Information page for written examinations at Linköping University

Examination date	2019-08-26
Room (1)	TER2(27)
Time	14-18
Edu. code	TDDB68
Module	TEN1
Edu. code name **Module name**	Concurrent Programming and Operating Systems (Processprogrammering och operativsystem) Examination (Tentamen)
Department	IDA
Number of questions in the examination	6
Teacher responsible/contact person during the exam time	Mikael Asplund
Contact number during the exam time	0700 895 827
Visit to the examination room approximately	15-16
Name and contact details to the course administrator (name + phone nr + mail)	Veronica Kindeland Gunnarsson 013-28 56 34 adm-gu@ida.liu.se
Equipment permitted	Engelsk ordbok / english dictionary, Miniräknare / pocket calculator
Other important information	
Number of exams in the bag	

TENTAMEN / *EXAM*

TDDB68

Processprogrammering och operativsystem / *Concurrent programming and operating systems*

2019-08-26

Examiner: Mikael Asplund (0700895827)

Hjälpmedel / *Admitted material:*

– Engelsk ordbok / *Dictionary from English to your native language*;
– Miniräknare / *Pocket calculator*

General instructions

- This exam has 6 assignments and 8 pages, including this one.
  Read all assignments carefully and completely before you begin.
- Please **use a new sheet of paper for each assignment**, because they will be corrected by different persons.
  Sort the pages by assignments and number them consecutively.
- You may answer in either English or Swedish. English is preferred because not all correcting assistants understand Swedish.
- Write clearly. Unreadable text will be ignored.
- Be precise in your statements. Unprecise formulations may lead to a reduction of points.
- **Motivate clearly** all statements and reasoning.
- Explain calculations and solution procedures.
- The assignments are *not* ordered according to difficulty.
- The exam is designed for 40 points. You may thus plan about 5 minutes per point.
- Preliminary grading thresholds:
  - 3: at least 20p
  - 4: at least 27p
  - 5: at least 34p

Good luck!

1. **Multiple choice quesions** (10p)

Below are 10 multiple choice questions. Please answer them by removing the last page of the exam, fill the appropriate boxes, and hand it in together with the rest of your answers. Please note that there might be more than one correct option. Each question below can give 0 or 1 point(s), and to get 1 point you must have identified exactly the right set of choices.

(a) Consider the program below. How many times will it outout the line "Hello"? (1p)

```
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>

int main()
{
  pid_t pid1 = fork();
  printf("Hello\n");
  if (pid1 == 0) {
    fork();
    printf("Hello\n");
  }

  return 0;
}
```

A) 2 times
B) 3 times
C) 4 times
D) 6 times

(b) Which of the following statements are true about Banker's algorithm? (1p)

A) A safe state implies that a deadlock can never occur in any future state.
B) Banker's algorithm checks the Coffman conditions to find deadlocks.
C) Banker's algorithm guarantees freedom from starvation.
D) Banker's algorithm requires a-priori knowledge of the resource demands of every involved process.

(c) Which of of the following events will take place when switching from one process (old) to another (new). (1p)

A) Changing the set of currently open files to match with the new process.
B) Copying data from the process control block (PCB) of the new process into the interrupt vector table.
C) Storing CPU register states into the PCB of the old process.
D) Updating registers relating to memory management (e.g., base address pointer).

(d) Which of the terms below refer to methods used for allocation of disk space for a file? (1p)

A) Virtual allocation
B) Indexed allocation
C) Segment allocation
D) Contiguous allocation

(e) Which of the following can be said to constitute vulnerabilities? (1p)

A) A denial-of-service attack
B) A program written in C
C) Use of weak or commonly used passwords
D) A bug in the OS that allows unauthorized privilege escalation

(f) Which of the following are classical scheduling algorithms (as described in the course literature): (1p)

A) Shortest-job first scheduling
B) Burst-first scheduling
C) Priority scheduling
D) Timer scheduling

(g) Which of the following operations will result in a system call for a normal operating system? (1p)

A) Reading the value of a variable stored on the stack
B) Creating a new file
C) Allocating memory on the heap
D) Performing a jump (JMP) instruction

(h) Which of the following concepts relate to multiprocessor scheduling? (1p)

A) Segmentation
B) Task migration
C) Hard vs. soft affinity
D) Co-scheduling

(i) Which of the following statements are true? (1p)

A) A process in a modern operating system has its own protected memory.
B) Different threads belonging to the same process share memory with each other.
C) A thread is only run when there is no process that wants to use the CPU.
D) Threads are guaranteed to be deadlock-free.

