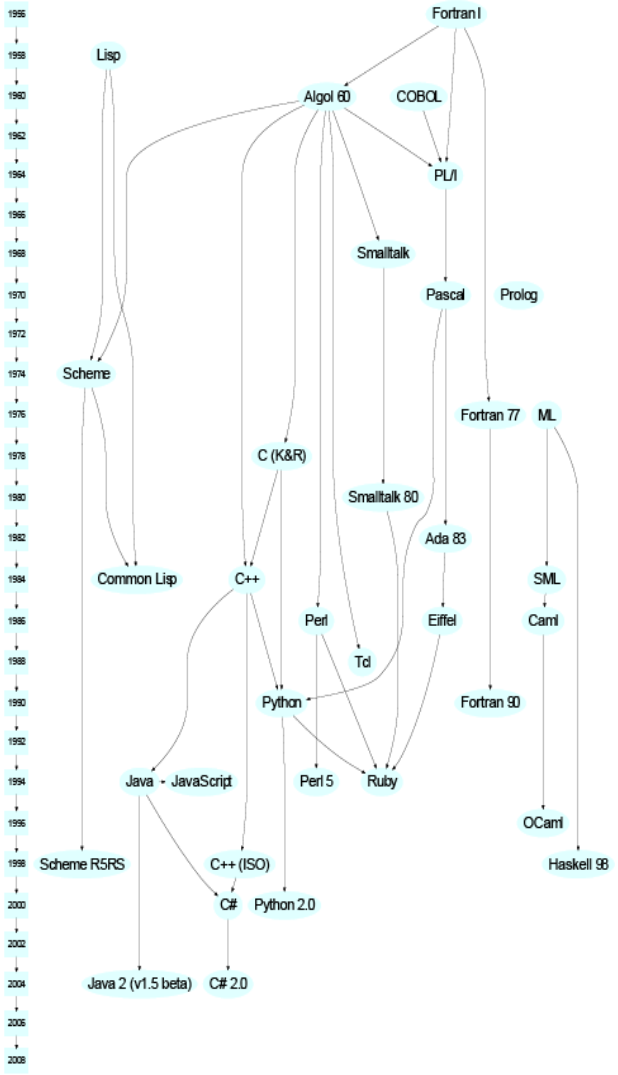
Introducing the C Language (for Java developers)

Antonio Barbalace
antonio.barbalace@ed.ac.uk

University of Edinburgh

Index

C Language History



- The C language development starts from ALGOL60.

  Mainstones and standards:

  1963 CPL
  1967 BCPL
  1969 B
  1971 C
  1978 C (Kernighan & Ritchie)
  1989 ANSI C (C89)
  1990 ISO C (C90)
  1996 ISO C (C95)
  1999 ISO C (C99)

- The Java language is quite new, inspired by Smalltalk-80, Cedar, Scheme84, Ada83 ANSI and Objective-C:

  1991 Oak
  1995 Java 1

  …

  2008 Java 2 (v1.4.2_18)

http://merd.sourceforge.net/pixel/language-study/diagram.html

Programming Languages

- Machine Languages
  instructions directly decoded by the CPU (i.e. strings of 1s and 0s);
- Assembly Languages
  the same as machine languages, but human readable;
- High Level Languages
  C, C++, Java, …

High level languages can be categorized in different ways depending not only on the language semantics…

Abstraction:

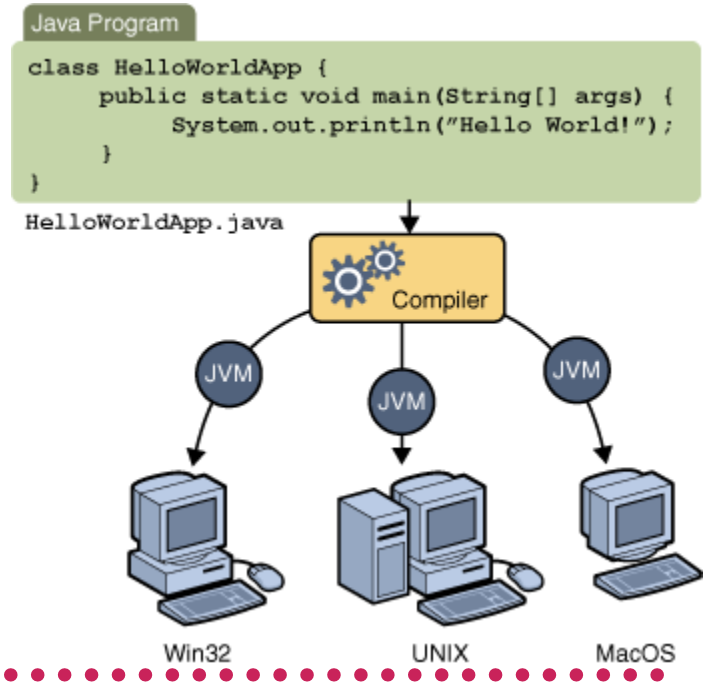
- Procedural (Fortran, C, Lisp, …)
- Object Oriented language (C++, Java, …)

