CS2040: Data Structures and Algorithm
General Java Program Compilation Guide
(Part 1 of Lab 0 Walkthrough)

Please note that:

Information for Lab 0:

- It is a purely take home lab, i.e. **no actual lab sessions will be conducted**.
  - You need to pass all the test cases of the 3 exercises to get the 1%
  - No deadline is set for lab0

Overview

Welcome to CS2040 Lab 0! This document serves as a self-guided explanation to doing Lab 0. It will guide you on the following topics:

- Connecting to the SoC Unix server
- Java program editing
- Java program compilation

Java Program Compilation on SoC Unix Server

This section gives you an introduction to the Unix machine as well as the basic Java programming environment available on this machine. The steps given will also help you to become familiar with the process of editing, compiling, and running simple Java programs.

The major steps

To compile and run a Java program on the Unix machine, we need to go through the following steps:

1. login to the **Unix** machine
2. create and modify programs using the Unix machine
3. compile and execute Java programs

Please check that you have the following before proceed:

1. A SOC unix account. You can create/re-enable your account by going to:

   `https://mysoc.nus.edu.sg/~newacct`

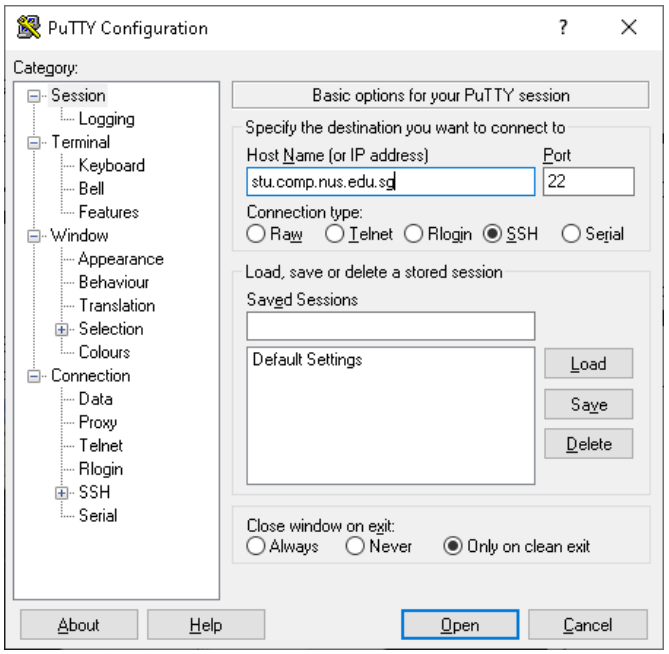
   Use your NUSNET id and password to log in.

2. An SSH Client. If you are using Windows, one option for this is the PuTTY software. You can find a copy of the software in CS2040 Luminus File under the "Lab Material" folder. Please download and install it.

Those of you working on Macs or Linux variants can use the "ssh" and "scp" commands in your "Terminal.app" window. The instructions below refer primarily to the Windows environment.

Section 1: Logging In

Left mouse click on the **Start** button at the bottom-left corner, select **All Programs**, look for **PuTTY** folder, and left click on **PuTTY**. A screen similar to the one shown below will appear.

