Written Exam

DM510: Operating Systems (Spring 2025)

August 6th, 2025

This exam consists of 6 pages in total, including this one, with 4 topics and several questions in each topic. There are 60 points in total. The number of points is given for each question. Along with your answers, also explain how you arrived there, in a clear, but concise way.

Topic 1: CPU Scheduling (16 points)

Consider the following CPU bursts by processes P1-P5.

Process	Burst Time	Priority
P1	10	5
P2	2	3
P3	7	4
P4	5	1
P5	12	2

1.a) ***[4 points]*** For each of the following algorithms produce a schedule for the given CPU bursts and compute the average response time (the time the CPU burst has finished):

1. Priority scheduling
2. Shortest-Job-First
3. Round-Robin with time quantum $q = 5$

1.b) ***[2 points]*** Apart from response time, name two other criteria for evaluating scheduling algorithms.

1.c) ***[2 points]*** Give an advantage of Round-Robin over Shortest-Job-First.

1.d) ***[3 points]*** Describe how the hardware clock is used to implement a *preemptive* scheduling algorithm. Which of the scheduling algorithms above are preemptive?

1.e) ***[2 points]*** You are using a text editor and experience significant delays on user inputs, for example, when pressing a key. When running the command "top" you see that the CPU utilization is at 100% due to a background process. Explain how the delay could be related to CPU scheduling and give a suggestion on how to improve the user experience.

1.f) ***[3 points]*** Describe in 2-3 sentences what a context switch is. Explain the role of context switches in the overall system performance.

Topic 2: Synchronization (14 points)

The following code is supposed to calculate the number of non-zero entries in an array:

```
int array[1024];
int count = 0;
initialize(array);

#pragma omp parallel for
for (int i=0; i < 1024; ++i) {
  count += !!array[i];
}
```

Here, note that `!!a` is 0 if `a` is 0 and 1 if `a` is anything else. Due to the openMP parallelization, the iterations of the loop may be executed in different threads.

2.a) ***[2 points]*** Elaborate the race condition that can occur in the code above.

2.b) ***[3 points]*** Describe how the race condition can be avoided using the atomic operation `int compare_and_swap(int* obj, int old, int new)`. Provide the corresponding code.

Another way to avoid the race condition is to use mutexes:

```
int array[1024];
int count = 0;
mutex_t m;
initialize(array);

#pragma omp parallel for
for (int i=0; i < 1024; ++i) {
  acquire(&m)
  count += !!array[i];
  release(&m)
}
```

2.c) ***[2 points]*** The solution using mutexes would not be faster than a naive sequential implementation. Explain why.

2.d) ***[3 points]*** Still using mutexes, modify the code so that it would run faster and argue why it would.

Hint: introduce a count variable for each parallel thread.

2.e) ***[4 points]*** Two alternative implementations of an mutex use (1) busy waiting or (2) moving the thread/process to the waiting queue of the CPU scheduler. Explain both approaches in 2-3 sentences and name one advantage of each.

Topic 3: Main Memory (14 points)

We consider a CPU architecture with 32-bit addresses and a naive page table implementation. The first 4 bits of the address specify the page or frame number. The current page table of a process contains the following entries (in hexadecimal; dashes "-" indicate that the invalid bit is set):

page	frame	page	frame
0	-	8	-
1	-	9	-
2	-	A	-
3	0	B	2
4	1	C	-
5	-	D	-
6	-	E	-
7	-	F	-

3.a) ***[2 points]*** Based on the page table above, translate the following logical addresses (given in hexadecimal) into physical addresses:

3A44
B40B
BE3F

3.b) ***[2 points]*** Describe one disadvantage of having too large page sizes. Describe one disadvantage of having too small page sizes.

3.c) ***[3 points]*** Explain in detail what happens when an instruction references the logical address 7AA0 with the page table above.

3.d) ***[4 points]*** Assume that process P's page table has the state above and there are 3 frames allocated to P (frames $0, 1, 2$). The last three pages that were referenced by P are (in this order) B, 4, 3. Consider the following page reference string, describing exactly which pages P references next and in which order:

$$1, 2, 4, 3, 2, 1, 2, 3, 6, 3, 2, 1$$

How many page faults would occur for the following replacement algorithms?

1. LRU replacement
2. Optimal replacement

3.e) ***[3 points]*** A research lab at SDU uses a computer to run 50 scientific simulations. Each of the simulations involves heavy computation on a large data set, requiring around 2GB of memory in each simulation. The lab computer has four CPU cores and 8GB of main memory. You consider three alternative setups:

1. Running the simulations after each other.
2. Running 5-10 simulation in parallel and starting a new one each time another has finished.
3. Running all simulations in parallel.

Which of these alternatives do you expect to finish all simulations in the shortest time? Justify your answer.

Topic 4: Network and Security (16 points)

4.a) ***[2 points]*** Video streaming is an application that requires a high network throughput. Explain how direct-memory access (DMA) can improves the performance of such an application compared to more naive device I/O.

In the following, we consider the example application of accessing a Linux server via a remote shell (for example the program `ssh`): here the server allows clients to send terminal input over the internet and executes it in a terminal/shell.

4.b) ***[2 points]*** Discuss whether TCP or UDP would be more suitable for implementing a remote shell and justify why.

4.c) ***[2 points]*** Describe two possible security violations that could occur if the remote shell connection is not encrypted.

4.d) ***[3 points]*** Describe what *symmetric* encryption is and how it could be used to secure a remote shell connection.

Consider a server that processes requests from a TCP connection line by line. The following code waits until a full line has been read (identified by the linebreak `'\n'`) and only then consumes it.

```
while (true) {
  char c;
  char line[1024];
  int pos = 0;
  while (true) {
    read(fd, &c, sizeof(char)); //read one byte from TCP connection
    if (c == '\n') {
      line[pos] = 0; // end of string
      break;
    } else {
      line[pos] = c;
      pos++;
    }
  }
  consume(line);
}
```

4.e) ***[3 points]*** Explain why this program is vulnerable to code injection and explain how such an attack would work.

4.f) ***[2 points]*** Modify the code so that it is safe.

4.g) ***[2 points]*** Describe two measures that an operating system can take to mitigate the risks of code injection.