Lab 2 (CS): Introduction to Operating Systems

Giancarlo Succi, Leonard Johard

Innopolis University
Course of Operating Systems

Week 02 – Lab

Recap.

- Which command is used to create a folder.

Recap.

- Which command is used to create a folder.
  - **mkdir** *folder_name*

Recap.

- Which command is used to create a folder.
  - **mkdir** *folder_name*
- How to execute a command as a background process.

Recap.

- Which command is used to create a folder.
  - **mkdir** *folder_name*
- How to execute a command as a background process.
  - *command_line* &

Recap.

- Which command is used to create a folder.
  - **mkdir** *folder_name*
- How to execute a command as a background process.
  - *command_line* **&**
- How to redirect the output stream of a command line to a file. What about appending the output?

Recap.

- Which command is used to create a folder.
  - **mkdir** *folder_name*
- How to execute a command as a background process.
  - *command_line* &
- How to redirect the output stream of a command line to a file. What about appending the output?
  - > *file_name* (overwriting)
  - >> *file_name* (appending)

Recap.

- Which command is used to create a folder.
  - **mkdir** *folder_name*
- How to execute a command as a background process.
  - *command_line* **&**
- How to redirect the output stream of a command line to a file. What about appending the output?
  - > *file_name* (overwriting)
  - >> *file_name* (appending)
- How do you compile a program in C?

Recap.

- Which command is used to create a folder.
  - **mkdir** *folder_name*
- How to execute a command as a background process.
  - *command_line* **&**
- How to redirect the output stream of a command line to a file. What about appending the output?
  - > *file_name* (overwriting)
  - >> *file_name* (appending)
- How do you compile a program ex.c in C?
  - **gcc** *ex.c* **-o** *ex*

Lab Objectives

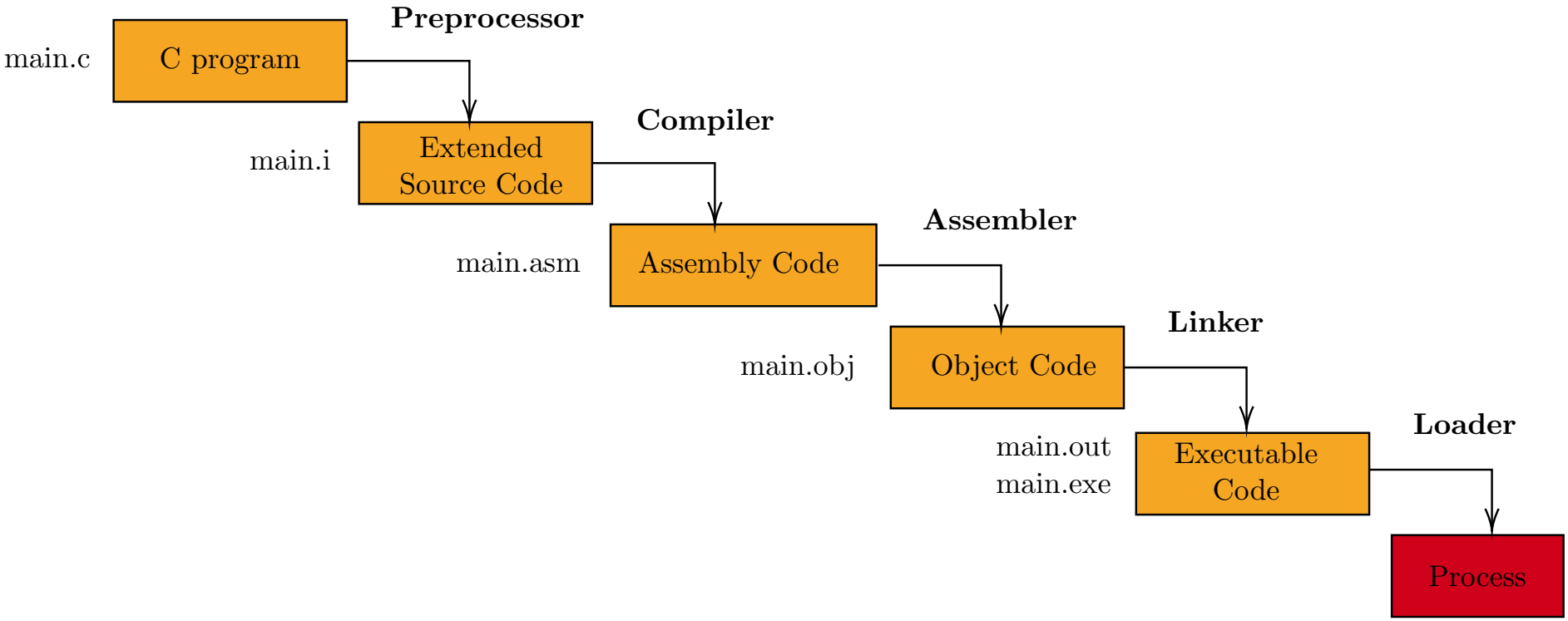
- You will learn how C program is executed from preprocessing to linking.
- You will learn how to read input from and write output in standard streams.
- You will learn how to define variables and work with control statements in C.
- You will do some exercises in C language.

