Challenge 1

Filip Strömbäck, Ahmed Rezine

February 5, 2026

Instructions

You have to do research on the parts you do not understand. When you solve problems, do not hesitate to:

- State your starting point. What do you know from start? Why is it important?
- State your goal. What do you need to know to reach it?
- State your assumptions. What are you taking for granted or assuming?
- Show step by step how you can go from what you have to what you want.

You will get one credit for each presentation you claim prepared. You may (at random) be selected to present any of the solutions you prepared. When you are selected for presentation:

- For each step in your solution, explain what you were thinking. How did you come up with this? What gave you the clues?
- Aim for 10 minutes.
- You should be able to present most of your solution without reading from your notes. You are, of course, free to reference your notes for calculations, figures, and important points.
- You are not required to have a correct solution to get credit, but **you are required to solve all parts of a problem, to make a serious attempt and to "believe" in your solution**.
- If you are not prepared you loose all credit for the seminar in question. As long as you have solved the problem yourself, and understood all steps in the solution, this is very unlikely to happen.

As audience you should think of how difficult it is to clearly present a solution. Be humble and supportive. There are multiple ways to solve most problems, so you are encouraged to ask questions. Compare the presentation to your solution and ask questions like "I did it in a different way, is there a difference?" This fosters discussions that are beneficial to everyone in the seminar.[1]

You are of course welcome to take notes.

[1] Do not hesitate to ask questions, even if you fear the presenter might not be able to answer. As long as they are able to answer questions about *their approach* and the concepts *they have used*, they run no risk of loosing points for not answering questions from the audience correctly.

Problem 1

Assume a system that is idle in the beginning. Consider the set of job arrivals as shown in Table 1.

Job	Arrival time	Execution time
J_1	1	9
J_2	4	7
J_3	6	3
J_4	10	2
J_5	11	1

Table 1: Job arrivals

(a) Illustrate the execution of the set of processes via two Gantt charts: one for Shortest Job First (SJF) with preemption, and the other one for SJF without preemption. Show the ready-queue content after every change. What are the average waiting times for each case?

(b) Assume a multi-level scheduler combining Round Robin with time quantum 5 and first-in-first-out (FIFO). Round Robin has higher priority and each job is initially inserted at the end of this queue. After executing for 5 time units, a process is preempted and inserted at the ready queue of FIFO. Illustrate via Gantt chart, how the given set of processes will execute using this multi-level scheduler.

Problem 2

A hypothetical CPU-scheduler $\mathcal{H}$, makes use of 3 preemptive schedulers R_1, R_2 and R_3. Preemption can for example be due to priorities. Each scheduler R_i (for $i \in \{1, 2, 3\}$) has a ready queue rd_i and a waiting queue wt_i. The resulting overall scheduler $\mathcal{H}$ first executes all jobs from the ready queue of R_1. $\mathcal{H}$ can execute jobs from the ready queue of R_2 only when the ready queue of R_1 is empty. In addition, $\mathcal{H}$ executes jobs from the ready queue of R_3 only when the ready queues of R_1 and R_2 are empty. A fresh CPU job is always put into the ready queue of R_1 first. If a CPU job is assgined to to scheduler R_i and relinquishes the CPU for some resource (e.g. performs an I/O request), it is moved to the waiting queue of R_i. If a CPU job is assigned to scheduler R_i (for $i \in \{1, 2\}$) and it is preempted from the CPU, the job is moved into the ready queue of scheduler R_{i+1}. Now, consider the following two variants of $\mathcal{H}$:

- V_1 : If a job is preempted from the CPU, while assigned to scheduler R_3, the job is moved into the ready queue of scheduler R_1.
- V_2 : If a job is preempted from the CPU, while assigned to scheduler R_3, the job is moved into the ready queue of scheduler R_3.

(a) For both V_1 and V_2, draw the process state diagrams. Your state diagrams must clearly distinguish the schedulers R_1, R_2 and R_3 in which processes are assigned. For instance, the diagrams must include states to distinguish waiting at scheduler R_2 or running at scheduler R_3.

(b) Is starvation possible in V_1? in V_2? Explain.

(c) Explain one scenario where variant V_1 will be a better choice compared to variant V_2.

(d) Explain a different scenario where V_2 will be a better choice compared to the variant V_1.

