TENTAMEN TDDB63
Operativsystem och Processprogrammering
2 poäng

October 22, 2003

LINJE: DI, EI, I, Ii
TID: 08.00 – 12.00
Antal uppg.: 7
Antal poäng: 60
Hjälpmedel: Engelsk-Svensk-Engelsk ordbok, Miniräknare

Betygsgränser: U, 3, 4 och 5
Underkänd: 0 – 29 p
3: 30 – 39 p
4: 40 – 49 p
5: 50 – 60 p

Betygsgränser: U, G och VG
Underkänd: 0 – 29 p
G: 30 – 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!
Var vänlig och skriv tydligt och läsbart. Poäng kommer inte att ges till oläsbbara svar.
Svaren får vara på svenska och/eller engelska.
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!
1 Operating Systems (10 points)

a) (2 points): What are the main tasks of an operating system? (Name at least four of them.)

**Solution:**
Processor management, resource management, communication, security and privacy.

b) (1 point): What is multiprogramming and its main advantage?

**Solution:**
Multiprogramming provides efficient use of the CPU by overlapping the CPU time with various I/O requests from different users. Multiprogramming attempts to increases the CPU utilization by having all the time a running process.

c) (4 points): Name the five major activities of an operating system in regard to process management. Draw the diagram of process state (that connects those five activities) for a time-shared operating system.

**Solution:**
The five activities regarding the process management are: process creation (new state), process scheduling (choosing a process from ready queue), executing a process (running state), performing an I/O request (processes are waiting for I/O completion), and finally terminating a process (terminated state). The diagram of process state is represented in Figure 1.

![Diagram of process state](image)

**Figure 1: Diagram of process state**

d) (3 points): What is a PCB (Process Control Block)? Give at least four entries of the PCB.

**Solution:**
A PCB is a data structure that records various information for a process that it is associated with. A typical PCB entries are: a) the status of a process (running, waiting, ...); b) the register contents (when a context switch occurs, the value of registers are copied into the PCB to be restored when later the process regain the
CPU; c) Memory limit information; d) Open file list recording all opened file for
the associated process. Each process has its own unique PCB.

2 Scheduling (12 points)

a) (1 point): A CPU-scheduling algorithm determines an order for the execution
of its scheduled processes. If you suppose a non-preemptive scheduling, given
n processes to be scheduled on one processor, how many different schedules are
possible? (express the number in terms of n.) Assume that all n processes are
ready to be scheduled at the same time.

Solution:
This is the number of permutation of all process, i.e. \( n! = n \cdot (n-1) \cdot (n-2) \cdot \ldots \cdot 2 \cdot 1 \).

b) (1 point): Describe the difference between a preemptive and non-preemptive sche-
duling.

Solution:
In a preemptive scheduling a coming task (new task) can interrupt the execution of
another task to perform rescheduling. In a non-preemptive scheduling algorithm,
incoming tasks are put into ready queue without interrupting execution of other
tasks.

c) (9 points): Consider the following set of processes:

<table>
<thead>
<tr>
<th>Process</th>
<th>Arrival time</th>
<th>Burst time (ms)</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>( P_0 )</td>
<td>5</td>
<td>8</td>
<td>3</td>
</tr>
<tr>
<td>( P_1 )</td>
<td>3</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>( P_2 )</td>
<td>0</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>( P_3 )</td>
<td>6</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>( P_4 )</td>
<td>2</td>
<td>10</td>
<td>3</td>
</tr>
</tbody>
</table>

i) (4 points): Draw Gantt charts illustrating the execution for the following
scheduling algorithms:

a) non-preemptive FCFS (First Come, First Served)

Solution:

b) non-preemptive SJF (Shortest Job First)

Solution:
c) preemptive priority (smaller number represents higher priority; for two candidates with the same priority apply FCFS for the next process to be scheduled)

Solution:

