Getting Started Guide

1 CS undergraduate environment

Whenever you are asked for a "host name", please use `linux.student.cs.uwaterloo.ca` instead of the name of a specific machine. CSCF periodically upgrades their servers, which changes the names. (See `https://cs.uwaterloo.ca/cscf/teaching-hosts` for the most up-to-date list of servers.) It also means that you'll get assigned to a machine so that the system loads are balanced, which hopefully means that your response time will be fast.

- To log into the CS undergraduate environment, you need to set up a password that is separate from your WatIAM/Quest password.
  - Go to `https://www.student.cs.uwaterloo.ca/password/` to set up your password.
    Read the instructions for choosing a password, then type your password in the first box. When you are finished typing in the box, read the message to the right of the box to determine if your password is acceptable according to the requirements.
    If it is not, you must try again until you get "Looks OK!". Retype your new password in the second box. When you get "Looks OK!", you can save your password by clicking the "Save" button.
- Connecting to the undergraduate environment requires good internet access (and can sometimes be a little slow) but it has several benefits:
  - Regular (hourly, nightly, weekly) backups of your files if CSCF has enabled snapshots.
  - Required software is pre-installed.
  - It is an exact replica of the environment in which our testing system, Marmoset, works. If your submission works in the undergraduate environment, it will work on Marmoset.

2 Connecting to the CS undergraduate environment

2.1 Linux

- If you primarily use Windows, you can still install Linux (e.g. ubuntu) on a second partition of your hard drive.
- Most Linux distributions come installed with typical applications that you will need (e.g. `vim`, `ssh`, `scp`).
- To log in to the undergraduate environment:
  - Open a terminal.
  - Use the Secure SHell command, `ssh`. Execute the command `ssh -Y userid@linux.student.cs.uwaterloo.ca` (replace *userid* with your university userid not student number e.g. `j2smith`).
  - Enter your CS environment password when prompted (you won't see the characters and your cursor will not move as a security feature).
  - Note that the `-Y` option allows for X11 forwarding (e.g. graphical applications).

2.2 Mac

- Every Mac has a Terminal application that runs a text interface for Linux. Open the Terminal application and then use the `ssh` command discussed above.

  You should be prompted to enter your password. Enter your password that you just set up previously and press enter. You will not see the characters entered, and your cursor will not move as a security feature.

2.3 Windows

You only need to use one of the following options but you could try a few to see which one you like.

1. Using `Command Prompt`
   - Every Windows installation comes with the `Command Prompt` application. Open this application and then use the `ssh` command discussed above.
   - Note that this `Command Prompt` is not running bash. While a few bash commands might work, it is not a substitute for bash.
2. PuTTY (Recommended as it is the oldest and most stable way.)
   - This is an SSH client that can be downloaded for free here: `https://putty.org/`
   - Open PuTTY.
   - In the "Host Name" field enter `linux.student.cs.uwaterloo.ca` (press the `Save` button to save this session for later use).
   - In the sidebar under "SSH", click "X11".
   - Click the box that says "Enable X11 forwarding".
   - Press the "Open".
   - Enter your CS username and password.
   - Again, it may appear that nothing is happening when you type your password but your keystrokes are being hidden for privacy.
   - Here is a link on how to setup `ssh` for PuTTY:
     `https://devops.ionos.com/tutorials/use-ssh-keys-with-putty-on-windows/`
3. Linux bash shell on Windows 10.
   - The 64-bit install of Windows 10 now supports a Linux subsystem.
   - You would need to enable the "Windows subsystem for Linux" and then install the Ubuntu app from the Microsoft Store.
   - Follow the instructions in the "SSHFS-WindowsInstructions.pdf" document.
     (Only mount your student account. Do not mount the course account.)

3 Basic Bash Commands

- Some basic commands that you may need for assignment 0 (a0):

Command	Example	Description
`ls`	`ls` `ls -alF`	Lists files in current directory List in long format

`cd`	`cd tempdir`
	`cd ..`
	`cd ~/cs246`
		Change directory to `tempdir`
		Move back one directory
		Move into the directory named `cs246` under your home directory

`mkdir`	`mkdir cs246`	Make a directory called `cs246`
`cp`	`cp file1 dir`	Copy `file` into a directory named `dir`
`mv`	`mv old.txt new.txt`	Move or rename files
`man`	`man ls`	Online manual (help) about command `"ls"`
`pwd`	`pwd`	See your current working directory
`cat`	`cat file1`	display the contents of a file named `file1`