Problem 3

Assume a Round Robin scheduler with a time quantum q and with a context switching overhead o. Suppose the average burst time for I/O bound jobs is t where t is much larger than o. Discuss the following choices of the time quantum (aim for 1–2 sentences per item):

- q smaller than o
- q equal to o
- q between o and t
- q equal to t
- q larger than t

Problem 4

Which statements are correct? Justify using examples or scenarios (aim for 2–4 sentences per item).

1. Real time systems tend to use preemptive schedulers.
2. Timesharing systems tend to use preemptive schedulers.
3. Assume a priority based preemptive scheduler where some jobs may need exclusive access to a resource. Suppose the system ensures that any lower priority job with access to the resource relinquishes the resource if a higher priority job asks for it.

   For instance: the low priority is running with the resource. It is preempted by a high priority. The high priority needs the resource, asks for it and is moved to a waiting queue. The low priority is scheduled and realises the resource is needed by a high priority. It relinquishes the resource, and is then preempted by the high priority job.

   Claim: "Such a scheduler will never witness a priority inversion", i.e., a situation whereby a higher priority job waits, directly or indirectly, for lower priority jobs (apart of course from the described case where the lower job does relinquish the exclusive access when asked for it).

Problem 5

The current state of a system:

- The system is idle. Ready queue is empty.
- Process P1 is waiting for keyboard input. It's next CPU burst is expected to be 2 ticks.
- Process P2 is waiting for disk data. It's next CPU burst is expected to be 5 ticks.
- Process P3 is waiting for network data. It's next CPU burst is expected to be 2 ticks.
- The initial CPU burst for any new processes are estimated to be 3 ticks.

A list of events that will take place starting at this time:

Tick	Event
0	Disk data is delivered.
1	Process P4 is created.
2	Network data is delivered.
3	Executing process asks to sleep until tick 6.
5	Process P5 is created.
6	Process from tick 3 wake up from sleep.
7	Keyboard input is delivered.
8	Executing process asks for keyboard input.

Table 2: Events in the system.

(a) Use preemptive shortest job first to schedule the processes in ticks 0–8. Draw a Gantt chart that shows the result and show how the system ready- and wait-queues change during every event.

(b) At tick 6, some process is executing. How will the kernel know to wake up the process from tick 3 when it (the kernel) is not executing? (Aim for 3–4 sentences.)

Problem 6

Can you see negative effects of each of the following programs? What typical Operating System mechanisms (e.g., dual mode, preemption, context switch, etc) are used to counter or limit each one of the effects? Explain (aim for 3–4 sentences per item):

(a)
```
int main()
{
  set interrupt flag to zero to isolate CPU from all devices
}
```

(b)
```
int main()
{
  use hardware io commands to erase hard drive
}
```

(c)
```
int main()
{
  execute infinite loop to hog CPU, no other program may run
}
```

(d)
```
int main()
{
  modify kernel memory to give oneself more resources
}
```

Problem 7

Assume you have a computer with one flat memory space addressed sequentially and one single core 1GHz central processing unit. You have three programs to execute:

```
int main() // sum 100 numbers
{
  initiate sum to zero
  repeat 100 times
    read one number from keyboard [waiting time = 500 ms]
    add the number to the sum
  display sum to screen
}
```

```
int main() // calculate n-factorial of 1000.000.000
{
  initiate factorial to one
  inititate i to 1000.000.000
  decrease i while larger than zero
    multiply i in to factorial [100 instructions per iteration]
  display n to screen
}
```

```
int main() // calculate checksum of 4MiB file on disk
{
  initiate global checksum to zero
  open file
  repeat for every 512 byte block in the file
    read block from disk to memory [waiting time = 10 ms]
    calculate and add block checksum [5000 instructions]
  close file
}
```

(a) If you execute your programs sequentially (one after the other), give an estimate on the total turnaround time.

(b) Explain the mechanism involved in allowing for concurrent executions on a single processor. (Aim for 3–4 sentences.)

(c) Estimate how much you can gain from executing the three programs concurrently?

Problem 8

Assume that a program that is running in user mode needs to read 10 bytes from the keyboard. Explain the different steps involved in order for an operating system to allow for such an operation while forbidding any direct access by user mode programs to hardware resources (such as keyboard, disk, etc). Aim for 4–5 senteces.