Försättsblad till skriftlig tentamen vid Linköpings Universitet

(fylls i av ansvarig)

Datum för tentamen	20 oct 2011
Sal	TER1, TERD
Tid	14-18
Kurskod	TDDB68
Provkod	TEN1
Kursnamn/benämning	Processprogrammering och operativsystem
Institution	IDA
Antal uppgifter som ingår i tentamen	8
Antal sidor på tentamen (inkl. försättsbladet)	6
Jour/Kursansvarig	Christoph Kessler
Telefon under skrivtid	0703-666687
Besöker salen ca kl.	16:00
Kursadministratör (namn + tfnnr + mailadress)	Gunilla Mellheden, 013-282297 e. 0705-979044, gunme@ida.liu.se
Tillåtna hjälpmedel	Engelsk ordbok, miniräknare
Övrigt (exempel när resultat kan ses på webben, betygsgränser, visning, övriga salar tentan går i m.m.)	

Linköpings universitet
IDA Department of Computer and Information Sciences
Prof. Dr. Christoph Kessler

TENTAMEN / *EXAM*

TDDB68

Processprogrammering och operativsystem / *Concurrent programming and operating systems*

20 oct 2011, 14:00–18:00 TER1, TERD

Jour: Christoph Kessler (070-3666687, 013-282406), visiting ca. 16:00

Hjälpmedel / *Admitted material:*

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

General instructions

- This exam has 8 assignments and 5 pages, including this one.
  Read all assignments carefully and completely before you begin.
- It is recommended that you use a new sheet of paper for each assignment, because they will be corrected by different persons.
  Sort the pages by assignment, number them consecutively and mark each one on top with your exam ID and the course code.
- You may answer in either English or 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.
- Grading: U, 3, 4, 5. The preliminary threshold for passing is 20 points.

  Students in international master programs and exchange students will receive ECTS grades. Due to the anonymization of written exam correction, ECTS grades will be set by one-to-one translation from swedish grades (5=A, 4=B, 3=C, U=FX), according to the regulations by Linköping university.

1. (6 p.) **Interrupts, processes and threads**

   (a) Define the terms *process* and *thread.*
   In particular, what are the main differences between processes and threads? Be thorough! (2p)

   (b) What is a *process control block (PCB)*?
   What is its purpose?
   What data is contained in the PCB of a single-threaded process? (at least 4 relevant items are expected) (2p)

   (c) Two main methods for inter-process communication in a computer are *shared memory* and *message passing*.
   For each of them, give a short explanation of how it works and how the operating system is involved, i.e., which important system calls are to be used and what they do.
   Which of the two methods is likely to have less overhead if two processes communicate frequently with each other, and why? (2p)

2. (6 p.) **Synchronization**

   Electronic voting server

   An electronic voting system uses as central data structure a global array

```
unsigned int votes[N];    // N = number of parties
```

   for counting the number of votes for each of the N parties, initialized to zeroes.

   The voting server program, originally single-threaded, processes one vote request received from an external client at at time, by calling the function

```
void cast_vote ( unsigned int i )
{
 if (i>=N)
    error();
 else
    votes[i] = votes[i] + 1;
}
```

   In order to scale up to many millions of voters, the voting program should be multithreaded (with the `votes` array residing in shared memory of the voting server process) and run on a multicore server running a modern multithreaded operating system with preemptive scheduling, so that multiple calls to `cast_vote` can execute concurrently. It is your task to make the program thread-safe and preserve its correct behavior.

