CMPE 250, Course Logistics

Fall 2016

Cagatay Yildiz

Computer Engineering Department
Bogazici University

September 2016

Outline

The Team

Teaching Assistant: Çağatay Yıldız, Çağatay Sönmez

- cagatay.yildiz1@boun.edu.tr, cagataysonmez@hotmail.com
- Room: BM 17-B (At the very corner of the first floor)
- Office Hours: ThTh 12, please send a mail beforehand or to setup time

Student Assistants: Rıza Özçelik, Abdullah Hanefi Önaldı

- rizaozcelik96@gmail.com

This Semester...

Things have changed in this semester. You will write your code

- in a virtual machine that we provide
- in Linux operating system
- in an IDE called CLion.

Last year you had Eclipse and all were already set for you. Consider this as your new setup.

- We would love to automatize your grading. This is why all of us, including those who are grading your projects, use the same setup.

Be Naughty?

What happens if you do not code in this setup? Well, we cannot guarantee your project grades. Though, you can

- write your code in your own OS
- move your code to the VM
- test your code in the VM
- finally submit if it does work

WARNING: No mercy if your code runs properly in your computer but not in the VM as it indicates there is something wrong in your code!

Outline

What is a Virtual Machine?

Simply, it is an operating system that can be used within your OS. Xubuntu is running in the machine and we have installed all you are going to need: git, GNU compiler, Java, CLion, etc. Feel free to install whatever software you like.

You can read wikipedia for more detail.

Installation Step-1: Download VM

In your own computer, go to http://79.123.177.215/cmpe250/CMPE250VM.ova and download *.ova* file (It is about 6.1 GB). Because the class is too crowded, server might be overloaded. Let me know if the link gets broken. You are actually recommended to get the VM from your friends who already downloaded BUT has not modify the OS.

Step-2: Download a Virtualization Software

Download a virtualization software such as VirtualBox, VMware or any other. I personally use VirtualBox and successfully imported the VM we provide.

If you download VirtualBox: Open the program, click *File* and *Import Appliance*. Then choose the *.ova* file you downloaded and finalize importing.

Step-3: Running

Start VM, enter *123456* as your password. You can change it later on.

Important Note: Depending on your BIOS settings, you may face with a problem while starting VM. In such a case, see http://www.sysprobs.com/disable-enable-virtualization-technology-bios. Even if your processor is not Intel, you are supposed to do something similar.

If your keyboard layout is in English, open the terminal and execute the following command:

```
setxkbmap tr
```

You will see how to execute commands in brief time.

Outline

Meet Xubuntu

If you successfully installed the VM, you will very probably see an OS that you have not seen before. It is called Xubuntu, which is just one flavor of the famous Ubuntu OS.

And what is Ubuntu? It is, according to wikipedia, *a Debian-based Linux operating system and distribution for personal computers, smartphones and network servers. ... It is based on free software and named after the Southern African philosophy of ubuntu (literally, 'human-ness'), which Canonical Ltd suggests can be loosely translated as "humanity to others" or "I am what I am because of who we all are".*

Meet Xubuntu

As you will see, it is just like other operating systems! You can use your keyboard and mouse to input, and you see the outputs on the screen! There is a file system installed that you can browse very much like in your Mac or PC.

BUT, what Xubuntu, or Unix-based operating systems, has is the thing called **terminal**.

Terminal = Command Line = Shell = Console

How to open terminal

You simply click the icon on the top-left and select **terminal emulator**.

Alternatively, press `Ctrl + Alt + T`

What is Terminal?

It is an interface in which you can type and execute text based commands. We use the terminal as it can be so handy when properly used.

Commands

You just write down some commands and receive a reply from the OS. Typical commands are

- **cd [some-directory]**: changes current directory
- **cd ..**: goes one directory up in the file system
- **pwd**: prints working directory
- **ls**: lists all the files in the current directory
- **ls -al**: lists all the files with a lot of details
- **mkdir [dir-name]**: makes a new directory
- **rm [file-name]**: removes a file
- **rm -rf [dir-name]**: removes a directory

Add `sudo` *to the beginning of commands if you need superuser permission, like* `sudo rm -rf [dir-name]`

Here is an excellent resource about terminal commands.

