl/prog/assignments/A2/myprog.c
    - ./spl/prog/assignments/
    - ../artim/SPL/tests/T1/questions.pdf
- Relative to the home directory. Examples:
  - ~/spl/prog/assignments/A3/
  - ~/sad/SPL/doc/T1soln.pdf

File/directory permissions

- Three types of users
  - Owner: the user who owns the file (u)
  - Other users of the group the file belongs to (g)
  - Other users (o)
- Three types of permission
  - Read permission (r)
  - Write permission (w)
  - Execute permission (x)
- Permissions are specified for each type. For example, you can give
  - Read, write, and execute permission for owner (so owner can both read, write, and execute)
  - Only read and write permission for group (so any user belonging to the group can read and write but not execute)
  - Only read permission for all (anyone can read, but cannot write or execute)

- Straightforward meaning for files
- For directories, the permissions mean:
  - Read permission: You can read the contents of the directory (for example, by ls command). With only read permission, you cannot access the files in the directory.
  - Write permission: You can create new files in the directory
  - Execute permission: You can go to the directory, and open and/or execute files in the directory (provided you know the names). With only execute permission, you cannot see the directory content

Commands and other things you must know

- Commands
  - File/directory creation/organization
    - cd, pwd, mkdir (including -p option), rmdir, mv, cp (including -r and -f option), rm (including -i, -r and –d option)
  - Listing contents of files/directories
    - ls command with the following options
      - -l : should know the meaning of each column
      - -a, -R, -t, -r, -d
    - cat, head (including -c and -n options), tail (including -c and -n options), more
  - Changing permissions: chmod
  - Other commands
    - wc (including -c, -m, -l, and -w options), diff (with -y option)
- Use of wildcards (*, ?)
- Use of redirection (<, >, and |)

Using manpage

- Gives detailed information about any command
  - Absolute must to know
- Basic usage: "man <command name>"
  - Example: man ls
- man is also a command, it can take options (a number). Default (if nothing specified) is 1. As an analogy
  - Using man is like looking up a manual with multiple "sections:
  - The commands are divided into different "sections" depending on their type
  - The numbers indicate this "section number"
  - Sometimes the same name can indicate two different things in two sections, you need to specify then which one you want. For example
    - "man printf" gives information about the printf command
    - "man 3 printf" gives information about the printf() function that you can call
  - Initially, you will mostly need simple man command only with no options. Later you may need the '3' option

- List of Sections
  - Just for information, no need to remember

Section	Contains
1	User Commands
2	System Calls
3	C Library Functions
4	Devices and Special Files
5	File Formats and Conventions
6	Games
7	Miscellaneous
8	System Administration

File/directory creation/organization commands

- cd <dirname>
  - Changes current working directory to directory named <dirname>
  - Name can be absolute or relative
- pwd
  - Shows the current working directory
- mkdir <dirname>
  - Creates a directory named <dirname> (name can be absolute or relative)
  - You should have write permission in the directory where this new directory is created
  - What does the -p option do?
- rmdir <dirname>
  - Removes the directory named <dirname> provided it is empty
  - You should have write permission in the parent of <dirname>
  - What does the -i option do?

File/directory creation/organization commands

- cp <file1> <file2>
  - Copies <file1> to <file2>
  - What will the -r and -f option do?
  - What if you want to copy multiple files to a directory in one command?
- mv <file1> <file2>
  - Moves (renames) <file1> to <file2>
  - mv <file> <dir> moves <file> to directory <dir> (with same name)
- rm <file1> <file2> ….
  - Deletes the files <file1> <file2> ….
  - What do -i, -r and -d options do?

Commands for Listing Files

- The basic command is ls. The listing is sorted with respect to the content names.
- Some options
  - –l Long listing
  - –a Show the hidden files also
  - –R Recursively list the subdirectories, the subsubdirectories, and so on
  - –t The sorting is with respect to last modification times (newest first)
  - –r Reverse the sorting order
  - –d Do not expand the directory contents
- Example: ls -lart shows a long listing of all files (including the hidden ones) sorted in the reverse order of modification times (oldest first)
- You may supply one or more directory or file names after the options in order to see the the listing of that/those file(s) or director(ies).
  - Example: ls -lR / makes a long listing of the entire directory tree (excluding the hidden files)

Explanation of the columns in ls –l output

```
drwxr-xr-x  2  agupta  faculty  4096  Jul 18 19:46  AdvancedOS
-rw-r--r--  1  agupta  faculty    47  Jul 18 19:47  list.txt
-rw-r--r--  1  agupta  faculty    38  Jul 18 19:47  readme.txt
drwxr-xr-x  2  agupta  faculty  4096  Jul 18 19:46  SysProgLab
```

- This is the ls -l output of a directory that has 2 subdirectories (AdvancedOS and SysProgLab), and 2 files (list.txt and readme.txt)

