1 Transferring your code to eniac

1. To upload your programs on one of the cslab machines you can use **sftp** in order to transfer your code to eniac. First, open a terminal window and navigate to the local directory where the files you want to upload are. For example, if your *.cpp* files are in the Downloads folder of your machine, do

   `cd ~/Downloads`

2. To **sftp** to the server type:

   `sftp your_username@eniac.cs.hunter.cuny.edu`

3. You can create or navigate directories here to organize your files (see review of shell commands below). Upload your files by typing:

   `put` *file*

   Or, to upload multiple files at the same time:

   `mput` *file1 file2 ...*

4. When you've uploaded all your files type:

   `exit`

   This will bring you back to your local machine.

2 Using SSH

1. Type:

   `ssh your_username@eniac.cs.hunter.cuny.edu`

   Usernames are case sensitive.

2. **Enter your password** (note that when entering a password, no characters appear on the screen). Passwords are case sensitive.

3. Now you're at a gateway machine called eniac. Don't do any processing on eniac – just ssh through eniac to one of the lab machines:

   `ssh your_username@cslabX.cs.hunter.cuny.edu` *#where X is a number 1 through 29.*

   You can pick any machine. If the machine is down, try another. For example, to log in to the 2nd machine type:

   `ssh your_username@cslab2.cs.hunter.cuny.edu`

4. All cslabX machines and eniac see the same directories for your account. That means that you see the same files on all machines.

5. Now that you are logged into a Linux machine in the lab, you can remotely compile and run your program with g++ (see next section).

3 Compiling your code with g++

Separate compilation:

We're now often working with multiple source files that must be compiled into a single executable.

Let's say your programming project consists of the following files:

```
ClassA.hpp ClassA.cpp ClassB.hpp ClassB.cpp program1.cpp main.cpp
```

You compile only the .cpp files.
To compile your program with g++, in a terminal window, navigate to where the files are, and type:

```
g++ -o myprogram ClassA.cpp ClassB.cpp program1.cpp main.cpp
```

This will produce an executable file named `myprogram`. To run the compiled program, type in the terminal window:

```
./myprogram
```

Alternatively, if you compile the program without giving the output file name (leaving out the command option -o myprogram), the executable file will be called a.out, which you can execute the same way:

```
./a.out
```

4 Review of shell commands

You need to know just a few commands to work comfortably in a Unix terminal:

```
ls, cd, pwd, mkdir, cp, mv, rm
```

A brief summary:

Command	**Use**
pwd	print the current working directory
ls	list files in the current directory
ls path/to/directory	list files in the directory
cd path/to/directory	change working directory
mkdir newdirectoryname	create a new directory
cp file1 file2	make a copy of file1 called file2
mv file1 file2	rename (move) file1 as file2
rmdir directoryname	remove an empty directory
rm file	remove file
chmod <options> file	change file permissions (read +r, write +w, execute +x)
man command	documentation about the command

These are some useful directory shortcuts:

```
.  #the current directory
..  #the parent directory (relative to the current directory)
~ #the home directory
```

For example:

```
cd ..  #go to the parent directory (one level up)
```

5 References

Unix Tutorial: https://info-ee.surrey.ac.uk/Teaching/Unix/unix1.html

Become a Command Line Ninja:
https://lifehacker.com/5743814/become-a-command-line-ninja-with-these-time-saving-shortcuts