Outline

CLion

In my experience, CLion is the best IDE to write C++, even better than Eclipse. It

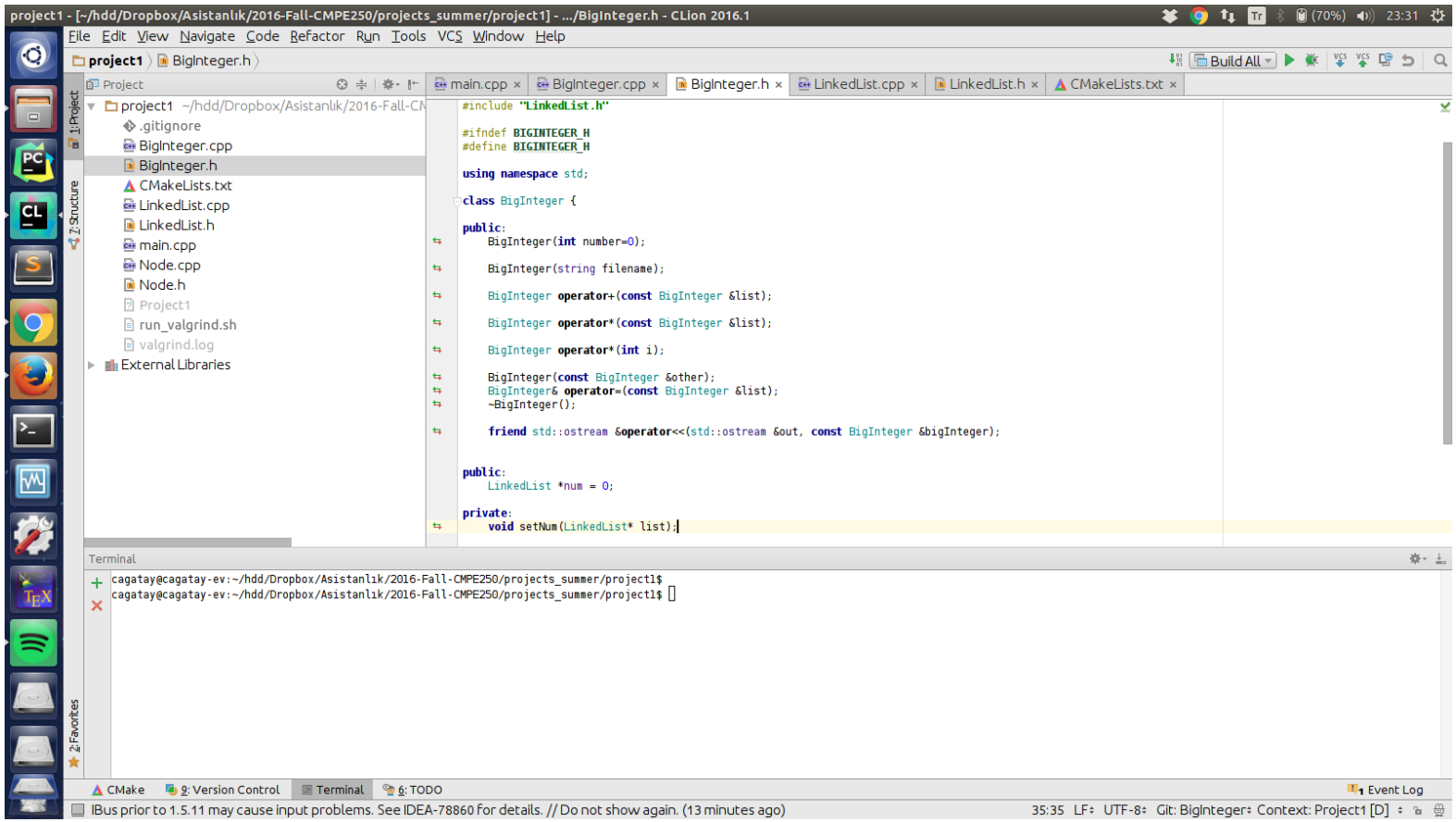
- auto-imports
- auto-completes
- compiles your code while you are writing and gives errors on-the-fly
- allows to create cmake files

and so on. But of course, you do not have to use CLion, if you prefer some other IDE.