Execution:

- Compiled (Fortran, C, C++, …)
- Interpreted (Java, C#, …)

Java Code Compilation and Execution

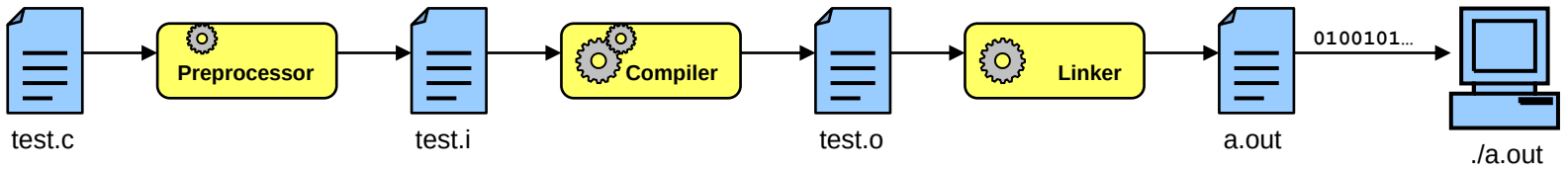
Java is an **interpreted** language. Source code is converted to architecture independent **bytecode**, which runs on any machine for which a Java Virtual Machine (JVM) exists.



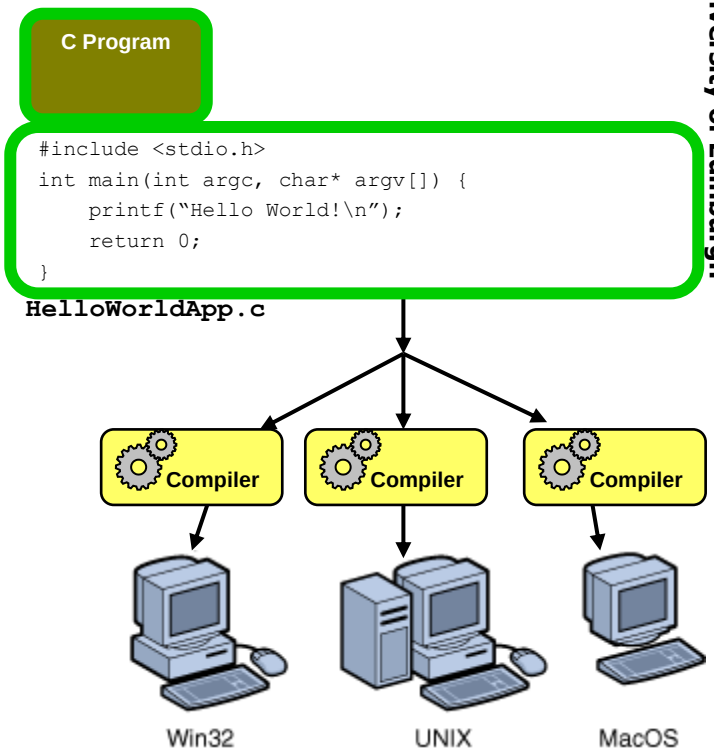
http://java.sun.com/docs/books/tutorial/getStarted/intro/definition.html

University of Edinburgh

C Code Compilation and Execution



C (and also C++) is a **compiled** language. It means that C code is converted, through more stages, into an architecture specific machine code (the same code will not run on different platforms).



