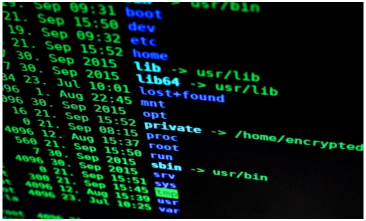
Specialized Numerical Methods for Transport Phenomena

C++ Programming and the Linux shell

Bruno Blais and Laura Prieto Saavedra

Associate Professor
Department of Chemical Engineering
Polytechnique Montréal

January 13, 2025



```
 21. Sep 15:50 boot
 19. Sep 09:32 dev
 21. Sep 15:52 etc
 30. Sep 2015  home
 30. Sep 2015  lib -> usr/lib
 23. Jul 10:01 lib64 -> usr/lib
  1. Aug 22:45 lost+found
 30. Sep 2015  mnt
 21. Sep 15:52 opt
 21. Sep 08:15 private -> /home/encrypted
 12. Aug 15:37 proc
 21. Sep 15:50 root
 30. Sep 2015  run
 30. Sep 2015  sbin -> usr/bin
 21. Sep 15:51 srv
 21. Sep 15:45 sys
 12. Aug 15:39 tmp
 23. Jul 10:25 usr
               var
```

Outline

Outline

The command line

BASH (Bourne Again Shell) is a command line language that is omnipresent on linux computers and clusters. It has numerous uses:

- Launch programs
- Copy, move, remove files
- Modify files
- Etc.

But why do we use such a primitive tool?

- Low resources required
- Omnipresent on numerous platforms (clusters all operate on BASH nowadays)
- Works remotely from any type of machines

We just need the basics!

We will indicate the start of the command line using:

```
$
```

To change the current directory to another one

```
$ cd destination
```

To copy files

```
$ cp source destination
```

To delete files

```
$ rm file
```

To remove folders

```
$ rm -r folder
```

The basics

List content of the current folder

```
$ ls  or  $ ll
```

List content of a folder

```
$ ls folder
```

Move a file (instead of copying it)

```
$ mv file destination
```

Find occurence of string in files

```
$ grep string file
```

Other useful commands: `pwd`, $\sim$, `cat file`

More information

The following website provides a very nice (and fast) tutorial to the command line:

```
https://ubuntu.com/tutorials/command-line-for-beginners#
1-overview
```

Do not worry if you have never used a command line before. It is easy to learn how to use a command line as you go. Many students have taken this class before without any previous command line experience and did not face any struggles.

Outline

Introduction to C++

C++ is powerful general-purpose programming language:

- Its syntax and its low-level management is derived from C
- Object oriented (or not, it is sometimes a matter of taste)

Language designed for performance, efficiency and flexibility:

- The flexibility of C++ is what makes it a better choice for large-scale simulation software than Fortran or regular C
- The same performance can be obtained, but the result is a much more flexible code structure

This remains a matter of choice, it is possible to design an efficient simulation code in any language.

Introduction to C++

C++ is powerful general-purpose programming language:

- Its syntax and its low-level management is derived from C
- Object oriented (or not, it is sometimes a matter of taste)

Language designed for performance, efficiency and flexibility:

- The flexibility of C++ is what makes it a better choice for large-scale simulation software than Fortran or regular C
- The same performance can be obtained, but the result is a much more flexible code structure

This remains a matter of choice, it is possible to design an efficient simulation code in any language.

Do not be scared if you have never programmed in C++ before. Some useful references are available on the moodle website to help you.

Philosophy in this class

More often than not, you will have to work with codes that are already designed:

- These codes can be open source
- They can also be in-house software

More importantly it is always the case that a building block of what you need already exists in the form of a library:

- Pre-processing (mesh reading)
- Post-processing (format to be read by post-processor)
- Sparse linear matrix and linear solvers:
  - Trilinos
  - PETSC
  - Eigen

Philosophy in this class

It is important to know how to code a simulation software!
Often more important to know how to modify existing codes...
We have a limited amount of time and we want to focus on the modeling, the science and the programming of this science. The post-processing, pre-processing part is important, but programming it can be very tedious.
Therefore in this class we will be working with code templates:

- This will make your life easier.
- When it comes to programming your own application, think about using a library for this. If you want to do it quickly, you can steal my routines, I'll be happy if I was of any help!