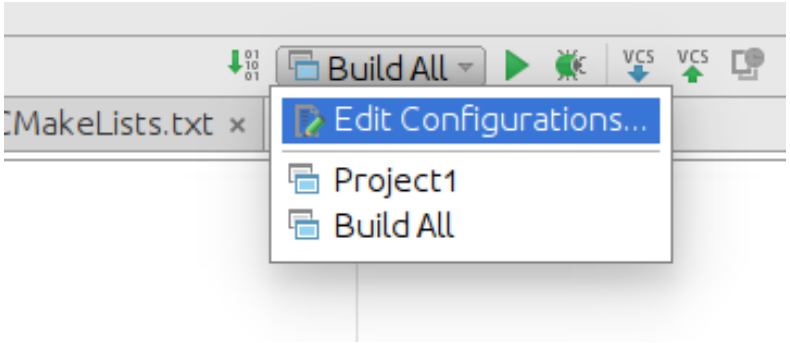
Click here to get a free JetBrains student license. See the shortcut to CLion on your Desktop. Run the IDE and register your account.

CLion

This is what CLion looks like:



Focus on the top-right of the screen:



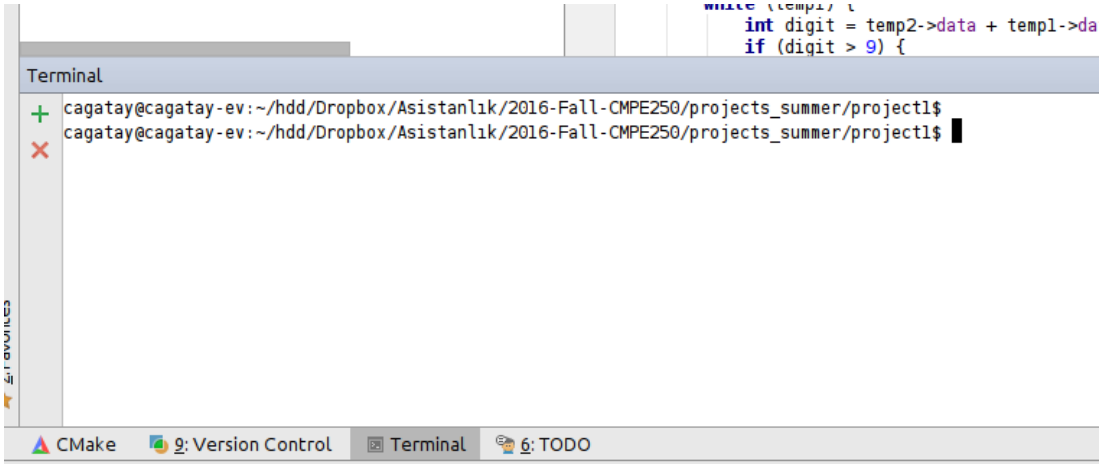
As you see, you can

- build
- run
- debug
- pick which executable to run (if you have multiple)

from here.

CLion

Focus on the lower-left corner of the screen:



