Lecture 4
Using the Unix Shell and Introduction to C++

Instructor: Ashley Gannon

ISC3313 Fall 2021

Unix shell and Basic Commands

Unix shell and Basic Commands   Redirecting output   Programming languages   Overview of C++   Hello World Program   Running First Program

00000000   000   0000   00000   0000   00000000

Using the Unix Shell

When we open the Bash shell (Windows), or the iTerm (Mac), we are first presented with a prompt:

Here we can enter many commands recognized by the Shell language.

Basic Commands

Here are some of the most basic commands:

- `ls`: list the contents of the current directory
- `pwd`: display the current path from root
- `cd`: change directory
- `cp`: copy a file/directory
- `rm`: delete (or remove) a file/directory
- `mv`: move a file/directory to a different name/path
- `touch`: create (or make) a file
- `mkdir`: create (or make) a directory

Special symbols

There are some special symbols that a user just *must* be familiar with:

- `.`: the period stands for 'current directory'
- `..`: two periods stand for 'previous directory'
- `/`: forward slash stands for 'root directory'
- `~`: tilde stands for 'home directory'

Beyond Navigation of Directories

We can interact with the Shell in many ways. We can define variables, view files, open a program for file editing, or do simple math right in our terminal! Here are some simple commands:

- `echo`: display argument to string or file
- `export`: create an environment variable
- `env`: list all environment variables
- `cat`: displays entire contents of file onto screen
- `find`: locates files and directories
- `grep`: searches for patterns within files

Command: cat

The cat command is widely used. It has three main uses: displaying files, concatenating files, and creating new ones. Here is displaying and concatenation:

```
$ cat file1
Never gonna give you up
$ cat file2
Never gonna let you down
$ cat file3
Never gonna run around
and desert you
```

```
$ cat file1 file2 file3
Never gonna give you up
Never gonna let you down
Never gonna run around
and desert you
```

Command: find

The find command is used to locate a file in the directory structure. In fact, it is an extremely versatile tool. You can search by filename (obvious), file type (.txt,...), date created, what permissions it has, etc. Here is a basic example:

```
$ mkdir mydir
$ touch file7
$ mv file7 mydir/
$ find . -name file7
mydir/file7
```

Command: grep

`grep` is a powerful tool. The computational scientist is nothing without it! Here we search `file3` for the word 'Never':

```
$ grep Never file3.txt
Never gonna run around and desert you
```

You can use `grep` to search for words in multiple files.

```
$ grep Never *
Never gonna give you up
Never gonna let you down
Never gonna run around and desert you
```

Redirecting output

Redirecting output

We can redirect the output of a program using the $>$ key.

```
$ cat file1 file2 file3 > file4
$ cat file4
Never gonna give you up
Never gonna let you down
Never gonna run around and desert you
```

Note that this will either create the file file4 or it will overwrite it if it already exists. To simply append to an existing file, use » instead.

```
$ echo 'Never gonna make you cry' » file4
$ cat file4
Never gonna give you up
Never gonna let you down
Never gonna run around and desert you
Never gonna make you cry
```

Class activity

Please download the files `sentence` and `file4.txt` from Canvas and perform the following:

1. Change directory to the home directory (hint: use `pwd` to see where you are)- If you are using Windows and set it to open in your user folder -which contains your downloads folder, skip this step.
2. Create a new directory `~/ClassActivity`
3. Use the `find` command to find the location of `sentence` and `file4.txt` (hint: `find . -name sentence`)
4. Move the files from this location to the new location `~/ClassActivity/`
5. Add sentence to the end of file4.txt in a new file `file5.txt`
6. Display the contents of the file to the screen

Programming languages

Unix shell and Basic Commands   Redirecting output   **Programming languages**   Overview of C++   Hello World Program   Running First Program

00000000   000   0●00   00000   0000   00000000

Why we need languages

Programming languages are used to help humans and computers communicate. They provide:

