Written examination in Operating Systems

February 05, 2024

Last name: _____

First name: _____

Student number: _____

Signature: _____

Written examination in Operating Systems

February 05, 2024

Please write only your student number — but **not your name** — on this or any of the following sheets. By omitting your name a pseudonymized correction of your exam can be achieved. The first page with your name will be removed before correction and consequently the corrector cannot be biased when correcting your exam. By putting your student number on all pages you make sure that even in the case the stapling gets lost each page can be attributed to your exam.

Student number: _____

Result:

Question:	1	2	3	4	5	6	7	8	9	10	11	12	Total
Points:	10	6	7	8	8	6	9	6	6	10	6	8	90
Score:													

1.0: 90-85.5, **1.3**: 85-81, **1.7**: 80.5-76.5, **2.0**: 76-72, **2.3**: 71.5-67.5,
2.7: 67-63, **3.0**: 62.5-58.5, **3.3**: 58-54, **3.7**: 53.5-49.5, **4.7**: 49-45, **5.0**: <45

Question 1

Points: ..... (max. 10 points)

Decide whether the following statements are correct or wrong and explain shortly why.

(a) Since operating systems based on a micro-kernel architecture are more robust, basically all relevant modern operating systems are based on this architecture.

☐ True ☑ **Wrong**

> **Solution:** Micro-kernels have also some drawbacks, e.g., complexity or process switching overhead. Linux is a very popular OS built on a monolithic kernel.

(b) In some scenarios a singletasking computer system can execute programs faster than a multitasking system.

☑ **True** ☐ Wrong

> **Solution:** If (pseudo-)parallelization is not required, singletasking can complete tasks faster because there is no overhead for context switching.

(c) A fork bomb is a problem for computers with very little resources, e.g., embedded systems.

☐ True ☑ **Wrong**

> **Solution:** Every computer may be affected by fork bombs.

(d) The kernel of an operating system may implement more than one scheduling algorithm.

☑ **True** ☐ Wrong

> **Solution:** Some OS, e.g., Linux allow to specify the scheduler on a per process level.

(e) Semaphores can be used to implement mutexes.

☑ **True** ☐ Wrong

> **Solution:** Binary semaphores are mutexes.

(f) Interrupts are used to simplify debugging. ☐ True ☑ **Wrong**

> **Solution:** The occurrence of interrupts makes debugging more difficult.

(g) Every x86 compatible CPU starts in Real Mode. ☑ **True** ☐ Wrong

> **Solution:** Only modern OS (or the firmware) switches into protected mode as early as possible.

(h) The rotational speed of a hard disk drive (HDD) is the only limiting factor of its performance. ☐ True ☑ **Wrong**

> **Solution:** The seek time and the cache also influence the performance.

(i) The block size of the storage devices defines an lower bound for the cluster size of a file system. ☑ **True** ☐ Wrong

> **Solution:** Blocks are the smallest addressable unit from the hardware perspective.

(j) The File Allocation Table of a VFAT file system grows over time when more files are created. ☐ True ☑ **Wrong**

> **Solution:** The FAT has a fixed size (which equals the number of available clusters).

Question 2

Points: ..... (max. 6 points)

Give a command that can be used to...

(a) print out the path of the present working directory in the shell.

Solution: `pwd`

(b) concatenate the content of different files or print out the content of a file.

Solution: `cat`

(c) modify the cron jobs for the current user.

Solution: `crontab -e`

(d) modify a certain pattern in a file.

Solution: `sed`

(e) print out lines from the beginning of a file in the shell.

Solution: `head`

(f) list the content of the current directory.

Solution: `ls`

(g) sort the lines of a text file.

Solution: `sort`

(h) create an archive file.

Solution: `tar`

(i) delete files or directories.

Solution: `rm`

(j) output a string in the shell.

Solution: `echo`

(k) create a hard link.

Solution: `ln`

(l) modify the permissions of files or directories.

Solution: `chmod`

Question 3

Points: ..... (max. 7)

(a) Explain why it is impossible to implement the optimal replacement strategy OPT. (1)

> **Solution:** Because it is impossible to predict the future and therefore the future request sequence is unknown.

