Part 0 (Set up)

Objective

- overview
- c++ command-line interface basics
- c++ environment setup (compiler and text editor)
- c++ hello world program (how to compile and run code)

Goal

- Know what, programming, source code, text editor, compiler and c++ is
- get comfortable on the terminal, capable of Creating, Reading, Updating and Deleting folders and files
- download, install and set up a c++ compiler
- compile and execute the first program in c++

Overview

This is a book that introduces programming concepts, using c++ as the means of communicating these great ideas to the computer. It will aid you in teaching yourself programming in c++ and also learning about computer programming and software engineering. I will try as much as I can to do away with unnecessary jargon and if I can not, I will try and paint it differently. This is my sole contribution to FOSS and the little guide I would use to teach anyone programming. This material stands to be questioned anyway.

At the end of sequentially, chapter after chapter, reading this material and practising with the projects, I believe that, at least, one would be kind of able, mentally and technically, write decent analyze problems, design and code a solution in c++. This material does not really require any prerequisite as such, though willing to learn and understand the hard way will be necessary and any prior knowledge is a plus.

So, what is c++? What is programming? Why should one learn to program? and why c++? Well, I shall leave you to answer those questions yourself, refer to cpp, Wikipedia.

We can create a lot of software just in C++, and such examples are compilers and interpreters, text editors and IDE's, databases and database management systems, programming languages, machine learning and AI systems, faster algorithms and efficient data structures and a whole lot but first, you have to know some C++ and more C++ and other tools.

C++ Command Line Interface Basics

Well, I said no prerequisites because I shall be filling in some of these holes and I shall direct you to a resource to learn more. Is that not awesome? Go to your applications and search for Terminal. I am using Ubuntu, and `Ctrl + T`, pops up my terminal. The terminal is the same as the shell and command line. For Window users, perhaps, using power-shell or downloading bash (my terminal is bash) will do but try power-shell in administrator mode first.

All that I want you to learn here is, how to create, read, update and delete, folders and files and their contents. This is very simple actually because it is the first thing most command-line/terminal based tutorials teach. You must know how to use the terminal, it will save you. Just know when it is appropriate to use the terminal.

CRUD Directories

When you open the terminal, by default, it will open in the directory (folder) - home of the current user. This is how mine looks like,

`otumian@bitwised-monkey:~/`

and when I enter the command, pwd, I see, `/home/otumian`, which tells me where I am in the terminal currently. `pwd` - means, current working directory.

Enter the command, `mkdir`, followed by a directory name, example, code and this will create a folder in the current working directory called code.

Example

```
mkdir code
mkdir shoes
```

You can create multiple directories at a go by just spacing out the name of the folders as in, `mkdir` code shoes horses.

If the name of your directory has space in it, it will be better you put the name of the directory into quotes, as in, `mkdir "cpp codes"`, else `mkdir cpp codes`, will create `cpp` and `codes` as folders.

If you remember the earlier command, `pwd`, which tells what the current directory is? When you enter `pwd` right now, it tells that you are not in any of the directories you may have just created but rather tells of the directory of the current user.

Use the command, `ls`, to see what's in the current working directory. This is what I see when I do, `ls` in my terminal.

```
 code         Downloads           Public      Videos
'cpp codes'   Music               shoes       Visual_Paradigm_CE_16.1
 Desktop      package-lock.json    snap        WorkStation
 Documents    Pictures             Templates   yEd
```

The `ls` command with the `-a` flag, displays what is in the current directory including hidden files. This is what I see when do, `ls -a`.

```
.                 .grip               shoes
 ..               .hello.py.swp       snap
 .audacity-data   .install4j          .ssh
 .bash_history    .java               .ssr
 .bash_logout     .local              .sudo_as_admin_successful
 .bashrc          .mozilla            .swp
 .cache           Music               Templates
 code             .npm                .thunderbird
 .config          package-lock.json   .tooling
'cpp codes'       Pictures            Videos
 Desktop          .pki                Visual_Paradigm_CE_16.1
 Documents        .profile            .vscode
 Downloads        Public              .wget-hsts
 .falcon          .pylint.d           WorkStation
 .gforth-history  .python_history     .yEd
 .gitconfig       .sample.py.swp      yEd
 .gnupg           .selected_editor
```

All those starting with a dot, `.`, are hidden files or directory. Compare the output of the `ls` and `ls -a` command. There is also, a `.` and `..` (single and double dot) which represents the current directory and the parent directory of the current directory, the double dot is sometimes said to be the previous directory.

