1806ICT
Programming Fundamentals

Introduction to C

1

1

C

- C is an old language, it was created in 1973 for the Unix operating system
- C is a very low level language that allows it to be used for systems programming
- C is a compiled language which is compiled into machine code that is run directly on the CPU

2

2

C

- The core of C is relatively small and as a result C is also useful when resources are limited such as:
  - CPU
  - Memory
  - Storage
- C is primarily used in high performance applications, systems programming, interfacing with hardware, and embedded applications

3

3

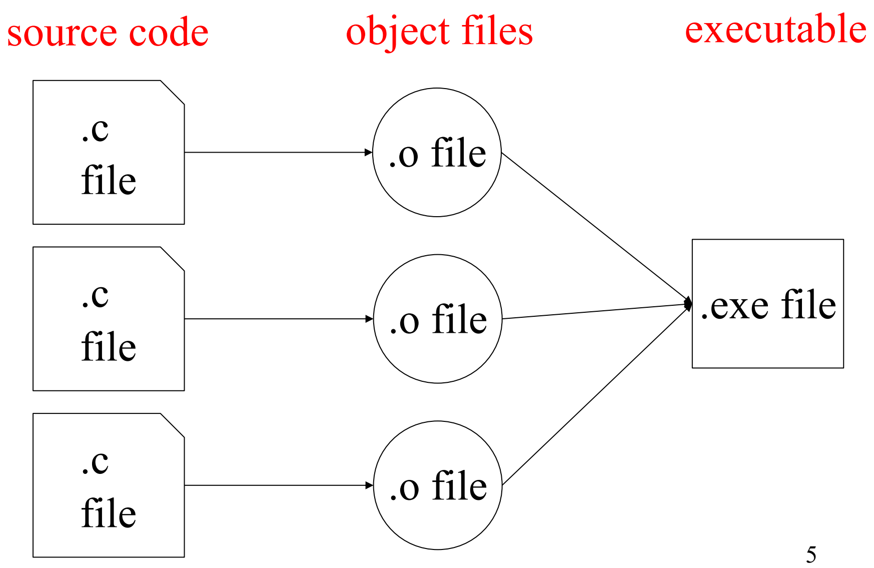
Source code to executable

- C programs are written in raw text files which we call source files or source code
- The source code is compiled into machine code which the CPU understands
- Each C source file is first compiled into an object file
- The object files are then linked together into an executable or library file

4

4

Source code to executable



5

5

First C Program

```
#include <stdio.h>

int main()
{printf("Hello World\n");
 return 0;
}
```

6

6

Syntax

- Programming languages have a syntax
- Syntax is the rules you must follow when writing code
- If you don't follow the syntax precisely the compiler will generate an error
- We will introduce new syntax throughout the course

7

7

Compiling a C file

- Integrated Development Environments (IDEs) are commonly used when developing software
- However, the command line is still commonly used and it is important to know how to compile a C program via the command line

8

8

GCC

- A commonly used command line compiler is GCC
- GCC stands for Gnu Compiler Collection
- Gnu (Gnu is Not Unix)
- There are other command line compilers such as clang and Intel and Microsoft compilers

9

9

Using GCC

- If we place our source code example on the previous slide into a file called main.c, we can compile it using the following (assuming we are in the same directory):

```
        gcc main.c
```

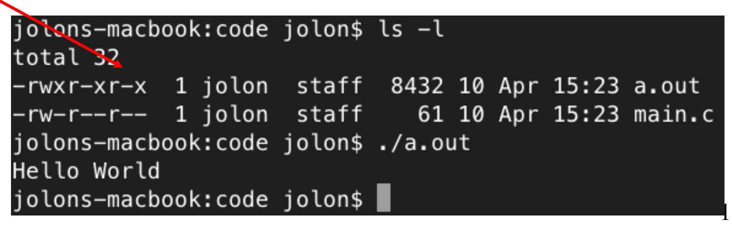
10

10

Using GCC

- This will produce an executable file called a.out
- On Unix we can execute it by typing ./a.out:

x means executable



```
jolons-macbook:code jolon$ ls -l
total 32
-rwxr-xr-x  1 jolon  staff  8432 10 Apr 15:23 a.out
-rw-r--r--  1 jolon  staff    61 10 Apr 15:23 main.c
jolons-macbook:code jolon$ ./a.out
Hello World
jolons-macbook:code jolon$
```

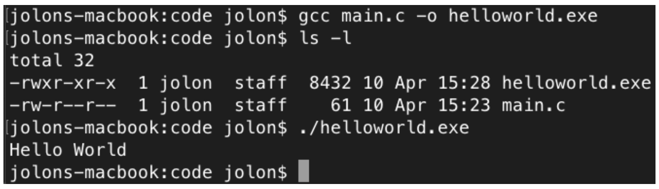
11

11

Using GCC

- We can instruct GCC to produce an executable with the specified name using -o:

```
gcc main.c –o helloworld.exe
```



```
jolons-macbook:code jolon$ gcc main.c -o helloworld.exe
jolons-macbook:code jolon$ ls -l
total 32
-rwxr-xr-x  1 jolon  staff  8432 10 Apr 15:28 helloworld.exe
-rw-r--r--  1 jolon  staff    61 10 Apr 15:23 main.c
jolons-macbook:code jolon$ ./helloworld.exe
Hello World
jolons-macbook:code jolon$
```

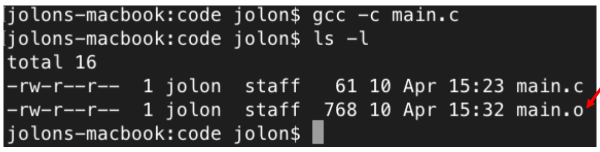
12

12

Object files

- Note that GCC skips the object files and compiles straight to the executable
- If we want to produce the object files we can use –c:

```
gcc –c main.c
```



```
jolons-macbook:code jolon$ gcc -c main.c
jolons-macbook:code jolon$ ls -l
total 16
-rw-r--r--  1 jolon  staff   61 10 Apr 15:23 main.c
-rw-r--r--  1 jolon  staff  768 10 Apr 15:32 main.o
jolons-macbook:code jolon$
```

object file

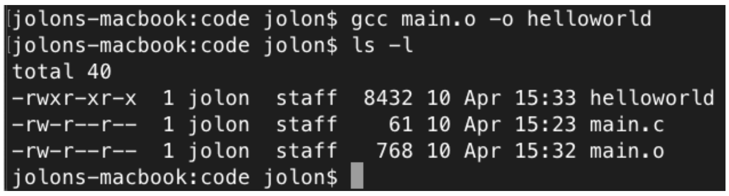
13

13

Linking

- We can link object files into an executable by simply compiling the object files:

```
gcc main.o –o helloworld
```



```
jolons-macbook:code jolon$ gcc main.o -o helloworld
jolons-macbook:code jolon$ ls -l
total 40
-rwxr-xr-x  1 jolon  staff  8432 10 Apr 15:33 helloworld
-rw-r--r--  1 jolon  staff    61 10 Apr 15:23 main.c
-rw-r--r--  1 jolon  staff   768 10 Apr 15:32 main.o
jolons-macbook:code jolon$
```

14

14