(b) Discuss whether the random strategy is a good or a bad choice for finding the next free block of memory compared to first fit, next fit, or best fit. (2)

> **Solution:** Since all the other approaches have their drawbacks, random may actually perform on average similarly well and is much easier to implement.

(c) Name one advantage and one drawback for larger page sizes. (1)

> **Solution:** Larger page sizes result in smaller page tables but more internal fragmentation.

(d) Explain in which situations a page fault exception occur. (1)

> **Solution:** A process tries to access a page, which is not located in the physical main memory.

(e) Explain in which situations an access violation exception or general protection fault exception occur. (1)

> **Solution:** A process tried to access a virtual memory address which it is not allowed to access.

(f) Explain in which situations the use of an SSD may be a bad choice. (1)

> **Solution:** If a lot of write and/or erase operations have to be conducted.

Question 4

Points: ..... (max. 8 points)

(a) Name the three sorts of process context information the operating system stores.

Solution: User context, hardware context and system context.

(b) Explain the task of the dispatcher.

Solution: It carries out the state transitions of the processes.

(c) Explain the task of the scheduler.

Solution: It specifies the execution order of the processes.

(d) Explain what the PID is.

Solution: the process identifier (PID) is an integer number used to uniquely identify a process.

(e) Explain what the PPID is.

Solution: The parent process identifier (PPID) is an integer number too. It is the process ID of a parent process of a process.

(f) Describe the effect of calling the system call `fork`.

Solution: If a process calls `fork`, an identical copy is started as a new process.

(g) Describe the effect of calling the system call `exec`.

Solution: The system call `exec` replaces a process with another one.

(h) Explain why some operating systems have one or more system idle processes.

Solution: If no process is in the state `ready`, the system idle process gets the CPU assigned. The system idle process is always active and has the lowest priority. Due to the system idle process, the scheduler must never consider the case that no active process exists.

Question 5

Points: ..... (max. 8 points)

(a) Explain the advantage of using the operations signal and wait compared with busy waiting.

Solution: When using busy waiting, computing time of the CPU is wasted because it is again and again occupied by the waiting process. Using signal and wait causes lesser CPU workload because the waiting process is blocked and later unblocked.

(b) Name two problems that can arise from locking.

Solution: Starvation and deadlock.

(c) Explain the difference between signaling and locking.

Solution: Signaling specifies the execution order of the critical sections of processes.

Solution: Blocking secures critical sections. The execution order of the critical sections of the processes is not specified. It is just ensured that the execution of critical sections does not overlap.

(d) Mark the scheduling method that is implemented by message queues.

- ☐ Round Robin
- ☐ LIFO
- ☐ SJF
- √ **FIFO**
- ☐ LJF

(e) Specify how many processes can communicate with each other via a pipe.

Solution: 2

(f) Explain the effect, when a process tries to write data into a pipe without free capacity.

Solution: The process that tries to write into the pipe is blocked.

(g) Explain the effect, when a process tries to read data from an empty pipe.

Solution: The process that tries to read from the pipe is blocked.

(h) Name the two different types of pipes.

Solution: Anonymous pipes and named pipes.

(i) Name the two different types of sockets.

Solution: Connection-less sockets (also called: datagram sockets) and connection-oriented sockets (also called: stream sockets).

Question 6

Points: ..... (max. 6)

(a) What are possible (direct or indirect) sources for an interrupt? (2)

√ **System Calls**
√ **Program Error**
√ **E-mail reception**
√ **Hardware Failure**
√ **Button press**

(b) What is the name of the data structure the OS uses to lookup which handler to run upon interrupt? (1)

□ ISR
√ **IVT**
□ IRQ
□ PIC

(c) Describe two different approaches to handle concurrent interrupts. (2)

Solution: Interrupts can be handled nested or sequentially.

(d) Name the bus that contains the line to signal the occurrence of an interrupt. (1)

Solution: Control bus.

