Exercise Sheet 7

Exercise 1 (Inter-Process Communication)

1. Describe what a critical section is.

2. Describe what a race condition is.

3. Describe how to avoid race conditions.

Exercise 2 (Synchronization)

1. What is the advantage of signal and wait compared with busy waiting?

2. Which two problems can arise from locking?

3. What is the difference between signaling and locking?

4. Which four conditions must be fulfilled at the same time as precondition that a deadlock can arise?
   - ☐ Recursive function calls
   - ☐ Mutual exclusion
   - ☐ Frequent function calls
   - ☐ Nested `for` loops
   - ☐ No preemption
   - ☐ Hold and wait
   - ☐ > 128 processes in `blocked` state
   - ☐ Iterative programming
   - ☐ Circular wait
   - ☐ Queues

5. Does a deadlock occur?

$$\text{Existing resource vector} = \begin{pmatrix} 8 & 6 & 7 & 5 \end{pmatrix}$$

$$\text{Current allocation matrix} = \begin{bmatrix} 2 & 1 & 0 & 0 \\ 3 & 1 & 0 & 4 \\ 0 & 2 & 1 & 1 \end{bmatrix} \qquad \text{Request matrix} = \begin{bmatrix} 3 & 2 & 4 & 5 \\ 1 & 1 & 2 & 0 \\ 4 & 3 & 5 & 4 \end{bmatrix}$$

Exercise 3 (Communication of Processes)

1. What must be considered, when inter-process communication via shared memory segments is used?

2. What is the function of the shared memory table in the Linux kernel?

3. What is the impact of a restart (reboot) of the operating system on the existing shared memory segments?

   ☐ The shared memory segments are created new during boot and the contents are restored.
   ☐ The shared memory segments are created new during boot, but they remain empty. This means, only the contents are lost.
   ☐ The shared memory segments and their contents are lost.
   ☐ Only the shared memory segments are lost. The operating system stores the contents in temporary files inside the folder `\tmp`.

4. According to which principle operate message queues?

   ☐ Round Robin ☐ LIFO ☐ FIFO ☐ SJF ☐ LJF

5. How many processes can communicate with each other via a pipe?

6. What is the effect, when a process tries to write data into a pipe without free capacity?

7. What is the effect, when a process tries to read data from an empty pipe?

8. Which two different types of pipes exist?

9. Which two different types of sockets exist?

10. Communication via pipes works...

☐ memory-based ☐ message-based

11. Communication via message queues works...

☐ memory-based ☐ message-based

12. Communication via shared memory segments works...

☐ memory-based ☐ message-based

13. Communication via sockets works...

☐ memory-based ☐ message-based

14. Which two sorts of inter-process communication operate bidirectional?

☐ Shared memory segments ☐ Message queues
☐ Anonymous pipes ☐ Named pipes
☐ Sockets

15. Name a sort of inter-process communication, which can only be used for processes, which are closely related to each other.

☐ Shared memory segments ☐ Message queues
☐ Anonymous pipes ☐ Named pipes
☐ Sockets

16. Which sort of inter-process communication allows communication over computer boundaries?

☐ Shared memory segments ☐ Message queues
☐ Anonymous pipes ☐ Named pipes
☐ Sockets

17. With which sorts of inter-process communication remains the data intact without a bound process?

☐ Shared memory segments ☐ Message queues
☐ Anonymous pipes ☐ Named pipes
☐ Sockets

18. For which sort of inter-process communication guarantees the operating system not the synchronization?

☐ Shared memory segments ☐ Message queues
☐ Anonymous pipes ☐ Named pipes
☐ Sockets

Exercise 4 (Cooperation of Processes)

1. What is a semaphore and what is its intended purpose?

2. Which two operations are used with semaphores?

3. What is the difference between Semaphores versus blocking?

4. What is a binary semaphore?

5. What is a mutex and what is its intended purpose?

6. Which type of semaphores has the same functionality as the mutex?

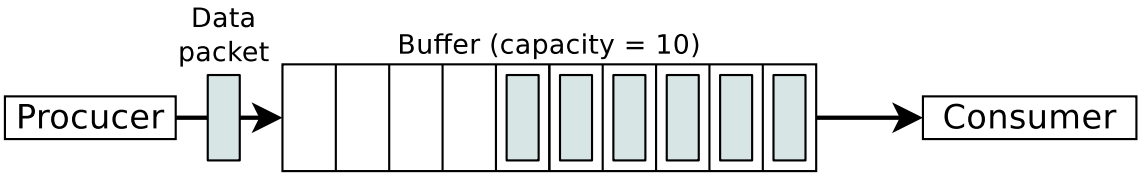
7. Which states can a mutex have?

8. Which Linux/UNIX command returns information about existing shared memory segments, message queues and semaphores?

9. Which Linux/UNIX command allows to erase existing shared memory segments, message queues and semaphores?

Exercise 5 (Producer/Consumer Scenario)

A producer should send data to a consumer. A buffer with limited capacity should be used to minimize the waiting times of the consumer. Data is placed into the buffer by the producer and the consumer removes data from the buffer. Mutual exclusion is necessary in order to avoid inconsistencies. If the buffer has no more free capacity, the producer must block itself. If the buffer is empty, the consumer must block itself.



For synchronizing the two processes, create the required semaphores, assign them initial values and insert semaphore operations.

```
typedef int semaphore;          // semaphores are of type integer




void producer (void) {
    int data;

    while (TRUE) {              // infinite loop
        createDatapacket(data);   // create data packet


        insertDatapacket(data);   // write data packet into the buffer


    }
}

void consumer (void) {
    int data;

    while (TRUE) {              // infinite loop


        removeDatapacket(data);   // pick data packet from the buffer


        consumeDatapacket(data);  // consume data packet
    }
}
```

Exercise 6 (Semaphores)

In a warehouse, packages are delivered constantly by a supplier and picked up by two deliverers. The supplier and the deliverers need to pass through a gate. The gate can always be passed only by a single person. The supplier brings three packages with every shipment to the incoming goods section. One of the deliverers can pick two packages with every pickup from the outgoing goods section. The other deliverer can pick only a single package per pickup from the outgoing goods section.

```
Supplier                  Deliverer_X               Deliverer_Y
{                         {                         {
  while (TRUE)              while (TRUE)              while (TRUE)
  {                         {                         {

    <Pass through gate>;      <Pass through gate>;      <Pass through gate>;



    <Enter incoming           <Enter outgoing           <Enter outgoing
    goods section>;           goods section>;           goods section>;



                              <Pick 2 packets>;         <Pick 1 packet>;
    <Unload 3 packets>;



                              <Leave outgoing           <Leave outgoing
    <Leave incoming           goods section>;           goods section>;
    goods section>;


                              <Pass through gate>;      <Pass through gate>;
    <Pass through gate>;
                              }                         }
    }                       }                         }
}
```

Exactly one process `Supplier`, one process `Deliverer_X` and one process `Deliverer_Y` exist. For synchronizing the three processes, create the required semaphores, assign them values and insert semaphore operations. These conditions must be met:

- Only a single process can pass through the gate.

- Only one of both existing deliverers can access the outgoing goods section.

- It should be possible that the supplier and one of the deliverers can simultaneously unload and pick goods.
- The capacity of the warehouse is 10 packages.
- No deadlocks are allowed.
- At the beginning, the warehouse contains no packets and the gate, as well as the incoming goods section and the outgoing goods section are free.

Source: TU-München, Übungen zur Einführung in die Informatik III, WS01/02