So the command, `ls`, will output the same result as `ls .` (ls dot).

To see what is in a directory, in the current working directory, use the `ls` command followed by the name of the directory. You can also do this for multiple directories.

Example:

`ls code/`

So far we have been able to create and read what is in a directory. It is about time we change into a different directory. I created the code directory and now I want to move into or change the current working directory (in)to `code` directory. This is done using the `cd` command followed by the name of the directory, as in, `cd code/`. So now if I enter the `pwd` command, I will have, `/home/otumian/code`, displayed, which means that I am currently in the code directory.

To go back to the parent directory of code or the previous directory I was in, I would do, `cd` followed by double dots, `..` with no space between the dots, as in `cd ..`. The `pwd` command will display, `/home/otumian`

If you have been entering the commands as I had, then you'd notices that the terminal is now cluttered with commands and outputs. Enter the `clear` command to clear the terminal.

So far I have created, `code`, `shoes` and `"cpp code"` as directories in the current working directory. So now let's move `code` and `shoes` into `"cpp codes"` and we can easily do that, using the `mv` command followed by the directory then the destination directory. Something like, `mv this_directory that_directory`, which you may read as move `this_directory` into `that_directory`. Generally, we have it has, `mv target destination`.

Now should move code directory into `"cpp codes"`. Enter the command, `mv code "cpp code"`, then follow this with the `ls` command. This is what I see after the `ls` command.

```
'cpp codes'   Music              shoes       Visual_Paradigm_CE_16.1
 Desktop      package-lock.json  snap        WorkStation
 Documents    Pictures           Templates   yEd
 Downloads    Public             Videos
```

`code` directory has been moved you see. Now you should move `shoes` into `"cpp codes"`.

Another you may write `"cpp codes"` is cpp\ code, where backslash followed by a space is the space between cpp and codes.

With the same `mv` command, for moving directories into another, we can rename a directory. Since we have moved the code and shoes into `cpp\ codes`, I believe it is about time we rename `"cpp codes"` by removing the space. This can be done by, `mv "cpp codes" cppcodes`. After this command followed by the `ls` command, this what my terminal displays.

```
 cppcodes          Public
Desktop            snap
Documents          Templates
Downloads          Videos
Music              Visual_Paradigm_CE_16.1
package-lock.json  WorkStation
Pictures           yEd
```

Now it's obvious that the directory has been renamed successfully. Know that when you move a directory into another that does not exist, will rename the original directory.

So we have been working hard and we fear that some dumb cat, which has access to our pc and its directories, would delete some of our directories and files that we need. So we want to create a backup of our directories. What we would do is make a copy of the original, right? Well, this is simple on the command line. Since we are dealing with a directory we would be interested in its contents as well right? So when copying, we have to get into the internal directories, thus copy recursively.

We do this as, `cp -r dir_name dir_dup_name` or `cp -r dir_name path_to_another_dir`.

I will make a copy of `cppcodes` and call it `cppcode_bkp` with the command, `cp -r cppcodes cppcodes_bkp`, followed by `ls`.

```
 cppcodes           Public
cppcodes_bkp       snap
Desktop            Templates
Documents          Videos
Downloads          Visual_Paradigm_CE_16.1
Music              WorkStation
package-lock.json  yEd
Pictures
```

`pwd` command tell me that I am in, `/home/otumian`, thus I made a copy of `cppcodes` into `/home/otumian`

What if I want to put the copy somewhere else. Create a folder called `cpp` and within `cpp` create another called `codes`. We shall put the duplicate into the last directory created. At this moment, `pwd` would tell me that I am in, `/home/otumian/cpp` which contains the `code` directory, after the `ls` command. I will enter `cd ..` to move back into the `otumian` directory then issue the duplicate or back up using the command, `cp -r cppcodes cpp/codes/`, where `cpp/codes/` is the path that I want to copy the `cppcodes` to. After the copying command, let us verify if there has been a copy of `cppcodes` into `cpp/codes` by doing ls `cpp/codes` and this is what displays on my terminal.

```
Cppcodes
```

What is the `pwd`? Mine is, `/home/otumian`.