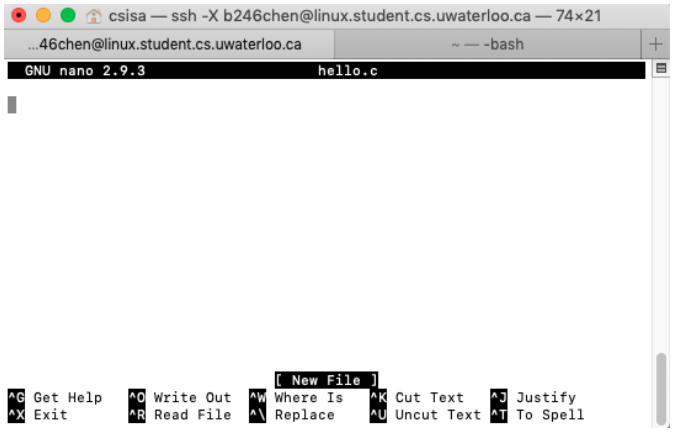
- Once you have completed a0, you will gain access to a Linux commands handout named `linuxCommands.pdf`.

4 Text Editors

There are several different command line text editors that you can use.

- We recommend `vim` (it is the replacement for `vi`).
  - `vim` has a pretty steep learning curve. However, once you master it, it greatly increases productivity.
  - We highly recommend looking up a "vim cheat sheet" and even printing it out.
  - Enter the command `vimtutor` on the command prompt to start the `vim` learning tutorial (Press `Esc :q` to quit).
  - To start `vim`, enter the command `vim file` on the command prompt to create or edit an existing file named `file`.
- Other options are `emacs`, or `nano`. `nano` is relatively easy to use if you find `vim` too difficult to learn.
- In addition, there are graphical text editors that are handy (e.g. gvim, gedit). These are likely to run slowly if run on a remote machine like the student Linux accounts.
- Do NOT use software like Microsoft Word and other Rich Text Editing software since they often embed characters not recognized by compilers.

4.1 A very short nano tutorial



The User Interface of nano

After successfully logging into your CS student Linux environment:

1. Make a `cs246` directory by using the following command `mkdir cs246` if the directory does not exist.
2. Now, change directories by using: `cd cs246`
3. Open an editor called `nano` using the command: `nano hello.c`

Write the code for assignment 0 using `nano`.

1. You can start typing your code right away!
2. Type out your code.
3. Press `Ctrl + O` to save your code. (Notice the prompts displayed at the bottom of Terminal; "^" means the Ctrl key.)
4. Press `Ctrl + X` to Exit.
5. Press `Y` to save.
6. Press the Enter key.

5 git: a Version/Revision Control System

- `git` is a popular version/revision control system that was developed by Linus Torvalds.
- Why are we using `git` rather than Subversion (SVN) or CVS?
  - It has become somewhat of an industry standard.
  - The functionality is largely the same as any other version/revision control system.
  - You get a local repository that you are free to modify, which allows you to rollback changes, branch code, etc. This functionality is not available on SVN/CVS.
  - Note: while you can modify your local repository/copy of the CS246 repository, you will not be able to push your modifications to the remote CS246 repository (only course staff can do that for obvious reasons).
- Instructions on how to clone your local repository for the course are found in assignment 0.
- To modify your local copy and take advantage of version control, you should look into the commands: add, commit, rm, checkout.
- However, you can create your own repositories on the UW git server that will allow you to push and pull to your heart's desire. See `git.uwaterloo.ca` for additional information.
- If you wish to make a repository on the UW git server, it *must* be set to **private**. Failure to do so can lead to severe academic integrity violations including course mark deduction, suspension or expulsion.

6 Copying files

If you are working in the Linux environment provided by the school, there might be times when you want to copy files from this **remote** machine to your **local** machine. How you do it will depend on which computer/OS you are using and how you chose to connect to the Linux environment:

6.1 Using the scp command

If you have access to a command prompt that supports the `scp` command you can use it to transfer files back and forth between a local and remote computer.

- Open a terminal and run the command `man scp`. This gives you the manual entry for the `scp` command. Read it.
- At its core, the `scp` command has the following format:

  ```
  scp source destination
  ```

  This will copy the file(s) from the source location to the destination location. Each of `source` and `destination` can be of the form `[[user@]host1:]file1`. Some examples follow:

  - `scp hello.c linux.student.cs.uwaterloo.ca:cs246/.`
    copies the file `hello.c` from the current directory of the local machine (on which this command is being executed) to the `cs246` directory on the `linux.student.cs.uwaterloo.ca` remote location. It assumes that the userid on both the local and remote machines is the same.
  - `scp jsmith@linux.student.cs.uwaterloo.ca:cs246/1225/a0/a0.pdf .`
    copies the file `cs246/1225/a0/a0.pdf` from `jsmith`'s student account to the current directory (note the `.` to indicate the current directory) on the computer upon which this command was executed.

