i Title page

Department of Computer Science

Examination paper for TDT4186 Operating Systems

Examination date: 25.05.2021

Examination time (from-to): 9.00–13.00

Permitted examination support material: A / All support material is allowed

Academic contact during examination: Michael Engel
Phone: 957 55 929

Technical support during examination: Orakel support services
Phone: 73 59 16 00

If you experience technical problems during the exam, contact Orakel support services as soon as possible before the examination time expires. If you don't get through immediately, hold the line until your call is answered.

OTHER INFORMATION

Make your own assumptions: If a question is unclear/vague, make your own assumptions and specify them in your answer. Only contact academic contact in case of errors or insufficiencies in the question set.

Cheating/Plagiarism: The exam is an individual, independent work. Examination aids are permitted, but make sure you follow any instructions regarding citations. During the exam it is not permitted to communicate with others about the exam questions, or distribute drafts for solutions. Such communication is regarded as cheating. All submitted answers will be subject to plagiarism control. *Read more about cheating and plagiarism here.*

Notifications: If there is a need to send a message to the candidates during the exam (e.g. if there is an error in the question set), this will be done by sending a notification in Inspera. A dialogue box will appear. You can re-read the notification by clicking the bell icon in the top right-hand corner of the screen. All candidates will also receive an SMS to ensure that nobody misses out on important information. Please keep your phone available during the exam.

Weighting: Points for each question are given next to the question title or subtitle. There are overall 70 points that can be achieved in this exam.

ABOUT SUBMISSION

How to answer questions: All question types other than Upload assignment must be answered directly in Inspera. In Inspera, your answers are saved automatically every 15 seconds. **NB!** We advise against pasting content from other programs, as this may cause loss of formatting and/or entire elements (e.g. images, tables).

File upload: When working in other programs because parts of/the entire answer should be uploaded as a file attachment – make sure to save your work regularly.

All files must be uploaded before the examination time expires.

The file types allowed are specified in the upload assignment(s).

30 minutes are added to the examination time to manage the sketches/calculations/files. The additional time is included in the remaining examination time shown in the top left-hand corner.

NB! You are responsible to ensure that the file(s) are correct and not corrupt/damaged. Check the file(s) you have uploaded by clicking "Download" when viewing the question. All files can be removed or replaced as long as the test is open.

How to digitize your sketches/calculations
How to create PDF documents
Remove personal information from the file(s) you want to upload

Automatic submission: Your answer will be submitted automatically when the examination time expires and the test closes, if you have answered at least one question. This will happen even if you do not click "Submit and return to dashboard" on the last page of the question set. You can reopen and edit your answer as long as the test is open. If no questions are answered by the time the examination time expires, your answer will not be submitted. This is considered as "did not attend the exam".

Withdrawing from the exam: If you become ill, or wish to submit a blank test/withdraw from the exam for another reason, go to the menu in the top right-hand corner and click "Submit blank". This cannot be undone, even if the test is still open.

Accessing your answer post-submission: You will find your answer in Archive when the examination time has expired.

1 Prosesser

```
// Code 1
pid_t pid1, pid2;
pid1 = fork();
pid2 = fork();
if (pid1>0 && pid2==0) {
  if (fork())
     fork();
}
```

These pieces of C code are executed on a Unix system. The missing code pieces (includes, definition of main) are not shown for brevity. Assume that all fork calls succeed.

Code 1 (4 points):

Give the **total number of processes** resulting from executing the code, including the process executing this code itself. **Explain your solution**.

Fill in your answer here

```
// Code 2
int main(void) {
   if (fork() == 0) {
      if (fork() == 0) {
         printf("3");
      } else {
         pid_t pid;
         int status;
         if ((pid = wait(&status)) > 0) {
            printf("4");
         }
      }
   } else {
      if (fork() == 0) {
         printf("1");
         exit(0);
      }
      printf("2");
   }
   printf("0");
}
```

Note:

We removed this part of the question from grading since a mistake in the code resulted in far too many possible generated outputs.