<table>
<thead>
<tr>
<th>Process</th>
<th>a) FCFS</th>
<th>b) SJF</th>
<th>c) PP</th>
<th>d) RR</th>
</tr>
</thead>
<tbody>
<tr>
<td>P0</td>
<td>29 - 5 = 24</td>
<td>25 - 5 = 20</td>
<td>29 - 5 = 24</td>
<td>31 - 5 = 26</td>
</tr>
<tr>
<td>P1</td>
<td>21 - 3 = 18</td>
<td>11 - 3 = 8</td>
<td>10 - 3 = 7</td>
<td>27 - 3 = 24</td>
</tr>
<tr>
<td>P2</td>
<td>4 - 0 = 4</td>
<td>4 - 0 = 4</td>
<td>11 - 0 = 11</td>
<td>4 - 0 = 4</td>
</tr>
<tr>
<td>P3</td>
<td>35 - 6 = 29</td>
<td>17 - 6 = 11</td>
<td>35 - 6 = 29</td>
<td>33 - 6 = 27</td>
</tr>
<tr>
<td>P4</td>
<td>14 - 2 = 12</td>
<td>35 - 2 = 33</td>
<td>21 - 2 = 19</td>
<td>35 - 2 = 33</td>
</tr>
</tbody>
</table>

ii) (4 points): What is the turnaround time of each process for each of the scheduling algorithm in part i)?

Solution:

Turnaround time is the interval from the time of submission of a process to the time of completion. The turnaround time includes waiting times in ready queue, doing I/O, loading (into memory) and execution time.

The turnaround times for each algorithm are summarized in the table below.

<table>
<thead>
<tr>
<th>Process</th>
<th>a) FCFS</th>
<th>b) SJF</th>
<th>c) PP</th>
<th>d) RR</th>
</tr>
</thead>
<tbody>
<tr>
<td>P0</td>
<td>29 - 5 = 24</td>
<td>25 - 5 = 20</td>
<td>29 - 5 = 24</td>
<td>31 - 5 = 26</td>
</tr>
<tr>
<td>P1</td>
<td>21 - 3 = 18</td>
<td>11 - 3 = 8</td>
<td>10 - 3 = 7</td>
<td>27 - 3 = 24</td>
</tr>
<tr>
<td>P2</td>
<td>4 - 0 = 4</td>
<td>4 - 0 = 4</td>
<td>11 - 0 = 11</td>
<td>4 - 0 = 4</td>
</tr>
<tr>
<td>P3</td>
<td>35 - 6 = 29</td>
<td>17 - 6 = 11</td>
<td>35 - 6 = 29</td>
<td>33 - 6 = 27</td>
</tr>
<tr>
<td>P4</td>
<td>14 - 2 = 12</td>
<td>35 - 2 = 33</td>
<td>21 - 2 = 19</td>
<td>35 - 2 = 33</td>
</tr>
</tbody>
</table>

iii) (1 point): What is the average waiting time for SJF scheduling algorithm?

Solution:

Average waiting time is the amount of time for each process spent waiting in the ready queue divided by the number of processes.

$$\sum_i w(P_i) = (17 - 5) + (4 - 3) + (0 - 0) + (11 - 6) + (25 - 2) = \frac{41}{5} = 8.2ms$$

d) (1 point): The SJF scheduling algorithm usually gives better performance than the other scheduling algorithms but is difficult to implement. What is difficult in its implementation?

Solution:

It is generally impossible to know the burst time of an incoming task. The amount of CPU time is then approximate using a statistical evaluation.

3 Synchronization (6 points)

a) (1 point): What is a deadlock?
Solution:
Process can only work with several resources, and allocate them one after the other. If they mutually allocate “away” what they need they will block themselves forever.

b) (2 points): What is a semaphore? Define in pseudo code (or C code) the \textit{wait()} (P()) and \textit{signal()} (V()) operations.

Solution:
A semaphore S is an integer variable that is modified with two atomic primitives: \textit{wait} and \textit{signal}. Corresponding C code for these operation is shown below:

```c
wait(S) {
    while(S <= 0); //no-op
    S--; 
}
```

```c
signal(S) {
    S++; 
}
```

c) (2 points): What are the methods (operations) on a lock? Write the (correct) C/C++ code for the lock you have implemented during the laboratory work.

Solution:
Ask your laboratory assistant.

d) (1 point): Show that, if the \textit{wait()} and \textit{signal()} operations are not atomic, the mutual exclusion may be violated.