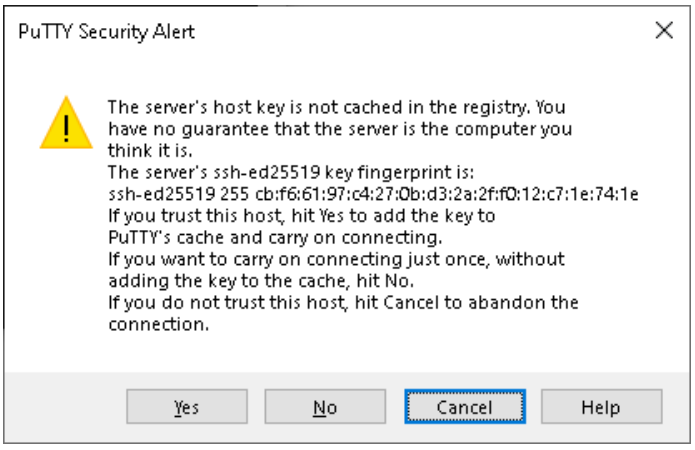


Under Host Name, enter **`stu.comp.nus.edu.sg`.** Please ensure that you have already created your Unix Account as shown above. Your Unix Account is not the same as your NUSNET Account. If connecting to **`stu.comp.nus.edu.sg`** still does not work, you need to install and connect to SoC first using SoC VPN (`refer to` https://dochub.comp.nus.edu.sg/cf/guides/network/vpn, You need your NUSNET ID to login to the page). Alternatively, you can also use the NUS VPN (refer to https://nusit.nus.edu.sg/eguides/ (look for documents referencing "nVPN"))

Ensure that the Port Number is 22 and click on the **Open** button.

After that, click on "Open", where you will be prompted to enter your account.

If you encounter the following window, just click "Yes".



When this is done, another window will appear to prompt you for your username. Enter your **SoC User ID** here. UNIX is case sensitive, so you must type "e01" and not "E01". After hitting "Enter", you will be prompted for your **password**. Note that the user ID and password are **case-sensitive** (i.e. an "a" is different from an "A").

The password will appear on the screen as asterisks (***) so you need to type slowly and carefully.

Once you have logged into the Unix server, you are automatically placed in your home directory. A directory is similar to a folder in Windows.

Notes before you proceed

- Do not click on the Close button (the button with a cross) at the top-right corner of the window. The correct way to log out of your UNIX account is explained in Section 8.
- The UNIX system is not GUI-based so the mouse is seldom used (if at all).
- You should see the UNIX prompt (read Section 5). If you don't, check that you have not scrolled off the screen with the mouse. Bring the prompt back into view by scrolling back.
- If you do see the UNIX prompt but whatever you type does not appear, you might have accidentally locked your screen (locking can be done by pressing Control-s or by pressing the "Scroll Lock" key). Unlock your screen by pressing Control-q (if you have pressed Control-s earlier), or by pressing "Scroll Lock" again (if you have pressed "Scroll Lock" earlier).
- If you get stuck after typing a command or in a middle of an application, sometimes typing 'q' (quit) may get you out.
- If 'q' does not work, press Control-c which is the interrupt key. Use this as the last resort. If it still doesn't work, seek help.
- Do not press Control-z, which is the interrupt key in DOS. Pressing Control-z in UNIX sends your job to the background, and if you have accidentally pressed Control-z, type 'fg' (foreground) to resume.

- Tip 1: You may use the up and down arrows to select a previously entered command. This saves you some typing time.
- Tip 2: You may use the tab key to fill out the complete path/filename. Ask your lab TA to show you how this can be done.

Section 2: User IDs and Passwords

When you have logged in successfully for the first time, you will be immediately asked if you would like to change your password. Enter 'n' (no) to skip it. Do **NOT** change your password as you are still new to the system and if you make any mistake, time may be wasted as you need to get it rectified at the Help Desk (located at level 1 of COM1 building).

You may want to change your password some other time, after you have thought of a good password. Choose a password that is **6 to 8** characters long.

There are guidelines to choosing good passwords which are safer and less susceptible to break-ins. When you change your password, the system will check that your new password satisfies its stringent requirements, failing which it will reject the change. Generally, the system tends to accept passwords that consist of at least a mix of uppercase and lowercase characters, plus some digits. It is advisable that you prepare **a few passwords** beforehand, so that if one fails to pass the test, you can pick another without wasting too much time. You can

change your password any time you like – in fact, you are advised to change it regularly. The command to change password is **`passwd`**.

Always **remember your password**! If you forget, you would need to approach the personnel at the Help Desk (COM1 building, Basement) to submit a request for a new password. Alternatively, you can go to https://mysoc.nus.edu.sg/~myacct/resetpass.cgi to change your password online (this requires your NUSNET ID and password).