Code 2 (6 points):

Give **all valid outputs** resulting from executing the code, including the process executing this code itself. **Explain your solution**.

Fill in your answer here

Maximum marks: 10

2 System calls

System call parameters (5 points)

The system call
ssize_t read(int fd, void *buf, size_t count);
reads *count* bytes from the file with file descriptor *fd* into the buffer pointed to by *buf*.

To enable safe and secure operation of the OS, the OS has to check properties of the parameters to the **read** system call. Describe which properties of the parameters should be checked before the OS performs the requested file access.

Assume that **buf != NULL** and **count >= 0**. Page faults are allowed to take place after the checks.

Fill in your answer here

Unix shell (5 points)

When you enter a command in a Unix shell, this command can either be an internal or an external command.

For an **external command**, explain the steps the shell takes to execute the command.

Give details on the following questions:

1. How does to shell find the command (in the file system) to run?
2. Which system calls does the shell use to execute the command and in which order?
3. What is the difference in the system calls executed by the shell between executing an external command in the foreground vs. in the background?

Fill in your answer here

Maximum marks: 10

3 Synkronisering 1

```
Semaphore S1 = ___;
Semaphore S2 = ___;
Semaphore S3 = ___;

f1() {
   // SYNC
   printf("3");
   // SYNC
   printf("5");
   // SYNC
}

f2() {
  // SYNC
  printf("2");
  // SYNC
  printf("13");
  // SYNC
}

f3() {
  // SYNC
  printf("7");
  // SYNC
  printf("11");
  // SYNC
}
```

Synchronization (5 points)

The following three functions of a program run in separate threads each and print some prime numbers. All three threads are ready to run at the same time.

Use synchronization using the **semaphores** S1, S2 and S3 and **wait/signal operations** on the semaphores to ensure that the program outputs the prime numbers in increasing order (2, 3, 5, 7, 11, 13).

Insert appropriate wait or signal operations in the code lines indicated with **// SYNC** and give the correct initial values for the semaphores. *Note that it might not be required to add a wait or signal operations in all of the places indicated.*

Enter the completed functions and the initial values for the semaphores in the code field below:

Fill in your answer here

Maximum marks: 5

4 Synkronisering 2

Semaphore implementation (3 points)

Is it possible to implement concurrency primitive such as mutexes **on modern multicore CPUs** without the use of special instructions such as **xchg**?

In other words, can such concurrency primitives be written purely in C on current multicore processors? Explain your answer.

Fill in your answer here

Deterministic process execution (2 points)

Given an integer variable **x** in a C program, assume that the instruction **x++** is executed by two threads of the program in parallel.

If you run the program multiple times, explain why in some cases the variable is sometimes only incremented by one instead of by two.

Fill in your answer here

Maximum marks: 5

5 Minnetildeling 1

Buddy algorithm (3 points)

A system has a memory of size 32 MB (1 MB = 2^{20} bytes). Four processes A, B, C and D request memory one after the other (in order A, B, C, D).

Use the **buddy algorithm** to perform dynamic memory allocation for the processes.

The following allocations take place in the given order. The first allocation for process A is already given in the table below (you can copy and paste the table into the answer test field):

1. Process A allocates 6 MB
2. Process B allocates 9 MB
3. Process C allocates 1 MB
4. Process D allocates 6 MB

Hint: If a memory allocation cannot be fulfilled, please indicate this appropriately next to the respective allocation.

Important: Note that *each field* in the table *represents two MB*!

State after the first allocation by process A:

Process A allocates 6 MB

0	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30
A	A	A	A												

Give the new allocation of the memory after each of the allocations 2, 3 and 4:

Fill in your answer here

Maximum marks: 3

6 Minnetildeling 2

LRU (4 points)

In a system with page addressing and the page replacement strategy **LRU (least recently used)**, a process performs accesses to memory pages in the given **reference sequence**:

Reference sequence: 5, 3, 5, 1, 2, 5, 4, 6, 1

The operating system provides **three page frames** to the process.

Give the number of the virtual memory page that is paged into the respective page frame at each given request in the reference sequence. You can use the fields under "control state" to note the age of the respective page (you can copy and paste the table into the answer test field).

