CSU NUPAC Tutorials
2019
Gabriel Palacino
Let’s review Lxplus for a moment

I have an alias for ssh lxplus.cern.ch

This is the path to my home area. All users have a home area in /afs/cern.ch/user

All users have a 10 GB quota in their home directory. You can monitor your quota using `fs lq`

Reaching 100% of your quota may cause some problems: impossibility to log in and run jobs, and slow down of the system.
Let’s review Lxplus for a moment

Users also have a Work area. I have defined an environment variable so that I don’t have to type the full path every time.

This is the path to my Work area. All users have a home area in `/afs/cern.ch/work`.

All users have a 100 GB quota in their work directory.

Ideally, you should keep all your code and big data files in your Work area. This way you won’t run into storage space issues.

• You also have 1 TB of storage available using your CERNbox.
  ‣ Activate it by going to `cernbox.cern.ch`
Let’s configure our shell

• Just to make sure that we all have the same system configuration let’s do a few things
  ‣ Identify which type of shell we are using
  ‣ Change if necessary
  ‣ Create some useful aliases

You can move to your home directory using the relative path identifier `~/`

Use `echo $0` to print out the name of your shell

Use `ls` and `grep` to find the hidden rc (run command) script

• If your shell is not zsh do the following
  ‣ Go to `www.cern.ch/it-div` —> Account Management —> My Accounts —> Services… —> LXPLUS and Linux —> Settings —> Unix Shell
  ‣ Choose `/bin/zsh` and save selection
Let’s configure our shell

- The script `.zshrc` is run automatically at the start of every session (every time you log in)
  - You should include here all the environment variables and aliases that you use frequently
  - You can also play with it to configure how your terminal looks like
  - Create some useful aliases

Use `#` to comment out lines in shell script that you don’t want to be executed

Add colors to your terminal and configure the information displayed in the prompt

_aliases and variables I want set up all the time_
Now let’s get to programming

Most of the content is stolen from Sam Meehan’s 2018 CSU NUPAC tutorial notes

- “Programming” - initially in c++
  - 1 - Going from knowing “cd, cp …” to running a program
  - 2 - Storage and operations - “doing stuff”
  - 3 - Factorizing thinking - Functions
  - 4 - Designing useful black boxes - Classes
  - 5 - Flow control - ifs and Loops
  - 6 - Preventing “Oh Sh$*!” moments - Version control and GitLab
  - 7 - Advanced Compilation - makefiles
  - 8 - Advanced grab-bag
  - 9 - The other 90% of your time - Debugging
What is “programming”?

Three ways to answer this question

- [Abstract] A way to solve questions based on creating a well defined procedure to arrive at a numerical answer
  - How many prime numbers exist between 0 and 100?
- [Abstract] A way to take a set of data, perform operations and manipulations on that data, and produce another set of data
  - Asking a bunch of people who they want to with the election ➔ determining on a state-by-state basis who will win each state
- [Concrete] Creating a human readable (text) file, converting this to a machine readable set of processes, and executing this set of processes with some number of inputs
  - This is done in front of the computer and is what you will do most of your time
Programming Paradigms

- **Functional**: Same input gives the same output
  - Pass data through “filters” that are independent of the data itself
  - Build up more complex manipulations via composition of simple operations \( \Rightarrow f( g(x) ) \)
Programming Paradigms

- **Object-Oriented**: Same input *might give* different output
  - Store the data and methods in a single “structure” (objects) and manipulate these structures
  - Build more complex manipulations by linking objects together
Compilation

- Basic concept: turn human readable text into a computer executable software
- What will you be using compilation for to begin with?
  - [1] Producing something you can run
  - [2] Getting hints at errors you have made while writing your program - “debugging”
Compilation

- In reality: there are two steps
  - Compiling: translation of human readable text (source code) into machine language
    - These look like nonsense and are not executable
  - Linking: bringing together many object files into a single executable
    - This produces the “a.out” executable - looks like nonsense but now its executable

Create object file from source code:
$ g++ -c hellocern.cxx
Compilation

- In reality: there are two steps
  - Compiling: translation of human readable text (source code) into machine language
    - These look like nonsense and are not executable
  - Linking: bringing together many object files into a single executable
    - This produces the “a.out” executable - looks like nonsense but now it's executable

- More correct term: “Build” = Compiling + Linking

Link object file(s) into executable:
$ g++ -o hellocern.exe hellocern.o

Run the executable command:
$ ./hellocern.exe
Language

- Take away message: communication
- Writing programs requires precise language
  - Learn a new way to speak/think
- I *hope* that you will be comfortable with all/most of these concepts
Helping Yourself ("the hoo ra ra")

