Università degli Studi di Padova

C++ source processing

Stefano Ghidoni

Agenda

- Compiling C++ code
- Tools for producing an executable
- Libraries in C++
  - Static
  - Dynamic

C++: motivation

- Efficient
- Low-level or high-level? What are the elements we can deal with?
  - High level: classes
  - High level: templates, inheritance
  - Low level: memory management
  - Low level: hard types
- Widely used, cross-platform

Producing software in C++

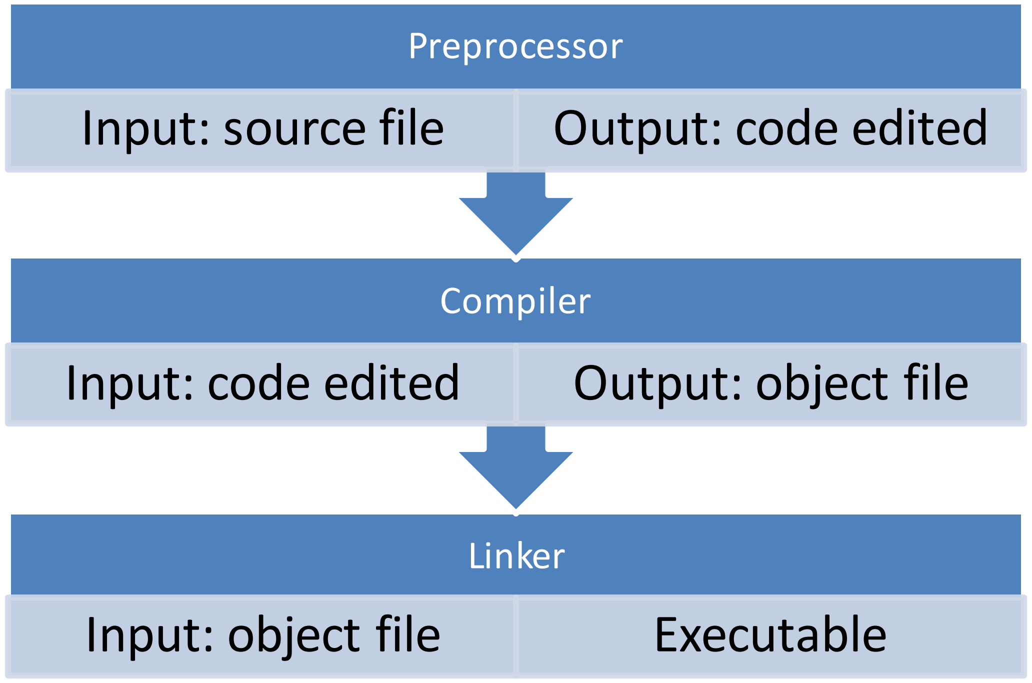
- Questions about your background:
  - What is a preprocessor?
  - What is a compiler?
  - What is a linker?

Producing software in C++

- Questions about your background:
  - What is an object file?
  - What is an executable?
  - What is a library?

Producing software in C++

- What is the software production process?
- Write source code
- Compile
- Link
- What's the difference between object and executable?



Toolchain

- "A simple software development toolchain may consist of a compiler and linker (which transform the source code into an executable program), libraries (which provide interfaces to the operating system), and a debugger (which is used to test and debug created programs)" (Wikipedia)

Functions in C++

```
int f(int i);
```

Function declaration

```
int main(void)
{
  int i = 0;
```

```
  i = f(i);
```

Function call

```
  return 0;
}
```

```
int f(int i)
{
  return i + 2;
}
```

Function definition

Multiple file SW projects

- What if we wish to distribute our SW among multiple files?

```
int f(int i);
```

Function declaration → Goes to **header file** (my_func.h)

```
int main(void)
{
  int i = 0;
```

```
  i = f(i);
```