Section 3: Setting up your account

If you are new to the Unix server, you are encouraged to perform the following setup program. Although the program is intended for CS1101C students, it contains a number of nice configurations which are general useful. Note that you can skip this step if you have performed this before.

Run the **setup** program as follows (you should do this only **ONCE**). Type the following commands in your SSH window:

```
~cs1101c/lab0/setup
```

Check that you enter the above command ACCURATELY (the tilde character ~ is usually found on the top left of your keyboard next to the "`1`" key).

After the setup program runs successfully, you must type:

```
source .bash_profile
```

Remember to type the SPACE in between "source" and ".bash_profile". "Respect the spaces". Note that if the above command ran successfully, you will see no output. With many UNIX commands, "*no news is good news*". This is very different from Windows or Mac where you are used to seeing a dialog box pop out. Try to get used to UNIX quickly.

Section 4: Getting Online Help

An online help facility is available in the system via the "`man`" command ("`man`" stands for **MAN**ual). Type "`man man`" (i.e. get the manual page on the command "man") to find out more about the facility. To look for more information about any UNIX command, for example, "`ls`", type "`man ls`". To exit "`man`", press "`q`".

Section 5: UNIX Crash Course

The following are a few commonly-used commands. This list is by no means exhaustive and you are urged to explore on your own. Note that UNIX commands are case-sensitive.

In the examples below, bold words are commands which you are expected to enter. Assume that you are `e0101234`, your default prompt may look like this (possibly a little different):

```
e0101234@stu1:~[xxx]$
```

`stu1` is the internal name for the server; `~` indicates that you are currently in your home directory.

a. Directory commands

pwd to **P**rint current **W**orking **D**irectory to show you which directory you are currently in

```
e0101234@stu1:~[xxx]$ pwd
/home/nusstu/e0101234
```

ls to **L**i**S**t files in your current directory

```
e0e01234@stu1:~[xxx]$ ls
java     doc
```

You may also use "`ls -F`" for more information (`-F` is one of the many options/flags available for the `ls` command. To see a complete list of the options, refer to the man pages, ie. "`man ls`".)

```
e0101234@stu1:~[xxx]$ ls -F
java/       doc/
```

The slash (/) beside the filename tells you that the file is a directory (folder). A normal file does not have a slash (/) beside its name when "`ls -F`" is used.

Note that the directories c/ and doc/ are created by the setup program in section 3. If you did not run the setup program, you will not see any file at all.

You may also use the "`ls -l`" command (dash L) to display almost all the file information, include the size of the file and the date of modification. Try it now!

cd to **C**hange **D**irectory from current directory to another

```
e0101234@stu1:~[xxx]$ cd c
e0101234@stu1:~/c[xxx]$ ls -F
ch1_1.c      ch2_1.c      ch2_2.c      ch2_3.c      ch2_4.c
```

Note that the prompt changes to ~/c to indicate that you are now in the `c` directory below your HOME directory.

Entering "`cd`" alone brings you back to your HOME directory, ie. the directory in which you started with when you first logged into the system.

```
e0101234@stu1:~/c[xxx]$ cd
e0101234@stu1:~[xxx]$
```

mkdir to **M**a**K**e a sub**DIR**ectory in current directory

```
e0101234@stu1:~[xxx]$ mkdir another
e0101234@stu1:~[xxx]$ ls -F
another/   java/     doc/
```

rmdir to **R**e**M**ove a sub**DIR**ectory in current directory -- note that a directory must be empty before it can be removed.

```
e0101234@stu1:~[xxx]$ rmdir another
e0101234@stu1:~[xxx]$ ls -F
java/     doc/
```

b. File commands

cp to **C**o**P**y files

```
e0101234@stu1:~[xxx]$ cd doc
e0101234@stu1:~/doc[xxx]$ cp abridged.txt anotherfile
e0101234@stu1:~/doc[xxx]$ ls
abridged.txt   anotherfile    faq.txt        tutor
```

mv to **M**o**V**e files from one directory to another; can also be used to rename files.

```
e0101234@stu1:~/doc[xxx]$ mv anotherfile afile
e0101234@stu1:~/doc[xxx]$ ls
abridged.txt   afile          faq.txt        tutor
```

rm to **R**e**M**ove files. Be **careful** with this command -- files deleted cannot be restored

