Mohamed Boudiaf University – M'sila
Department of Computer Science
3rd year, Computer Science - SI

Practical Session N°1

Process Handling in C under Linux

Objective: The aim of this practical session is to help students learn how to create and manipulate processes in Linux, understand parent–child relationships, and recognize the importance of process synchronization.

Basic Concepts :

fork() : Process Creation	**sleep() :** Sleep for a Specified Duration
getpid() – Get Process ID	**getppid()** – Get Parent Process ID
pause() : Sleep Until Signal	**kill() :** Send a Signal to a Process
wait() : Wait for Child Termination	**exit() :** Terminate a Process

- **fork()** may fail if there is insufficient memory or if the user has already created too many processes. In such cases, no child process is created, and fork() returns -1.
- To compile a C program on Linux, use the **gcc** compiler:

Compilation: gcc -o your_program your_program.c
Execution: ./your_program

Exercise 1:

Write a C program that creates 3 child processes using a **for loop**. For each iteration, the program should display the following information: **i = value of i, I am process: pid, my parent is: ppid**
The **parent process** should print the message "END"**,** after all child processes have finished**.**

Exercise 2:

Write a C program that creates two child processes; Child 1 prints the integers from 1 to 50, Child 2 prints the integers from 51 to 100. The parent process prints "End of processing".

- What do you notice about the output?
- Modify the program so that the numbers are printed in order from 1 to 100.

Note: Add **fflush(stdout)**; after each **printf** in your program to force immediate output and observe how the processes interleave.

Exercise 3:

Write a C program with a global counter starting at zero and create two child processes Child 1 and Child 2. Each child should increment the counter 5 times and print its value. Each increment should be printed along with the child's identifier. The parent process should wait for both children to finish and then print the final counter value.

- What do you notice about the output?
- Modify the program so that the counter increments in a fixed order, alternating between Child 1 and Child 2.

Good luck