Introduction to Standard C++

Lecture 00: Administravia + Language basics

Massimiliano Culpo[1]

[1]CINECA - SuperComputing Applications and Innovation Department

07.04.2014

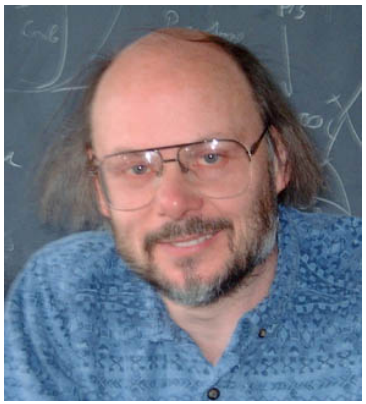
Lecture timetables

Day 1	Language basics	• Overview of the language • Memory and object models
	The C core in C++	• Quick review of basic C concepts • Scope, overloading and name look-up • Types, standard conversions
	Introduction to OOP	• Motivation for object orientation • Basic design principles

Lecture timetables

Day 2	A primer on classes	• Class members • Overloaded operators
	Class hierarchies	• Derived classes • Abstract classes and polymorphism
Day 3	C++ templates	• Template functions • Template classes
	C++ standard library	• Structure of the standard library • Iterator concepts • Overview of containers and algorithms

A little history



C with classes [1979]

B. Stroustrup began his work on "C with Classes" at AT&T:

- enhancement of C with Simula-like features
- C was chosen because it was general-purpose, fast, portable and widely used

C++ [1983]

- Added run-time polymorphism, overloading, references, ...
- First edition of *"The C++ Programming Language"* released in 1985
- *"The Annotated C++ Reference Manual"* was published in 1990
- ISO Standard in 1998, 2003, 2011

The latest major revision of the C++ standard was approved by ISO/IEC on 12/08/2011.

What is C++?

The easiest way to view C++ is not as a single language but as a federation of related languages:

C

Way down deep, C++ is still based on C.

Object oriented C++

This part of C++ is what C with Classes was all about.

Template C++

This is the generic programming part of C++.

The STL

You need to be sure to follow its conventions.

- C++ is a multi-paradigm programming language
- Keep the sub-languages in mind, change strategy where needed
- Rules for effective C++ programming vary, depending on the part of you are using

Why using (or not) C++?

All the good things...

Properly used, C++ can be a joy to work with:

- it permits an uncommon range of power and expressiveness
- designs can be *directly expressed* and *efficiently implemented*
- *generic-programming* is supported through templates
- it fosters *object-orientation* with virtual classes
- it is not prohibitive to write effective C++ programs
- it really helps in managing large software projects

...come at a price

Used without discipline, however, C++ can lead to code that is incomprehensible, unmaintainable, inextensible, inefficient, and just plain wrong. (Scott Meyers, "Effective C++")

Where can I found information about C++?

The "ISO/IEC Standard" is the final reference on the language:

C

Chapter 1-8 Core language
Chapter 16 Preprocessor

Object oriented C++

Chapter 9-12 Classes, run-time polymorphism
Chapter 15 Exception handling

Template C++

Chapter 13 Overload resolution
Chapter 14 Templates

The STL

Chapter 17-30 Description of each library

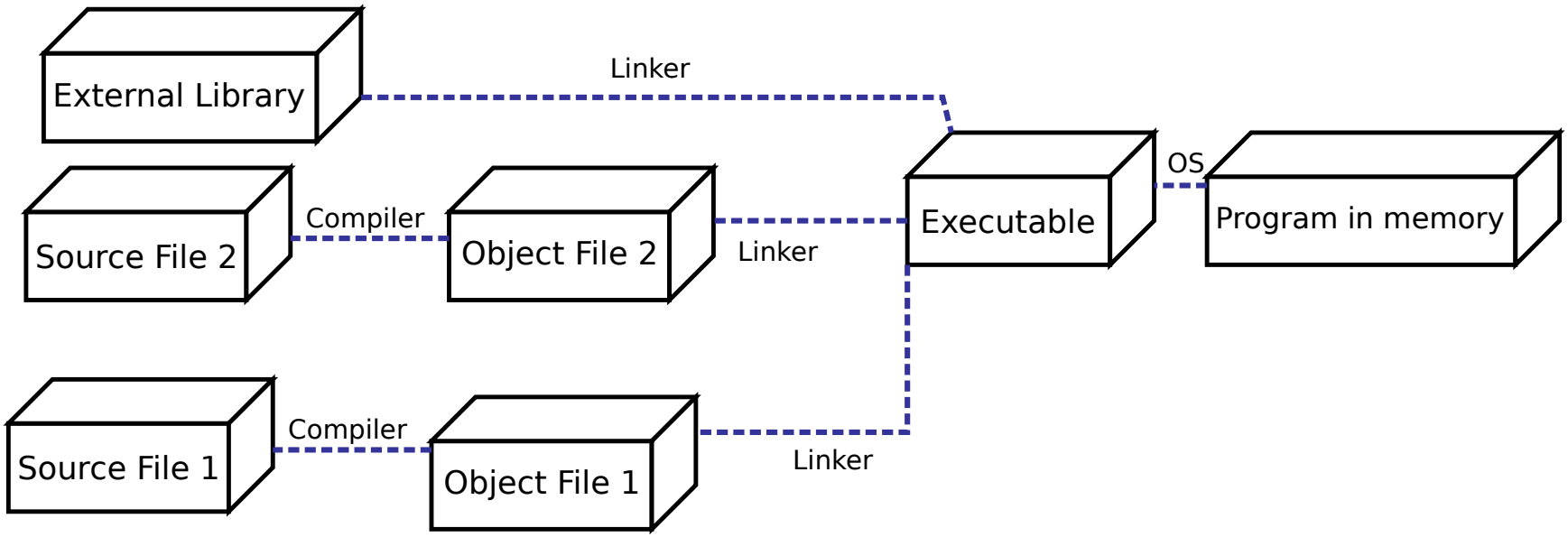
It is written in *legalese*, and therefore is quite difficult to interpret.