Question 7 **Points:** ..... (max. 9)

(a) Show Belady's anomaly by performing the access sequence with the replacement strategy FIFO once with a cache with a capacity of 3 pages and once with 4 pages. Also calculate the hit rate and the miss rate for both scenarios. (8)

Requests:	**3**	**2**	**1**	**0**	**3**	**2**	**4**	**3**	**2**	**1**	**0**	**4**
Page 1:	**3**	3	3	**0**	0	0	**4**	4	4	4	4	4
Page 2:		**2**	2	2	**3**	3	3	3	3	**1**	1	1
Page 3:			**1**	1	1	**2**	2	2	2	2	**0**	0

Hit rate: 3/12 = 25%
Miss rate: 9/12 = 75%

Requests:	**3**	**2**	**1**	**0**	**3**	**2**	**4**	**3**	**2**	**1**	**0**	**4**
Page 1:	**3**	3	3	3	3	3	**4**	4	4	4	**0**	0
Page 2:		**2**	2	2	2	2	2	**3**	3	3	3	**4**
Page 3:			**1**	1	1	1	1	1	**2**	2	2	2
Page 4:				**0**	0	0	0	0	0	**1**	1	1

Hit rate: 2/12 = 16.66%
Miss rate: 10/12 = 83.33%

(b) Explain why fragmentation in memory management is irrelevant for modern operating systems. (1)

Solution: Because of the virtual memory concept.

Question 8

Points: ..... (max. 6 points)

Take a look at the given file system tree.

```
/
├── bin
│   ├── bash
│   ├── dash
│   └── ...
├── mnt
├── src
│   ├── factory
│   ├── main
│   │   ├── worker.py
│   │   └── app.py
│   ├── resources
│   └── util
└── test
    ├── main
    │   ├── test_factory.py
    │   └── test_save.py
    └── misc
```

(a) Write down the absolute path to the file `test_save.py`:

Solution: `/test/main/test_save.py`

(b) Write down the relative path from `src` to the file `app.py`:

Solution: `main/app.py`

(c) Write down the relative path from the `factory` directory to the file `test_save.py`:

Solution: `../../test/main/test_save.py`

(d) Another file system gets *mounted* at `/mnt`. The tree of this file system looks like this:

```
/
├── backups
│   ├── user1
│   │   └── archive1.tar
│   └── user2
│       └── archive2.tar
└── data
    ├── audio
    │   └── sound.mp3
    └── video
```

Write down the absolute path to the file `archive1.tar`:

Solution: `/mnt/backups/user1/archive1.tar`

(e) A symbolic link to `sound.mp3` shall be created in the directory `resources`. Describe the information that needs to be added to the file system.

Solution: The symbolic link contains the relative or absolute path to the file `sound.mp3`.

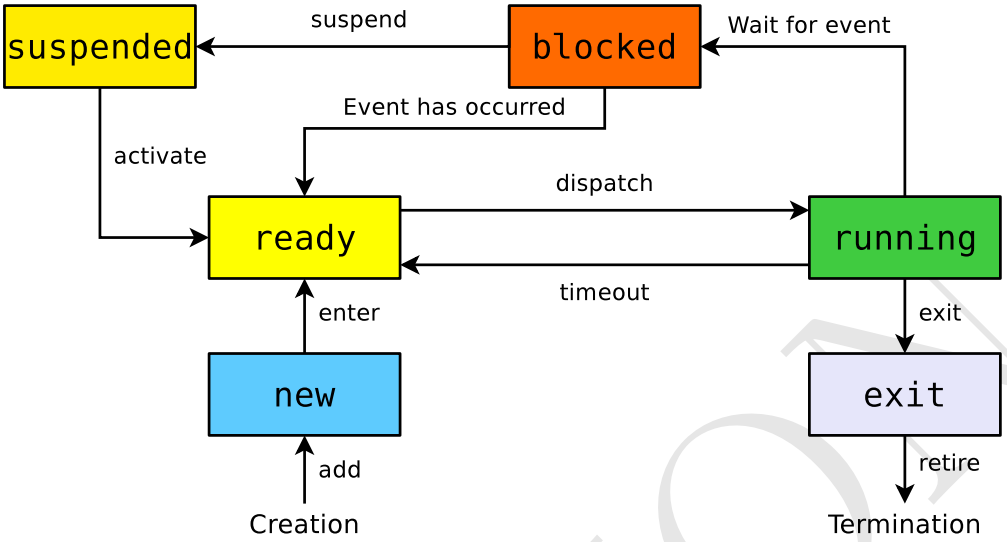
(f) A hard link to `sound.mp3` shall be created in the directory `resources`. Describe the information that needs to be added to the file system.

