Exam Operating Systems	**Prof. Dr. Tobias Lauer**
Summer term 2021	**14/07/2021**

Name: _____ Matriculation Number: _____

Program/semester: _____ Signature: _____

Question	**1**	**2**	**3**	**4**	**Sum**
Maximum points	**18**	**12**	**16**	**14**	**60**
Achieved points					

1. Please check this exam for completeness (8 pages).

2. You can write all your solutions directly below the respective questions. If you need additional space, you use the back sides too, or ask for extra paper. Please put your name and matriculation number on all extra paper.

3. If a question asks for calculations, make sure the way of solution is clear. Always specify units of measurement.
   For memory sizes we use: **1 KB = 2^{10} Byte**; **1 MB = 2^{10} KB = 2^{20} Byte**; etc.

4. Write clearly with a **pen** or **ball-pen**. Answers written with pencil or in red color will not be graded.

5. As supplementary material, you can use **one standard A4 sheet of paper with notes handwritten** by yourself, and a standard **calculator** that is not programmable and not connectable to a network or other devices.

6. Any cheating attempt will lead to a failure of the exam.

7. The duration of the exam is 60 minutes.

Best of luck!

Question 1 (18 points)

(a) Three processes P_1, P_2 and P_3 are using shared resources R_1, R_2, R_3, R_4 and R_5.
At time t_0 we have the following situation:

a. P_1 is holding R_5 and waiting for R_2.
b. P_2 is holding R_4 and waiting for R_3
c. P_3 is holding R_1 and R_3 and waiting for R_5

Is this a deadlock situation? Justify your answer by drawing the resource graph (circles for processes, squares for resources). (3 points)

(b) We assume that in the next step (time t_1) process P_1 has gotten its requested resource R_2 and is now requesting another resource. The other two processes P_2 and P_3 have not made any progress.
Which of the above resources should P_1 **not** request, as it would lead to a deadlock? Name that resource/those resources and draw the requests into the graph with dashed arrows. (2 points)

(c) Besides the **detection** of deadlocks, what can an operating systems use the resource graph for? Hint: Think of part (b). (1 Punkt)

(d) The following attempt to protect a **critical region** uses a global variable `flag`, which is initially set to 0 and which indicates by a value of 1 that a thread/process is currently inside the critical region. To achieve this, each thread calls the method `beginRegion()` directly **before** entering the critical region, and directly **after** leaving the critical region, the method `endRegion()` is called. These two methods are defined as follows:

```
void beginRegion() {
    while (flag == 1) {
        // do nothing and wait until lock != 1
    }
    flag = 1;       // Now set flag to 1
}


void endRegion() {
    flag = 0;       // Set flag to 0
}
```

Explain why this attempt **does not** protect the critical region (even if it is guaranteed that the variable `flag` can only be changed by the above methods and that those will always be called correctly). In the above code, mark the lines that are problematic with an arrow and describe **in detail** what can happen there. (3 points)

(e) Even if the attempt in (d) could protect the critical region, it would **not be efficient**. Why? Also mark the problematic part of the code and explain. (1 point)

(f) A **binary semaphore** also consists of a variable that can have the values 0 and 1, and therefore is quite similar to the attempt in (d).
What are the differences? In your answer, explain the meaning of each value of the semaphore variable, who can see and change this value, and how the "waiting" of a thread/process is achieved. (4 points)

(g) A „rendezvous" is a **barrier** for **two concurrent processes**. **Both** processes must have completed all instructions in front of the barrier before they are allowed to proceed (i.e. carry out the instructions after the barrier). Hence, the processes are "meeting" (waiting for each other) at the barrier.

Using pseudocode (or precise words), describe how such a barrier can be implemented. You can choose to do this by using either **semaphores** or **message exchange** (inter-process communication). Explain how the individual constructs (semaphores, variables, messages, buffers, etc.) have to be initialized and give the sequence of semaphore operations/message exchanges by which the barrier is implemented in each process. (4 points)

Question 2 (10 points)

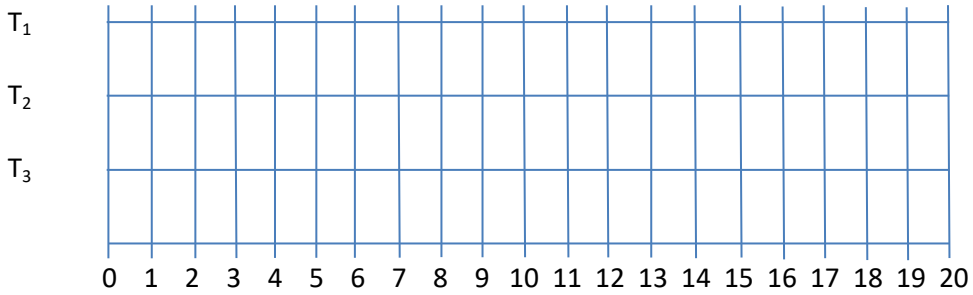
The following periodic tasks T_1, T_2 and T_3 have to be scheduled in a real-time operating system (on a computer with only one processor, no hyperthreading etc.), using **Rate Monotonic Scheduling (RMS)**