I am kind of fed up with the creation of folders, now I want to delete something. Being fanatical as the devil. The command is simply, `rmdir dir_name ...`, which is remove directories, the ellipse indicates that we may remove multiple directories at a go.

Lets remove `code/` directory in `cppcodes/` directory.

First move into `cppcodes` and list (display/read) the content of `cppcodes/`. You should know what or how to do this by now, using the `ls` command. This is what I see in my terminal.

```
 code
shoes
```

Now we know `code` is in `cppcodes`, but is it empty? Check with the `ls` command if `code/` is an empty directory. By the command, `rmdir code/`, followed by `ls`. I see,

```
 shoes
```

Now remove `shoes` as well, now `cppcodes` will be empty? right?

Move to the previous directory or move to the parent directory or move up the directory implies, do `cd ..`, remember? At this point, I am in `/home/otumian`. Now there is a small problem, though I do not mind moving into a directory and deleting directories and so on, the `rmdir` command deletes only empty directories. Now, what if the directory is not empty?

Try it and see, make directory `fish/`, and within `fish/` make `tuna/` or after making `fish/`, make `fish/tuna/` without moving into `fish/`. Now remove `fish`. What did you notice?

Well, let me show you a very simple approach but be careful.

The command, `rm -r dir0/` deletes the content of `dir0` and `dir0` itself. The command, `rm -r dir0/dir1/dir2/`, deletes `dir2` and all it's content.

So let us consider that there is `dir0` which contains `dir1` and `dir1` contains `dir2` and `dir2` contains `dir3`. Something like, `dir0/dir1/dir2/dir3`.

From the other two commands, it dictates that `rm -f dir_path/` removes the content of the last directory and the directory itself. `rm -r dir0/` will delete all the content of `dir0`, in the reverse order, `dir3`, `dir2`, `dir1` and `dir0` will be removed. `rm -r dir0/dir1` will remove, in the reverse order, `dir3`, `dir2` and `dir1`.

I have had enough, now disintegrate your folders! (Delete).

With the command, `rm -r cpp cppcodes cppcodes_bkp`, I will delete all the contents of the directories and the directory itself will be removed. Yes!!

This is a gist of the commands we have learnt so far

```
pwd, current working directory
ls, list the content of the pwd
ls -a, including hidden files
mkdir dir_name(s), create/make directories where the dir_name(s) is (are) the name(s) of the directory(ies)
cd dir_name, change into another directory that is in the pwd
cd path_to_dir, change in another that may not necessarily be in the pwd
cd .., moves up the directory (to the parent directory of the pwd)
ls path_to_dir, list the content of the directory at path_to_dir
ls -a path_to_dir, list the content of the directory at path_to_dir, including hidden files
clear, clears the screen of the terminal
mv dir1 dir2, renames a directory called dir1 to dir2 in the pwd
mv dir1 dir_path/, moves a directory called dir1 to another path but does not rename it
cp -r dir another_dir_name, copies the content of dir into another_dir_name
cp -r dir dir_path/, duplicate dir into another path
rmdir dir(s), removes empty dir(s)
rm -r dir(s), removes dirs and their contents
```

You better take a break, it is not necessarily how much you learn by sitting for long hours - even in coding (practising) for exhaustive long hours can cause abnormalities in your code. !!; Try one hour study, 30 minutes or more break. Make sure you have enough break and nap if possible.

CRUD Files

Remember this, that ..., sorry I kind of forgot what I wanted to say. I guess I wanted to define what a file is but I shall leave that to you, oh great student!

Well, we did create, read, update and delete folders and their content, now we shall move to files, the sole containers of our source codes. The program we write is a source code.

Coding is awesome. You should know how to deal with the opening of the terminal by now. Open it, I have got some cool tricks to show you, you will like it, assuming you like ice cream.

Anyway, let us create some files.

There are several ways to create files from the terminal and the one we would be using is `touch`, the `touch` command. The `touch` command, creates an empty file, in `pwd`. The name of the file or files follows after the touch command.

So my `pwd` says that I am in the `/home/otumian` directory. If you are not in the `/home/<user>` directory then, enter the command, `cd`, and hit the enter key, then followed by the `pwd` command. Verify if you are in the user's directory.