(j) Which of these parameters will affect the effective access time (EAT) of demand paging? (1p)

A) Page fault rate (probability of page fault)
B) Segment table length
C) Time to swap page in from disk
D) Disk size

2. **Synchronization** (9p)

Servers can be designed to limit the number of open connections. For example, a server may wish to have only N socket connections at any point in time. As soon as N connections are made, the server will not accept another incoming connection until an existing connection is released.

The current number of available connections could be kept track of by a counter in shared memory, initialized to N:

```
shared int N_available_connects = N;
```

Basically, each incoming connection request could create a new thread that would execute the following function:

```
connection *void handle_connection_request( void )
{
  while (N_available_connects == 0)
     ;    // wait...
  N_available_connects = N_available_connects - 1;
  return make_new_connection();
}
```

and upon closing a connection, the thread would call the following function:

```
void close_connection( connection *c )
{
   release_connection(c);
   N_available_connects = N_available_connects + 1;
}
```

(a) Show by an example scenario that, without using additional synchronization, the above code can lead to a race condition, resulting in undesired behavior. (1p)

(b) Give a properly synchronized solution (pseudocode), using either *test&set* or *atomic_swap* instructions. Explain why your solution guarantees absence of race conditions. (3p)

(c) Explain whether your solution is fair or not, i.e., whether it guarantees that no connection request could be postponed indefinitely. (1p)

(d) Suggest (pseudocode/English) how to extend your solution to avoid busy waiting in the case where no connections are available. (1p)

(e) Give a solution using a *monitor* abstraction. Use C or pseudocode notation with appropriate keywords to identify the monitor components, and explain your code. (3p)

3. **Deadlocks** (4p)

(a) Consider the following resource allocation problem in a system with 3 resources (R1-R3), and 4 processes (P1-P4). The table indicates the currently allocated resources and in parenthesis the maximum possible demand. (4p)

	R1	R2	R3
P1	0 (4)	2 (2)	0 (1)
P2	0 (3)	2 (4)	0 (1)
P3	3 (3)	1 (2)	3 (4)
P4	1 (3)	0 (0)	2 (3)

The currently available resources are: [1, 2, 4]. Use Banker's algorithm to determine if the request [0, 0, 1] from Process P1 should be granted.

4. **Filesystems and scheduling** (7p)

(a) Can, in principle, the same file be opened by multiple processes? (3p)
If yes, explain (draw a commented figure) how the internal data structures for opened files in the operating system provide this possibility.
If not, explain why it is not possible.

(b) Given a single-CPU system and the following set of processes with arrival times (in milliseconds), expected maximum execution time (ms), and priority **(5 is highest, 1 is lowest priority)**. (4p)

Process	Arrival time	Execution time	Priority (as applicable)
P_1	0	3	1
P_2	2	2	2
P_3	4	4	3
P_4	7	1	5
P_5	9	2	4

For each of the following scheduling algorithms, create a Gantt chart (time bar diagram, starting at $t = 0$) that shows when the processes will execute on the CPU. Where applicable, the time quantum will be 2 ms. Assume that a task will be eligible for scheduling immediately on arrival. If you need to make further assumptions, state them carefully and explain your solution.

(i) FIFO

(ii) Shortest-job first scheduling without preemption

(iii) Priority scheduling *with* preemption

(iv) Round-robin scheduling

5. **Memory management** (5p)

(a) Consider a page-based virtual memory system with a page size of $2^{10} = 1024$ bytes where virtual memory addresses have 32 bits. If using *multi-level paging*, determine how many levels of paging are required (assuming each table must fit in one page), and describe the structure of the virtual addresses (purpose, position and size of its bit fields). (3p)

(b) Explain (annotated figure) how the physical address from (a) is calculated by multi-level paging from a virtual address. (2p)

6. **Security** (5p)

(a) What is a buffer-overflow attack? Describe the vulnerability, the factors that contribute to it, and give a scenario of how it can be exploited by an attacker to hijack the control of a running server program. (4p)

(b) How can using *virtual machines* increase the security of a system? (1p)

Intentionally empty page.

AID-nr: **Date:** **Page number:**

Course code: TDDB68 **Module:** TEN1

Multiple choice form for answering question 1. Please put X:s in the appropriate cells:

	A	B	C	D
1 a)				
1 b)				
1 c)				
1 d)				
1 e)				
1 f)				
1 g)				
1 h)				
1 i)				
1 j)				

Optional: if you feel you need to clarify your interpretation of the question you can do so here. This is not needed if your answer is correct.

1 a)	
1 b)	
1 c)	
1 d)	
1 e)	
1 f)	
1 g)	
1 h)	
1 i)	
1 j)	