

COMP10002 – Assignment 2

1	Submission Process
2	Today's Work & Advices

Computing environment in your laptop: scp, ssh, VPN

- Make sure that your [Unix/minGW](#) shell have commands [scp](#) and [ssh](#)
- If your [minGW](#) does not accept [scp](#) and [ssh](#), open the [minGW Installation Manager](#), mark line “[msys_openssh bin](#)” for installation and install.
- At home: install [VPN](#) for connecting to uni’s server from home by following the link from Submission Instructions

- Sample copying a file or a folder to **H:**

```
scp ass1.c bob@dimefox.eng.unimelb.edu.au:
```

```
scp -r ass1 bob@dimefox.eng.unimelb.edu.au:
```

- Login into [dimefox /nutmeg](#)

```
ssh bob@dimefox.eng.unimelb.edu.au
```

Assignment 2: 4C process

1. **CREATE**: Create a directory, say `ass2`, download all related files into `ass1`, then create `ass2/ass2.c` that satisfies the requirements 😊
2. **COPY**: Copy the whole directory `ass2` to your university's drive `H:`. Note: if you work in lab computers and use `H:`, you don't need to do this step.
3. **CHECK**: login into the server to do the testing/checking `dimefox.eng.unimelb.edu.au`, then on that server, navigate to the directory `ass1`, compile and test your program.
4. **COMMIT**: while in `dimefox`, submit your `ass2.c`, and verify.

Today Work

Create a simple (perhaps near empty, perhaps just for Stage 0), then try all 4 steps. Make sure that you can submit, at least from a lab PC.

Then, incrementally **CREATE** your `ass1.c`, do **COPY-CHECK-COMMIT** after every major development.

1. The CREATE step (on lab PCs or your laptop)

CREATE: Create an assignment's directory, say `ass1`, under your `comp10002`. To this directory:

- download all the data files mentioned in point 2 of FAQ, namely, all files `test?.txt` and all files `test?-out.txt`
- copy `ass2-skel.c` to `ass2/ass2.c`, and:
 - sign the authorship declaration, and
 - add comment `Algorithms are fun` at the end
- then compile & test to make sure it “works”.

2. The COPY step (from your laptop)

COPY: Copy the whole directory `ass2` to your university's drive `H:.`

1.

2. To copy:

- If yours is a Mac: open a `Terminal`. If it's a PCs: open a `minGW` window
- Navigate to the parent directory of your `ass2`
- Run the following command for copying the whole directory `ass2`:

```
scp -r ass2 bob@dimefox.eng.unimelb.edu.au:
```

(note: replace `bob` with your `loginname`, and don't forget the `colon :` at the end of the line)

3. The CHECK (TEST) step

- **login into the server** `dimefox.eng.unimelb.edu.au`: From Mac `Terminal`, or Windows' `MinGW` window, run command:
`ssh bob@dimefox.eng.unimelb.edu.au`

- **Then**, when you are with `dimefox`:
 - Navigate to your `ass2` directory
 - Compile your program
 - Test, at least with all data Alistair supplied.

- **Example testing** using `test0-out.txt`:

```
$ ./ass1 A < test0.txt > mytest0-out.txt
```

```
$ diff mytest0-out.txt test0-out.txt
```

The “`diff`” command will find the difference between 2 files. If it produces no output at all, then the 2 files are absolutely identical (Bravo!). If not, then you need to open both files using `jEdit` and try to figure out what's wrong in your output.

You can also do testing on your laptop, but remember that a final test in `dimefox` is a need!

4. The COMMIT (SUBMIT) process

When you are working on `dimefox`, and already navigated to your `ass1` directory, run:

```
submit comp10002 ass2 myass2.c
```

then, wait a few minutes and verify by:

```
verify comp10002 ass2 > my-receipt-ass2.txt
```

```
more my-receipt-ass2.txt
```

The “`more`” command will display the content of the receipt.

Alternatively, you can use `jEdit` to open `my-receipt-ass2.txt` for a careful viewing.

When to submit? Submit now, submit today, scp and submit after any session you work with the assignment. Think about submission as a way to backup your work!

Assignment 2: advices

- *Regularly submit and backup your work (for example, by securing a copy **with timestamp** on **dimefox** every day).*
- *Read the specifications and marking rubric carefully.*
- *It's better to design a data type for graph before writing stage 0. When designing, make sure that the data type supports the operations needed for Stage 0 and Stage 1.*
- *A diagram of your data structure on paper could be very helpful.*
- *Test your program carefully, at least with all supplied data. Do the testing not only in your computer, but also on **dimefox**.*
- *This time, it's very likely that your program works well on your laptop but crashes on **dimefox** → **TEST YOUR PROGRAM ON dimefox**.*
- *For checking memory leaks, use **valgrind** on dimefox.*

Algorithms are fun!

Good Luck!