Solution: A hard link cannot be created beyond file system boundaries.

Question 9

Points: ..... (max. 6 points)

(a) Enter the names of the states in the diagram of the process state model with 6 states.



Question 10

Points: ..... (max. 10)

(a) Explain which problem may occur when static priorities are used for scheduling. (1)

> **Solution:** Processes with low priority may starve.

(b) Some systems implement one or more idle process. Explain what idle processes are good for. (1)

> **Solution:** The system idle process is always active and has the lowest priority. It gets the CPU assigned when no other process is ready. Due to the system idle process, the scheduler must never consider the case that no active process exists.

(c) The two processes P_A (4 ms CPU time) and P_B (26 ms CPU time) are both in state `ready` at time point 0 and are to be executed one after the other. (6)
Fill the table with correct values. *(Hint: Runtime = Lifetime)*

Execution order	**Runtime** P_A	P_B	**Average runtime**	**Waiting time** P_A	P_B	**Average waiting time**
P_A, P_B	4	30	17	0	4	2
P_B, P_A	30	26	28	26	0	13

(d) Explain what can be observed from the values you filled into the table in (c). (2)

> **Solution:** If a short-running process runs before a long-running process, the runtime and waiting time of the long process process get slightly worse. If a long-running process runs before a short-running process, the runtime and waiting time of the short process get significantly worse Therefore, it is favorable to run short processes first, if a low average waiting time and average runtime is desired.

Question 11

Points: ..... (max. 6)

(a) Explain what the following code is doing and whether it is correct and complete. State the value of the variable `ret` and explain its meaning. (3)

```
1  #include <stdio.h>
2  #include <unistd.h>
3  #include <stdlib.h>
4
5  void main() {
6    int ret = fork();
7
8    if (ret > 0) {
9      printf("Parent.\n");
10     exit(0);
11   }
12   else {
13     printf("Child.\n");
14     exit(0);
15   }
16 }
```

Solution: The variable `ret` will contain the PID of the child process in the if clause. The else case may indicate that fork() has returned 0, in which case the child process would be running. However, it is not checked whether the function returns an error.

(b) Describe what will be printed when executing the following program (after compiling and linking it). State the return value of the program. Explain your expectations. (3)

```
1  #include <unistd.h>
2  #include <stdio.h>
3
4  int main(void)
5  {
6      printf("Execute ls...\n");
7      execlp("/bin/echo", "/bin/echo", "now", NULL);
8      printf("done.\n");
9      return 5;
10 }
```

Solution: The program will print
`Execute ls...`
`now`
The program will return 0 (the return value of echo). The process will be replaced by echo.

Question 12

Points: .....(max. 8 points)

(a) Perform the deadlock detection with matrices and check if a deadlock occurs.

$$\text{Existing resource vector} = \begin{pmatrix} 9 & 6 & 8 & 7 & 6 & 7 \end{pmatrix}$$

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

Solution: The existing resource vector and the current allocation matrix are used to calculate the available resource vector.

$$\textit{Available resource vector} = \begin{pmatrix} 1 & 1 & 2 & 2 & 3 & 2 \end{pmatrix}$$

Only process 1 can run with this available resource vector. The following available resource vector results when process 1 has finished execution and deallocates its resources.

$$\textit{Available resource vector} = \begin{pmatrix} 3 & 1 & 4 & 5 & 5 & 2 \end{pmatrix}$$

Only process 3 can run with this available resource vector. The following available resource vector results when process 3 has finished execution and deallocates its resources.

$$\textit{Available resource vector} = \begin{pmatrix} 4 & 4 & 6 & 6 & 5 & 3 \end{pmatrix}$$

Only process 4 can run with this available resource vector. The following available resource vector results when process 4 has finished execution and deallocates its resources.

$$\textit{Available resource vector} = \begin{pmatrix} 7 & 5 & 6 & 7 & 6 & 4 \end{pmatrix}$$

Process 2 is not blocked.
No deadlock occurs.