(unless they have been backed up during the normal backup cycle).

```
e0101234@stu1:~/doc[xxx]$ rm afile
rm: remove `afile'? y
e0101234@stu1:~/doc[xxx]$ ls
abridged.txt   faq.txt        tutor
```

c. Command to display text files

ca to string together or display (**CAT**enate) the contents of files onto the screen

```
e0101234@stu1:~/doc[xxx]$ cat abridged.txt
```

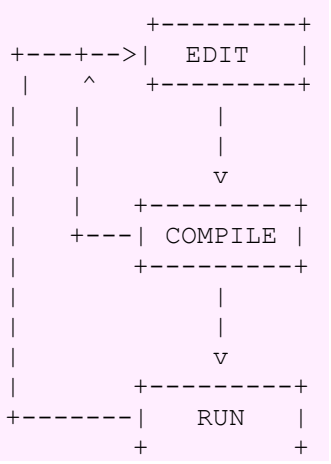
less variant of "`cat`" (includes features to read each page leisurely)

```
e0101234@stu1:~/doc[xxx]$ less -e abridged.txt
```

In "`less`", use <space> to move down one page, 'b' to move **B**ack up one page, and 'q' to **Q**uit from "`less`". You can also use the up/down arrow keys to move one line at a time.

Section 6: Running a Java Program

The process of creating a working Java program involves the following steps:



1. Use an editor of your choice, type in the source code (filename must have extension `.java`, eg: `Helloworld.java`)
2. Compile your program to obtain the executable file. If there are compilation errors, identify them and re-edit the source code before you proceed.
3. Run your executable file by typing the name of the executable file.
   If there are run-time errors, you must identify them and re-edit the source code, and compile again.

Section 7: Creating your own Java Programs

Create your first Java program "HelloWorld.java"

There are a number of editors available in sunfire: *vim, vi, emacs, joe, pico, nano* etc. The more powerful one requires a longer time to learn. For this lab, you will use `vim` -- a powerful editor with many commands, but even with the knowledge of a few simple commands it is quite easy to use and very powerful. It is YOUR responsibility to pick an editor and master it, and in future labs we will assume that you are familiar with your editor and its various functions.

It is recommended that you create a directory to store all your Java programs. Placing all programs under your home directory can get messy real quick. Follow the following steps to create a new directory:

1. Enter "**`cd`**" (this will get you back to the home directory)
2. Enter "**`mkdir java`**"(creates a new directory java/ under home directory)
3. Enter "**`ls -F`**"
   You should see the new directory **`java/`** along with other files and directories.

Go to the **java/** directory. Enter "vim HelloWorld.java", then press the "i" key. You will see that the words "-- INSERT --" appear on the bottom left corner of your screen. You are now in INSERT mode. While you are in INSERT mode, you may use the arrow keys (Up, Down, Left, Right) to move around your program, as well as the Backspace key and Delete key to delete text. This is contrary to what I mentioned during the briefing session. You can use the arrow keys in Insert mode after you set up your .vimrc profile. The PageUp and PageDown keys do not work, so do not use them. Do not use the scroll bar as it does not always work as expected.

Notice that line numbers (1, 2, 3, etc.) are displayed on the left. This helps you to easily identify a number by its line. Line numbers are NOT part of the actual Java code that you write, but are provided by *vim* to assist you in coding. This is extremely useful when identifying the location of compilation errors.

To copy text, you may use the mouse to highlight blocks of text, then right-click on the mouse and choose Copy. To paste your text, you must use the cursor keys to move the cursor to the desired position, then right-click on the mouse and choose Paste.

