TENTAMEN / EXAM

TDDB68
Processprogrammering och operativsystem /
Concurrent programming and operating systems

TDDB72
Processprogrammering, operativsystem och realtidsystem /
Concurrent programming, operating systems and real-time systems

18 dec 2006, 14:00–18:00

Jour: Christoph Kessler (070-3666687, 013-282406)

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 for each assignment. Number all your sheets, and mark each sheet on top with your name, personnummer, 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.
  For exchange students (with a ’P’ in the personnummer), ECTS marks will be applied.
OBS C:are antagna före 2001: Om du vill ha ditt betyg i det gamla betygsystemet (U, G, VG) skriv detta tydligt på omslaget av tentan. Annars kommer vi att använda det nya systemet (U, 3, 4, 5).
1. (3 p.) Processes and threads
   
   (a) Define the three terms process, kernel thread and user thread, and explain the differences between these. (3p)

2. (7 p.) Synchronization
   
   Consider a shared stack of positive integers, to be implemented as a shared array of sufficiently large size and a shared stack pointer. The stack operations push and pop can be called concurrently by multiple threads. When the stack is found to be empty, operation pop should return an error code (−1).

   (a) Write (unprotected) code for the stack initialization and for the operations push and pop. (Use C, Java or pseudocode.)
   
   Describe a contrived scenario with two threads where the concurrent execution of these unprotected operations leads to an unexpected result (i.e., a race condition). Identify precisely the variables and statements that could potentially be involved in any race condition (that is, describe the critical section(s)). (3p)

   (b) You are given a single-processor system where multi-tasking is implemented by a preemptive scheduling algorithm, and where an atomic test-and-set operation is available. Here, there exist actually two different hardware-supported mechanisms to protect critical sections. Name and explain these two variants, and show for each case how your code of part (a) is to be modified to protect its critical section(s) against race conditions. (2p)

   (c) Write a monitor solution for the shared stack (based on your code above). Use C or pseudocode notation with appropriate keywords to identify the monitor components, and explain your code. (2p)

3. (6 p.) CPU Scheduling
   
   (a) 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).

<table>
<thead>
<tr>
<th>Process</th>
<th>Arrival time</th>
<th>Execution time</th>
<th>Priority (as applicable)</th>
</tr>
</thead>
<tbody>
<tr>
<td>P₁</td>
<td>0</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>P₂</td>
<td>2</td>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>P₃</td>
<td>3</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>P₄</td>
<td>5</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>P₅</td>
<td>6</td>
<td>1</td>
<td>4</td>
</tr>
</tbody>
</table>

   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 3 ms. (5p)

   (i) FIFO;
   (ii) Round-robin;
   (iii) Shortest Job First without preemption;
   (iv) Priority Scheduling without preemption.
   (v) Priority Scheduling with preemption.
(b) What is the purpose of
   (i) the long-term / medium-term scheduler and
   (ii) the short-term scheduler in an operating system? (1p)

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) You are given a system with 4 types of resources, A, B, C and D. There are 3 instances of A, 5 instances of B, 2 instances of C and 6 instances of D. Currently, 4 processes P₁...P₄ are running, and for each process, the resources currently held and its total maximum resource need for each type are given as follows:

<table>
<thead>
<tr>
<th>Process</th>
<th>Already held</th>
<th>Maximum total need</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A B C D</td>
<td>A B C D</td>
</tr>
<tr>
<td>P₁</td>
<td>1 2 2 0</td>
<td>2 2 2 3</td>
</tr>
<tr>
<td>P₂</td>
<td>0 1 0 1</td>
<td>0 2 0 4</td>
</tr>
<tr>
<td>P₃</td>
<td>1 0 0 2</td>
<td>2 2 0 3</td>
</tr>
<tr>
<td>P₄</td>
<td>0 0 0 2</td>
<td>1 0 1 4</td>
</tr>
</tbody>
</table>

   (i) Show that the system is currently in a safe state (calculation). (1.5p)
   (ii) Process P₂ now asks for 1 instance of B and 1 of D. Can the request be granted? Why or why not? (calculation) (1.5p)

(c) How can the occurrence of a deadlock be detected when only one instance of each resource type exists in a system? (1p)

5. (7 p.) Memory management
   (a) Define internal fragmentation and external fragmentation in allocation of (physical) memory to processes. Give an example of a memory management technique that completely avoids internal fragmentation, and one that completely avoids external fragmentation. (2p)
   (b) 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, 4, 3, 1, 6, 4, 5, 2, 3, 2.

   (Justify your answer. Just guessing the right number is not acceptable.) (1.5p)
   (c) What is thrashing in a virtual memory system? How does it occur? And what can be done about it? (1.5p)
   (d) Describe the principle of segmentation. Why would one prefer a segmented memory model instead of a paged memory model? (2p)

6. (4 p.) File systems
   (a) What is the difference between hard links and soft (symbolic) links? (1p)
   (b) Describe the file allocation methods contiguous allocation and indexed allocation. What are their strengths and weaknesses? Describe one technique to extend indexed allocation for large files. (3p)
7. (2 p.) Disk I/O
   
   (a) What is the purpose of disk scheduling? (1p)
   
   (b) Name and describe one disk scheduling algorithm of your choice. (1p)

8. (5 p.) Protection and Security
   
   (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. (3p)
   
   (b) Suggest at least one (software or hardware) measure to prevent such buffer-overflow vulnerabilities. (0.5p)
   
   (c) Unix uses password salting in user authentication. What does that mean, and what is its purpose? In particular, which security problem can be (partly) avoided by salting? (1.5p)

Good luck!