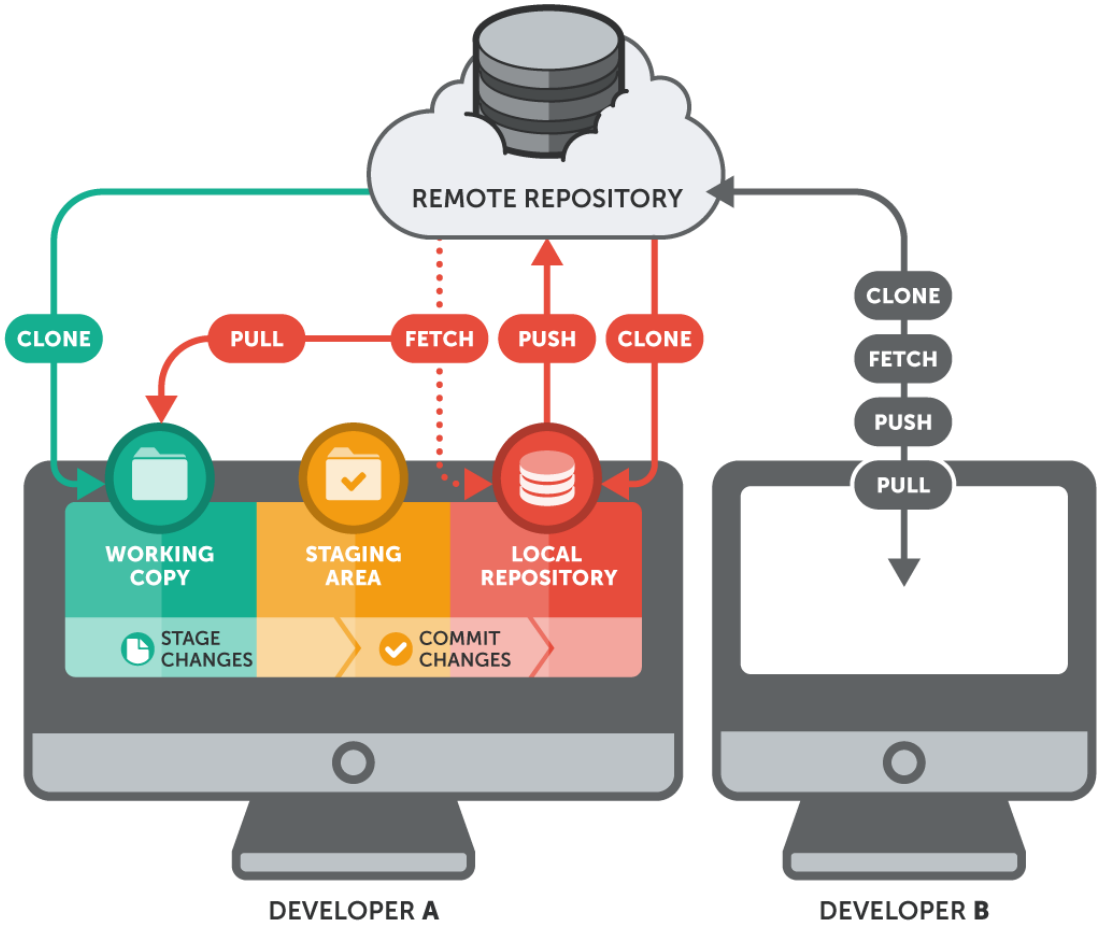
There, you have a number of views. My favorite among all is the terminal. You will see that you will make your submission via terminal - and you need to run the terminal commands in the same directory as your project files. So, do not ignore this window!

Outline

What is git?

- According to wikipedia, *git is a version control system that is used for software development and other version control tasks*.
- Git was created by Linus Torvalds in 2005 for development of the Linux kernel.
- Mainly used for large projects, where undoing your changes, simultaneous code development in different branches and versioning are vital.
- Why do we use git: To submit and collect projects!
  Git is already installed in the VM.
- **Click** for an excellent git cheat sheet.

What is git?



[figure taken from http://www.amygori.com/gdi-rdu-git-github/#/]

Some Terminology

This tutorial covers what git is. Below are very brief information:

- git projects usually have more than one contributors.
- There is a **remote** storage where all contributors submit codes.
- A new contributor should **clone** the repository to join the development. When one clones the repo, s/he becomes the owner of a **local** copy.
- Anyone that has cloned the repo should **commit** changes to your local repo and **push** to the **remote**.
- One should **pull** the remote repo to retrieve other's contributions.
- Applied to our context,
  - I will push the projects to your repo.
  - You will clone the project, work on it, commit and push.
  - I will then pull it again and grade.

Outline

6-Step Procedure

Below is a 6-step procedure for project submissions:

- Step 0: Configuring git
- Step 1: Retrieving Projects
- Step 2: Importing Projects into CLion
- Step 3: Committing/Pushing Your Changes
- Step 4: Testing Your Code
- Step 5: Checking Your Submission

You are expected the follow steps 0, 1, 3 and 4. If you want to use another IDE instead of CLion, just ignore step 2. If you are so sure that you submit your project without any problem, skip the last step.

