Exercise sheet 6 – Process synchronisation 1

Goals:

- Understand synchronisation issues
- Use semaphore for mutual exclusion
- Use Lock-Files for mutual exclusion

Exercise 6.1: Synchronisation problem analysis: theoretical

Consider two processes that counts information. Each process works independently. There is a `counter` file that hold the current state of the `counter`. Every time a process counts something, it opens the `counter` file, reads the current value, increases the value by one, and finally writes the new `counter` value into the `counter` file.

(a) Create a drawing that illustrates the situation (as simple as possible).

(b) Write pseudocode to further illustrate the work of each process (as simple as possible).

(c) What will happen if both processes works as described?

(d) How could you solve the issue? Extend your pseudocode to solve the issue. Hint: you may use `P()/V()` operations.

Exercise 6.2: Synchronisation problem analysis: practical

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

(b) Change into the `OS_exercises/sheet_06_process_sync1/counting_sem` directory.

(c) Inspect the `counting_process.c` file.

(d) If you would start two processes of the `counting_process` and the initial value in the `counter` file was 0, which value should be in the `counter` file after both processes ended?

(e) Print the value of the `counter` file on the shell.

(f) Start two processes of `counting_process` in parallel on the shell (if it takes too long: reduce the number inside the for loop and compile again). Use the provided `start.sh` for that by calling `./start.sh`. The `start.sh`

- Resets the `counter` file to 0
- Starts two processes of `counting_process`
- Waits until both processes have finished
- Prints the value of the `counter` file

(g) What is the value of the `counter` file? Have you expected that?

Exercise 6.3: Synchronisation with a semaphore

(a) Make sure you are in the `OS_exercises/sheet_06_process_sync1/counting_sem` directory.

(b) Compile your `counting_process.c` into `counting_process`, just to make sure everything compiles. Use the provided `Makefile` for that.

(c) Use a semaphore to fix the synchronisation issue in `counting_process.c`.

(d) Compile your `counting_process.c` again.

(e) Start two processes of `counting_process` in parallel on the shell (if it takes too long: reduce the number inside the for loop and compile again). Use the provided `start.sh` for that by calling `./start.sh`.

(f) What is the value of the `counter` file? Have you expected that?

Exercise 6.4: Synchronisation with a lock file (optional)

(a) Change into the `OS_exercises/sheet_06_process_sync1/counting_flock` directory.

(b) Compile your `counting_process.c` into `counting_process`, just to make sure everything compiles. Use the provided `Makefile` for that.

(c) Use a `counter.lck` lock file and the `flock()` function to fix the synchronisation issue in `counting_process_flock.c`

(d) Compile your `counting_process.c` again.

(e) Start two processes of `counting_process` in parallel on the shell (if it takes too long: reduce the number inside the for loop and compile again). Use the provided `start.sh` for that by calling `./start.sh`.

(f) What is the value of the `counter` file? Have you expected that?