C++heatsheet(s)

Paolo Burgio
paolo.burgio@unimore.it

UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA High Performance Real Time Lab

“

Programming is a skill best acquired by practice and example rather than from books.

ALAN TURING

Outline

- My first *Hello world* in C++/Unix
  - GCC compiler flags
- Headers, libraries
  - Compilation chain
  - Namespaces
  - Other feature (macros)
- Arrays, vectors (stdc++), ..
  - Static vs. Dynamic memory
- AoB

My first *Hello world* in C++/Unix

Write your .cpp file, compile with GCC/G++

- `$ gcc helloworld.cpp –o helloworld`
- In Cygwin, produces helloworld.exe

GCC useful flags

- `–I <INCLUDE-FOLDER>` (capital ‘i’)
- `–l <LIBNAME>` (‘l’ di Livorno) : link specific library (libstdc++ => `–l stdc++`)
- `-o <EXEC–NAME>` [Default `a.out`]
- `–Wall` : enable all Warning messages

Headers, libraries

Include headers to let compiler find the symbols

- Es: We want to use `cout, endl`
- `#include <iostream>`

Grouped in namespaces

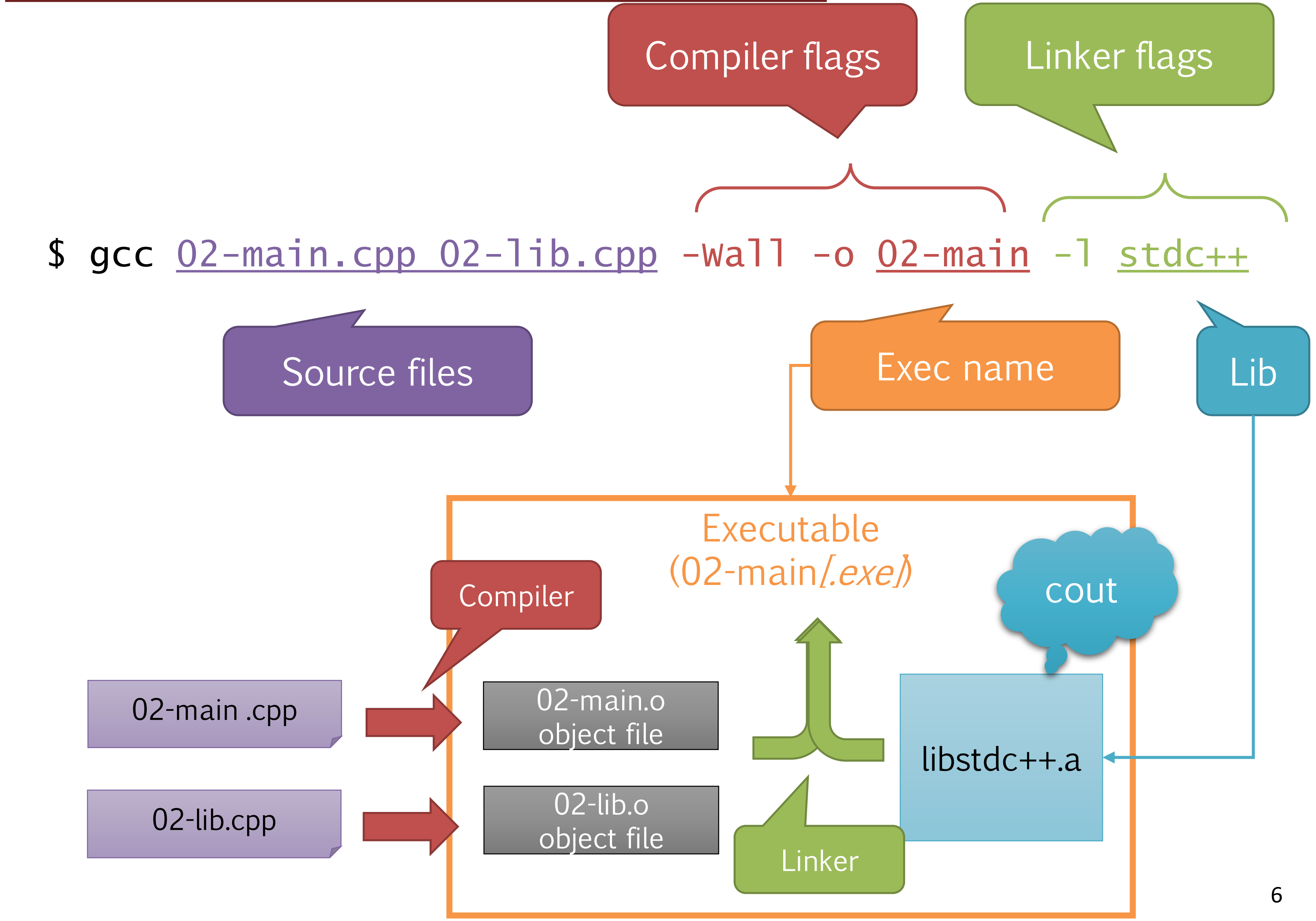
- `using namespace std;`
- `std::cout std::endl`

Then, link to library

- Es: to let linker (**ld**) find the **libstc++**
- `-l stdc++`
- `-l` **always last flag** in compilation line!