Solution:
Assume the wait and signal are translated in the following pseudo assembly instructions (only the integer update part is shown):

<table>
<thead>
<tr>
<th>wait()</th>
<th>signal()</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. ( r_1 = \text{counter} )</td>
<td>2. ( r_2 = \text{counter} )</td>
</tr>
<tr>
<td>2. ( r_1 = r_1 - 1 )</td>
<td>3. ( r_2 = r_2 + 1 )</td>
</tr>
<tr>
<td>3. ( \text{counter} = r_1 )</td>
<td>( \text{counter} = r_2 )</td>
</tr>
</tbody>
</table>

In a concurrently executing producer and consumer process, the concurrent execution of wait and signal, one possible interleaving is, where initially the counter value is 5:

|  \( T_0 \)  |  \( r_1 = \text{counter} \) (producer, \( r_1 = 5 \)) |
|  \( T_1 \)  |  \( r_1 = r_1 + 1 \) (producer \( r_1 = 6 \)) |
|  \( T_2 \)  |  \( r_2 = \text{counter} \) (consumer, \( r_2 = 5 \)) |
|  \( T_3 \)  |  \( r_2 = r_2 - 1 \) (consumer, \( r_2 = 4 \)) |
|  \( T_4 \)  |  \( \text{counter} = r_1 \) (producer, \( \text{counter} = 6 \)) |
|  \( T_5 \)  |  \( \text{counter} = r_2 \) (consumer, \( \text{counter} = 4 \)) |

One would expect the counter to be unchanged after wait and signal.
4 Memory Management and Virtual Memory (12 points)

a) (1 point): What is the difference between internal and external fragmentation?

**Solution:**
Processes are loaded and removed from memory. The free memory space is broken into little pieces. External fragmentation occurs when there is enough total memory to satisfy a request, but it is not contiguous.

In a paging system memory is allocated in term of virtual pages (or physical frames). If a page is 4kB large, and the process requires 9kB, the OS will allocate 3 pages for that process. However, the last page contains only 1kB of useful data. The remaining 3kB represents the internal fragmentation. Remark that external fragmentation cannot occur in a paging system.

b) (2 points): What is paging? (Gives some characteristics: pros and cons).

**Solution:**
Paging is a memory-management schema that allows to break the physical-address of a process to be non-contiguous.

Pros: Simpler allocation of memory for various amount of memory requirements; external fragmentation does not occur;
Cons: Suffer from internal fragmentation; additional computation is required to access data at a given offset (require additional hardware).

c) (1 point): What is Belady’s anomaly?

**Solution:**
In a page replacement algorithm the number of page faults does not necessary decreases if the number of frames increases. FIFO page replacement algorithm suffers from Belady’s anomaly.

d) (3 points): In 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.

i) (1 point): How many frames uses a program of 42Mb ($M = K \cdot K; K = 1024 = 2^{10}$), and the page tables if everything is in memory.

**Solution:**
A two-level paging system is depicted in Figure 2.
First, since the instruction is 32 bit long, we assume that data is stored in double words (32 bits). The offset is of 10 bits, *i.e.* the frame size has $2^{10}$ entries of 32 bits double word, 4kB.

To hold 42MB of information of our program (effective data), we need $\frac{42 \cdot 2^{20}}{4 \cdot 2^{10}} = 42 \cdot 2^8$, *i.e.* 10752 frames.

The second level page table can address $2^{14}$ (16384) frames. Thus we only need one second level page table that is of size $2^{14} \cdot 4 = 64kB$ and the first
level of size $2^8 \cdot 4 = 1kB$. Thus, the number of frames needed to store the second level table tables is: $\frac{64}{4} = 16$ and $\frac{1}{4} = 0.25$ (i.e. 1 since we can only allocate full frames) for the first level page.

Finally, the total number of frames to store the table and the program is: $10752 + 1 + 16 = 10769$.

ii) (1 point): What is the memory overhead due to the paging in case i) in terms of data accesses?

**Solution:**

To access a data you need to perform 3 memory accesses before you can get useful data.

iii) (1 point): What is the memory overhead due to the paging in case i) in terms of storage?

**Solution:**

The overhead is expressed as the ration between the total usage memory (tables + program) and useful data (program). $overhead = \frac{42 \cdot 2^{10} + 64 + 1}{42 \cdot 2^{10}} = \frac{43073}{43008} = 1.00151135$

Thus, the overhead is only of 0.15% for storing the data structures. This is quite a small overhead, but the program is rather big.

e) (4 points): Consider the following page-reference string:

$$1, 3, 2, 5, 4, 2, 1, 5, 6, 3, 2, 1, 2, 6, 3, 2, 1, 2, 3, 6$$
How many page faults would occur for the following replacement algorithms assuming one, four, and six frames?

i) LRU (Least Recently Used) replacement;

**Solution:**
1 frame: each time a page need to be replaced, since we have room only for a single page. Thus 20 page faults will occur for the reference string above. Similarly for the case with 6 frames, for this reference string there will be 6 page faults (loading each page into a frame), but not more. The same situation will occur for other algorithms. Thus we only need to provide the answers for the case of 4 frames.