Type in the following program:

```
import java.io.*;

public class HelloWorld {

   public static void main(String [] args) {

       System.out.println("Hello World!");

   }
}
```

Note that for simplicity, the above program has no documentation. A good program should include documentation, at least the identity of the author, the purpose of the program and other relevant information. Keep this in mind when you write your own programs.

When you are done, press <ESC> then ZZ. You may also press "<ESC>:x<ENTER>" (more clearly seen as pressing the following four keys one after another: <ESC> : x <ENTER>) to save your program and exit from the *vim* editor. <ESC> means press the Escape key, then press the colon key (shift-;), then press x (the x is a lowercase x), and finally <ENTER> means press the Enter key. This option is not recommended as you might accidentally press X instead of x. You might get your file encrypted and no one will be able to open the file for you. This happened to some students during PE and get zero for that problem.

If you want to save your file without exiting from the *vim* editor, press "<ESC>:w<ENTER>", then press the "i" key again to go back into INSERT mode. It is a good habit to save your file periodically so that if the network or the system goes down for any reason, you will not lose your hard work.

When you startup *vim*, it begins in COMMAND mode. One way to go into INSERT mode is to press the "i" key. While we are in INSERT mode, we can type in our Java code. To switch back to COMMAND mode, we press the "<ESC>" key.

The following shows you a list of useful commands in *vim*:

- <ESC>:x<ENTER> : Saves your program and exits from *vim*. (don't use this option)
- <ESC>:wq<ENTER> : Saves your program and exits from *vim*.
- <ESC>ZZ : (Note that the Zs are uppercase) Saves your program and exits from *vim*.

- `<ESC>:q!<ENTER>` : Exits from *vim* without saving your program.
- `<ESC>ZQ` : Exits from *vim* without saving your program.

If you would like to learn more about *vim* http://tnerual.eriogerg.free.fr/vim.html

Compiling and running your program

```
e0101234@stu1:~/c[xxx]$ javac HelloWorld.java
```

There will be error messages if your program has errors. Go to Step 7.1 to make the necessary corrections and re-compile. If there are no compilation errors, a class file helloworld.class will be created. Proceed with program execution as follows:

```
e0101234@stu1:~/c[xxx]$ java HelloWorld
```

If you feel comfortable with the above steps, you can try out some bigger java programs from the textbook. Do not be discouraged by the compilation errors (you are bound to get a few ☺), it is more important to understand those errors quickly (what they mean and how to fix them).

When you write program that input a lot of data, it is not logical to type the data every time you run the program, you can create a text file to contain the data and then use it as the standard input by doing the following re-direction command.

```
e0101234@stu1:~/c[xxx]$ java HelloWorld <  hello1.in
```

the '<' is the command that tell java to treat hello1.txt as the standard input file. There is no need to change your program to do file input.

Most of the time, output of your program will appear on the screen. When your program produces a lot of output, it is not easy to look at them. If this is the case, we can use the other re-direction command to output the data to a test file which can be read separately.

```
e0101234@stu1:~/c[xxx]$ java HelloWorld < hello1.in > hello1.out
```

When you are testing your program, to check whether the program passes all the test data, you can use the unix command diff to compare the output file from your program to the standard output file provided. For example, if the standard output file given my your TA is Hello1.out and your program produced the output file as indicated, then the command

```
e0101234@stu1:~/c[xxx]$ diff Hello1.out  hello1.out
```

will compare the two files character by character. If there is no difference between the two files, then you will get back the command prompt. If they are differences, you will receive an error message.

Section 8: Logging Out

When you have finished, you must always log out of the computer system as follows:

```
e0101234@stu1:~/c[xxx]$ logout
```

Typing "`exit`" (or pressing Control-d) will also have the same effect. You MUST remember to log out, otherwise anyone who comes by may delete all your valuable files away!

Appendix: Transferring files to and from Unix server

PuTTY uses a separate executable to handle file transfers called PSFTP. Start by opening PSFTP from the start menu (it is in the same folder as PuTTY). At this point you will be prompted to connect to a server. Type "`open stu.comp.nus.edu.sg`". Once again, you will be prompted to enter your username, followed by your password.