Outline

Features of C++

Compiled

The source code is compiled by a compiler to make efficient machine code.

Statically typed

Variables have a type that is defined at compile time.
The type of variables must be declared explicitly.

Large ecosystem

Compilers, testing toolset, automatic indentation, IDEs, etc...

HPC-ready

Native support for the Message Passing Interface (MPI). C and Fortran are the other languages with native support.

Hello world!

```
#include <iostream>

// This function is required in every C++ program
int main()
{
    std::cout << "Hello World!";
    return 0;
}
```

Outline

Numerical data types

Double precision (64 bits):

```
double a;
```

Integers (32 bits):

```
int i;
```

Boolean (8 bits):

```
bool condition;
```

Mathematical operations

Addition

```
double c = a + b;
```

Substraction

```
double c = a - b;
```

Multiplication

```
double c = a * b;
```

Division (be careful about this one)

```
double c = a / b;
```

Strings

C++ strings are extremely flexible. They are very similar to python strings. They have a few characteristics.

- In namespace std
- Requires a specific include `<string>`
- `http://www.cplusplus.com/reference/string/string/`

```
std::string word;
```

Outline

Decision (control) structures

Control structures in C++ are similar to those encountered in most programming languages:

Conditional structures

They will be `if-else` statements or `switch`

Loop (iteration) structures

They will be `for` loops (when the number of iterations is known a priori) or while statements.

If conditions

```
#include <iostream>

int main()
{
    int age;
    std::cout << "Please input your age: ";
    std::cin >> age;
    std::cin.ignore(); // Throw away enter
    if (age < 100) {
        std::cout << "You are pretty young!\n";
    }
    else if (age == 100) {
        std::cout << "You are old\n";
    }
    else {
        std::cout << "You are really old\n";
    }
    return 0;
}
```

For loops

```
#include <iostream>

int main ()
{
    // for loop execution
    for (int a = 10; a < 20; a = a + 1)
    {
        std::cout << "value of a: " << a << std::endl;
    }
    return 0;
}
```

While loops

```
#include <iostream>

int main ()
{
    // Local variable declaration
    int a = 10;

    // while loop execution
    while(a < 20)
    {
        std::cout << "value of a: " << a << std::endl;
        a++;
    }
    return 0;
}
```

Do..While loops

```cpp
#include <iostream>

int main ()
{
    // Local variable declaration:
    int a = 10;

    // do loop execution
    do
    {
        std::cout << "value of a: " << a << std::endl;
        a = a + 1;
    } while( a < 20 );
    return 0;
}
```

Outline

A sample program

```
#include <iostream>

int main()
{
    int n;
    int factorial = 1.0;

    std::cout << "Enter a positive integer: ";
    std::cin >> n;
    for(int i = 1; i <= n; ++i)
      {
        factorial *= i;
      }
    std::cout << "Factorial of " << n << " = " << factorial;
    return 0;
}
```

A program with a function

```cpp
#include <iostream>

double square(double x)
{
    return x*x;
}

int main()
{
    double y = square(10);
    std::cout << "The square value is " << y << std::endl;
    return 0;
}
```

C++ is a strongly typed language. Functions must have a return type. if a function does not return anything, it is declared as `void`.

A real program

Real programs combine multiple files that interact amongst themselves. To achieve this, C++ uses two type of files:

Header files (`.h`)

They contain the declaration of the functions and the classes. They generally don't contain the implementation of the functions. By including the correct header files, other part of the code can be made readily aware of the functions without knowing the code of the function.