We represent only the page fault in the following diagrams for 4 frames.

```
1 4 6
3 1 2
2 3
5 2
```

That is, 10 page faults occur for this reference string if LRU replacement algorithm is used.

ii) FIFO (First In First Out) replacement;

**Solution:**

```
1 4 2
3 1
2 6
5 3
```

FIFO replacement algorithm produces 9 page faults with 4 frames.

iii) Optimal replacement.

**Solution:**

```
1
3 4 3
2
5 6
```

The optimal replacement algorithm requires 7 page faults in the case of that reference string.

f) (1 point): Describe briefly the second-chance page-replacement algorithm.

**Solution:**

The second-chance page-replacement algorithm is based on a FIFO replacement algorithm. When a page has been selected for replacement the second chance algorithm inspects the reference bit. If the reference bit is set 1, the algorithm sets it to 0, and choose the next candidate in FIFO manner. If the reference bit is clear (set to 0) then the page is selected and become a victim for replacement. The second chance algorithm is implemented using a circular list.
5 File System (8 points)

a) (2 points): Explain the differences between linked allocation and indexed allocation of a file. To which abstract data types do these implementation methods correspond?

Solution:
Linked allocation sequences the blocks of a file as a list on the disk. Indexed allocation uses index arrays (in index blocks) to index data blocks.

b) (2 points): Consider the UNIX inode with the following entries:

<table>
<thead>
<tr>
<th>Entry</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>mode (4 bytes)</td>
<td></td>
</tr>
<tr>
<td>owners (8 bytes)</td>
<td></td>
</tr>
<tr>
<td>timestamps (12 bytes)</td>
<td></td>
</tr>
<tr>
<td>size block (4 bytes)</td>
<td></td>
</tr>
<tr>
<td>count (4 bytes)</td>
<td></td>
</tr>
<tr>
<td>direct blocks (4 bytes)</td>
<td></td>
</tr>
<tr>
<td>direct blocks (4 bytes)</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>single indirect (4 bytes)</td>
<td></td>
</tr>
<tr>
<td>double indirect (4 bytes)</td>
<td></td>
</tr>
<tr>
<td>triple indirect (4 bytes)</td>
<td></td>
</tr>
</tbody>
</table>

Assume that pointed indirect blocks have only direct pointers or indirection pointers, and the size of an inode is of the block size. Draw the inode representation on which you base your calculations. What is the maximum size of a file that an inode can address on a system with a block size of 4kB? (1k = 2^{10})

Solution:
To calculate how large file can be address by this inode we need to know how many blocks can be addressed. See the text book (p.429) for the graphical representation of UNIX inode.

Remark that each address requires 4 bytes (this can be observed on the direct pointers in the inode). Thus, the inode can address (with direct pointers):
\[
\frac{4 \cdot 2^{10} - (4 + 8 + 12 + 4 + 4 + 12)}{4} \text{ blocks.}
\]
The single indirection pointer points to an inode entry that has only direct pointers to data, i.e. \(\frac{4 \cdot 2^{10}}{4}\). The double indirection pointer points to an inode whose entries points to other inodes which have then pointers to data. For the double indirection chain we have \(\frac{4 \cdot 2^{10}}{4}\) pointers to data blocks. Following one more indirection, for the last, triple indirection pointer we have \(\frac{4 \cdot 2^{10}}{4}\) data pointers. Thus the maximum file size is obtained by multiplying the total number of blocks addressable with the inode by the block size:

\[
(2^{10} + 2^{10^2} + 2^{10^3} + 2^{10} - 11) \cdot 4kB = 4.0TB
\]

\((1TB = 1024 \cdot 1Gb)\)
c) (4 points): Consider a file currently consisting of 50 blocks. How many disk I/O operations are required for i) contiguous, ii) linked, and iii) indexed (single-level) allocation strategies for the following operations? (each operation starts from the same initial status)

- The block is added at the beginning.
- The block is added in the middle (after the block 25th block).
- The block is added at the end.
- The block is removed from the beginning.
- The middle block (25th) is removed.
- The last block is removed.