Reference sequence		**5**	**3**	**5**	**1**	**2**	**5**	**4**	**6**	**1**
Main memory	frame 1	5								
	frame 2									
	frame 3									
Control states	frame 1	0								
	frame 2									
	frame 3									

Fill in your answer here

Maximum marks: 4

7 Virtuelt minne 1

Segmented memory (4 points)

Determine the corresponding physical addresses for memory accesses to logical addresses (hexadecimal):

- 0x0000BEEF
- 0x1CEB00DA

using memory segmentation.

In the logical address, the most significant 8 bits of the address indicate the position in the segment table. Indicate in your answer if one of the memory accesses would result in an access violation.

Segment table:

Index	Start address	Length
00	0x0000 00E0	0x21 20FF
01	0xB542 0000	0x01 0000
02	0x0515 0000	0x20 0000
03	0x0006 0000	0x00 FFFF
...		
0x1C	0x0001 0000	0xFF FFFF

Fill in your answer here

Maximum marks: 4

8 Virtuelt minne 2

Paging (4 points)

In a system with page addressing (paging), the page frame table is in the state given below. The length of an address is 16 bits. The 12 least significant bits of an address are the offset inside the page (page size = 4096 bytes).

Determine the physical addresses for the logical addresses **0x6AB1** and **0xF1B7**.

Page table:

Page number	Start address
0	0xF000
1	0x3000
2	0x8000
3	0x1000
4	0xC000
5	0x2000
6	0x4000
7	0xB000
...	...
0xF	0x5000

Fill in your answer here

Logical address structure (2 points)

In a system where virtual address 0x722B2104 is mapped to physical address 0x16AB2104, what is the largest page size that could be used for this mapping? Explain your calculation.

Fill in your answer here

Maximum marks: 6

9 Planlegging 1

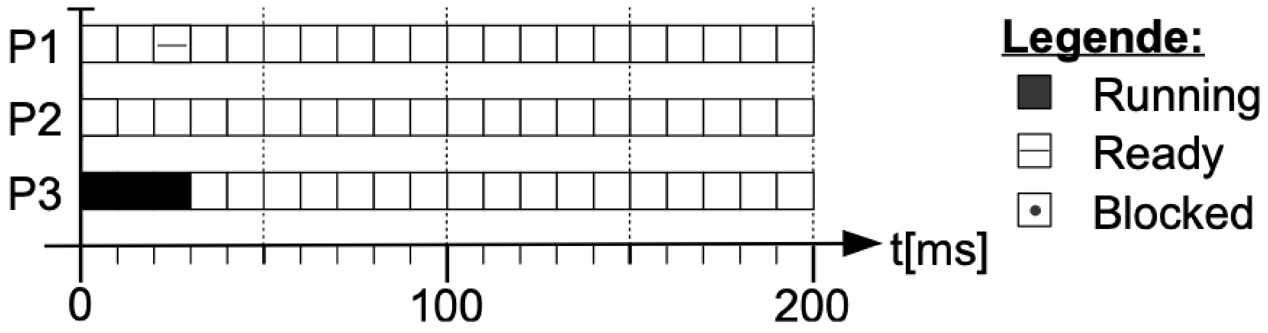
FCFS Scheduling (5 points)

An operating system has three cyclically executing processes P1, P2 and P3 (i.e. the process starts from its beginning after it ran through a CPU- and I/O-burst each).

The processes arrive (are ready to execute) at the time points given in the table below. All times are given in milliseconds (ms).

Process	Arrival time	CPU time	I/O time
P1	20	40	20
P2	30	20	10
P3	0	30	40

Enter the execution order and process state of the processes P1, P2 and P3 in the given Gantt diagram for a **first-come, first-served (FCFS) scheduler**.



Every process is starting from the beginning after the successful execution of one CPU- and I/O-burst each. After each CPU-burst of a process, a I/O-burst always follows.

Process switch times can be ignored. Each I/O operation can run in parallel with other I/O operations and CPU-bursts.

