Scientific Computing (Phys 2109/Ast 3100H)
I. Scientfic Software Development

SciNet HPC Consortium

University of Toronto

Winter 2013

Part I

Introduction to Software Development

Lecture 2

C++ Introduction

- Objects
- Libraries

Building with Make

Version Control

- Git

Homework

C++ Introduction continued...

C++ Intro: Const

A type modifier

- const is a type modifier.
- It means the value of that type is fixed.
- Useful for constants, e.g.

```
const int arraySize = 1024;
```

- Useful to show read-only arguments to functions:

```
int f( const Type &in, Type &out );
```

- const is contageous!
- Now everything has to be “const correct”.

C++ Intro: Classes and objects

Object oriented programming (OOP)

- **Non-OOP:** functions and data accessible from everywhere.
- **OOP:** Data and functions (methods) together in an object. Implementation details hidden.

What are classes and objects?

- Classes are to objects what types are to variables.
- Using a class, one can create one or more instances of it, called objects:

```
class object(arguments);
```

C++ Intro: Classes and objects

Syntax:

```
class object(arguments);
```

Usage

- Different from regular variables are the possibility of argument, supplied to construct the object.
- An object has members (fields) and member functions (methods), which are accessed using the "." notation.

```
object.field;
object.method(arguments);
```

C++ Intro: Classes and objects

Example (member function/method)

```
#include <string>
std::string s("Hello");
int stringlen=s.size();
```

Example (member/field)

```
#include <utility>
std::pair<int,float> p(1, 0.314e01);
int   int_of_pair   = p.first;
float float_of_pair = p.second;
```

C++ Intro: Templates

Templates

- In *generic programming*, specific types are not specified initially, but instantiated when needed.
- In C++, generic programming uses templates.
- Many templated functions and classes in the standard library.

Usage

- To create an object from a template class *templateclass*:

```
templateclass<type> object(arguments);
```

Examples:

```
std::complex<float> z; //single precision complex number
std::vector<int> i(20);//array of 20 integers
```

C++ Intro: Libraries

Usage

- Put an include line in the source code, e.g.

```
#include <iostream>
#include "mpi.h"
```

- Include the libraries at link time using `-l[libname]`.
  Implicit for the standard libraries.

Common standard libraries (Standard Template Library)

- string: character strings
- iostream: input/output, e.g., `cin` and `cout`
- fstream: file input/output, e.g., `ifstream` and `ofstream`
- containers: vector, complex, list, map, . . .
- cmath: special functions (inherited from C), e.g. `sqrt`
- cstdlib, cstring, cassert, . . . : C header files

Streams

IO

In C++, stream object are responsible for I/O.

You can output an object `obj` to a stream `str` simply by

```
str << obj
```

while you can read an object `obj` from a stream `str` simply by

```
str >> obj
```

The stream will encode these object in ascii format, provided a proper operator is defined (true for the standard c++ types).

Standard streams

- `std::cout` For output to the screen (buffered)
- `std::cin` For input from the keyboard
- `std::cerr` For error messages (by default to the screen too)

These are defined in the header file `iostream`

Streams - File IO

- Classes for file IO are defined in the header `fstream`.
- The `ofstream` class is for output to a file.
- The `ifstream` class is for input from a file.
- You have to declare an object of these classes first.
- Then you can use the streaming operators << and >>.
- Use member functions `read`/`write` to read/write binary.

Example

```
std::ofstream fout("output.txt");
int x = 4;
float y = 1.5;
fout << x << ' ' << y << std::endl;
fout.close();
std::ifstream fin("output.txt");
int x2;
float y2;
fin >> x >> y;
fin.close();
```

C++ Intro: Multidimensional arrays

- Automatic multidimensional arrays are easy to define

```
double a[6][6];
```

  and are equally easy to use

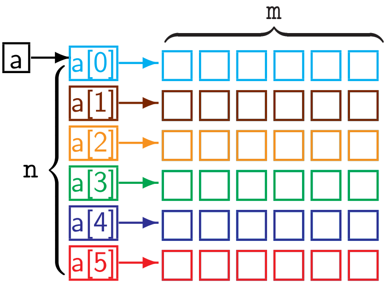
```
a[2][0] = 15.7;
```

- This repeated bracket business means that `a` is a kind of pointer-to-a-pointer:
  Each `a[i]` for `i=0..5` points to a 1d array.
- But we know this is BAD:
  - Memory is statically allocated;
  - Cannot check for successful allocation;
  - and can pose limits on memory usage.
- How to do this dynamically?

