Submissions for comp10002 Assignment 1, 2019s2

Instructions for submitting programming projects using the `submit` and `verify` systems.
Note, you will need to install if the University's VPN software Cisco AnyConnect in order to connect to `dimefox` from outside the University network (that is, from home).

Everyone Read This First!

There are two machines involved:

- The machine your file is on (might be the shared drive accessible from the lab computers, might be your own PC, might be your own Mac machine); and
- The server that you run `submit` on, and that we compile and test your programs on.

So there are thus two steps that need to be performed:

- The COPY step: Copy your file over to the `submit` machine; and then
- The SUBMIT step: Login (in a terminal shell) to the server and run the `submit` command.

How you do these two steps depends on what kind of machine you are on. But everyone has to do the two steps.
There are system-dependent differences between the `gcc` on the lab computers and the `gcc` on the server. In particular, the server is very unforgiving of uninitialized variables. If your program fails on the server, look carefully for this kind of bug.

Don't forget the `-lm` on the end of the compilation line if you have used functions from the math library. There might also be some small differences in the availability of functions like `strncmp()` and etc for string handling. *You are strongly advised to compile your program on the server and test it as a routine part of your development. Only then will you have the confidence that it will work correctly when we test it. Use a normal gcc command (with -lm at the end if you need the math library) while you are connected, and then execute the compiled program on that machine with some suitable data.*
After you `submit`, wait for a few minutes, and then, before you disconnect from the server, `verify` your submission, to make sure that your program compiled smoothly and has executed on the first set of simple test data. (We will re-run all of the programs again after the due date, using further test data; you will not be able to access that test data in advance.)
A complete transcript of a session from a Mac appears below, there are only minor differences between this and what you would see on a PC, and the commands that are typed to the `dimefox:` prompt are completely independent of which machine you started at.
It is assumed throughout that your source program is in a file `myass1.c`; if not, replace `myass1.c` in the instructions by the name of your file. In the `submit` and `verify` commands, the first argument must be `comp10002` and is case-sensitive; and the second argument must be `ass1`, the name that is being used for this particular project. The third argument to `submit` is the name of your program, and must be the C source file (not the executable).
You can submit as often as you like. Indeed, the more frequently you lodge a submission, the better off you will be if you have a disaster and would like to recover a previous version of your program. Submitting once a day is not at all unreasonable. And the sad truth is, most of the people who stuff things up at the last moment (9:57am on the due date) do so because they are floundering around trying to understand what they are supposed to be doing. Practice in advance…

From a PC, including the Lab machines

0. Install the `PuTTY` and `pscp` programs from here. They are already installed on the lab computers, find them in the menus or use Explorer to locate them.

And also make sure (if you are at home) that you have started the VPN software and used your login name/password details to establish a secure connection to the domain `remote.unimelb.edu.au/student`, *before* you try and do steps 1 and 2. (No need to use the VPN if you are already on a computer within the University network.)

1a. From your home PC or your own laptop: do the COPY step using `pscp`, by starting up a `cmd` shell (like you did to compile the program), `cd`'ing to the directory that has your `myass1.c` file in it, and typing

```
    pscp myass1.c my-username@dimefox.eng.unimelb.edu.au:
```

(don't miss the final colon, it is required). You might get asked to accept an encryption key (type "yes"), and will need to type your University password. The file will then copy over to your home directory on the server. Note that if you have downloaded `pscp` and `PuTTY`, they are probably just sitting on your Desktop. They need to be moved to a place where

they can be executed, either a folder that is in your path, or into (perhaps via a shortcut) into the directory where you have your C programs.
1b. On a lab machine: your files are on the shared drive, and are already accessible from `dimefox`. All you have to do when you login to `dimefox` is navigate to the right place using `cd` commands.
2. The SUBMIT step is done by using `PuTTY` to remote login to one of the student servers. Use "All programs > Network Apps > PuTTY > PuTTY" on the lab machines, or the location that you placed it on you home computer (perhaps the Desktop), to start PuTTY running. On your home computer, it may be necessary for you to "accept to execute a program that has been downloaded from the internet" first, and note again, this will not work at home if you haven't installed and started the University VPN, instructions here.
A dialog box will open, type `dimefox.eng.unimelb.edu.au` as the "destination you want to connect to", and check that the "Port" is set to 22. Then click the "Open" button, and wait for a connection to be established; click "Yes" to accept the server's host key if you get asked. If `dimefox` is unavailable, try `nutmeg`, they are equivalent and all share the same file systems. (You can always use either of these machines, `dimefox` is used here as the example for consistency only.)
You will then be issued with a login prompt; type your University username and University password to connect to the Unix server. Note that it may look like the password characters aren't getting registered, because the cursor doesn't move. Just type the whole password carefully and correctly, and then "return".
Once you have logged in, check that your file is indeed sitting there waiting to be submitted (use the command `ls`), check that it compiles and executes cleanly on the server (use `gcc` and etc), and then type the `submit` command:

```
    submit comp10002 ass1 myass1.c
```

Wait a few minutes, and then carry out the verify and check steps:

```
    verify comp10002 ass1 > my-receipt-ass1.txt
    more my-receipt-ass1.txt
```

Look through `my-receipt-ass1.txt` carefully to make sure that (a) your program compiled; (b) it executed on the single initial test file; and (c) that the listing that is shown is the right version of your program.
You have to have a network connection to make all of this work, of course!
Then, logout from the server using `logout`, and exit `PuTTY` if it doesn't close by itself.