Mark the states of the three processes (running, ready, blocked) in the diagram according to the legend. It is sufficient to give the states for the first 200 milliseconds.

Hint: The first three time units are already filled in.

Upload your file here. Maximum one file.

All file types are allowed. Maximum file size is **50 GB**

Select file to upload

Maximum marks: 5

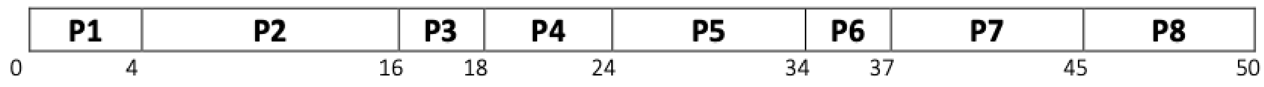
10 Planlegging 2

Scheduling algorithms and waiting time (6 points)

The following set of processes is given along with their arrival time and the length of their CPU-bursts:

Process	Arrival time (ms)	Burst time (ms)
P1	0	4
P2	2	12
P3	5	2
P4	6	6
P5	8	10
P6	12	3
P7	15	8
P8	22	5

For the FCFS scheduling algorithm, the **Gantt diagram** giving the sequence and timing of process execution looks as follows:



The **waiting time** of a process is defined as the different between the actual termination of the process and its earliest possible termination time if no other processes would interfere. In our example, the waiting time for P3 would be (18 ms - 7 ms) = 11 ms – it could be finished at (P3 arrival time + P3 burst time) = (5 ms + 2 ms) = 7 ms, but actually finishes at 18 ms.

The **average waiting time** is the arithmetic average over all waiting times. For FCFS in our example, this is = (0+2+11+15+32+6+17+5) ms / 8 = 88/8 ms = 11 ms.

Draw the respective **Gantt diagrams** and calculate the **average waiting time** for the following scheduling algorithms:

1. Non-preemptive SJF (shortest job first)
2. Preemptive RR (round robin) with a time slice (quantum) of 6 ms
3. Preemptive SRTF (shortest remaining time first)

Upload your file here. Maximum one file.

All file types are allowed. Maximum file size is **50 GB**

Select file to upload

Maximum marks: 6

11 Planlegging 3

Scheduling and semaphores (2 points)

We have learned in the course that if the semaphore operations **wait** and **signal** are not executed atomically, then mutual exclusion may be violated.

Assume that **wait** and **signal** are implemented as follows and that **semaphores are used as mutexes** (i.e. they can only have values 1 and 0):

```
void wait (Semaphore S) {
   while (S.count <= 0) {}
   S.count = S.count - 1;
}
void signal (Semaphore S) {
   S.count = S.count + 1;
}
```

Describe a scenario of context switches (i.e. at which points in the given code does a context switch occur) in which two threads, T1 and T2, can both enter a critical section guarded by such a semaphore as a result of a lack of atomicity.

Fill in your answer here

Maximum marks: 2

12 I/O- og diskplanlegging 1

```
void read_input(int fd, char **buf) {
  *buf = malloc(1024);
  memset(*buf, 0, 1024);
  lseek(fd, 3, SEEK_SET);
  read(fd, *buf, 1023);
}
```

Unix file I/O (3 points)

Consider the following function **read_input()**. Assume that the file which the descriptor *fd* refers to contains the following sequence of characters:
abcd

Which string does the buffer buf contain after the **read()** call returns? Assume that no errors occur.

Fill in your answer here

What is the purpose of the memset() call in the code above?

Fill in your answer here

Maximum marks: 3

13 I/O- og diskplanlegging 2

Disk scheduling (3 points)

A disk has 16 tracks. The related I/O scheduler receives a number of read requests for a certain set of tracks.

Initially, the read requests in set L1 are already known to the I/O scheduler. The requests in set L2 arrive after the I/O scheduler has processed one requests; the requests in L3 arrive after the I/O scheduler has processed three additional requests (so overall four).

Initially, the disk read/write head is located at track 0.