C Program Compilation

Using GNU/GCC tools on UNIX like platforms:

```
# gcc -E hello.c
```

Output on the stdout the **preprocessed file** (`hello.i`)

```
# gcc -c hello.c
```

Output the **object file** `hello.o`

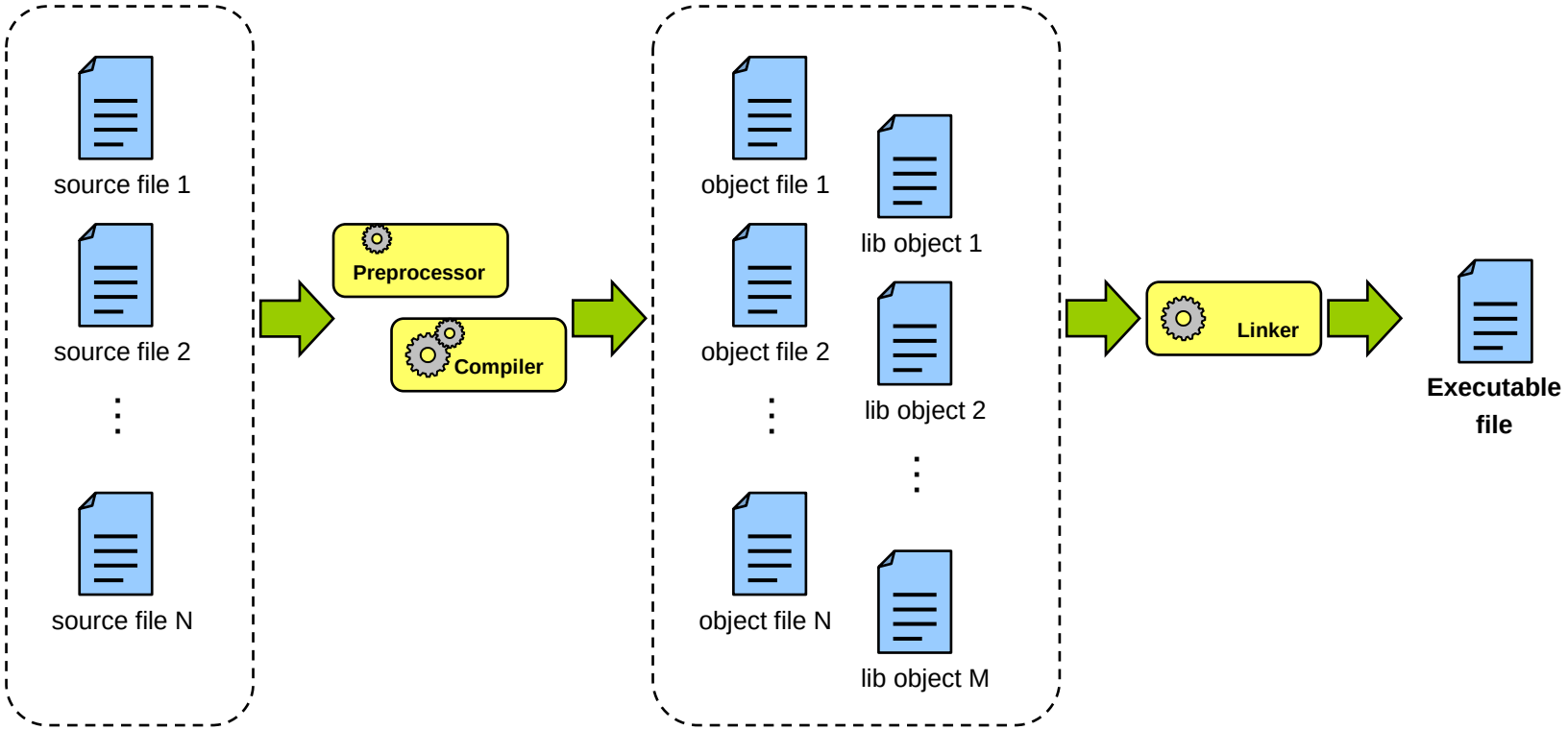
```
# gcc hello.c
```

Output **executable file** `a.out`

```
# gcc -o hello hello.c
```

Output **executable file** `hello`

C Program Compilation



Using GNU/GCC tools on UNIX like platforms:

```
# gcc -c source_file_1.c
# gcc -c source_file_2.c
# gcc -c source_file_N.c
# gcc -lmat -lpthread source_file_1.o source_file_2.o \
   source_file_N.o
```

