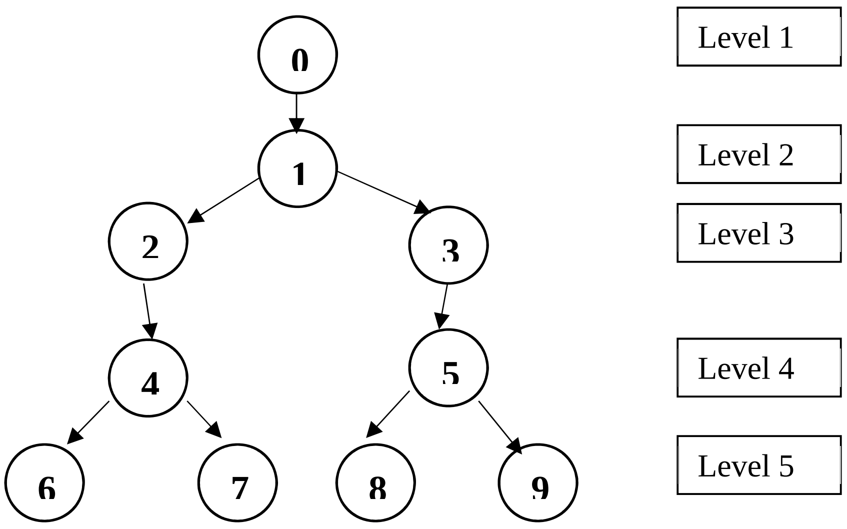
Operating Systems

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

Learning goals: this laboratory activity is useful to understand how to create child processes using the fork function, and to understand which variables are shared among processes.

Exercise 1

Implement a C concurrent program that generates a process tree of height `n`. Each process at the odd level of the tree (1, 3, 5, etc.) must generate 1 process, and terminate. Each process at an even level of the tree (2, 4, 6, etc.) must generate two processes, and terminate.
Therefore, the initial process (at level 1) must generate a single process, which is at level 2 must generate 2 processes. Each of these two processes generated (at level 3) must generate one process, and so on so forth. **The processes on the last level of the tree (the leaves) must print their PID.** The program receives the value of `n` from the command line. Example with `n=5`



```
> process_tree 5
Process 2204
Process 2206
Process 2208
Process 2211
```

Exercise 2

Implement a C program, `generation_tree_number.c`, which receives a command line parameter: `n`. The program must create the same tree of Exercise 1, but the main process, pushes its PID in a vector, creates its child, and terminates. The child process inherits the vector, and pushes its own PID in this vector, creates another two children, and terminates. Each created process performs the same steps, according to its level. **Process creation stops after `all processes of level n 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 5
Process tree: 5343 5344 5347 5349 5353
Process tree: 5343 5344 5347 5349 5352
Process tree: 5343 5344 5346 5351 5355
Process tree: 5343 5344 5346 5351 5356
```