Hello World

"Hello world" program in C language.

```c
#include <stdio.h>
int main(void)
{
   printf("Hello World!");
   return 0;
}
```

Execution flow of C program



Execution flow of C program

- Example
  - Write "hello world" program in C. Save the program in the file *main.c*.
  - Only preprocess the source code and save the output in a file *main.i*.
  - Then compile only (do not assemble or link) the source code and save the output in a file *main.asm*. Look at the assembly code of your program.
  - Then compile and assemble but do not link the source code and save the output in a file *main.obj*. Try to execute the output file. Is it a text file or binary file?
  - Then compile, assemble and link the source code and save the output in a file *main.out*. Can you execute it now?

Datatypes in C

- Primitive
  - [*unsigned* | *signed*][*short* | *long* | *long long*] **int**
  - [*unsigned* | *signed*] **char**
  - **float**
  - [*long*] **double**
  - **void**
- Derived
  - Arrays
    - String
  - Structures
  - Pointers
  - ...etc

Note: You can get the size of any datatype by using the keyword ***sizeof*** which acts as a function that accepts a parameter and returns its size in bytes.

Exercise 1

- Create a program *ex1.c* that declares integer, unsigned short integer, signed long int, float and double variables.
- Find out how to assign maximum values for each variable (hint: use `INT_MAX` for integer, etc.)
- Print sizes and values of each variable.
- Write a script *ex1.sh* to run your program.
- **Note:** you should submit the file *ex1.c* and the script *ex1.sh*.

Input/Output in C

Output:

- Character output
  - putchar
  - puts
  - fputs
  - fputc
- Formatted output
  - printf
  - fprintf

Input:

- Character input
  - getchar
  - gets
  - fgets
  - fgetc
- Formatted input
  - scanf
  - fscanf

Hint: you can get the user manual of these functions on your shell program by running the command line **man 3 function_name**

Reference guide: https://cplusplus.com/reference/clibrary/

Input/Output in C

Example:

- Write a program which reads the user name using the function *gets* and prints it. Is it safe to use this function? Suggest a safer way to read a string from *stdin*?
- How to declare a string which holds the value "OpSys"? (Declare it in two ways, as a single literal "*string**0*" and as an array of characters {'*ch1*','*ch2*',...etc, '*0*'})
- How to read a float number from *stdin* and how to write to *stdout*? How to read a hexadecimal number from *stdin* and how to write to *stdout*?
- Can we use formatted input to read a string from *stdin*?
- How to convert the string "2022" to an integer 2022 and vice versa?