Assume the following: the FCB (File Control Block) in the case of indexed allocation is already in memory; in the contiguous-allocation case there is no room to grow in the beginning, but room to grow in the end; finally, the block to be added is already in memory.

You can present your answers in a matrix, where column entries represent the strategies, and rows different operations.

<table>
<thead>
<tr>
<th>Operation</th>
<th>i)</th>
<th>ii)</th>
<th>iii)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Solution:
We assume that in the case of linked allocation we have simply linked blocks, and the FCB does not have a pointer to the last block.

<table>
<thead>
<tr>
<th>Operation</th>
<th>i)</th>
<th>ii)</th>
<th>iii)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$50R + 50W + 1W = 101$</td>
<td>$1R + 1W = 2$</td>
<td>$1W = 1$</td>
</tr>
<tr>
<td>2</td>
<td>$25R + 25W + 1W = 51$</td>
<td>$25R + 1W(U) + 1W = 27$</td>
<td>$1W = 1$</td>
</tr>
<tr>
<td>3</td>
<td>$1W = 1$</td>
<td>$50R + 1W(U) + 1W = 52$</td>
<td>$1W = 1$</td>
</tr>
<tr>
<td>4</td>
<td>$49R + 49W = 98$</td>
<td>$1R = 1$</td>
<td>$0$</td>
</tr>
<tr>
<td>5</td>
<td>$25R + 25W = 50$</td>
<td>$25R + 1W(U) = 26$</td>
<td>$0$</td>
</tr>
<tr>
<td>6</td>
<td>$0$</td>
<td>$50R + 1W(U) = 51$</td>
<td>$0$</td>
</tr>
</tbody>
</table>

6 Protection and Security (6 points)

a) (2 points): Explain the concept of access matrix. Give its entries.

Solution:
An access matrix is an abstract model of protection. The rows of an access matrix
represent domains, and the columns represent objects. Each entry consists of access rights. The entry \( a_{i,j} \) defines the set of operations that a process, executing in domain \( D_i \) can invoke on object \( O_j \).

b) (2 points): What is the need to know principle? Does the access matrix follow the need to know principle?

**Solution:**
A process that follows the need to know principle should be only access only those (authorized) resources that it currently requires to complete its task. The access matrix follows the need to know principle since each object in a specific domain have a defined number of authorized resources.

c) (2 points): What are two advantages of encrypting data stored in the computer system?

**Solution:**
Encrypted data are guarded by the operating system’s protection facilities, an well as a password that is needed to decrypt them.

7 Quiz (6 points)

In this multiple choice questions correct answer gives 0.5 point. No subtraction in case of a wrong answer.

Please, cross the circles and return this paper sheet.

<table>
<thead>
<tr>
<th>Right</th>
<th>Wrong</th>
<th>Question</th>
</tr>
</thead>
<tbody>
<tr>
<td>○</td>
<td>✗</td>
<td>A child process (created by fork system call) has the same process identifier as its parent process.</td>
</tr>
<tr>
<td>○</td>
<td>✗</td>
<td>A separate process control block (PCB) is associated with each process’ state.</td>
</tr>
<tr>
<td>✗</td>
<td>○</td>
<td>With Priority Scheduling, processes are subject to starvation.</td>
</tr>
<tr>
<td>✗</td>
<td>○</td>
<td>A typical mobile phone has an operating system.</td>
</tr>
<tr>
<td>○</td>
<td>✗</td>
<td>The setuid bit indicates if a user has or has not a valid id.</td>
</tr>
<tr>
<td>✗</td>
<td>○</td>
<td>Aging is a technique to prevent starvation.</td>
</tr>
<tr>
<td>✗</td>
<td>○</td>
<td>An I/O instruction is usually privileged.</td>
</tr>
<tr>
<td>✗</td>
<td>○</td>
<td>A device controller is controlled by a device driver.</td>
</tr>
<tr>
<td>✗</td>
<td>○</td>
<td>Switching the CPU to another process requires saving the state of the old process and loading the saved state for the new process.</td>
</tr>
<tr>
<td>○</td>
<td>✗</td>
<td>A process is thrashing if it is spending more time in the garbage collection than executing.</td>
</tr>
<tr>
<td>✗</td>
<td>○</td>
<td>In a hard disk, a track is divided into sectors of (generally) equal size.</td>
</tr>
<tr>
<td>✗</td>
<td>○</td>
<td>The distinction between the monitor mode and user mode is indicated by the mode bit.</td>
</tr>
</tbody>
</table>