Consider the line -rw-r--r-- 1 agupta faculty 47 Jul 18 19:47 list.txt

-	The first character indicates the type of file A dash (as in here) means a regular file, a 'd' indicates it is a directory
rw-r--r--	Access rights to the file. The first three characters are the access rights for the file's owner, the next three are for members of the file's group, and the final three are for all
1	File's number of hard links (ignore for now)
agupta	The username of the file's owner
faculty	The name of the group that the user agupta belongs to
47	Size of the file in bytes
Jul 18 19:47	Date and time of the file's last modification
list.txt	Name of the file

- cat <file1> <file2> ….
  - Prints (concatenates) the contents of <file1>, <file2>, … in that order
- head <textfile>
  - Prints the first few lines of <textfile>
  - What do -c and -n options do?
- tail <textfile>
  - Prints the last few lines of <textfile>
  - What do -c and -n options do?
- more <file1>
  - Gives a page-by-page display of a file
  - Useful for viewing long files

Command for file permission change

- Only the owner (and the root user) can change the permission of a file/directory.
- The command for that is chmod
- Symbolic change: Add (+) or remove (–) a permission (r, w, x) for user (u), group (g), others (o), or all (a).
  - chmod g+x /home/sad/spl/prog/libstaque/a.out
  - chmod o-rwx /home/sad/spl/prog/libstaque/static
  - chmod a+w /home/sad/spl/prog/libstaque/shared
- Numeric change: Set the permission bits as a three-digit octal number.
  - chmod 755 /home/sad/spl/prog/libstaque/a.out
  - chmod 666 /home/sad/spl/prog/libstaque/Makefile.txt
  - chmod 700 ~sad/spl/prog/libstaque/shared ~sad/spl/prog/libstaque/static

Other commands

- diff <file1> <file2> …
  - Compares the files line-by-line and lists their differences
  - What does the -y option do?
- wc <file1> <file2> ….
  - Shows the count of lines, words, and characters (in that order) for each file
  - How do you print only one of them? Check the options.

Use of wildcards (*, ?)

- Allows replacement by “any” character
  - ‘*’ allows replacement by any sequence of characters
  - ‘?’ allows replacement by any ONE character
- Example use:
  - ls -l *.pdf
    - Will list all (and only) files that have a .pdf at the end of the name, irrespective of what is before the .
    - a.pdf, b.pdf, xy.pdf, pqr.pdf, ….
  - ls -l ?.pdf
    - Will list only files that have a .pdf at the end of the name, and only ONE character before the .
    - So will show the file a.pdf, b.pdf, but not the file xy.pdf or pqr.pdf

Redirection (<, >, >>, |)

- Recall basic C programming
- Suppose you wrote a program a.out that requires a 10x10 integer matrix as input, which you will read with proper scanf() calls inside your program
- Every time you run it, you have to type in 100 integers from the keyboard!!
  - Not a good way to work
- Redirections help in specifying where your input is going to come from or where your output is going to go
- In the above case, you will just type the matrix once in a text file (say input.txt), and redirect the program's input (using <) to read from the file instead of the keyboard!
  - $ ./a.out < input.txt

Redirection (<, >, >>, |)

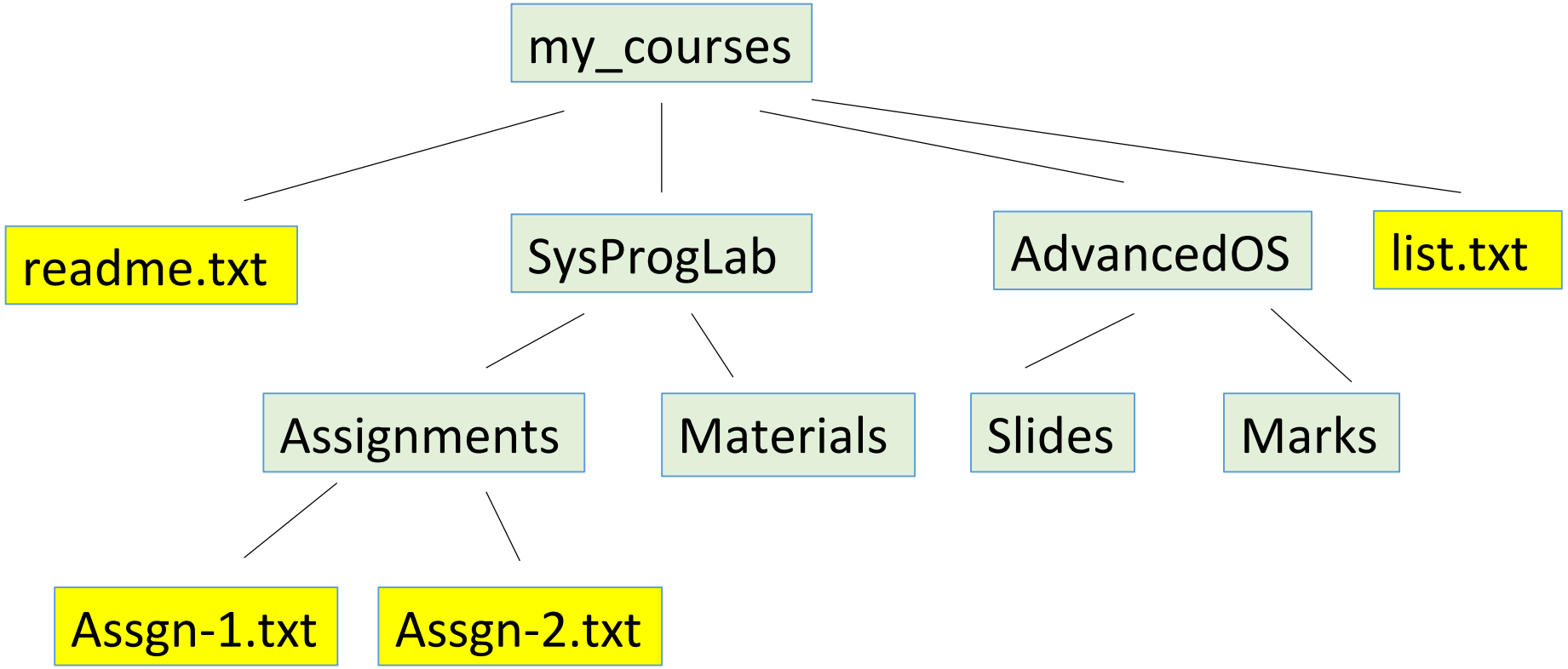
- Allows redirecting a command's input/output from/to a file/command
- Input redirection (<)
  - The command will read whatever it is supposed to read from the keyboard from the file specified
  - Example: see previous slide
  - Very useful for running commands/executable files with large inputs that is difficult to type from keyboard every time

