TENTAMEN TDDB63 & TDDB68 – Operativsystem och Processprogrammering, 2 poäng

LINJE: D, DI, EI, I, i, C, Y

DATUM: 29 maj 2001

TID: 14.00 – 18.00

Hjälpmedel: Engelsk-Svensk-Engelsk ordbok
Miniräknare

Antal uppg.: 6
Antal poäng: 60

Betygsgränser: U, 3, 4 och 5:
Underkänd: - 30 p
3: 31 - 40 p
4: 41 - 50 p
5: 51 - 60 p

Betygsgränser för U, G och VG
Underkänd: - 30 p
G: 31 - 45 p
VG: 46 - 60 p

Skriv namn, klass och personnummer på samtliga lösa blad. I övrigt så gäller även följande:
1) Endast ett svar per blad. Använd endast framsidan (delfrågor kan vara på samma sida).
2) Sortera inlämnade svar med avseende på uppgiftsnummer i stigande ordning.

Poängavdrag kommer att göras om detta inte åtföljs!


Dina svar skall tydligt visa lösningsmetod. Enbart rätt svar kommer inte att ge poäng. I det fall du är osäker på frågeställning, skriv ner din tolkning och lös uppgiften utifrån din tolkning.

Lycka till!

Uwe & Jörgen
Q1: I/O and Resource Management

a) How does synchronous I/O differ from asynchronous I/O. Which is to be preferred? (3 points)

b) Give a detailed explanation what happens when a context switch occurs in the system. Also discuss how the time for context switching can be reduced. (5 points)

c) Resource Management: Consider the following snapshot of a system:

Allocated resources (ALLOC)

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>P2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>P3</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>P4</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

Maximal required resources (MAX)

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>P2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>P3</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>P4</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

Available resources (AVAIL)

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

(i) What is the total number of available resources (i.e.: before allocation)? (1 point)

(ii) Calculate the matrix Need. (1 point)

(iii) Is the system in a safe state? If not, how many instances of the different resources have to be added in order to make sure the system is in a safe state? Further, If a request from process P3 arrives for (0,0,0,0,1), can the request be granted immediately? Show the single steps that lead to your solution. Show the single steps that lead to your conclusions! (4 points)

(6 points)

Q2: Scheduling

a) Consider a variant of round-robin called *progressive round-robin*. In this algorithm, each process has its own quantum. This starts out at 50ms and increases by 10ms each time it goes through the round-robin queue. Hence, long jobs keep getting longer and longer time slices. Discuss the advantages and disadvantages of this variant over ordinary round-robin. (3 points)

b) Consider the following set of processes (low priority number = high priority):

<table>
<thead>
<tr>
<th>Process</th>
<th>Arrival Time</th>
<th>Execution time</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>0</td>
<td>10 ms</td>
<td>3</td>
</tr>
<tr>
<td>P2</td>
<td>5</td>
<td>6 ms</td>
<td>3</td>
</tr>
<tr>
<td>P3</td>
<td>5</td>
<td>7 ms</td>
<td>2</td>
</tr>
<tr>
<td>P4</td>
<td>8</td>
<td>4 ms</td>
<td>1</td>
</tr>
<tr>
<td>P5</td>
<td>12</td>
<td>10 ms</td>
<td>4</td>
</tr>
</tbody>
</table>
For each scheduling algorithm listed below, you are asked to draw and label Gantt charts, with arrival times and execution times, clearly outlining the execution order. Further, compute the average waiting time and the turnaround time for each case (time for context switching can be ignored).

(i) Round robin scheduling with time quantum 4 ms.
(ii) Non-preemptive SJF.
(iii) Preemptive SJF
(iv) Preemptive priority scheduling (use SRTF for equal priorities)

(8 points)

Q3: Swapping

a) Three things take up time in a swapping system: (1) swapping out the old job, (2) swapping in the new job, and (3) running the new job. When the new job is ready to be swapped out again, it becomes the old job, and the cycle starts over again. If we can overlap these activities, we can speed up the system. There are three levels of overlap: (i) no overlap; (ii) you can overlap running a program with either swapping or swapping out (but not both); and (iii) you can overlap all three things.

Suppose we have a computer system where each job is 400 Kbytes long, the CPU can execute 1M instructions per second, the I/O channels can transfer 3 Mbyte per second, and a job runs exactly 600K instructions before being swapped. (K=1024; M=K*K). We only consider the CPU time spent on running the jobs to be useful. The effective computation rate is the ratio of useful computation time to total time. Hence, if the CPU was busy only half of the time, the effective computation rate would be 0.5. (Assume that processes are in the system, i.e. no cost of initialisation)

(i) For each of the three overlap cases above, compute the effective computation rate. (3 points)
(ii) Now assume that you can speed up the CPU or the I/O channels by 100 percent. For each of the three overlap cases, motivate which one would be better to speed up (you can only speed up one, not both) and how that speedup will affect the effective computation rate. If your results seem anomalous, explain why! (6 points)

(9 points)

Q4: Paging

a) There are primarily three ways of implementing a page table. Either by placing it in dedicated registers, main memory, or using associative registers. Explain the differences between these three approaches.

(3 points)

b) Virtual Memory Management: Consider the following reference string: 4, 4, 2, 3, 4, 2, 4, 2, 3, 1, 5, 5, 1, 3, 4. Determine the number of page faults that would occur using the replacement algorithms: (i) LRU using 5 frames, (ii) LFU using 3 frames, (iii) OPT using 4 frames, and (iv) FIFO using 4 frames (frames are initially empty).

(6 points)

c) What is Belady’s anomaly?

(2 points)

d) Suppose you have a two-level paging system where the first level uses 8 bits, the second level uses 14 bits, and the page offset uses 10 bits. If you have a program that uses 14 Mbytes of memory, how many page frames will it use for the program and the page tables if everything is in memory? What is the memory overhead due to paging in this case? (Assume that processes are in the system, i.e. no cost of initialization)

(3 points)
Q5: Protection

Explain the concept of a protection domain. Further, explain how an access matrix works, and how it relates to protection domain. In your description, emphasize the degree of flexibility with respect to design and how easy it is to dynamically change the configuration later. Further, indicate if the principles are following the need to know principle or not. 
(5 points)

Q6: Semaphores

a) Semaphores: Define the wait operation \texttt{wait(S)} and the wakeup operation \texttt{signal(S)}. Give an explanation and an example when and how they are used. 
(4 points)

b) Discuss three strategies for handling deadlocks. 
(3 points)