

System Programming

2nd Laboratory (6 and 8 March 2019)

During this laboratory students should read and understand the man pages of several commands and functions. These functions are listed in the end of each exercise

I

Implement a program that creates 10 children processes.

Each of these children should sleep for a random number of seconds (between 0 and 10 seconds).

This random value is generated in the child process.

At the end of the execution, each child should print its PID and the number of seconds it was asleep.

Use the following functions:

- **fork**
- **random**
- **sleep**

II

Change the first program so that at any given time there are always 10 children processes running.

The parent process should also print the total sleep time for each child (immediately after its dead):

- Each child should return the number of seconds it was asleep.
- The parent should wait for the death of each of its child and retrieve the returned value.
- The parent should print the information of the dead children (PID and sleep time) before starting a new process.

Use the following functions:

- **exit**
- **wait**

III

Implement a program that continuously prints increasing numbers, and at random intervals (from 0 to 10 seconds) resets the counter.

Use the following functions:

- **alarm**
- **sigaction** or **signal**

IV

Modify the last exercise of the previous laboratory so that 4 child processes are created and each child is responsible for finding and printing the multiples of one number:

- The first child will print the multiples of 2
- The first child will print the multiples of 3
- The first child will print the multiples of 5
- The first child will print the multiples of 7

The parent will wait for the completion of all the children.

Observe the execution time of the two versions of the program:

- Serial version developed in last laboratory
- Parallel version developed in this exercise

Use the following command:

- **time**

V

Modify the previous program so that it is possible to know how many milliseconds each process was running.

Use the following functions:

- **clock_gettime**

VI

Observe the **lab-2-mult-debug.c** file.

What does it do?

Compile the file: **gcc lab2-mult-debug.c -g -o mult-debug**

Run the program using the command **mult-debug &**

Run the command **ps f**

What is the state of the **mult-debug** processes?

Send the signal SIGCONT (kill -SIGCONT xxxxx) to the various processes. After each kill observe the result of **ps f**

Send the other signals to the various processes. After each kill observe the result of **ps f**

Look at the following man pages:

- **kill**
- **ps**

VII

Since each of the processes are stopped, it is now possible to run the debugger, attach it to any of such processes and resume computation following the following steps:

- start the debugger running **ddd mult-debug**
- start the processes issuing the command **run**
- take note of the PID of the process you want to debug
- execute **ps f** in the terminal
- detach from the current process issuing the command **detach**
- attach to one of the processes command **attach** or:
 - **menu File -> menu attach -> select process**
- press **Step** (3 times) until the program restarts executing step by step just like in a regular debug session

if it is impossible to attach to a running p[rocesse read the follwomg page:

- <https://github.com/Microsoft/MIEngine/wiki/Troubleshoot-attaching-to-processes-using-GDB>