   (a) Using a simple contrived scenario (hint: use $N = 2$ parties and few votes), show with a concrete example (interleaving of shared memory accesses) that, without proper synchronization, race conditions are possible that could even cause that the wrong party wins the election. (1p)

(b) Identify the critical section(s) and protect the program against race conditions with a *single* mutex lock. (1p)

(c) Explain why the solution in (b) might have a scalability problem, and propose a better lock-based (race-free) solution. (1.5p)

(d) Assume that the server machine provides a transactional memory implementation (in either hardware or software). How does transactional memory work in general, what are its possible advantages over a lock-based approach, and how would it be used with the above `cast_vote` function? (2.5p)

3. (5 p.) **CPU Scheduling**

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

Process	Arrival time	Execution time	Priority (as applicable)
P_1	0	5	4
P_2	1	7	2
P_3	3	2	5
P_4	9	3	3
P_5	10	1	1

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 4 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. (5p)

(i) FIFO;

(ii) Round-robin;

(iii) Shortest Job First *without* preemption;

(iv) Priority Scheduling *without* preemption.

(v) Priority Scheduling *with* preemption.

4. (6 p.) **Deadlocks**

(a) There are four conditions that must hold for a deadlock to become possible. Name and describe them briefly. (2p)

(b) What is the difference between *deadlock* and *starvation*? (1p)

(c) You are given a system with 4 types of resources, A, B, C and D. There are 4 instances of A, 5 instances of B, 1 instance of C and 6 instances of D. Currently, 4 processes $P_1 ... P_4$ are running, and for each process, the resources currently held and its total maximum resource need (including the already held ones) for each type are given as follows:

Process	Already held				Maximum total need			
	A	B	C	D	A	B	C	D
P_1	0	1	0	1	0	2	0	4
P_2	1	2	1	0	2	2	1	3
P_3	1	0	0	2	2	2	0	3
P_4	1	0	0	2	2	0	1	3

(i) Show that the system is currently in a safe state (calculation). (1.5p)

(ii) In the situation given above, Process P_1 now asks for 1 instance of B and 1 of D. in addition to the B and D it already has. Is it safe to grant the request? Why or why not? (calculation) (1.5p)

5. (6.5 p.) **Memory management**

(a) Several 32-bit processors use *three-level paging* to address the large page table problem, where the page table itself is stored in main memory. (If you don't recall three-level paging, you may answer, with reduced points, this and the following question for one- or two-level paging.)

Explain three-level paging for a 32-bit virtual address space and a memory page size of 1024 bytes. Show the structure of virtual (logical) addresses and explain (with a well commented drawing) how to compute physical memory addresses. (2.5p)

(b) Explain how paging supports sharing of memory between processes. (1p)

(c) Describe the principle of *segmentation*.

Why would one prefer a segmented memory model instead of a paged memory model?

And which drawback does segmentation have, compared to paging? (2p)

(d) How can segmentation and paging be combined? (1p)

6. (4.5 p.) **File systems**

(a) What information is usually contained in a *file control block* (FCB)? (At least 4 different items are expected) (1p)

(b) Where is the FCB contents stored after a file has been opened? (0.5p)

(c) Is the file allocation method *indexed allocation* susceptible to *external* fragmentation? Give a short explanation. (1p)

(d) Describe one case where the file system is *not* an appropriate abstraction for secondary storage, and explain why. (1p)

(e) Name and describe one disk scheduling algorithm of your choice (but *not* FIFO/FCFS, which is a trivial one). (1p)

7. (3.5 p.) **OS Structures and Virtualization**

(a) What is the main idea of the *microkernel* approach to OS structuring, what is its main advantage and what is its main drawback? (2p)

(b) What does a *hypervisor* (also known as *virtual machine monitor, VM implementation*) do?

Illustrate your answer with a commented figure that shows where the hypervisor is positioned in the system software stack and with which other system entities it interacts. (1.5p)

8. (2.5 p.) **Protection and Security**

(a) Name at least one (software or hardware) measure to prevent *buffer-overflow* vulnerabilities. (0.5p)

(b) Unix offers the so-called `setuid` flag for (executable) files.

(i) How does the `setuid` mechanism work?

(ii) What is its (intended) purpose?

(iii) Why is it problematic from a security perspective?

(iv) Suggest a better alternative (technical term, no details). (2p)

Good luck!