	Time of request A_i	**Execution-time C_i**	**Period-P_i**
Task T_1:	0	3	8
Task T_2:	1	1	4
Task T_3:	2	2	6

(a) Determine (mathematically) whether a successful scheduling of the above tasks with RMS is possible at all. State the **necessary condition** and check whether it is satisfied. (2 points)

(b) Which condition has to be satisfied in order to **guarantee** that RMS meets all deadlines when scheduling the above tasks? State this **sufficient condition** and check whether it is satisfied in this case. (2 points)

(c) In the following diagram, draw the scheduling sequence of RMS for the above tasks during the first 20 time units. Mark the times at which a task is preempted with a „P“, and times where a deadline is not met with an „X“. (4 points)
(**Hint:** We ignore time that the scheduler and dispatcher themselves are using.)



(d) Name two scheduling goals/requirements that are particularly relevant for a **real-time scheduler**, and give a short explanation why. (2 points)

Qusetion 3 (16 points)

Assume a computer system with **32-bit** addressing, **1 GB** of physical memory (RAM) and a page and frame size of **16 KB**.

a) What is the number of frames in the physical memory (decimal number or power of two)? (1 point)

b) How many entries does the page table of the virtual memory have (decimal number or power of two)? (1 point)

c) A program wants to access memory address `0x1B32FFE0`. What is the **page number** of this address? Give it as a hexadecimal number. (3 points)
Hints: Translate the address into the binary system. That way it is easy to divide it into page number and offset.
When retranslating into hexadecimal, only translate the page number, beginning with its rightmost bit.

d) In a paged memory system with 3 frames R1, R2, R3 of physical memory, the pages

5, 1, 2, 3, 1, 4, 5, 1, 5, 4, 2, 1, 5

are requested (in the given order). In the below diagram, enter the sequence of page replacements that are occurring in the **LRU strategy** (**Least Recently Used**). Mark each page fault with an „F". The first three pages 5, 1, 2 have already been entered. (3 points)

		3	1	4	5	1	5	4	2	1	5
R1	5										
R2	1										
R3	2										
Page fault	F										

e) As an alternative, the strategy **LFU** (**Least *Frequently* Used**) is proposed. Here we always replace that page that has been accessed least frequently, i.e. **the lowest number of times so far**. For this, it is necessary to keep a count for each page how often it has been requested yet. (In the case where more than one page have the same access frequency we use the FIFO strategy as a tie breaker).
In the diagram, entert he sequence of page replacements that occurrs with this strategy. Again, mark page faults with an „F". (4 points)

		3	1	4	5	1	5	4	2	1	5
R1	5										
R2	1										
R3	2										
Seitenfehler	F										

f) A precondition for effective paging is the **locality principle**. Explain this principle, mentioning both the spatial and temporal aspects of it. (3 points)

g) Which of the following also make(s) use of the locality principle? Tick the die correct answer(s): (1 point)

◯ Caching ◯ Locking ◯ Spooling

Question 4 (14 points)

The hard drive of a computer system has **2 GB** of storage capacity and allows access to blocks of size **1024 bytes**. We want to store a file with **100.000.000 Bytes**. For this, we are comparing 3 different file systems *A*, *B* and *C*.

Hints: In parts (a)-(c) we ignore the space used for the directory entry of the file, but we take into account all other space required for storing the file. First determine the number of blocks that are required, and from that, calculate the answer. Remember that a block is completely used whenever a file requires it.

(a) File system *A* uses the **contiguous (or sequential) allocation** of blocks. How many bytes does the above file take on the hard drive? (2 points)

(b) File system *B* uses a **linked list** (without table). Each reference tot he next block consists of a 32-bit number.
How many bytes does the file take on the hard drive when using system *B*? (3 points)

(c) File system *C* uses **index nodes** (I-Nodes). Each I-Node contains **eleven** list entries: **eight** direkte Blockadressen, **one** address of a block for **single indirect** addressing (storing 256 block addresses), **one** address for **double indirection** Adressierung and **one** address for **triple indirection**.
The space required for each I-Node as well as for 256 addresses is 1 block. How many bytes does the file take on the hard drive when using system *C*? (3 points)

(d) The above file should provide **random access**, i.e. we want to be able to read directly at an arbitrary position of the file (rather than having to read from the beginning).
Sort the file systems *A*, *B* and *C* (from fast to slow) regarding their efficiency for random access. Justify your order. (6 points)
Hint: For comparing the systems, use the example of direct access to the last block of the file. How many blocks have to be loaded from disk to achieve this with each system?