Compilation chain

```
$ gcc 02-main.cpp 02-lib.cpp -Wall -o 02-main -l stdc++
```

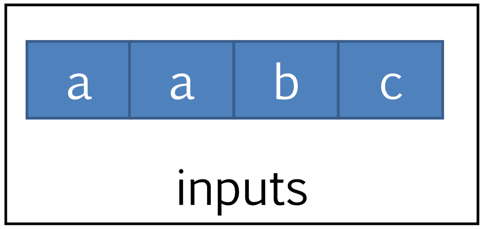


Arrays vs. pointers

```
const char inputs[]  = {'a', 'a', 'b', 'c'};
```

Type

Size is implicit in declaration



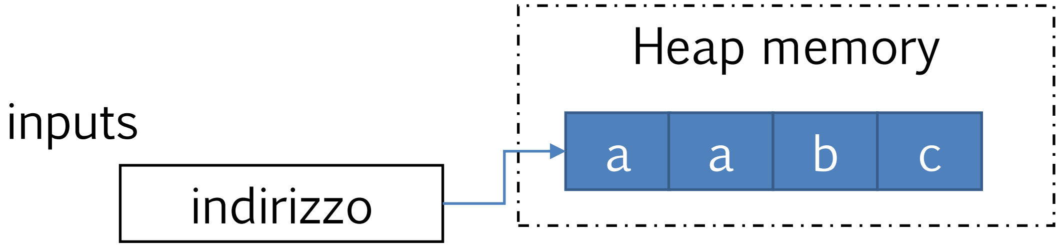
```
inputs[1]  = ...
```

Object on heap memory

```
char *inputs = new char[];
```

```
inputs[1]  = ...
```

Another variable (32- or -64-bit pointer type)



C vs. C++

	C	C++
Memory allocation	`malloc`	`new`
Memory disposal	`free`	`delete, delete[]`
Stdout	`printf`	`cout << …`
stdin	`scanf`	`cin >> ..`
Includes	`#include <stdio.h>`	`#include <iostream>`
Namespaces	N/A	`Using namespace std;`

Preprocessor macros

`#include <`*`SYSTEM-HEADER`*`>`

- Located in (default) system folders
- `/usr/include – /usr/local/include – /usr/share/include`

`#include "MY-HEADER.H"`

- Relative (to where you **compile**) or absolute path

`#define` ***`SYMBOL [VALUE]`***

- You can use this symbol: it's a macro/replacement, not a variable!
- You can check it exists
  - `#ifdef` *`SYMBOL`* `–` `#ifndef` *`SYMBOL`* `–` `#if defined(`*`SYMBOL`*`)`
  - `#endif`
  - Can comment away portions of code!!
- Also, can use **`–DSYMBOL`** in compilation line

Other stuff

› Ternary operator (used mainly in assignments)

```
<COND> ? <VALUE-IF-TRUE> : <VALUE-IF-TRUE>
```

```
bool b = true;
std::string s = b ? "TRUE" : "FALSE"; // s is "TRUE"
```

› Passing (and consuming) arguments to (within) your program

```
$ ./myprogram parameter
```

```
int main(int argc, char **argv)
{
  // argc is 2
  // argv[0] is "myprogram"
  // argv[1] is "parameter"
```

How to run the examples

Let's code!

- Find them in `Code/` folder from the course website

For C++: compile

```
$ gcc code.cpp -o code -Wall -lstdc++
```

Run (Unix/Linux)

```
$ ./code
```

Run (Win/Cygwin)

```
$ ./code.exe
```

References

Course website

- http://hipert.unimore.it/people/paolob/pub/Industrial_Informatics/index.html

My contacts

- paolo.burgio@unimore.it
- http://hipert.mat.unimore.it/people/paolob/

Resources

- Programmazione I course @FIM UNIMORE
  - https://algogroup.unimore.it/people/paolo/courses/programmazione_I/
- Practice, practice, practice
- A "small blog"
  - http://www.google.com