From a Mac (and Linux)

It's only a little different. Remember, this will not work outside the University's network if you haven't installed and started the University VPN, instructions here.
Linux users should look here for guidance on installing the VPN.
0. Make sure (if you are at home, and perhaps even if you are at the University coming in via UniWireless) that you have started the VPN software and used your login name/password details to establish a secure connection to the domain `remote.unimelb.edu.au/student`, *before* you try and do steps 1 and 2. (No need to use the VPN if you are already on a computer within the University network.)
1. To do the COPY step, start a terminal window ("Terminal" on a mac), then navigate to the directory that contains your file, then execute a `scp` command (already installed on a Mac as a standard tool):

```
    scp myass1.c my-username@dimefox.eng.unimelb.edu.au:
```

(don't miss the final colon, and don't forget that you need to `cd` to the right directory first). Type your University password when prompted for it.
2. Then to do the SUBMIT step, you stay in the terminal window, and use `ssh` to create a network terminal connection through to the server:

```
    ssh my-username@dimefox.eng.unimelb.edu.au
```

(without a final colon) and then once you have typed your University password (and typed "yes" if it asks you to accept the encryption key) everything you type in that window is being executed on the server. So now you can check that your file is there by using `ls`, compile it using `gcc`, run it by typing `./myass1`, and presuming that it is all ok, run the `submit` command:

```
    submit comp10002 ass1 myass1.c
```

followed a few minutes later by the verify and check steps:

```
    verify comp10002 ass1 > my-receipt-ass1.txt
    more my-receipt-ass1.txt
```

Then, logout from the server using `logout`; you can then close the local terminal window that you were using.

From A Home Computer

It is exactly the same, *except* that before you can use `ssh` , you need to have installed the University's VPN (virtual private network) software, see the instructions linked from here.

Complete Transcript

With a few liberties shown, here is a complete transcript for a person whose login name is *uname*, showing the steps to be followed on a mac, assuming that they have installed the VPN (or are on a lab computer) and have connected to `remote.unimelb.edu.au/student`. From a PC, `pscp` will have been used to do the COPY step rather than `scp`. The commands shown in red are what the user typed; the commentary in blue is intended to help you understand what is happening.
Note that this careful user is checking that their program compiles and executes on the server before doing the submission. *You should do this as a matter of routine, every time you submit, but especially including the last time you submit!* Submissions that don't compile will be heavily penalized.

```
mac: cd 10002/ass1
mac: ls
myass1.c
myass1
test0.txt
test1.txt
test2.txt

mac: scp myass1.c uname@dimefox.eng.unimelb.edu.au:
uname@dimefox.eng.unimelb.edu.au's password: XXXXXXX
# Note: typing the password will not move the cursor or give XXXX's

mac: scp test1.txt uname@dimefox.eng.unimelb.edu.au:
uname@dimefox.eng.unimelb.edu.au's password: XXXXXXX

mac: ssh uname@dimefox.eng.unimelb.edu.au
Do you wish to accept security key: yes
uname@dimefox.eng.unimelb.edu.au's password: XXXXXX
# Note: typing the password will not move the cursor or give XXXX's

# You should now be "talking" to dimefox, and the prompt will have changed

dimefox: ls
myass1.c
test1.txt
# plus maybe other stuff there as well

dimefox: gcc -Wall -o myass1 myass1.c -lm

# Note: need the -lm at the end for the math library, and don't use -ansi if you
rely on M_PI

dimefox: ./myass1 < test1.txt

# Should see the same output that you got on your own computer
# if not, you have a bug, look first for uninitialized variables

dimefox: submit comp10002 ass1 myass1.c
This is part of the legacy submit system.
Connecting to dimefox.eng.unimelb.edu.au ... OK
Your submission is continuing in the background.
Don't forget to VERIFY later.

# wait for a minute or two

dimefox: verify comp10002 ass1 > my-receipt-ass1.txt
This is part of the legacy submit system.
Connecting to dimefox.eng.unimelb.edu.au ... OK
dimefox: ls
myass1
myass1.c
mytest.txt
```

```
my-receipt-ass1.txt

dimefox: more my-receipt-ass1.txt
============================================
vis/uname-submit.txt
13:04:02_Friday_30_April_2019
============================================

# and so on, should show the output, and then your source code
# (press "space" to step through the output screens , that's how "more" works)

# look carefully at this file to check that:
# (a) you submitted the right program;
# (b) that the program compiled properly, and
# (c) it then executed correctly on the simple test data that was used

dimefox: logout
Connection to dimefox.eng.unimelb.edu.au closed.

mac:
```

Last updated: August 30, 2019
Alistair Moffat, ammoffat@unimelb.edu.au