- Output redirection (>)
  - The command will write whatever it is supposed to write to the display to the file specified. If the file exists and has some content, it will be overwritten. If the file does not exist, it will be created.
    - Example: "ls -l > contents.txt" will write the output of ls -l to the file contents.txt, nothing will be printed on the display
  - Using >> instead of > will append to a file if it exists instead of overwriting it
  - Very useful for storing outputs of commands/executable files for further processing
- Redirecting output of one command to input of another (|)
  - Example: "ls -l | wc" will print the number of characters, words, and lines in the output generated by the command ls -l

- < and > are actually redirecting from/to something more generic called STDIN and STDOUT, not important for you now.

Practice in lab today

- Find the absolute name of your home directory using the pwd command
- Create the following directory tree under your home directory using the appropriate commands and verify that it is created correctly. Green boxes are directories, yellow boxes are files. Remember that Linux is case sensitive. You can create the files using any text editor. Put few lines (at least 7-8) of arbitrary contents in the files.



- List the contents of each directory and subdirectory separately using ls -l and understand the meanings of the lines printed
- List the contents of all subdirectories under (and including) the subdirectory my_courses using a single command.
- Remove all subdirectories and files under (and including) the subdirectory my_courses/SysProgLab using a single command. Verify that it is removed using ls command.
- Remove all remaining files and directories. Verify that all files are removed using ls command
- Recreate the entire directory tree again as before

- Copy the file readme.txt to the subdirectory Materials with a new name readme-copy.txt. Use ".." to specify name. Verify that the copy is done.
- Use cat command to print the contents of readme.txt
- Use head command to print the first 3 lines only of readme.txt
- Use tail command to print the last 3 lines of readme.txt
- Use wc command to see the number of characters, words, and lines in readme-copy.txt
- Change (add/modify a few words) a few lines in readme-copy.txt
- Use wc again on it to see the changes
- Use diff command to see the difference between readme.txt and readme-copy.txt
- Concatenate the contents of Assign-1.txt and Assgn-2.txt (in that order) and store it in a new file called Assgns.txt in the subdirectory Materials (use redirection)
- Print only the no. of lines in the ls -l output of the directory my_courses (use with wc and redirection)
- Print only the 3rd line in the file readme.txt (use head, tail, and redirection)

- Copy the entire subdirectory SysProgLab with all its contents to a new subdirectory called SysProgLab-Copy under my_courses using a singe command
- Create the subdirectory my_courses/additional/AGT using a single command from the my_courses directory
- Look at all commands and options you are supposed to know. Exercise any option you have not tried yet.
- Since all these directories/files are created by you, you are the owner of them all, so have all permissions on them (verify from ls -l output). So you should have no permission related problem so far.
- We will now try changing some permissions using chmod.

- Change the permission for user on the SysProgLab directory as below using chmod. For each case, try listing the content under it using ls and creating a new subdirectory under it using mkdir and see what happens. Explain why it happens.
  - Only Read and Execute
  - Only Read and Write
  - Only Read
  - Only Execute
  - Only Write
- Revert the permissions to rwx for user

- Run the command "ls -l /user/include"
  - Check the owner of a random file in it
  - Check the permission given to owner (u) and all (o) for the file and understand them
  - Open the file in a text editor. Can you see its contents?
  - Try changing something in the file and save it in the text editor. Can you do it? Explain what you see
- Write a small C program to print "Hello World" in your home directory. Compile it to an executable file named a.out
  - What permissions do the file a.out has?
  - Run a.out. What happens?
  - Now change the owner's permission to "only read and write". Now try running it again. What do you see now?