L1 = {4, 7, 11, 3}
L2 = {2, 13, 1}
L3 = {15, 5, 6}

Give the order of tracks that are read for an I/O scheduler that uses the **First In First Out (FIFO)** strategy:

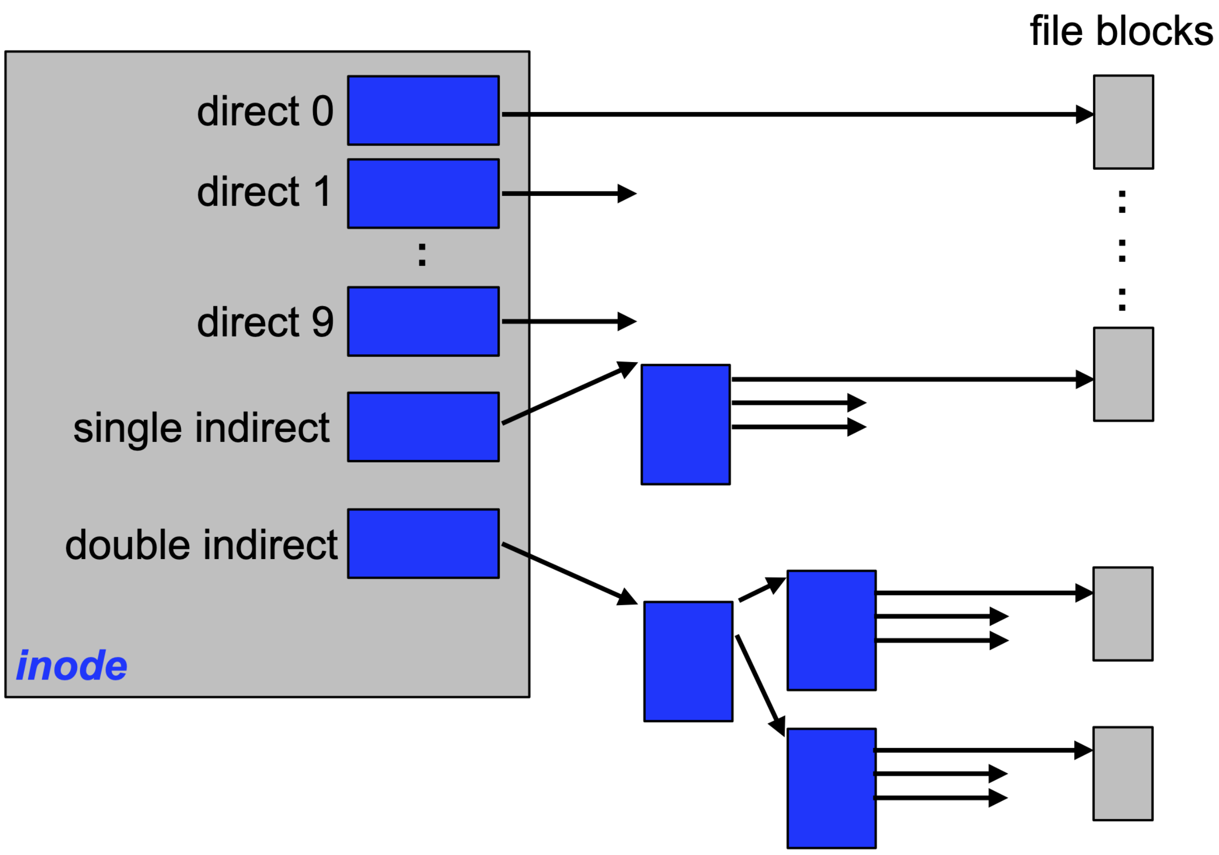
Fill in your answer here

Maximum marks: 3

14 Filsystemer

Inode-based file systems (4 points)

A file system uses inodes as shown in the picture – direct, single and double indirect inodes (but no triple indirect ones).



Calculate the maximum possible file size in this file system under the given assumptions:

- The block size is 1024 bytes
- A block number requires four (4) bytes of storage

Describe all steps of your calculation and give the value for the maximum possible file size.

Fill in your answer here

Maximum marks: 4