In-Class Work: C++

1. Copy & Compile & Run

This problem guides you through the first steps of writing and running a program:

Log into linuxcomp1 (redo the steps 1.-4. of the Unix/Linux Exercise). Get into your capstone directory. Next make a directory for C++ sample files, so

```
mkdir C++samples.dir
```

Get into this directory (using `cd`) and copy all C++-sample programs by typing

```
cp ~kvollmay/sunhome/classes.dir/capstone_s2008.dir/unix_C++_intro.dir/C++sample* .
```

Look at C++sample_inout.cc with

```
emacs C++sample_inout.cc &
```

This is the text file ("source code") of a program. This textfile needs to be converted to a format which the computer will understand. To do so you "compile" the source code (text file):

```
g++ -o C++sample_inout.out C++sample_inout.cc
```

This creates a file C++sample_inout.out (the "executable file") which you can now use to run with

```
C++sample_inout.out
```

You will be asked to type in your first and last name separated by blanks, do so and hit enter and then answer the next question. Check in the source code that you understand what exactly the program was doing.

2. Input/Output

2a. Write a program that reads in (from screen) the name of a fruit (e.g. banana) and its price (e.g. 0.5), and prints out for this example: `One banana costs $0.5`.

2b. Copy ~kvollmay/sunhome/classes.dir/capstone_s2008.dir/unix_C++_intro.dir/groceries.data into your working directory. Write a program which reads in 10 groceries and their prices from the file groceries.data using fstream and prints a sentence for each item as in 2a.

2c. Same as 2b. but with $<$ on the command line. To do so use `cin` in the program and run the program with `executable.out < groceries.data`

2d. Next you learn how you will share your programs with others. Go back to your home directory and create therein a directory called `share.dir`. While still being in your home directory make this directory accessible for others with

```
chmod 755 share.dir
```

Next go into share.dir and copy your program from 2c. into share.dir. Let's assume you had called your program for 2c "C++2c.cc", then you can make this file readable (accessible to be copied) for others by typing

```
chmod 644 C++2c.cc
```

Whenever I will ask you to make your programs available for others (e.g. at the next daily assignment) please copy your programs into your share.dir and make the corresponding program readable with `chmod 644` *filename.cc*

In-Class Work: C++

3. Repetitions

3a. Read in the 10 groceries (item and price) from the file groceries.data. Print out each item and its price, the sum of the prices, the sales tax (use 5%, i.e. sum*0.05), and the total cost (= sum + tax).

3b. Now read in any number of groceries. The last item and its price is followed by a line with 0. Test your program by adding a line with 0 to the file groceries.data.

4. Decisions

Read in a date in 2008 (as two integers) and check if it is during spring break (March 9 - March 16).

5. Traffic Model

Copy the file
~ kvollmay/sunhome/classes.dir/capstone_s2008.dir/unix_C++_intro.dir/road.data
into your working directory. The file contains 100 integers describing a road of 100 lattice sites at a certain time. "-1" mean an empty site and a number $>= 0$ means a car with the corresponding speed in mi/h.

5a. How many cars are on this road?

5b. How many cars have a speed larger or equal 30 mi/h?

6. Factorial 6a. Write a program which calculates $N! = 1 * 2 * 3 * \ldots * N$ and prints out N and $N!$ for $N = 1, 2, \ldots, 20$. What happens for $N \geq 14$?

6b. Write a program which reads in an integer $N < 13$, determines $N!$ via a function, and prints out the result. The the program by print N and $N!$ for $N = 1, 2, \ldots, 14$, i.e. by calling the function in main 14 times.

7. Time

7a. Write a program which reads in some time length as 3 integers for hours, minutes and seconds. Write a function which converts this into the total amount of seconds (for example 3 h 4 min 2 s = (3*3600 + 4*60 + 2)s = 11042 s). Print out the result.

7b. Write a program which reads in seconds and uses a function to convert the seconds to hours, minutes and seconds (i.e. the reverse of 7a.). How can you test your program?
Hint: Use both integer devision and the modulo function %.

In-Class Work: C++

Announcement: Solutions to in-class work 1. – 5. are in
~ kvollmay/sunhome/classes.dir/capstone_s2008.dir/unix_C++_intro.dir/C++2a.cc
C++2b.cc etc.

7. Time (User-Defined Functions)

The following task was once needed by a student whose project was the prediction of triathlon athletes performance based on their performances in the past.

7a. Write a program which reads in some time length as 3 integers for the number of hours, minutes and seconds and which converts this into the total amount of seconds (for example 3 h 4 min 2 s = (3*3600 + 4*60 + 2)s = 11042 s). Print out the result.

7b. Now do the task of the convertion not in main but in a user-defined function.

7c. Write a program which reads in a total number of seconds and uses a user-defined function to convert the seconds to hours, minutes and left over seconds. Print out the result for hours, minutes and seconds. (So this is the task of 7a. almost backwards. For simplicity we do not determine hours.) For example 133 s = 2 min and 13 s.
Hint: You get the minutes by using integer division (division of two integers), e.g. (# minutes) = 133/60 = (total # seconds)/60 and you get the left over seconds with the modulo function %, e.g. (# left seconds) = 133 % 60 .

8. Traffic Model (Arrays) As you did for the in-class work 5., copy the file
~ kvollmay/sunhome/classes.dir/capstone_s2003.dir/unix_C++_intro.dir/road.data
into your working directory. The file contains 100 integers describing a road of 100 lattice sites at a certain time. "-1" mean an empty site and a number $>= 0$ means a car with the corresponding speed in mi/h.

8a. Either use your program for the in class work 5a. or copy the solution program
~ kvollmay/sunhome/classes.dir/capstone_s2003.dir/unix_C++_intro.dir/C++5a.cc
into your working directory. The program reads in the road data and determines how many cars are on this road. Change the program such that the road data are first read into an array and then this array is used to determine the number of cars.

8b. Now change the program such that the number of cars is determined via a function.

8c. (Tool For Fun) You can look at the street with

```
cat road.data | DynamicLattice -nx 100 -ny 1 -matrix
```

or with

```
DynamicLattice -nx 100 -ny 1 -matrix < road.data
```

9. Population Dynamics (if time)

One of the simplest model for population growth of one species (e.g. humans) is the so called "logistic map." You describe the population with one number N and say from year t to year $t+1$ that N changes:

$$N_{t+1} = aN_t(1 - N_t) \tag{1}$$

where a is a constant. If you rewrite the equation to $N_{t+1} = aN_t - aN_t * N_t$ you see that from year t to $t+1$ you get aN_t more people (because they are born) and you get $aN_t * N_t$ fewer people (because they die). Write a program which starts with $N(t=0) = 0.5$ and then you do 50 timesteps $t = 1, \ldots, 50$, so a loop which does Eq.(1) again and again. Print for each time step t and N_t (and use the `endl`).

Run your program for $a = 2.5$ (let's say your program is "logmap.out" and look at the output with

```
logmap.out | xgraph -m -nl
```

Then run your program with $a = 2.7, a = 3.2, a = 3.5$ and $a = 3.7$. Describe your results.