CS 241 – Week 1 Tutorial

Setup & C++ review

Fall 2017

1 Summary

1. Setup
2. C++ code style review
3. Standard Template Library (STL) review

2 Setup

To access the student environment via the command line, you must ssh into your user account. Linux/Mac users can use Terminal and execute the command `ssh userid@linux.student.cs.uwaterloo.ca`. Windows users should use an ssh client called PuTTY, or Cygwin, or use a virtual machine.

2.1 Password-less ssh

Entering your password every time is tedious, but there are simple steps you can take to use password-less ssh. Instructions for those on Windows machines using PuTTY can be found at https://www.getfilecloud.com/blog/ssh-without-password-using-putty/. For Linux/Mac users, you can enter the two following commands:

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

Keep hitting Enter until this completes. This will generate the two keys and put them in `~/.ssh/` with the names "`id_rsa`" for your private key, and "`id_rsa.pub`" for your public key.

```
scp ~/.ssh/id_rsa.pub userid@linux.student.cs.uwaterloo.ca:~/.ssh/authorized_keys
```

This will copy your public key to a new file called `authorized_keys`. Subsequent public keys can be appended to this file. This means that if you wanted to add another public key for another computer on this server, you would copy the contents of the second `id_rsa.pub` file into a new line on the existing `authorized_keys` file.

You now can now access the student environment from your machine without a password.

2.2 .profile

- When you log into the CS environment there are a number of files that get executed. One of these files is `~/.profile`.

- For convenience, edit .profile to execute the command `source /u/cs241/setup`. This will save you from having to source CS241 tools every time you ssh.
- Aliases, for commands you use often, should also be added to `.profile`. For example:

  alias xxd4='xxd -bits -cols 4'
- You can also write your own functions and make them available in .profile to simplify repeated tasks.
- Bonus tip for Mac users: You can similarly edit (or create) `~/.bash_profile` on your machine if you use bash and add the following alias to simply logging into to the student environment.
  `alias lse='ssh userid@linux.student.cs.uwaterloo.ca'`

2.3 Further reading

When it comes to editing files/programs, some of you prefer to use the student environment using Vim or Emacs (or Nano), while others prefer to edit on their local machines using a graphical editor such as Atom or Sublime Text. Note that CS241 tools are only available on the student environment.

For those of you who wish to edit files on their local machine, you may want to mount the student environment as a network drive on your computer, to avoid having to copy files to/from the student environment for testing. There are a variety of options, not limited to SSHFS (`https://www.digitalocean.com/community/tutorials/how-to-use-sshfs-to-mount-remote-file-systems-over-ssh`), Mountain Duck (paid), and Cyberduck (free). You are not required to use any of these, but it has been helpful and time-saving for students in the past.

3 C++ Review

You can use several different languages in CS241 for assignments, namely C++, Racket, and Scala. For the most part we'll avoid talking about specific languages whenever possible in this course, but since a little more than half of you typically elect to do the assignments in C++, here's a bit of C++ review.

3.1 Code Style

There are a number of flaws in the following code snippet. How can this piece of code be improved? Think about both stylistic improvements and performance improvements.

```
#include <iostream>
#include <vector>
#include <set>
#include <map>
#include <string>
#include <algorithm>
using namespace std;

bool foo(vector<string> v) {
    if (v.size() > 16) {
            return true;
    } else {
            return false;
        }
```

```
}

int bar(map<string, map<vector<string>, int> >  m, string w) {
        return m[w].size();
}

bool baz(string fruit) {
        return fruit == "apple" || fruit == "pear" || fruit == "mango" ||
                fruit == "coconut" || fruit == "kiwi" || fruit == "pepper";
}


string temp;
bool qux(pair<vector<string>, int> p) {
        int count = 0;
        for (int i = 0; i < p.second; ++i) {
                if (p.first[i] == temp) count++;
        }
        if (count > p.second/2) {
                return true;
        } else {
                return false;
        }
}

int main() {
        vector<string> fruits;
    map<string, map<vector<string>, int> > fruitMap;
        while (true) {
                string fruit;
            cin >> fruit;

            fruitMap[fruit][fruits] = fruits.size();

                int mode;
                if (fruit == "apple") {
                    mode = 0;
                } else if (fruit == "banana") {
                    mode = 1;
                } else if (fruit == "tangelo") {
                    mode = 2;
                } else {
                    throw 143;
                }

                fruits.push_back(fruit);
        if (foo(fruits)) {
                    cout << "Many fruits" << endl;
                }

        int val = bar(fruitMap, fruit);
```

```
                bool flag1 = false, flag2 = false;
                if (val > 12345) {
                    flag1 = true;
                } else if (fruits.size() > 8 && fruits.size() < 12) {
                    flag1 = true;
                } else if (mode == 1) {
                        flag2 = true;
                }

                if (flag2 || flag1 && baz(fruit)) {
                        break;
        }
        }


    for (map<string, map<vector<string>, int> >::iterator it = fruitMap.begin()
          it != fruitMap.end(); ++it) {
        temp = (*it).first;
                cout << count_if((*it).second.begin(), (*it).second.end(), qux);
        }
}
```

3.2 Standard Template Library (STL)

The STL is a collection of generic containers and algorithms. Getting used to using these can make your code shorter, faster and easier to understand.

- containers: vector, list, map, set, and pair.
- algorithms: find, count, copy, for_each, transform, accumulate (and more!)

Try using the STL as much as possible to complete the following exercises:

1. Write a short C++ program which reads in a sequence of numbers separated by whitespace and prints the sequence twice: once forwards, then once backwards.
2. Modify your solution to the previous problem to read in a sequence of name and ID pairs, where the name and ID are separated by whitespace, and each pair is separated from the next pair by whitespace. Print 5 pairs per line, where each pair is formatted as `[name,ID]`. Avoid using global variables.
3. Write a short C++ program which reads from standard input line by line, and prints the number of times each character appears in that line. You may omit the count of newline character. For simplicity, print each character followed by its frequency on its own line. Make as much use of the STL as you can.