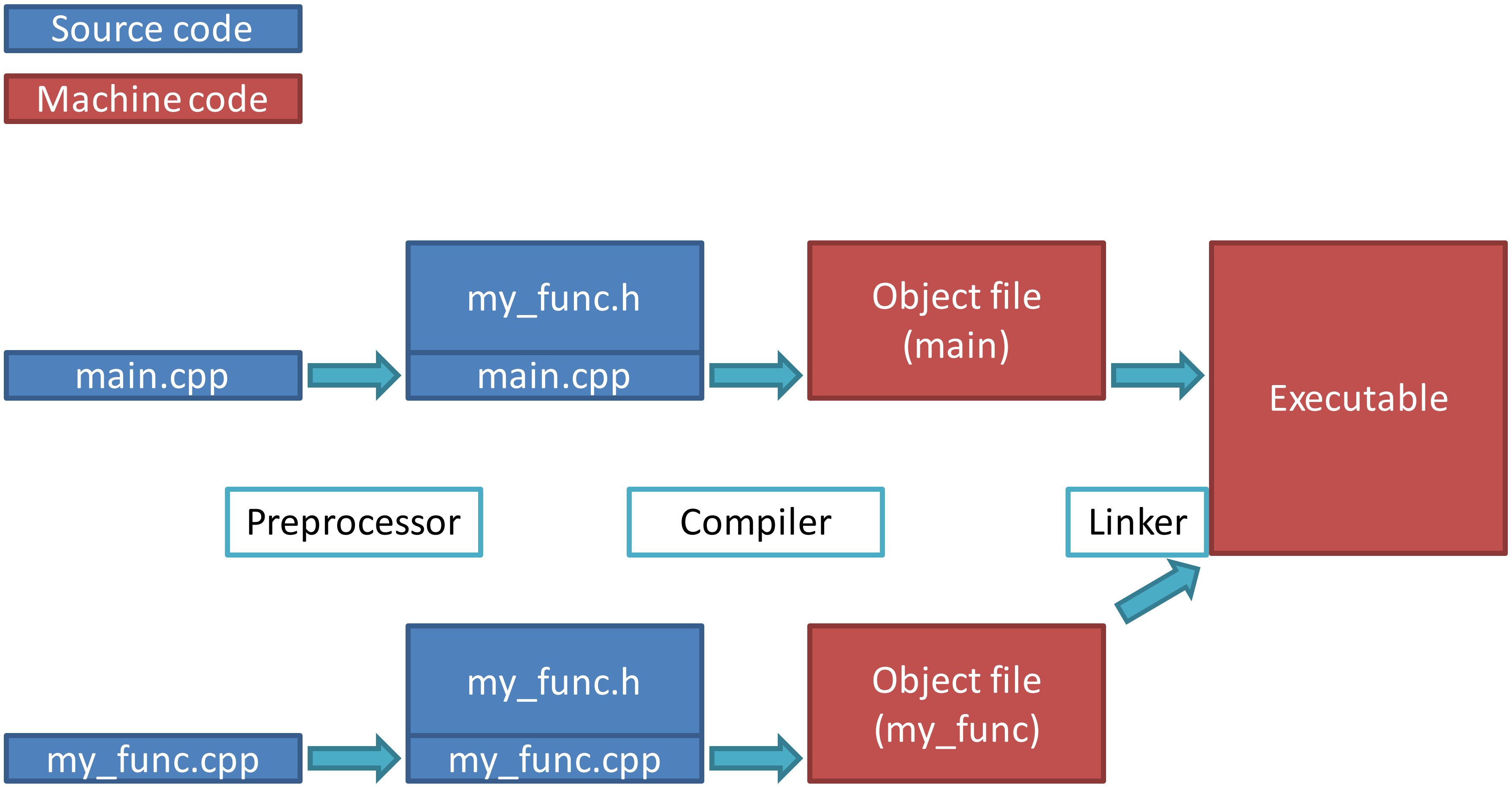
Function call

```
  return 0;
}
```

```
int f(int i)
{
  return i + 2;
}
```

Function definition → Goes to **library source file** (my_func.cpp)

From source to executable



External libraries

- Previous case: one project divided into multiple files
- Libraries can be provided by third party
  - Example: OpenCV!
- The same mechanism holds when user and library are different projects
  - The .cpp file(s) are often shipped already compiled