C Program Execution

On UNIX like platforms:

```
# ./a.out
```

Or

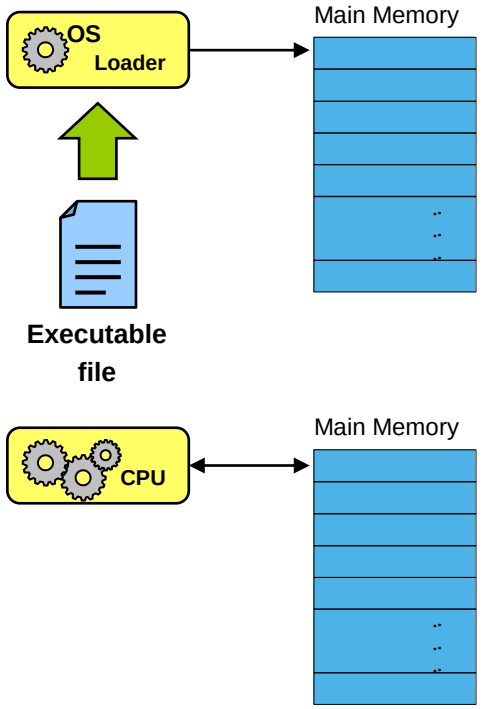
```
# a.out
```

Or

```
# ./ProgramName
```

Or

```
# ProgramName
```



The executable file is first **loaded** from the physical media into main memory and then the CPU **executes** the instruction at the starting address of the compiled code.

Compilation vs Interpretation

Compilation advantages

- Greater run time performance: generally much faster than Scheme or Java for comparable code (because it is optimized for a specific architecture);
- Better compilation time: enhancements in compilation procedure (`Makefile`) allow only modified files to be recompiled.

Compilation disadvantages (interpretation advantages)

- All compiled files (including executables) are architecture specific, depending on both the CPU type and the operating system;
- Compiled executables must be rebuilt on each new system; whereas interpreters of the same code may exist any platform.

Hello World

Hello World in Java

filename hello.java

```
public class hello {
  public static void main(String argv[]) {
    System.out.println("Hello World!");
    System.exit(0);
  }
}
```

compile as `# javac hello.java`

execute as `# java hello`

Hello World in C

filename hello.c

```
#include <stdio.h>
int main(int argc, char* argv[]) {
  printf("Hello World!\n");
  return 0;
}
```

compile as `# gcc hello.c`

execute as `# ./a.out`

Hello World

Hello World in Java

filename hello.java

```
public static void main(
String argv[]
argv.length
System.out.println("Hello World!");
System.exit(0);
```

Hello World in C

filename hello.c

```
int main(
char* argv[]
int argc
printf("Hello World!\n");
return 0;
```

- C doesn't need a class that contains the main procedure
- C code directly calls library functions
- Unlike Java, C has an explicit integer return value

C vs Java™ Overview

Java	C
Object Oriented	Doesn't know about objects; no built-in object abstraction; data is separate from methods
A Java program is a collection of classes	A C program is a collection of functions and global variables
Methods	Functions
References	Pointers
Automatic memory management (Garbage collector)	Manual memory management
Arrays are initialized to zero	Arrays are not initialized

C vs Java™ Overview

Where C lacks…

- In C arrays are not checked run time for index-out-of-bound access
- In C it is not possible to overload methods
- `boolean` and `byte` are not primitive data types in C
- No exception handling (library functions `setjmp()` and `longjmp()` may be used)

C vs Java™ Overview

What C adds...

- C has a preprocessor:
  - it is possible to declare MACROs, including header files (`*.h`)
  - conditional compilation of code segments may be specified ;
- C has the structure, the union and the enumeration abstraction;
- C has variable length argument lists.

Exercises

- Try to compile the `hello.c` file in your preferred C development environment

Bibliography

- http://merd.sourceforge.net/pixel/language-study/diagram.html
- http://en.wikipedia.org/wiki/C89
- http://en.wikipedia.org/wiki/C99
- http://home.tiscalinet.ch/t_wolf/tw/c/c9x_changes.html
- http://publications.gbdirect.co.uk/c_book/
- http://cslibrary.stanford.edu/
- http://cm.bell-labs.com/who/dmr/
- http://www.science.unitn.it/~iori/java/jindex.html

- Brian W. Kernighan, Dennis Ritchie, *The C Programming Language, 2° Edition,* Prentice Hall, 1988
- *JAVA in a Nutshell, O'Reilly (also via http://web.deu.edu.tr/doc/oreily/java/javanut/index.htm)*