Now, since we are transferring files to and from the server, we need to be able to keep track of our local working directory as well (this is the directory on your computer which PSFTP will use to find files to upload to the server, as well as store files downloaded from the server). All of the following commands will also be shown if you enter "`help`" in the PSFTP prompt.

lpwd to (**L**ocal) **P**rint **W**orking **D**irectory

```
psftp> lpwd
Current local directory is C:\Program Files (x86)\PuTTY
```

lcd to (**L**ocal) **C**hange **D**irectory

```
psftp> lcd ../..
New local directory is C:\
```

!dir to list local **DIR**ectory

```
psftp> !dir
```

'!' is a special symbol that specifies that the command should be sent to Windows instead of being parsed by PSFTP. In this case, we use it to ask Windows to list the files in the local directory. To handle directories on the server, you can use the same commands as on PuTTY (eg. `pwd`, `cd`, `ls`, `mkdir`, `rmdir`)
To upload and download files (and directories) we will need to use the put and get commands:

get to **get** file from server

To download a file from the server and store it on your local PC, you use the `get` command. In its simplest form, you just use this with a file name:

```
get myfile.dat
```

If you want to store the file locally under a different name, specify the local file name after the remote one:

```
get myfile.dat newname.dat
```

This will fetch the file on the server called `myfile.dat`, but will save it to your local machine under the name `newname.dat`.

To fetch an entire directory recursively, you can use the `-r` option:

```
get -r mydir
get -r mydir newname
```

(If you want to fetch a file whose name starts with a hyphen, you may have to use the `--` special argument, which stops `get` from interpreting anything as a switch after it. For example, '`get -- -silly-name-`'.)

put to **put** file onto server

To upload a file to the server from your local PC, you use the `put` command. In its simplest form, you just use this with a file name:

```
put myfile.dat
```

If you want to store the file remotely under a different name, specify the remote file name after the local one:

```
put myfile.dat newname.dat
```

This will send the local file called `myfile.dat`, but will store it on the server under the name `newname.dat`.

To send an entire directory recursively, you can use the `-r` option:

```
put -r mydir
put -r mydir newname
```

(If you want to send a file whose name starts with a hyphen, you may have to use the `--` special argument, which stops `put` from interpreting anything as a switch after it. For example, '`put -- -silly-name-`'.)

mget to **get m**ultiple files from server
mput to **put m**ultiple files onto server

`mget` works almost exactly like `get`, except that it allows you to specify more than one file to fetch at once. You can do this in two ways:

- by giving two or more explicit file names ('`mget file1.txt file2.txt`')
- by using a wildcard ('`mget *.txt`').

Every argument to `mget` is treated as the name of a file to fetch (unlike `get`, which will interpret at most one argument like that, and a second argument will be treated as an alternative name under which to store the retrieved file), or a wildcard expression matching more than one file.

The `-r` and `--` options from `get` are also available with `mget`.

`mput` is similar to `put`, with the same differences.

Appendix 2: Changes between this doc, and previous version (5 Aug 2021)

Page 4: still run **`~cs1101c/lab0/setup`** as instructed, but for students who did so before this document was uploaded, they may want to run it again, as the old version had "`sunfire`" hardcoded as part of the prompt. You will not lose any Java files already in your account, so this script is safe to run again even if you have started coding.

Alternatively, you could change the prompt by typing "**`vim .bash_profile`**", and changing the line with:

```
export PS1='\u@sunfire [\t] \w \$ '
```
to
```
export PS1='\u@\h [\t] \w \$ '
```

Page 5-6: modified prompts to read "`stu1`" instead of "`sf3`"

Page 6: changed the whitespace for "**`mkdir java`**" to be more visible (present in the old version, but barely visible unless the space was specifically highlighted)

This is the end of this walkthrough document. If you encounter or discover any problem, please let us know.