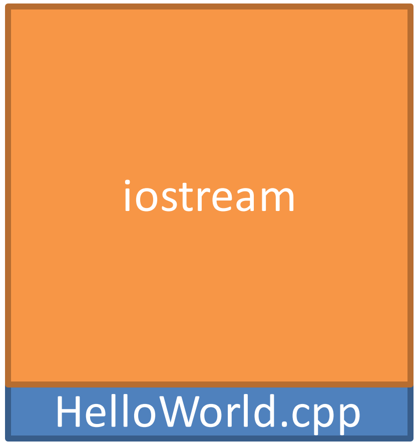
A simple case

- One single source file (e.g.: hello world)

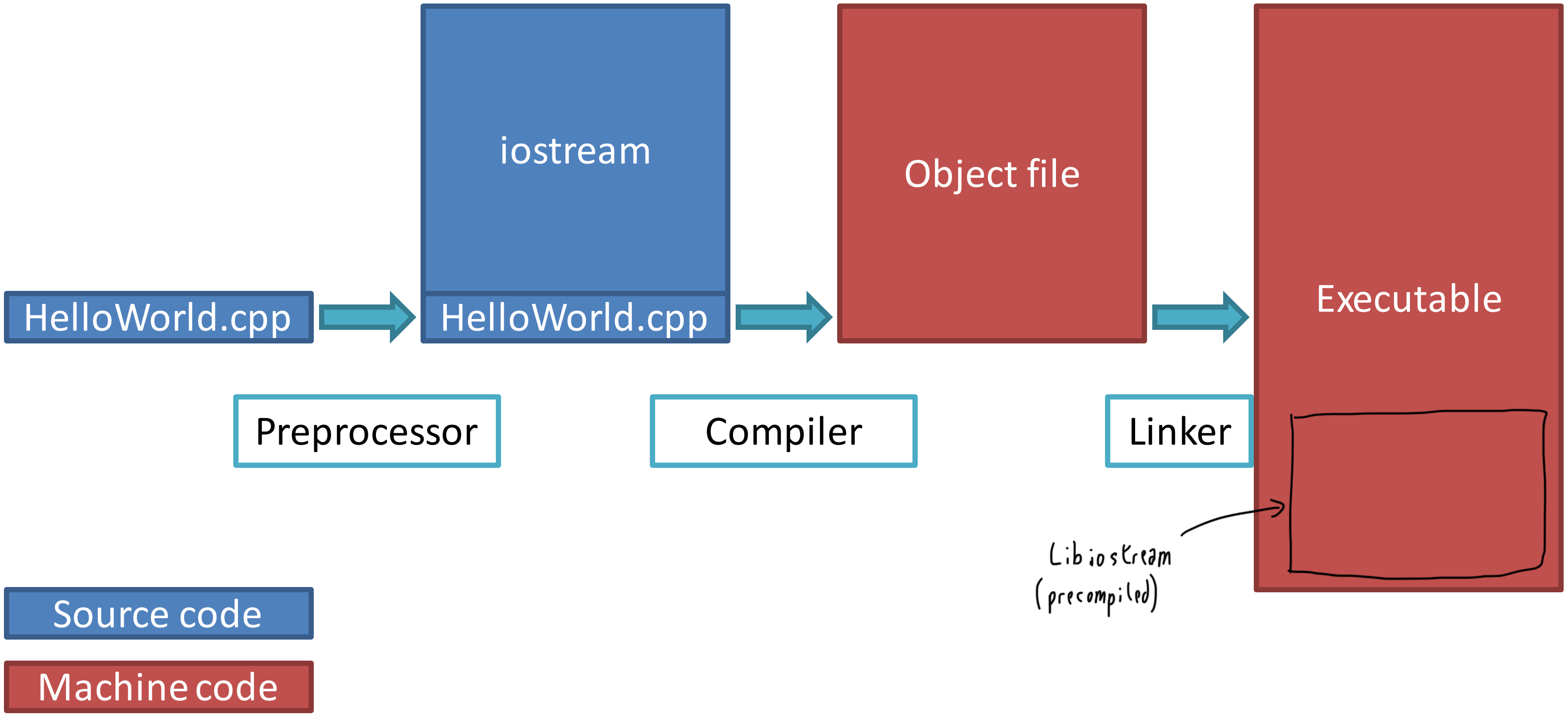
```
#include <iostream>

int main(void)
{
  std::cout << "Hello world!\n";

  return 0;
}
```