- conciseness: high-level functions may each contain millions of little instructions.
- maintainability: smaller code base allows for easy modification of old functionality
- portability: different processors have different instructions; the high-level language can be adapted
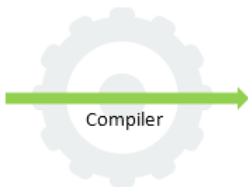
Compilation

Compilation is the conversion of the human-readable program to machine-readable instructions.

```
#include
<stdio.h>
int main()
{
printf("Hello")
;
return 0;
}
```
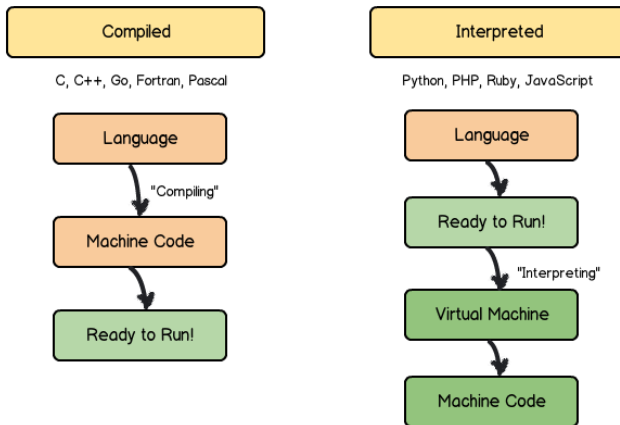Source code

Compiler

```
1000101010101010
0001001010101011
0111111001100000
0010110011010100
0101110111000110
0111110011110000
0001100111101010
0100100101010000
```
Executable code

Compiled versus interpreted language

Overview of C++

Overview of the language

C++ is a compiled language designed to offer the programmer high-level functionality, while retaining low-level capability.

- C++ was written by Bjarne Stroustrup at Bell labs in 1980s
- Based on the successful C language
- Designed around the use of objects (object-oriented)

Overview of the language

C++ is a compiled language designed to offer the programmer high-level functionality, while retaining low-level capability.

- C++ was written by Bjarne Stroustrup at Bell labs in 1980s
- Based on the successful C language
- Designed around the use of objects (object-oriented)

Objects in programming are similar in concept to real-world objects. They have *states* and *behaviors*. They are a useful concept for a number of reasons:

- modularity
- infromation-hiding
- code re-use

These features make C++ great for large-scale software projects.

Picking an Editor

All computer code is typed up in some sort of text editor. Some text editors are better than others. For example Microsoft Word would be a terrible computer language text editor. Notepad could work, but there are much better editors out there.

Some that I've seen students use before:

- emacs
- geany
- Vi/Vim
- Nano
- Sublime
- https://en.wikipedia.org/wiki/Comparison_of_text_editors

Opening the editor

I'll let you choose which editor you would like to use later, for now we'll use vim.

Navigate to your class examples folder. In your terminal, type `vim HelloWorld.cpp`. This will create a file `HelloWorld.cpp` and open it in the vim editor.

Hello World Program

Simple Hello World Program

```
//My first C++ Program

#include <stdio.h>

int main()
{
        printf("Hello World!! ");
        return 0;
}
```

//

Denotes that the rest of the line is a comment. The computer ignores comments.

#include

Tells the computer to include a particular package. In this case we are including the package stdio.h

Simple Hello World Program

```
//My first C++ Program

#include <stdio.h>

int main()
{
        printf("Hello World!! ");
        return 0;
}
```

int main()

This is where the program starts. It executes everything between the two curly braces { }.

printf();

This is a function that takes some text (in this case "Hello World") and prints it to the terminal. (We will learn more about functions later)

Simple Hello World Program

```
//My first C++ Program

#include <stdio.h>

int main()
{
        printf("Hello World!! ");
        return 0;
}
```

return 0;

This statement ends the program.

;

Notice the semicolons at the end of the two statements, return 0; and printf(); above.
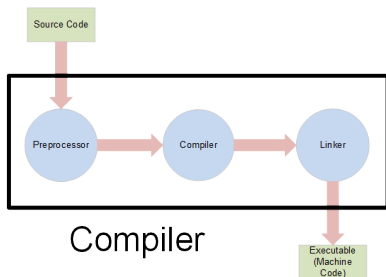These are critical, and belong at the end of most statements.

