Logging into the system.

Log on turing.cs or hopper.cs. Using any of your favorite ssh programs. One recommendation is putty.

Some useful commands:

"pwd" to tell where you are. When you login, you are at your home directory: for example: /home/turing/z074989

How to change password? Command "passwd" will change the NIS+ password, you will be asked to type the password twice for verification.

```
% passwd
```

ps: get a list of processes (running programs) that are currently executing. More useful command: is

```
% ps –ef   -- To get a complete list of all uses in foreground and  background.
%ps –ef   | grep z??????   -- To get those from you.
```

quota -v:

get your quota limit and usage.

ls: list files in a given directory.

```
%ls –a   all hidden files too.
%ls –l  longer list with date, time, permissions(useful for security, more later.)
```

mkdir: create a new directory

cd: change directory

rm -- delete file. Using it carefully, since Linux does not have anything like undelete! Commonly:

```
% rm  -i    -- ask you to confirm before it is deleted.
```

rmdir -- delete empty directory

mv -- "move" files and directories. It is useful if you want to rename a file or directory. There is no "rename" command.

```
% mv prog1.c program1.c  -- ren file
% mv program1.c 340     -- move into directory
% mv 340 csci340    -- ren directory name
```

cp -- copy file

less/more - list a file on standard-put while pausing at each screenful of text.

- Enter key for one line.
- Space for next page.
- Q/q to exit

  % more prog1.c

chmod -- important!

% ls –l

total ..

-rw-r—r-- z?????? csci 154 Jan 6 14:55 assin1.cc

The first ten characters are the access modes of the file/dir. It is divided into 4 sections

Position	
1	whether a dir
4.	permission of owner, for reading/writing/executing
5-7	permission for group users.
10.	permission for others

For every type of users, you can have 8 possible permission (0-7)

000 001 010 011 100 101 110 111

So rw-r—r-- is 110 100 100 -> 644, the owner can read and writer, others can only read.

“chmod 644 *” will give every file in current dir this permission.

In our class, you do:

chmod 700 on your home directory.

```
% cd
% cd ..
% chmod 700 /home/turing/z012345
% cd     --- go back to home directory
```

Linux shell provides many more commands (e.g. grep, wc); It also provides powerful I/O redirection and piping. Study them on your own.

How to find out the usages of other commands: online reference manual:

```
%man  ls
```

Makefile

It is a file that can be used to maintain and build programming projects. A makefile consists largely of rules, which specify how various intermediate files in the projects are constructed. Rules are separated by blank lines. Makefile rules consist of three parts --- a target, a dependency list and a command which are laid out in the following manner:

```
target:  dependency list
   command
```

Example makefile:

```
# Some variables needed to access private libraries for this project
HOME_DIR = /home/z123456
INCLUDES = -I$(HOME_DIR)/include
LIB_DIR = $(HOME_DIR)/lib/

# Standard compiler variables
CC = g++
CCFLAGS = -Wall -g

# Rules start here

pxform: pxform.o similar.o translation.o perspective.o incremental.o panoramic.o
        $(CC) -o pxform $(CCFLAGS) pxform.o panoramic.o incremental.o perspective.o similar.o \
                translation.o \
                -L$(LIB_DIR) -lrender -lmatrix -lm

incremental.o: incremental.cc incremental.h pxform.h
        $(CC) -c $(CCFLAGS) $(INCLUDES) incremental.cc

panoramic.o: panoramic.cc panoramic.h pxform.h
        $(CC) -c $(CCFLAGS) $(INCLUDES) panoramic.cc
```

```
perspective.o: perspective.cc perspective.h pxform.h
	$(CC) -c $(CCFLAGS) $(INCLUDES) perspective.cc

pxform.o: pxform.cc panoramic.h incremental.h perspective.h similar.h \
		translation.h pxform.h
	$(CC) -c $(CCFLAGS) $(INCLUDES) pxform.cc

similar.o: similar.cc similar.h pxform.h
	$(CC) -c $(CCFLAGS) $(INCLUDES) similar.cc

translation.o: translation.cc translation.h pxform.h
	$(CC) -c $(CCFLAGS) $(INCLUDES) translation.cc

clean:
	-rm *.o pxform
```

Once written, to use it is easy:

% make -- maintain/build the final target.

If your file is not named makefile or Makefile, you can use options of the make command

% make –f makefilename

% make target -- only rules for that target are executed.

Editing, compiling and running and debugging a program.

Editor: up to you. You can use pico or vi on Unix/Linux server. You can also use an local editor to edit your source, and sftp it over to the server.

Compiling: g++. Your assignments will be compiled and run by TA using g++. It does NOT count if they run on visual C++, Borland C++ etc.

%g++ test1.cc → create a.out as the executable file.

%g++ -o test1 test1.cc -> create test1 as the executable file.

This is essentially two step: compile and link.

%g++ -c test1.cc → create test1.o

%g++ -Wall test.cc -> turn on all warnings. Useful for compiling.

%g++ -g test1.cc → required for using debugger such as gdb.

%g++ -v → print version

For 480 assignments, you will need a makefile.

Other

Secure ftp tool: FileZilla

Further Reading:

https://linuxcommand.org/index.php