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

TENTAMEN / *EXAM*

TDDB68

Processprogrammering och operativsystem /
Concurrent programming and operating systems

TTIT61
Tema 1, Processprogrammering och operativsystem

26 aug 2010, 14:00–18:00 TER1

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. (7 p.) **Interrupts, processes and threads**

   (a) Define the terms *process*, *kernel thread* and *user thread*, and explain the differences between them. (3p)

   (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) What kind of hardware support (2 technical entities) in a processor is required for enabling multitasking with preemptive scheduling? (1p)

   (d) The *system call API* (application programming interface) is a software abstraction for invoking OS functionality. Name one kind of technical details that it abstracts from.
   Why is such abstraction important for application programming? (1p)

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

   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. (0.5p)

(b) Give a properly synchronized solution (C/pseudocode), using either *test&set* or *atomic_swap* instructions.

Explain why your solution guarantees absence of race conditions.

Explain whether your solution is fair or not, i.e., whether it guarantees that no connection request could be postponed indefinitely. (3p)

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

(d) Give a solution using a *monitor* abstraction. Use C or pseudocode notation with appropriate keywords to identify the monitor components, and explain your code. (1.5p)

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	3
P_3	4	2	5
P_4	9	3	2
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 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. (5p)

(i) FIFO;

(ii) Round-robin;

(iii) Shortest Job First *without* preemption;

(iv) Priority Scheduling *without* preemption.

(v) Priority Scheduling *with* preemption.

4. (5 p.) **Deadlocks**

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

Consider the Dining Philosophers problem as discussed in the lectures, where each philosopher first tries to obtain the chopstick to her left. Describe how each of the four conditions applies in this scenario, thereby showing that a deadlock could occur. (3p)

(b) (2p) Construct a resource allocation graph with four processes and four resource instances such that

(i) the graph has a cycle and the processes on the cycle are deadlocked;

(ii) the graph has a cycle and the processes on the cycle are *not* deadlocked.

5. (9.5 p.) **Memory management**

(a) Which kind of fragmentation (external or internal) can occur in contiguous memory allocation? Explain your answer. (1p)

(b) Given a paged memory system with a translation lookaside buffer (TLB). What is the effective memory access time if the physical memory access time is 90ns, a TLB lookup takes 10ns and the average TLB hit rate is 90%? Explain your calculation. (1p)

(c) Paging for large virtual address spaces and small page sizes leads to the large-page-table problem. Explain *one* technique how this problem can be solved. Be thorough! (2.5p)

(d) Given a virtual memory system with 4 page frames, how many page faults occur with the *Least-Recently Used* replacement strategy when pages are accessed in the following order:

1, 2, 3, 4, 5, 1, 3, 2, 1, 5, 6, 2, 5.

(Justify your answer. Just guessing the right number is not acceptable.) (1.5p)

(e) For the same access sequence, what would be the theoretical *minimum* number of page faults, assuming the clairvoyant optimal replacement strategy? (Justify your answer. Just guessing the right number is not acceptable.) (1.5p)

(f) What is *thrashing* in a virtual memory system? How does it occur? And what can be done about it? (2p)

6. (4 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) What is the basic idea and motivation of the Unix *inode* structure? (1.5p)

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

7. (1.5 p.) **OS Structures and Virtualization**

(a) What is the main idea of the *microkernel* approach to OS structuring, and what is its main advantage? (1.5p)

8. (2 p.) **Protection and Security**

   (a) How does memory segmentation support protection? (1p)

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

Good luck!