Exercise sheet 5 – Process/Thread

Goals:

- Process management
- Thread management

Exercise 5.1: Process management

(a) List all running processes.

(b) What is the meaning of the ‘`x`’ flag?

(c) What information do you find for each process?

(d) How many processes are running?

(e) Which processes were created first?

(f) Why are gaps between the PIDs (process IDs)?

(g) What is the lowest PID, and what is the meaning of this process?

(h) What is the meaning of the ‘`-p`’ flag of `pstree`?

(i) What are the parent and grand parent processes of `pstree`?

Exercise 5.2: Process information

(a) Update the `OS_exercises` repository with `git pull`

```
cd OS_exercises
git pull
cd ~
```

(b) Start the program `OS_exercises/sheet_05_processes/demo_program`.

(c) Find the process ID (PID) of the running `demo_program`. You may need a separate shell for that.

(d) How many CPU percentage and memory does the process use?

(e) Try to stop the `demo_program`.

Exercise 5.3: Process creation

The file `OS_exercises/sheet_05_processes/process/processCreation.c` provides a skeleton for this exercise.

(a) Create `N` processes.

(b) Each process works something: we simulate that by calling the `work()` function, which sleeps for 20 seconds.

(c) Before a process ends, it increases the `counter`.

(d) The main (parent) process waits until all its child processes have been finished.

(e) After all processes have been finished: it prints the value of the `counter` and exits.

(f) Change into the folder `OS_exercises/sheet_05_processes/process` (if you aren't already) and compile the program with `make`

(g) Start the program with `./processCreation N` (N stands for the number of processes to create). What is the value of the counter and what have you expected?

Exercise 5.4: Thread creation

The file `OS_exercises/sheet_05_processes/thread/threadCreation.c` provides a skeleton for this exercise.

(a) Create `N` threads. Each thread calls the `work()` function, which simulates working by sleeping for 20 seconds.

(b) Before a thread ends, it increases the `counter`. Add this to the `work()` function.

(c) The main thread waits until all its created threads have been finished.

(d) After that: it prints the value of the `counter` and exits.

(e) Change into the folder `OS_exercises/sheet_05_processes/thread` (if you aren't already) and compile the program with `make`

(f) Start the program with `./threadCreation N` (N stands for the number of threads to create). What is the value of the counter and what have you expected?

(g) Can you identify some problems that may occur, if the threads access the `counter` variable in parallel?