Running First Program

Compiling

Before the computer can execute/run our simple program we first have to compile it. Basically we put it through a process that translates it from our English text to computer understandable bits (ones and zeros)

- Once your code has been successfully compiled it will now run on your machine.
- We will learn more about the compiler in future lessons.
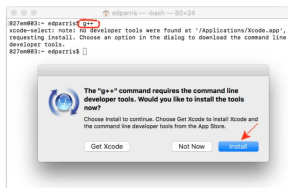
Installing the g++ compiler

Windows 10: Bash on Ubuntu

1. In your terminal, go home using the `cd ~` command
2. Type `sudo apt-get install g++` - This will install the g++ compiler.
3. Check that it was installed correctly by typing `which g++` or `g++ -version`. The outputs should be similar to:

```
ag12s@DESKTOP-1FNKE80:/mnt/c/Users/ag12s$ which g++
/usr/bin/g++
ag12s@DESKTOP-1FNKE80:/mnt/c/Users/ag12s$ g++ --version
g++ (Ubuntu 5.4.0-6ubuntu1~16.04.9) 5.4.0 20160609
Copyright (C) 2015 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.
```

Mac Terminal

1. In the Terminal type `g++`.
2. If this alert box pops up, click "Install". You do not need Xcode unless you want Xcode.

3. check that it has installed by typing `g++` in the terminal. This will return the error message `no input files`.

Running our Program

Now that we have installed the g++ compiler,

- In the terminal, navigate to the directory which contains our HelloWorld.cpp file

Running our Program

Now that we have installed the `g++` compiler,

- In the terminal, navigate to the directory which contains our `HelloWorld.cpp` file
- Compile your program. Run the command
  `g++ HelloWorld.cpp -o HelloWorld`

Unix shell and Basic Commands   Redirecting output   Programming languages   Overview of C++   Hello World Program   **Running First Program**

00000000   000   0000   00000   0000   00000●00

Running our Program

Now that we have installed the `g++` compiler,

- In the terminal, navigate to the directory which contains our `HelloWorld.cpp` file
- Compile your program. Run the command
  g++ HelloWorld.cpp -o HelloWorld

g++ HelloWorld.cpp -o HelloWorld

The `g++` command stands for compile. The Second option `HelloWorld.cpp` is the name of the file you want to compile. The third option `-o` stands for outfile and the fourth option `HelloWorld` is what you wish the out file (executable file) to be called. Thus this command takes `HelloWorld.cpp` and creates a file called `HelloWorld` which is executable.

Running our Program

Now that we have installed the `g++` compiler,

- In the terminal, navigate to the directory which contains our `HelloWorld.cpp` file
- Compile your program. Run the command
    g++ HelloWorld.cpp HelloWorld

g++ HelloWorld.cpp HelloWorld

The `g++` command stands for compile. The Second option `HelloWorld.cpp` is the name of the file you want to compile. The third option `-o` stands for outfile and the fourth option `HelloWorld` is what you wish the out file (executable file) to be called. Thus this command takes `HelloWorld.cpp` and creates a file called `HelloWorld` which is executable.

- Execute your program. Run the command.
    ./HelloWorld

Running our Program

Now that we have installed the g++ compiler,

- In the terminal, navigate to the directory which contains our HelloWorld.cpp file
- Compile your program. Run the command
  g++ HelloWorld.cpp HelloWorld

g++ HelloWorld.cpp HelloWorld

The g++ command stands for compile. The Second option HelloWorld.cpp is the name of the file you want to compile. The third option -o stands for out file and the fourth option HelloWorld is what you wish the out file (executable file) to be called. Thus this command takes HelloWorld.cpp and creates a file called HelloWorld which is executable.

- Execute your program. Run the command.
  ./HelloWorld

./HelloWorld

The ./ command allows you to execute. It should be followed immediately by the executable file. No space.

Congratulations! You have just compiled and executed your program.