
PL1 Class Script

- Explain how the classes will work
    - Lab exercises -> 20%
    - Project -> 30%
    - Groups of 2 students should commit their code on a git repository (bitbucket preferably) so that their work can be assessed
    - Commits considered for evaluation are the ones that are made until the end of weeks 4, 8, 12 (Sunday being the last day of the week)

- Topics to be covered during the semester:
    Weeks 2-4: Processes/Threads (27/02/2023 - 19/03/2023)
    Weeks 5-8: Synchronization   (20/03/2023 - 16/04/2023)
    Weeks 9-12: Kernel Development (17/04/2023 - 14/05/2023)

- Process management; its importance and how it works in Linux
- Present the most important system calls to manage the processes (fork(), wait(), waitpid(), exit(), getpid(), getppid())

- contact: crr@isep.ipp.pt

Exercises
---

1. Consider the following code:

```
int main(void)
{
    int x = 0;
    pid_t p = fork(); /*pid_t: sys/types.h; fork(): unistd.h*/

    if (p == 0) {
        x = x+2;
        printf("Step 1. x = %d\n", x);
    } else {
        x = x-2;
        printf("Step 2. x = %d\n", x);
    }
    printf("Step 3. %d; x = %d\n", p, x);
}
```

a) What is the output of this code in the terminal? Justify your answer.
b) Please explain the order in which the printf() calls appear in the terminal.

2. Consider the following code:

```
int main(void) {
    fork();
    fork();
    printf("MESCC");
    fork();
    printf("MESCC");
}
```

a) How many processes are created when executing the code above?
b) Draw a process tree that represents the execution flow of the code above.
c) How many times is "MESCC" printed?

3. Consider the following code:

```
int main(void) {
```

```
    int a=0, b, c, d;
    b = (int) fork();
    c = (int) getpid(); /* getpid(), getppid(): unistd.h*/
    d = (int) getppid();
    a = a + 5;
    printf("\na=%d, b=%d, c=%d, d=%d\n", a, b, c, d);
}
```

a) What are the values of the variables defined in the program above?
b) Do you observe any relation between the values of the variables? Please justify your answer.

4. Consider the following code:

```
void main()
{
    int i;
    int status;

    for (i = 0; i < 4; i++) {
        if (fork() == 0) {
            sleep(1); /*sleep(): unistd.h*/
        }
    }
    printf("End of execution!\n").
}
```

a) How many processes are created by the above code?
b) Please modify the code so that exactly 4 child processes are created?
c) Assuming the changes in b), modify the code so that the parent process waits for child processes with an odd process ID.
d) Assuming the changes in b) and c), modify the code so that the child processes return a number that shows their creation order
(the first child process returns 1, the second returns 2, and so on).

5. Write a program that initializes an array of integers with 500 random integers in the range [0,255].

a) The program should create 5 child processes that will concurrently find the local maximum value of a part of the array and return this value to the parent process;
b) The parent process should sum the local maximum values and print the result on the terminal.
c) What is the difference between this approach and an appraoch with a single process. Which one is more advantageous? Justify your answer.

6. Write a program that does the following:
    - the parent process creates a new child process;
    - the new child process generates a random number between 0 and 5 and returns that value to the parent;
    - upon reading the return value, the parent process should create that number of child processes;
    - each child process should print the number of its creation and exit;
    - the parent should wait for all child processes and exit.