- Reminder: I am not a programming guru / I’m a physicist
  - Requires me to “parameterize my ignorance” to make scientific progress
- To go from 0 to 60 you must take the lead
  - student ➔ self-educator ➔ researcher
- Copying code?: “I approximately don’t care”
  - Use your academic judgement about using the work of others
  - Anything meaningful in research will not be found on the internet
- You will get out of this what you put into it
  - The more preparation you do NOW
  - ... the more enjoyable and fruitful your summer
Helping Yourself

- Many wonderful resources exist for self-education in programming
  - **Google**: “how to (what is) &lt;FILL IN THE BLANK&gt; in c++”
    - &lt;FILL IN THE BLANK&gt; requires the correct language
    - The first hit will commonly be cplusplus.com, StackOverflow
  - **StackOverflow**: Forum online to answer questions
    - If you think you are the first one to see an error, you probably aren’t
  - **cplusplus.com**: The **OFFICIAL** reference
    - Hugely helpful number of examples
    - Much of what I will say comes from here
  - **YouTube**: Nice people make amazing video lectures on this stuff
    - E.g. I learned the concept of “inheritance” from saurabhschool just one year ago
Let’s do some programming

- A common C++ program includes
  - Header files. Usually .h
  - A `main` function that returns an `int`
  - Functions that will be executed in the `main` function

```cpp
#include <iostream>

using namespace std;

int main()

    string name;
    cin >> name;
    cout << "Hello there. What's your name?" << endl;
    cout << "Hi " << name << ". Welcome to the CSU NUPAC ATLAS tutorials!" << endl;
    return 0;
```
Let’s do some programming

Comments in C++ start with //

```cpp
#include <iostream>
using namespace std;

int main()
{
    string name;
    cout << "Hello there. What's your name?" << endl;
    cin >> name;
    cout << "Hi " << name << ". Welcome to the CSU NUPAC ATLAS tutorials!" << endl;
    return 0;
}
```

- `#include <iostream>` declares the use of the std library headers
- `using namespace std;` declares the use of the std library headers
- `int main()` defines the main function
- `string name;` declares a variable containing text
- `cout << ...` performs some operations

`iostream` provides input and output services
Compiling and running the code

• We will use **g++** to compile our code
  ‣ GNU C++ compiler-driver
  ‣ Automatically links standard C++ libraries. Other compiler-drivers don’t do this

Compilation command: `g++ -o <output name> <source code name>`

Execute output

This is the result of your program
Let’s get a bit more sophisticated

- I have created now a header file which will include all the necessary libraries and where all the functions (except) main will be declared.
- I have implemented two functions one for for retrieving the your name and another for greeting me.
- Although it looks like more code had to be written, this is a better way to write code. Factorization leads to more efficient and understandable code.

Header file

```c
// Here I declare my functions
std::string returnName()
{
    std::cin >> name;
    return name;
}

void greet()
{
    cout << "What’s your name?";<br />
    cout << "Welcome to the CSU HEPAC ATLAS tutorials!";
    return;
}
```

Source file

```c
// This is a basic C++ program
#include "HelloWorld.h"

// Function to return name
string returnName()
{
    string name="";
    cin >> name;
    return name;
}

// Function to greet me
void greet()
{
    cout << "What’s your name?";
    cout << "Welcome to the CSU HEPAC ATLAS tutorials!";
    return;
}
```
Compilation again

- Compilation is done in the same way as before
- Now we have one additional header file in the directory
- Running the code produces the same output.

```
ls -lrth
-rw-r--r-- 1 palacino 94 Jan 29 10:47 HelloWorld.h~
-rw-r--r-- 1 palacino 441 Jan 29 10:53 HelloWorld.cxx~
-rwxr-xr-x 1 palacino 11K Jan 29 10:53 Hello
-rw-r--r-- 1 palacino 94 Jan 29 10:55 HelloWorld.h
-rw-r--r-- 1 palacino 439 Jan 29 10:56 HelloWorld.cxx
```

```
g++ -o HelloWorld HelloWorld.cxx
```

```
What's your name?
```

```
Gabriel
```

```
Hi Gabriel. Welcome to the CSU NUPAC ATLAS tutorials!
```

```
ls /afs/cern.ch/work/p/palacino/CSUTutorial/2019/Programming
```

```
g++ -o HelloWorld HelloWorld.cxx
```

```
./Hello
```

```
Hi Gabriel. Welcome to the CSU NUPAC ATLAS tutorials!
```