Let us create a file, called, `helloworld.cpp`, using the command, `touch helloworld.cpp`. I don't know the text editors that you have or use. I have `nano`, `vim`, `vscode` and `gedit`, I travel lightly. I am sure you may have others of your own. Now I shall open `helloworld.cpp` with `vscode`, from the terminal, using the command, `code helloworld.cpp` (code is the name of vscode from the terminal) and when that opens, I will add the following lines of code.

```
// helloworld.cpp
#include <iostream>

using namespace std;

int main() {
    cout << "Hello world!!" << endl;

    return 0;
}
```

Save this file and we are done. Well, if you can not open yours from the command line, no worries. We will do nothing with the file we just created. All we are to take home is how to open a file from the terminal. More generally, you can use the command, `<name of editor> filename`. Also know that you can create multiple files with the `touch` command, as in `touch file1.cpp file2.cpp` and so on.

There is the `cat` command also, unlike the `touch` command, which overwrites the content of a file if the file already exists. Let me show you how we would have used the `cat` command in place of the `touch` command.

With the command, `cat > helloworld.cpp`, we create a file name helloworld.cpp and also ready to write into. When you are done hit, `Ctrl+C` to exit. Now, the best part is this, so not to overwrite the content of a file, but to append the new content to the old, we best do, `cat >> helloworld.cpp`. The double greater than sign means appends new content to the old one.

Create some files and feel free. Of course, we will delete them, no worries.

Do you remember the `cp` command? Of course, you do and if you don't, well ... I will use it here with the `mv` and `rm` command. No sweat but scroll up and see how we used them.

The `cp` command is used to copy files, as in `cp path_to_file another_path_to_copy_to`. The reason is so that, you can also assume you can `cp` from some folder somewhere and duplicate it somewhere. The same works for `mv` and `rm`.

Please refer to the above. Know that `rmdir` is for directories and `rm` is for files but when we want to show our technological-geeky might, we use `rm` followed by some flags and we delete not important files and folders.

Let me just show you something cool. Now I assume you are in `/home/<user>`

`touch somefile`, creates a file called `somefile` in the `pwd`. `cp somefile whatfile`, copies `somefile` into the `pwd` and names the duplicate, `whatfile`. `cp somefile Desktop/`, copies `somefile` onto the `Desktop` but the duplicate may have the same name as the original. `cp somefile Desktop/whatfile`, copies `somefile` onto the Desktop and names the duplicate, `whatfile`.

So we have 4 files now, `somefile`, `whatfile`, `Desktop/somefile` and `Desktop/whatfile`

`mv somefile shoefile`, renames `somefile` in the `pwd` to `shoefile` `mv shoefile Desktop/`, moves `shoefile` to the `Desktop`

Now we should delete some of the files `rm whatfile Desktop/shoefile Desktop/somefile Desktop/whatfile`

And all those files will sadly be removed.

So, command-line ends here, it is now, that the torch has been passed to you, read more at, unix-workbench

This is a gist of the commands we have learnt so far

```
touch file(s), creates files
editor_name file, opens the file in the editor
cat > file, creates a file and overwrites its content. opens the file for writing
cat >> file, opens the file for writing but appending to the end
echo data > file, just like the cat command, writes data into file
echo data >> file, appends data to the end of file
cp file1 file2, duplicates file1 as file2
cp file another_path, duplicates file to another_path, but the file name remains the same
mv file1 file2, renames file1 to file2
mv file dir_path/, moves file from pwd to dir_path
rm file(s), removes files(s)
```

C++ Environment Setup (Compiler And Text Editor)

I do not speak computer and I am sure you do not also speak computer. How then do we communicate our ideas to the computer? We simply need an interface, a translator. We need a translator that may understand and speak our language, C++. One of those guys is the GNU C++ compiler.