C++ Intro: Multidimensional arrays

```
float **matrix(int n,int m) {
   float **a = new float * [n];
   a[0] = new float [n*m];
   for (int i=1; i<n; i++)
      a[i] = &a[0][i*m];
   return a;
}
void free_matrix(float **a) {
   delete[] a[0];
   delete[] a;
}
void fill(int n,int m,
.          float **a,float v){
   for (int i=0; i<n; i++)
      for (int j=0; j<m; j++)
         a[i][j]=v;
}
```

"Row major order"

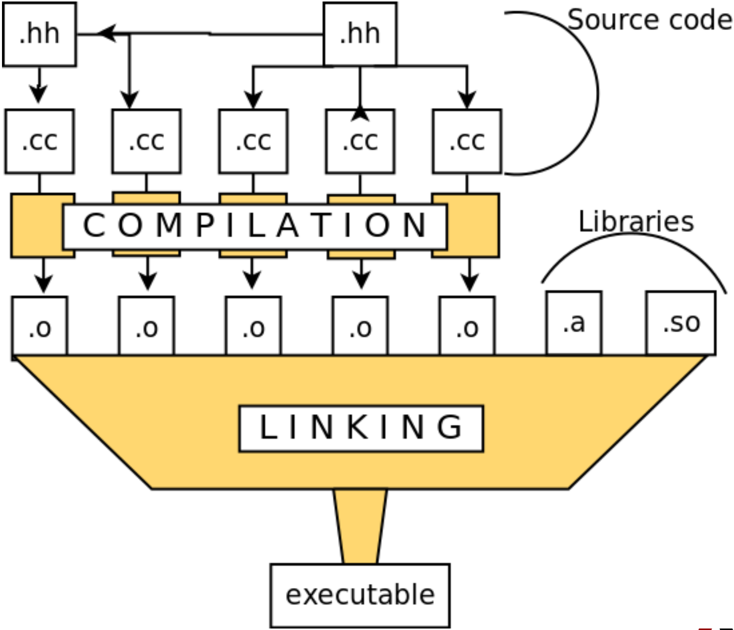


Why not `std::vector<std::vector<float> > a;`?

Make:

Building from multiple source files

Compilation workflow



Basic modular setup: .cc/.hh

The compilation workflow is caused by a modular setup

- Implementation of a module goes into the .cc file.
- Interface of a module goes into the header (.hh) file:
  This includes declarations of function, but not their definitions (i.e. not the statements etc.).
- The header file gets included in other files.

We'll discuss the advantages of such a modular design next week, but it is good for (re)compilation speeds and for tracking down and preventing bugs.

Compiling with make

What does make do?

- make takes a 'makefile' and does what it specifies.
- makefile contains variables, rules and dependencies.
- makefile often called Makefile or makefile.
- if one file depends on another one that is newer, the rule is applied.
- There are default rules for e.g. c and c++ grograms.

Compiling with make

Single source file

```
# This file is called Makefile
CXX      = g++
CXXFLAGS = -I$(GSLINC) -O2
LDFLAGS  = -L$(GSLLIB)
LDLIBS   = -lgsl -lgslcblas

all:  main

main:  main.o
    $(CXX) $(LDFLAGS) -o main main.o $(LDLIBS)

main.o:  main.cc
    $(CXX) $(CXXFLAGS) -c -o main.o main.cc
```

Compiling with make

Multiple source file application

```
CXX      = g++
CXXFLAGS = -I$(GSLINC) -O2
LDFLAGS  = -L$(GSLLIB)
LDLIBS   = -lgsl -lgslcblas

all:  main

main:  main.o mylib.o
    $(CXX) $(LDFLAGS) -o main main.o mylib.o $(LDLIBS)

main.o:  main.cc mylib.hh
    $(CXX) $(CXXFLAGS) -c -o main.o main.cc

mylib.o: mylib.hh mylib.cc
    $(CXX) $(CXXFLAGS) -c -o mylib.o mylib.cc

clean:
    rm -f main.o mylib.o main
```

Compiling with make

When typing `make` at command line, make start buiding the first rule, here, `all`:

- Checks if `main.cc` or `mylib.cc` or `mylib.hh` were changed.
- If so, invokes corresponding rules for object files.
- Only compiles changed code files: faster recompilation.

Compiling with make

- Make does not detect changes in compiler, or in system.
- But .o files are system/compiler dependent, so need to be recompiled.
- Always specify a “clean” rule in the makefile, so that moving from one system or compiler to another, you can do a fresh rebuild:

```
  $ make clean
  $ make
```

- First and foremost, need to get dependencies between object, source, and header files right.
- `g++ -MM` can help:

```
$ g++ -MM main.cc
main.o:  main.cc mylib.hh
```

See Mike Nolta’s slide set

Homework 1: Multi-file C++ program to create a data file.

- Start a git repository, and begin writing a C++ program to
  - Get an array size and a standard deviation from user input
  - Allocate a 2d array,
  - Store a 2d Gaussian with a maximum at the centre of the array & given standard deviation (in units of grid points).
  - Outputs that array to a text file, free the array, and exit.
- The output text file should contain just the data in text format, with a row of the file corresponding to a row of the array and with whitespace between the numbers.
- The 2d array creation/freeing routines should be in one file (with an associated header file), the gaussian calculation be in another (ditto), and the output routine be in a third, with the main program calling each of these.
- Use a makefile to build your code (add it to the repository).

Homework 1: Multi-file C++ program to create a data file.

- You can start with everything in one file, with hardcoded values for sizes and standard deviation and a static array, then refactor things into multiple files, adding the other features.
- As a test, use the ipython executable that came with your Enthought python distribution to read your data and plot it. If your data file is named data.txt, running the following:

```
$ ipython --pylab
In [1]:  data = numpy.genfromtxt('data.txt')
In [2]:  contour(data)
```

  Should give a nice contour plot of a 2-dimensional gaussian.
- Email in your source code and the git log file of all your commits by email to rzon@scinethpc.ca and ljdursi@scinethpc.ca by next Thursday at 9:00 am.