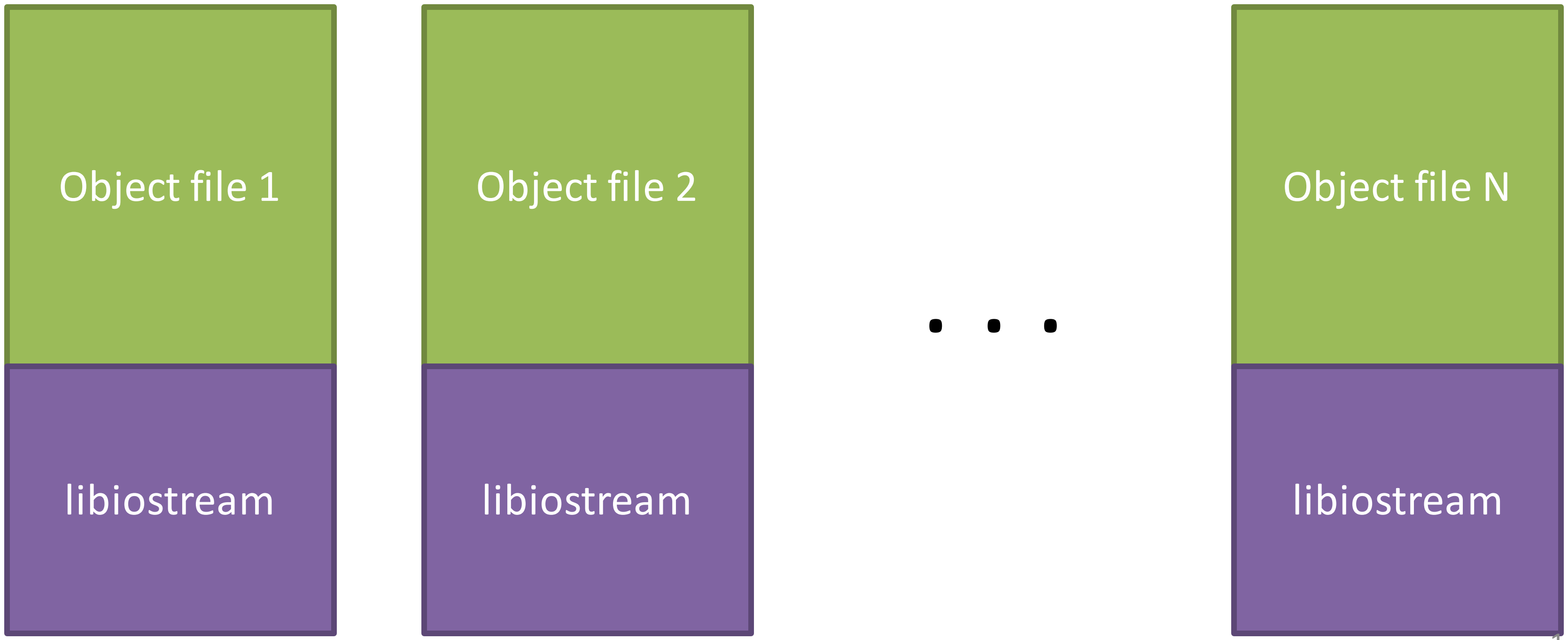


From source to executable



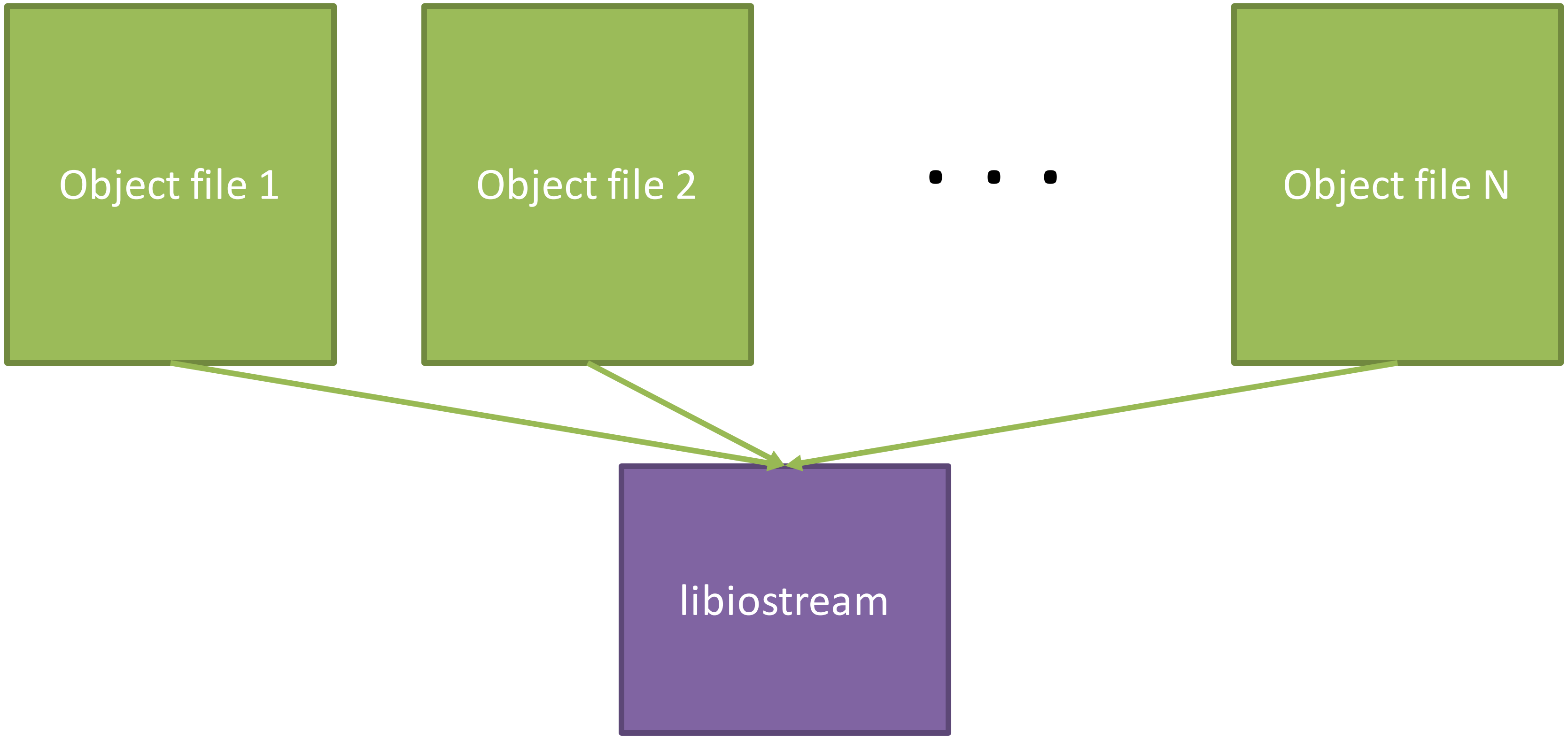
Static libraries

- What we have just seen is a static link (static library)
- What if multiple programs link the same library?



Dynamic libraries

- One single copy of the library



Must be located properly, or the executable can't find it at runtime

Static vs dynamic

- Reflect on this: What are the pros/cons?
- Dynamic libraries
  - save disk space (one installations serves all)
  - Can be recompiled without touching the executables
  - Called SO (Shared Object) under Linux or DLL (Dynamic Linking Library) under Windows
- Static libraries
  - Generate execs that can't be broken at a later stage
  - Are self-contained

UNIVERSITÀ DEGLI STUDI DI PADOVA

C++ source processing

Stefano Ghidoni