6.2 Graphical Interfaces for Secure File Transfer

Every OS has some form of secure graphical file transfer application. Just search for "secure file transfer application". For example, if you had a Mac, you could search for "Mac OS X secure file transfer application".

Popular applications for Windows are `winscp` and FileZilla.

6.3 Using SSHFS to mount your student account as a drive

SSHFS (a filesystem client based upon SSH) mounts the specified account as a drive/folder like Samba, but it uses SSH. You will be able to work with the account's files in a Finder window.

Note: This an optional step, and only makes sense if you find yourself copying files back and forth from the remote server *very* often. SCP and/or a GUI interface are usually sufficient for working with CS 246.

6.3.1 Getting Access:

You just need to be able to SSH into the account.

If you want to use this on your local machine, you will have to install SSHFS first.

See `https://tinyurl.com/sshfs-tutorial` for instructions covering Ubuntu/Debian, and Mac OS X machines. There are instructions for Windows 10 in the "SSHFS-Instructions" PDF document (the process is a little lengthier).

The following instructions are for the Mac, but will be similar for the other operating systems. If you are on Solus Linux (and apparently some Mac systems), you may need to modify the instructions below slightly. See the box below the "Explanation".

1. Open a Terminal and run these 3 commands.

   You don't need to SSH into anything, just run them when the Terminal opens. The third command will ask for your password because it connects to your own account.

   You can avoid this step by setting up an SSH key (see below).

   (a) `mkdir -p ~/yourQuestID`

   (b) `sshfs yourQuestID@linux.student.cs.uwaterloo.ca: ~/yourQuestID -o volname=yourQuestID-ssh`

   (c) Type in your password to your account in the CS student environment if you haven't set up an SSH key. (If you need to reset your password, go to `https://www.student.cs.uwaterloo.ca/password/` first.)

   Explanation: The first command creates a folder where your account will be mounted. Your files will show up in these folders. The `-p` means "do nothing if the folders already exist." This only needs to be done once, unless you're working in the Mac labs where your environment is temporary.

   The second command mounts your own account to the folder `~/yourQuestID`. After connecting, an icon may appear on your desktop with the name `yourQuestID-ssh` that you can double-click upon to see your own files.

2. Your account should show up as drives on the `Desktop`. If not, they should show up in your home folder (go to the Finder menu bar, click Go then Home). You can disconnect from the account by right-clicking the drive/folder, and choosing Eject.

3. (optional): To save time, you can copy-paste the 3 commands into a file such as `sshfs.sh` (if you've set up an SSH key, you'll only need the first 2 commands). Then run the commands by typing:

   `bash sshfs.sh`

   This way, you won't have to type the commands each time you log back in to a lab Mac. If you're not using the Mac lab, it's probably easier to just set up the second command as an `alias`.

If you are using Solus Linux:

- When installing `sshfs`, note that it may be called "sshfs-fuse":

  ```
  sudo eopkg install sshfs-fuse
  ```

- The same `mkdir` step works.
- Instead of the option `volname`, use `fsname` instead:

  ```
  sshfs yourQuestID@linux.student.cs.uwaterloo.ca: ~/yourQuestID -o fsname=yourQuestID-ssh
  ```

7 Skipping Two-Factor Authentication

In order to avoid having to perform Two-Factor Authentication (2FA) every time that you connect to the student environment, you will need to create an SSH key. We will use the RSA protocol, though there are many other possible ones to use. Be sure to keep the generated key `.ssh/id_rsa` private! Anyone with that file can access the student server with your account. You will instead copy the public key, `.ssh/id_rsa.pub` to your student account `.ssh` directory.

7.0.1 Windows

First create an SSH key if you don't already have one:

```
ssh-keygen
```

Then press enter at the prompts to use default values, unless you want to type in a code or password every time you SSH.

Note that Windows 10/11 has `ssh` by default so you can also use this command to create a SSH key on Windows (in `cmd`).

If you get a warning saying a key already exists, DO NOT create a new key, just use Ctrl+C to kill the process and use your existing one.