Step 0: Configuring git

- As the zero'th step, set up your git account with the following commands:

```
git config -global user.email "asd@asdf.com"
git config -global user.name "Your Name"
```

- You can see the current settings with:

```
git config -global user.email
git config -global user.name
```

Do not forget! These commands are to be executed in command line = terminal.

Step 1: Retrieving Projects

- You then clone the project. Just like in Eclipse, you are supposed to work in a single workspace. There is actually already one. The path to this directory is `/home/student/workspace`
- So, open the terminal, change the current directory into `workspace` with the command "`cd workspace`". Alternatively, go to the folder, right click and click on "Open Terminal Here".
- Now, you are good to clone the project. Use the following command:

```
git clone
http://[stu-num]:[password]@178.62.229.16/cmpe250/fall2016/[stu-num]/project[project-num]
```

This is actually one liner but does not fit to the page.

Step 2: Importing Projects into CLion

- Open up CLion, click on *File* from the menu and select *Import Project*.
- The project you imported should be ready to go. Now, switch to the *Terminal* tab, located below the text editor. Execute
  `git status`
  to make sure that you cloned the repo.

Step 3: Committing/Pushing Your Changes

- As you make a progress, push your changes so that you make sure that your changes will not get lost (even if you delete the local repo).
- To push the changes, open the terminal and go to the project folder. Alternatively, you can use the terminal in CLion, which is already in the correct folder. Then execute the following commands one by one, with the given order.

```
git status
git add -A
git commit -am "some-message"
git push
```

to see your changes, add all your changes to the stashed area (where files to be committed are recorded), commit to local repo and push to the remote, respectively

Step 4: Testing Your Code

- We want you to test your code through command line, not from within your IDE because we will test your code using terminal.
- Again, open the terminal and change to project directory. Then run the following commands one by one:

  ```
  cmake CMakeLists.txt
  make
  ```

  The first command creates a `Makefile` and the second one executes the `Makefile` = compiles and links the code.
- If everything goes okay, your executable should be ready. Its name would be *project[id]*, such as *project1* or *project4*. Finally execute

  ```
  ./project[id] [some-arguments-if-there-are-any]
  ```

  command and see the output.

Step 5: Checking Your Submission

To check whether you submitted your code or the changes are still in your local repo, follow the steps below:

- Open the terminal in any directory other than the one that contains your project
- Clone the project's repo with the command given before.
- The files that you are looking at are the ones that we will grade.
- If you are done, just delete the directory so that you do not get confused later. If there is a problem with your submission, find a way to push your code to the remote. Please do not submit your code via e-mail.

Warnings

- You are always allowed to work on multiple repos. That is to say, you can clone your repo anytime or in any computer you want and work on your project. Crucial thing is that you should not forget to push your code when you are done and to pull the code at the beginning.
- If you create new files, you need to update `CMakeLists.txt` file. A `CMakeLists.txt` file looks like the following:

```
cmake_minimum_required(VERSION 2.8)
project(project1)
set(CMAKE_CXX_FLAGS "$CMAKE_CXX_FLAGS -std=c++11")
set(SOURCE_FILES main.cpp Node.h Node.cpp)
add_executable(project1 ${SOURCE_FILES})
```

What you need to do is just append the names of new files to the fourth line.

Warnings

- Your code will be tested automatically. This is why you need to follow the input and output format strictly.
- Last semester, many students did not produce output files but instead they printed results to the console. This semester, you will lose credit in such cases.
- There will be several test cases and associated timeout values. This is to say, if your program does not terminate before the timer expires, you will fail the corresponding test case. So, you must write your code as efficiently as possible. Make sure that you find the most efficient algorithm (the one with the smallest complexity) and do not perform unnecessary operations. Printing stuff to the console consumes a lot of time, please delete all print statements while submitting.

Warnings

- Do not spend a great deal of time on precision issue. It does not really matter if you print 4 or 6 digits after the decimal of a floating point number. Not printing an integer in a double format or vice versa is good enough.
- Do not use `float` for storing floating point numbers, always use `double`.
- In cases where `int` may be insufficient, consider using `long` or `long long`.
- Always use absolute paths. If you are to input a file, the argument should not be **case1.txt** but it should include the whole path, such as **/home/student/workspace/project1/testcases/case1.txt**