Code files (`.cc or .cpp`)

They contain the actual implementation of the functions.

Outline

Memory in C++

For a C++ program, the memory of a computer is like a succession of memory cells, each one byte in size, and each with a unique address. These single-byte memory cells are ordered in a way that allows data representations larger than one byte to occupy memory cells that have consecutive addresses.

When a variable is declared, the memory needed to store its value is assigned a specific location in memory (its memory address).

This way, each cell can be easily located in the memory by means of its unique address. For example, the memory cell with the address 1776 always follows immediately after the cell with address 1775 and precedes the one with 1777, and is exactly one thousand cells after 776 and exactly one thousand cells before 2776.

Pointers

In C++, a pointer is a variable that stores the memory address of another variable. Pointers are used to store the addresses of memory locations that are used to store values of variables.

Pointers - Example

```
#include <iostream>
int main()
{
    // Declare a variable
    int x = 10;

    // Print the value of x
    std::cout << "x = " << x << std::endl;

    // Print the memory address of x using the address-of operator (&)
    std::cout << "&x = " << &x << std::endl;

    // Print the value stored in the address using the dereference operator (*)
    std::cout << "*(&x) = " << *(&x) << std::endl;

    // Declare a pointer variable that stores the memory address of x as its value
    int* ptr = &x;

    // Print the value stored in the memory address pointed to by ptr
    std::cout << "*ptr = " << *ptr << std::endl;

    // Print the memory address stored in ptr
    std::cout << "ptr = " << ptr << std::endl;

    return 0;
}
```

Pointers - Example

In this example, `x` is a variable of type int that stores the value 10. ptr is a pointer variable that stores the memory address of x. The & operator is used to get the memory address of `x`, and the * operator is used to get the value stored in the memory address pointed to by ptr.

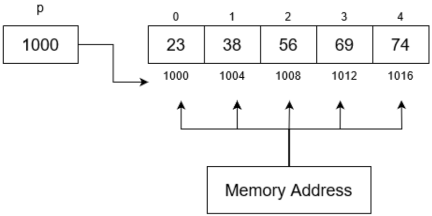
Pointers are useful because they allow you to modify the value of a variable indirectly, through the memory address. They are also used to dynamically allocate memory at runtime, and to pass large amounts of data to functions more efficiently.

```
#include <iostream>
int main()
{
    // Change value of a variable through a pointer
    int x = 10;
    int* ptr = &x;
    std::cout << "Value of x: " << x << std::endl;
    std::cout << "Value of ptr: " << *ptr << std::endl;
    *ptr = 6;
    std::cout << "Value of x: " << x << std::endl;
    std::cout << "Value of ptr: " << *ptr << std::endl;
    return 0;
}
```

Pointers - Arrays

Pointers are useful for many things (which we will see during the class). One basic usage of them is in the declaration of arrays:

```
int main () {
    double p[5];
    p[0]=23;
    p[1]=38;
    p[2]=56;
    p[3]=69;
    p[4]=74;
}
return 0
```



Pointers - Arrays

The size of arrays can also be set dynamically using new

```
int main () {
    double* p;
    int n=5;
    p = new double[n];
    p[0]=23;
    p[1]=38;
    p[2]=56;
    p[3]=69;
    p[4]=74;
}
return 0
```

More conviently - Vectors

An alternative is to see the vector class (we will discuss classes in the next course)

```
#include <vector>
int main ()
{
   std::vector p;
   int n=5;
   p.resize(n);
   p[0]=23;
   p[1]=38;
   p[2]=56;
   p[3]=69;
   p[4]=74;
}
return 0
```

Outline

In Conclusion

Complex

C++ is a very complex programming language. However, it is used in the quasi totality of scientific computing software (alongisde Fortran)

Our usage of it will be minimal

C++ has a large breadth. We will use a small part of it. Remember to look online and/or ask questions when there are things you don't understand. There are many available resources online. If you find one that is interesting, tell me and I will share it with others on the Moodle website.