Exercise 2

- Write a program *ex2.c* that asks the user to type a string character-by-character until **dot (.)** character is entered and prints its reverse with double quotation.
  - Example

Input	**Output**
C language**.** is easy and simple.	"egaugnal C"

Example on Exercise 2 Input/Output

- Write a script *ex2.sh* to run your program.
- Hints:
  - a string in C is an array of chars (more about arrays next week)
  - use `strlen()` function to get the length of a string
- **Notes:**
  - You should submit the file *ex2.c* and the script *ex2.sh*.
  - The maximum size of the string is 256.
  - If the user did not type **dot (.)** character, then pressing **Enter key** will be accepted as terminating character.

Control Statements and functions in C

- Control statements
  - if/else
  - for (initialization; condition; step)
  - while (condition)
  - do ... while(condition);
  - break; continue; goto;
- function

  return_type *func_name*(params){
  ...
  function_body
  ...
  }

Exercise 3 (1/2)

- In computer systems, the data is stored in the memory as 0s and 1s which forms a number in binary system. We use different numeral systems since reading these binary numbers is difficult for humans. The most common number systems are binary, decimal, octal, and hexadecimal systems.
- In this exercise, you have to write a function **convert** which converts a given number x from a numeral system s to another numeral system t where t, s are numbers in the range [2-10]. If the given number is wrong or s or t are out of the previous range then we should print the error message **"cannot convert!"**.

  For instance, the function call **convert(1234, 8, 2)** will convert the given number *1234* from the octal (8) system to the binary (2) system and prints *1010011100*.

Exercise 3 (2/2)

- Write a program *ex3.c* which uses the function **convert**, reads a long long number and the source and target number system specifiers from the user, then it should print the converted number or error message in case of errors.
- Write a script *ex3.sh* to run your program.
- **Notes:**
  - You should submit *ex3.sh* and *ex3.c* which contains the function **convert**.
  - We assume that the user enters a non-negative number.
  - using arrays, structures or pointers is not allowed in this exercise. But you can use an array of characters which is a string.
  - The numerals in the number systems are represented by decimal numbers [0-9]. For instance, in the number system 7, we have the numerals [0, 1, 2, 3, 4, 5, 6].
  - We did not specify the return type of the function **convert**.
- **Hint: use `sscanf()` to convert string to int and `sprintf()` to convert int to string.**

Exercise 4 (1/2)

- Write a function **count** which returns the number of occurrences of an input character in a string.
- Example:

Input	Output
Innopolis, i	i:2
Innopolis, m	m:0

Example on **count** function input/output

- Write a function **countAll** which prints the number of occurrences of each character in the input string.
  - **Hint:** Use your function **count**.
- Example:

Input	Output
Innopolis	i:2, n:2, n:2, o:2, p:1, o:2, l:1, i:2, s:1

Example on **countAll** function input/output

Exercise 4 (2/2)

- Write a program *ex4.c* that accepts an input string from the command line and prints the number of occurrences of all characters in the input string.
- Write a script *ex4.sh* to run your program.
- **Notes:**
  - using arrays, structures or pointers is not allowed in this exercise. But you can use an array of characters which is a string.
  - The maximum size of the string is 256.
  - your program should be case-insensitive. This means that characters 'i' and 'I' are treated as a single character.
  - you should submit *ex4.sh* and *ex4.c* which contains the functions **count** and **countAll**.

Exercise 5

- The Tribonacci sequence T_n is defined as follows:

$$T_n = \begin{cases} 0 & n = 0 \\ 1 & 1 \leq n \leq 2 \\ T_{n-1} + T_{n-2} + T_{n-3} & n \geq 3 \end{cases}$$

- Write a function **tribonacci** that takes as argument n and returns the value of T_n $(0 \leq n \leq 37)$
- You are neither allowed to use arrays nor function recursion
- Write a program *ex5.c* which calls the above function with arguments 4, 36 and print the output to standard output.
- Submit your ex5.c file accompanied by an ex5.sh file to compile and execute ex5.c

Exercise 6 (optional)

- Write a program which prints the following patterns. examples are:

*	*	*******
**	**	*******
***	***	*******
****	****	*******
*****	***	*******
******	**	*******
*******	*	*******

End of lab 2 (OS)