Simon Myhre

Lab 8

Exercise 1:

- First parent process asks “My child asks”. Then it goes to child that says “Are you my parent”. Then back again to parent that says “And the returned” + the wexitstatus the child process returns “42”
  Out writes to file. Then in reads from it. Places it in in. In is placed in buffer. Buffer is printed. Then the returnvalue is from the status using wexitstatus.
- Nothing is different. Result are the same.

Exercise 2:

- It says “Hi fifo”
- It says “Hi fifo” again. That’s because the hi Fifo is still stored in the fifo_test. So whenever cat fifo_test is run, it will return with the “Hi fifo” whenever it says that in the file. If there are no “Hi fifo” or whatever else text, it will wait.

“where is the data is sent through FIFO store» = “Where is the date stored while using FIFO?”

- Its passed internally in the Kernel. There are no contents in the file. Its only a reference point.

Exercise 3:

- No, we do not need to do that. Since the while loop will run forever, it will also listen forever

Exercise 4:

- When trying to quit using CTRL + C the program catches the signal and sends it to a function to write an output.
- The number of the signal is 2. That should be process interrupted.

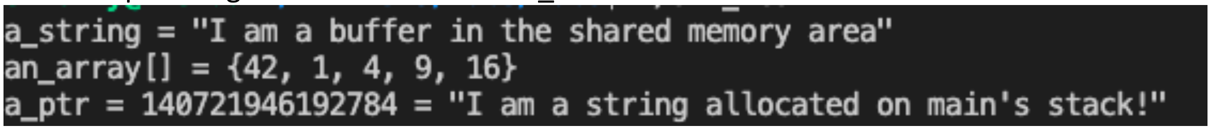
Exercise 5:

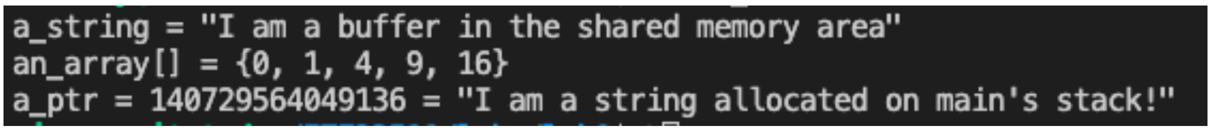
- It prints out 1 handler 1. Then 100 in handler 2. Then 101 in main handler
- First when SIGUSR1 is catched it will do the halder1 function = Print count = 1.
  Then it will fork.
  The child process will also catch SIGUSR1 but send it to handler 2 = print count 10.
  Then it will go to the parent that waits for the child. When child is finished the parent will print count 101

Simon Myhre

Exercise 7
Sorry for messy code. Hade to comment the program to teach myself how it works

- a-prt is segmentation fault in shm_test2 both times


```
a_string = "I am a buffer in the shared memory area"
an_array[] = {42, 1, 4, 9, 16}
a_ptr = 140721946192784 = "I am a string allocated on main's stack!"
```


```
a_string = "I am a buffer in the shared memory area"
an_array[] = {0, 1, 4, 9, 16}
a_ptr = 140729564049136 = "I am a string allocated on main's stack!"
```

- The segmentation fault comes from "shared_mem->a_ptr" in the printf close to the end. The reason is because a_ptr is an absolute pointer. Using that means that the processes doesn't share the same address. That's why it should not be used. The solution is to not use it, or to allocate a big shared memory segment for the shared_mem
- Nothing special happends. It's the one that does the task latest, that will write the last result. So if task2 is slower than task1, the text in task2 will be shown
- Shmctl() will remove a shared memory

Exercise 8
- It will last until the system is shut down
- Using sem_destroy will destroy the semaphore
- Sem_open. Sem_init. Sem_wait. Sem_post. Sem destroy. Code goes inbetween sem_wait and post (usually). Like in my lab7

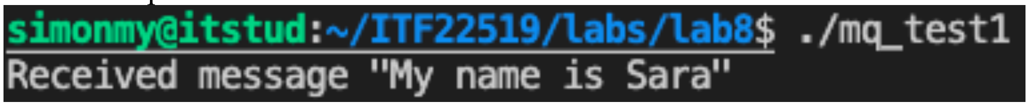
Exercise 10
1. Socket: Creates a socket
2. Connect: Connect a socket to a remote socket address
3. Bind: Binds a socket to.a local socket address
4. Listen: tells the socket that new connections shall be accepted
5. Accept: Get a new socket with a new incoming connection
6. Socketpair: returns two connected anonymous sockets (only for a few local families)

https://man7.org/linux/man-pages/man7/socket.7.html

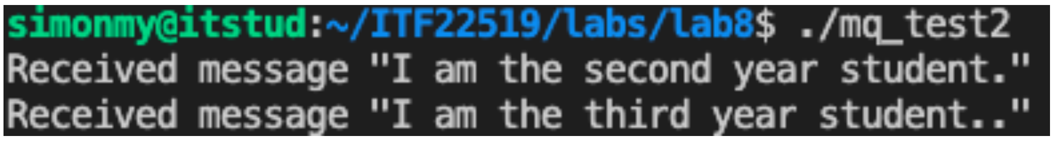
- Socketpair and bind

Exercise 11

- Output:


```
simonmy@itstud:~/ITF22519/labs/lab8$ ./mq_test1
Received message "My name is Sara"
```

Simon Myhre



```
simonmy@itstud:~/ITF22519/labs/lab8$ ./mq_test2
Received message "I am the second year student."
Received message "I am the third year student.."
```

- When running test2 first and test1 after, the result is still the same

Exercise 12

- mq_receive makes sure to block if there are no messages in the queue. Therefor nothing more synchronizing is needed.