cppreference provides a nice quick-reference to the standard libraries.

Some specialized Q&A sites, like stack**overflow**, may provide help in case you want to pose a question related to your code.

And of course there are books!

The compilation process in a nutshell



All you need to know about memory and objects therein

The C++ memory model (§1.7)

1. The fundamental storage unit in the C++ memory model is the byte
2. The memory consists of one or more sequences of contiguous bytes
3. Every byte has a unique address
4. A memory location is an *object* of scalar type
5. Two threads of execution can
   - update
   - access

   separate memory locations without interfering with each other

Address	Type	Value
0		
1	...	...
2	char	0x28
3	float	0x00
4		0x00
5		0x00
6		0x00
⋮	⋮	⋮

All you need to know about memory and objects therein

The C++ object model (§1.8)

1. An object is a region of storage
2. An object is created by either:
   - a definition
   - a new-expression
   - the implementation
3. An object has a:
   - type
   - storage duration
4. An object may have a name
5. Objects can contain other objects, called subobjects
6. An object that is not a subobject is called a complete object

Address	Type	Value
0		
1	...	...
2	char	0x28
3	float	0x00
4		0x00
5		0x00
6		0x00
⋮	⋮	⋮

All you need to know about memory and objects therein

```
1  struct A {
2     char c;
3     int  i;
4  }
5
6  A obj;
7
8  a.c = 'H';
9  a.i = 1024;
```

Address	Type	Name	Value
0	...	...	...
1	...	...	...
2	A char	a a.c	0x28
3	int	a.i	0x00
4			0x00
5			0x04
6			0x00
⋮	⋮	⋮	⋮

Objects in presence of multiple threads (C++11)

```
// Thread 0
flag[0] = true; // A
if( flag[1] )   // B
// resolve contention
else
// critical section

flag[0] = false;
```

```
// Thread 1
flag[1] = true; // C
if( flag[0] )   // D
// resolve contention
else
// critical section

flag[1] = false;
```

Q: If flags are shared and atomic, could both threads enter the critical region?

A: Not in a sequentially consistent (SC) memory model

Sequentially consistent data-race free (SC-DRF)

The C++ standard guarantees a SC system if it is free of data-races

Storage duration of objects (§3.7)

Storage duration is the property of an object that defines the minimum potential lifetime of the storage containing the object.

Static storage duration

Storage lasts for the entire duration of the program

Thread storage duration

Storage lasts for the entire duration of the thread in which they were created

Automatic storage duration

Storage lasts until the block in which they were created exits

Dynamic storage duration

Storage is managed by the programmer with **new** *and* **delete** *operators*

Storage duration of objects (§3.7)

```
// Automatic storage duration
void foo() {
  int ii = 0;
  { float jj = 0.0f; } // storage for jj expires
} // storage for ii expires

// Static storage duration
void bar() {
  static int ii = 0;
} // lasts until the end of the program

// Dynamic storage duration
void foobar() {
  int * pii = new int[3];
  delete[] pii; // lasts until corresponding delete
}
```

Q: What is the storage duration of the objects below?

```
1  void foo(float* workspace,const int size)
2  {
3    static int ncalls = 0;
4    ncalls++;
5    for (int ii = 0; ii < size; ii++)
6      workspace[ii] = ii;
7  }
8  int main() {
9    float * pf = new float[10];
10   foo(pf, 10);
11   delete[] pf;
12 }
```

Key points so far...

General guidelines

1. C++ is a complex, multi-paradigm programming language
2. To master C++ you need to know:
   - where a paradigm needs to be applied
   - how to combine different paradigms
3. Discipline is a pre-requisite for an effective use of C++

Memory and object models

1. The constructs in a C++ program create, destroy, refer to, access and manipulate objects
2. An object is a region of storage with an associated type, storage duration and possibly a name
3. The C++ standard guarantees a SC-DRF memory model