Exercise Sheet 6

In this exercise sheet you will practice shared memory communication and mutual exclusion with semaphores.

Task 1

Write a program that creates a shared memory segment and stores an integer with value equal to zero there. After that, the program reads an integer from a named FIFO called “RESULT_FIFO” and prints the obtained value on the standard output.

Write another program that creates 100 processes. Each of these newly created processes should attach the shared segment created before, and executes a loop of 100 iterations. In each of these iterations, the value in shared memory should be increased by one. The parent process waits for all the child processes to finish and then it does two things: (1) it reads the value in the shared memory segment and prints it on the standard output; and (2) it writes that value to the FIFO called “RESULT_FIFO”.

Analyze the obtained output. Is the computed result the expected value? Does the computed value change across different runs? Explain why.

This exercise MUST be implemented using the System V shared memory mechanisms reviewed in the lecture (Teil 7).

Task 2

Implement the same exercise as before. However, in this case, the access to the value in shared memory should be mutually exclusive across processes. Use a semaphore to implement this behaviour.

This exercise MUST be implemented using the System V semaphore facility reviewed in the lecture (Teil 7).

Task 3

Research and analyze the POSIX alternatives for shared memory and semaphores. Describe the differences between them and the System V mechanisms used in the previous tasks, and highlight advantages and disadvantages of both.

Task 4

Re-implement Task 2 using only the POSIX alternatives to shared memory and semaphores.