Operating Systems

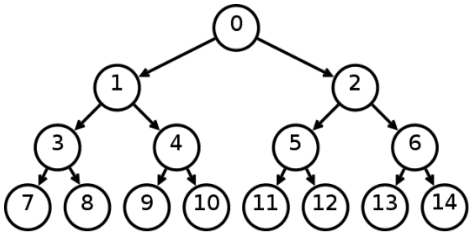
Lab 3 Exercise – Processes, child processes and fork()

Learning goals: this laboratory activity is useful to understand how to create child processes using the fork function. fork() creates a new process by duplicating the calling process.
The new process, referred to as the child, is an exact duplicate of the calling process, referred to as the parent.
On success, the PID of the child process is returned in the parent, and 0 is returned in the child.
On failure, -1 is returned in the parent.

Exercise 1

Implement a C program that receives a command line parameter: **n**. The parent process creates two children and terminates. Each child creates another two children, and terminates.

Process creation stops after 2^n leave processes have been created. For example, if **n=3**, the parent creates two children, and each child creates another two processes for a total number of 8 leaves processes. At this point process creation stops.



15 processes

Initially the main process prints its PID.
Then, only the last set of children, the leaf children, prints their PIDs
Which is the order of process termination? Is the order always the same?

Exercise 2

Implement a C program, **`generation_tree.c`**, which receives a command line parameter: **n**.
The parent process, pushes its PID in a vector, creates two children and terminates. Each child, pushes its PID in its vector, creates another two children, and terminates.

Process creation stops after 2^n leave processes have been created.

Each leaf child pushes its PID in its vector, and print its generation tree, i.e., the sequence of its saved PIDs. Example:

```
> generation_tree 3
Process tree: 5904 5906 5908 5911
Process tree: 5904 5906 5908 5912
Process tree: 5904 5906 5907 5913
Process tree: 5904 5905 5910 5915
Process tree: 5904 5905 5910 5916
Process tree: 5904 5905 5909 5917
Process tree: 5904 5905 5909 5918
Process tree: 5904 5906 5907 5914
```

Exercise 3

Implement a C program, **`generation_tree_number.c`**, which receives a command line parameter: **`n`**. The parent process, pushes its number (1) in a vector, creates two children and terminates. Each child, pushes its in its vector, creates another two children, and terminates.

Process creation stops after 2^n leave processes have been created.

Each leaf child pushes its number in its vector, and print its generation tree, i.e., the sequence of its saved PIDs. Example:

```
> generation_tree 3
Process tree: 1 3 7 15
Process tree: 1 2 5 11
Process tree: 1 3 7 14
Process tree: 1 2 5 10
Process tree: 1 3 6 13
Process tree: 1 2 4 9
Process tree: 1 3 6 12
Process tree: 1 2 4 8
```

Hint:
Draw the tree for n=3 and check that the number of the process at the second level begins by 2, the number of the process at the third level begins by 4, number of the process at the fourth level begins by 8 etc..)