Linux and Mac may already have c++ compiler installed. Verify by entering the command, `c++ -v` or `g++ -v` or `clang++ -v` or `clang++-10 -v`, onto the terminal and hit enter. This is mine, showing I have a c++ compiler.

```
otumian@bitwised-monkey:~/$ c++ --version
c++ (Ubuntu 9.3.0-10ubuntu2) 9.3.0
Copyright (C) 2019 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

If there is an error, install the c++ compiler with the command, `sudo apt install build-essential`, on Ubuntu. You have to enter your password.

On Mac download the latest Xcode from the apple-website, with a couple of scrolls down or do, `brew install gcc`, on the command prompt.

For Window, Download it from mingw-web, run the executable, after the extraction you can delete the executable. Now, add the compiler to the environment variable path by:

1. first copy the path to the bin folder of the compiler downloaded
2. On the Start menu, right-click Computer.
3. On the context menu, click Properties.
4. In the System dialogue box, click Advanced system settings.
5. On the Advanced tab of the System Properties dialogue box, click Environment Variables.
6. In the System Variables box of the Environment Variables dialogue box, scroll to Path and select it.
7. Click the lower of the two Edit buttons in the dialogue box.
8. In the Edit System Variable dialogue box, scroll to the end of the string in the Variable value box and add a semicolon (;).
9. Add/paste the new path, after the semicolon.
10. Click OK in three successive dialogue boxes, and then close the System dialogue box.

More information on [how to add path to environmental variable path path-web.

After the installation, enter the `c++ --version` or `g++ --version`, on the terminal.

So one software down, another to go. We now need a text editor for our source codes. The default text editors such as notepad, gedit and Xcode, will do the job right. I mean in the beginning but in the future, it might be kind of a hindrance so we may need another fully-fledged and capable text editor. There are many editors to choose from such as vscode, atom, [sublime][sublime-web], bracket, and what I may not have even heard of. I use vscode so I'd say download vscode. You are free to do as you please.

You may also get an IDE, which is a bundled software, made up of a text editor and a translator, awesome right? It won't kill to use an IDE but you have to see what lies behind the IDE. I won't use an IDE.

C++ Hello World Program (How To Compile And Run Code)

So now I will not be writing command as I did in the command-line/Terminal section.

I trust your wits now. I will just tell of what to be done and I shall assume that it will be done.

We have already done this, but we shall do it again, for what it is worth.

Note, at this time, I encourage you to create folders for each chapter and if the chapter or work demands a different folder for a particular chapter, then do it. It is better that way.

First, create a folder to hold all the codes for this project, one from this material and the best part from you. Now inside this folder create yet another folder for this hello world project. Be happy, all you have learnt up to now is going to pay off.

`cd` into the helloworld directory and create a file, helloworld.cpp, and open the file (You see I did not dictate how you should open the file?). Add these lines of code in your text editor.

```
// helloworld.cpp - project
#include <iostream>

using namespace std;

int main() {
    cout << "Hello world!!" << endl;
    return 0;
}
```

Save the file and go to the terminal now. We have to compile this code.

The simple way to compile code is to call the compiler, followed by the name of the file to be compiled, as in, `compiler_name path_to_file_to_be_compiled`, and you are done. If it was successful, you'd see a new file, `a.out` r `a.exe` on Windows. All you have to do now is run `a.out` as `./a.out` from the terminal.

Now the name of the compiler is `c++`, `g++` or `clang++` and the file is `helloworld.cpp`. `g++ helloworld.cpp -o app` will compile `helloworld.cpp` and save the binary (output) as `app`. The `-o app` indicates that the binary should be named, `app`. Then run `./app` to execute the binary. Make sure you are in the folder that contains the `helloworld.cpp` file. Finally, compile and execute the source code. So now you know what a source code is?

Exercises for part 0

Create File and Folder

- create a directory, `cmdexercise` in `/home/username/`
- create a file, `exercise` in `cmdexercise`

Rename File and Folder

- rename `exercise` to `Exercise`
- rename `cmdexercise` to `cmdExercise`
- in `cmdExercise`, create a folder, `newcmdexercise`

Copy and Rename

- duplicate `Exercise` as `Exercise.bkp` (this is like a backup so you copy and give a new name to the copy)
- move `Exercise.bak` into `newcmdexercise`
- duplicate `newcmdexercise` as `newcmdexercise.bkp`

Delete

- delete `Exercise`
- delete `newcmdexercise`
- delete `cmdExercise`

project 0

In this project, you will create a folder, create the *helloword* application but this time add your name, compile and execute the binary then delete the binary after the application has worked as expected. This application will create `Hello world <YourName>!!!` on the screen, followed by a new line. The output will be similar to `Hello world, Dennis Ritchie!!!` if *Dennis Ritchie* is your name.

- create a folder, `helloworldcpp`
- create a file, `helloworld.cpp`, in `helloworldcpp`
- Now, write a hello world program in cpp in `helloworld.cpp` file
- compile the cpp file as, `c++ helloworld.cpp -o app`
- run the cpp binary as, `./app`
- finally, delete `app`, the cpp binary

#