On Windows, run this command (note that for readability purposes, the line has been split after the second semicolon, but should be entered on a single line):

```
ssh YOURUSERID@linux.student.cs.uwaterloo.ca "umask 077;mkdir -p .ssh;
cat >> .ssh/authorized_keys || echo 'Whoops that failed'" < %HOMEDRIVE%%HOMEPATH%\.ssh\id_rsa.pub
```

Finally, to skip DUO authentication on the Linux server when you `git pull`, run:

```
ssh-copy-id localhost
```

and go through DUO one last time.

7.0.2 Mac/Linux

Open a terminal window and type:

```
ssh-keygen -t rsa
```

To copy your key to the linux servers run this command:

```
scp ~/.ssh/id_rsa.pub [Quest ID]@linux.student.cs.uwaterloo.ca:~/.ssh/authorized_keys
```

8 Recovering Missing Files

If you ever accidentally delete something you didn't want to delete, here are several ways to recover it.

1. `git checkout`

   This applies to materials provided to you via the repository. If you are missing `filename` then you can recover the file using `git checkout filename` in the directory that the file is suppose to be located within.

   This also works for entire directories, and will recursively recover the whole directory. This method does not work for files you've created, as it only works on files we upload to the repository.

2. `.snapshot`

   This method is for files that you've created, but it only applies if you've done your work in the CS student environment. Due to filesystem issues, this feature is currently disabled by CSCF.

   You can go into the snapshot folder using `cd .snapshot` in the folder where the file was, and then using `ls`, you can see weekly, daily, and hourly backups of that directory. Note that the `.snapshot` directory will not be listed if you use the `ls -a` command.

   You can then `cd` into those backups, to see a read-only copy of the directory at the time the snapshot is made. From there, you can `cp` the past version of the file back to your original folder.

   Since the most frequent update is hourly, you may lose up to an hour of work.

3. You can use a version control to make more frequent, on demand backups. If you choose to do so, we strongly recommend UW Gitlab. If you use version control, you must set your repositories to private, or risk violating academic integrity. This is especially important if your code is hosted on external servers such as GitHub.

   Note that submitting to Marmoset also acts a a backup system, since you can download your submissions from there.

9 Zipping files

If you are submitting more than one files, you can use the command `zip` to create a ZIP file. ZIP is a compression and file packaging utility for Unix. Each file is stored in single `zip` file with the extension `.zip`. `zip` compresses the files to reduce file size and is also used as file package utility. `zip` is available in many operating systems like Unix, Linux, Windows etc. See `man zip` for the manual page on how to use it.

Be careful how you create the ZIP file; Marmoset generally doesn't want the entire directory structure to be zipped, just the files, and you will fail the Marmoset tests. The best way to ensure that you've done it correctly is to copy your ZIP file to a clean directory and `unzip` it. Also, ZIP by default adds files to an already existing ZIP file, so if your ZIP file contains files it shouldn't, you will need to delete it and create it again.

Syntax :

```
zip [options] zipfile files_list
```

Syntax for creating a zip file:

```
zip myfile.zip filename.txt
```

10 Testing

See `https://student.cs.uwaterloo.ca/~cs246/W24/testing.shtml` if you need a refresher on how to use the `produceOutputs` and `runSuite` programs for testing your test suite.

Notes:

- The provided binaries in the `tools` folder of your CS246 Git repository will only work in the CS student environment.
- `produceOutputs` has an additional optional `-e` flag to produce `.ret` (program execution return code) and `.err` (standard error output) files.

- `runSuite` supports the additional optional `-e` flag used to compare those files in addition to the standard output files. It also supports the additional optional `-v` flag that produces `.mem` files containing `valgrind` output for each test case in the suite.

11 Debugging

See `https://student.cs.uwaterloo.ca/~cs246/W24/debugging.shtml` if you need a refresher on how to use `gdb` and `valgrind` for debugging.

11.1 `valgrind` usage

11.1.1 `valgrind` Basics

`https://www.youtube.com/watch?v=D4G2JcA0UXE`

11.1.2 Using `valgrind` to debug the Assignment Operator

`https://www.youtube.com/watch?v=8762-C1ng_g`

12 Miscellaneous

- Press on the up arrow to see previous commands.
- Press the Ctrl-r keys to search backwards through the bash history. (Useful for repeating previous commands.)
- The command `clear` will clear the terminal.
- When typing in a command or file name, you can press tab to autocomplete the word if the remainder of the word is not ambiguous. Otherwise, it will fill in part of the word and pressing tab again will show the options for what word it could be.