

CS4411 Intro. to Operating Systems Final Fall 2005

200 points – 12 pages

Name: _____

- Most of the following questions only require very short answers. Usually a few sentences would be sufficient. Please write to the point, however. If I don't understand what you are saying, I believe, for most of the cases, you don't understand the subject.
- *Justify your answer with a convincing argument.* If there is no justification when it is needed, you will receive no credit for that question even though you have provided a correct answer. *I consider a good and correct justification more important than a right answer.*
- Repeated problems are marked with asterisks, and will be graded with the *all-or-nothing* policy. If you made any mistakes, you receive no credits.
- Except for the last two problems which are programming related, all problems test if you have understood the basic concepts and can be answered quickly. **Please go through all problems once and do those you know how to do first.**
- The programming related problems are easy, if you understand the merit of channels and semaphores.

1. Threads and Synchronization

- (a) [8 points]* What is *thread cancelation*? Define the term and discuss the two commonly used versions and their differences.
- (b) [8 points] A good solution to the critical section problem must satisfy three conditions: *mutual exclusion*, *progress* and *bounded waiting*. Explain the meaning of the *progress* condition. Does starvation violate the progress condition? Why? **Note that there are two problems. For each problem, you should provide a clear answer with convincing argument. Otherwise, you will receive no credit.**

2. CPU Scheduling

- (a) [8 points]* What are *preemptive* and *non-preemptive* scheduling policies? Elaborate your answer.

- (b) [8 points] What is *priority inversion*? How could it happen? How can it be overcome? **Note that there are three questions here.**

3. Deadlock

- (a) [8 points] Consider a system consisting of four resources of the same type that are shared by three processes, each of which needs at most two resources. Show that the system is deadlock free.
- (b) [8 points] A system has four processes P_1, P_2, P_3 and P_4 , and two resource types R_1 and R_2 , each of which has two instances. Suppose further P_1 is requesting an instance of R_1 and allocated an instance of R_2 , P_2 is allocated an instance of R_1 , P_3 is requesting an instance of R_2 and allocated an instance of R_1 , and P_4 is allocated an instance of R_2 . Do the following two problems: **(1)** Draw the resource-allocation graph, and **(2)** Does this system have a deadlock? **Only providing an answer and/or vague elaboration will receive no credit.**

4. Memory Management

- (a) [8 points] A system has 8 page frames and each process has a page table of four entries in which a “X” indicates the corresponding page is not in physical memory. Suppose a snapshot of the page tables at certain moment is shown below:

Process 1	
0	X
1	5
2	1
3	X

Process 2	
0	X
1	0
2	X
3	4

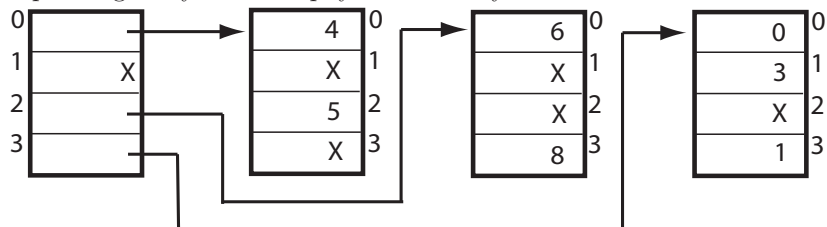
Process 3	
0	2
1	7
2	3
3	6

Convert the content of these three page tables to an inverted page table below.

Inverted Page Table

0	
1	
2	
3	
4	
5	
6	
7	

- (b) [10 points] Let the page size be 4K. Each page table has four entries. Suppose we have the following two-level page tables in which only page frame numbers are shown, where an X means the corresponding entry is not in physical memory.



Translate the following virtual addresses to the corresponding physical addresses (in decimal). Fill in the table with your answers. Please only fill the table entries with decimal values. **Expressions are not acceptable.**

Virtual Addr.	1st level page #	2nd level page #	Offset	Page frame #	Physical Addr.
46,000					
58,000					

5. Page Replacement Algorithms

Let the page reference string be 3, 1, 2, 4, 3, 2, 5, 2, 1, 3, 2, 4. The computer on which this page reference string is run has three page frames. Run this page reference string with the LRU and the Optimal algorithms. Initially, the memory is empty. Give the content *after* each reference and compute the number of page faults, miss ratio and hit ratio. **You should circle those pages that cause page faults.**

(a) [10 points] LRU algorithm:

Page	3	1	2	4	3	2	5	2	1	3	2	4
Memory												
Page faults												
Miss ratio												
Hit ratio												

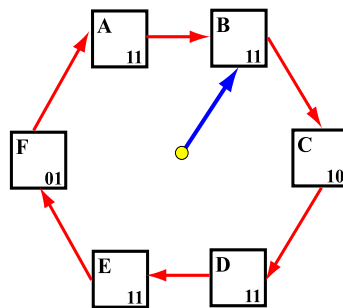
(b) [10 points] The Optimal algorithm:

Page	3	1	2	4	3	2	5	2	1	3	2	4
Memory												
Page faults												
Miss ratio												
Hit ratio												

(c) [10 points] Working set of window size $\theta = 3$:

Page	3	1	2	4	3	2	5	2	1	3	2	4
Memory												

(d) [8 points] Suppose the CLOCK page-replacement algorithm is used to approximate the LRU algorithm. Currently a system has six page frames with RM (Reference/Modified) bits as shown below (*i.e.*, the first and second bits are the reference and modified bits, respectively):



If the clock pointer is at the page frame as shown and a page fault occurs, after running the CLOCK algorithm, what are the new RM bits? Which page frame will be selected as a victim? Is a page-out required in this case? Why? Which page frame will be selected if a new page fault occurs immediately? **Note that there are five questions!**

- (e) [8 points] What is Belady’s anomaly? What is the *inclusion property*? Show that a page replacement algorithm that satisfies the inclusion property does not have Belady anomaly. **Note that there are three questions!**
- (f) [8 points] What is *thrashing*? How can it happen? How can it be overcome? **Note that there are three questions. Elaborate your answer. Otherwise, you will receive no credit.**

6. File System and Storage Management

- (a) [8 points] Suppose we have an access matrix as shown below, where ‘o’, ‘r’, ‘w’ and ‘e’ mean “owner”, “read only”, “write only” and “execution only”, respectively:

	File 1	File 2	File 3
User A	orwe	rw	rw
User B	r	orwe	e
User C	re	rw	orwe

From this access matrix, construct a *access control list* and a *capability list*. **Make sure you will indicate clearly which answer is for the access control list and which is for the capability list. Otherwise, you will receive no credit.**

(b) [16 points] Consider a file currently consisting of 10 blocks, from block 0 to block 9. Assume that the FCB is already in memory. There are a few more assumptions as shown below:

- In the contiguous allocation case, assume that there is no room to grow in the beginning, but room to grow in the end, and the directory entry has a start pointer and a length value.
- In the linked allocation, a directory entry has a start pointer and an end pointer.
- In the FAT allocation, the FAT table is in memory and a directory entry has a start pointer to the FAT table.
- In the case of indexed allocation, the index block is in memory and is large enough for further expansion.

How many disk I/O operations are required for contiguous, linked, FAT, and indexed (single-level) allocation strategies, if a new block is to be inserted between block 4 and block 5. Assume that the block information to be added is already in memory. **Note that there are four questions in this problem. Note also that you have to provide a complete elaboration of your calculation. Only providing an answer or vague elaboration will receive no credit.**

(c) [16 points] Suppose a disk has 30 cylinders numbered 0 (outer-most) to 29 (inner-most). In a sequence of I/O requests, the cylinders, in the order received, are 10, 6, 15, 18 and 9, and the disk head is currently on cylinder 10 and moves inward. Use the following disk arm scheduling algorithms to determine the cylinders visited in correct order and the total number of cylinders crossed.

<i>Algorithm</i>	<i>Cylinders Visited</i>					<i>Cylinders Crossed</i>
	1	2	3	4	5	
FCFS	10					
SSTF	10					
SCAN	10					
LOOK	10					

7. Problem Solving:

- (a) [20 points] We mentioned in class that semaphores, monitors and channels are all equivalent in the sense that any one of them can be used to implement the other two in a “shared memory” system. Suppose a system supports channels but not semaphores. This system has a set of non-specific sender and non-specific receiver channel functions as follows:

- `CHAN_t channel_name(int BUF_SIZE)`: This creates a channel with name `channel_name` and capacity `BUF_SIZE`. `BUF_SIZE` must be a non-negative integer and if `BUF_SIZE` is zero, this is a synchronous channel. Channel `channel_name` may have an infinite capacity if `BUF_SIZE` is set to a pre-defined value `INFINITY`.
- `CHAN_SEND(channel_name, (void *) addr, int size)`: This function sends a message of `size` bytes stored in a location with beginning address `addr` to channel `channel_name`.
- `CHAN_RECV(channel_name, (void *) addr, int size)`: This function receives a message `size` bytes from channel `channel_name` and stores the information to a location with beginning address `addr`.

Use the above channel primitives to design a semaphore system. More precisely, define a type `SEM_t` and functions `SIGNAL()` and `WAIT()` as follows:

- `SEM_t sem_name(int value)`: This creates a semaphore with name `sem_name` and initial value `value`.
- `WAIT(sem_name)`: This function performs a semaphore wait on semaphore `sem_name`.
- `SIGNAL(sem_name)`: This function performs a semaphore signal on semaphore `sem_name`.

You should present the definition of `SEM_t` and the detailed implementation of all functions. An elaboration of each component is required. Otherwise, you will receive low or even no credit. However, exact and correct C/C++ syntax is not required. Note that you can only use the channel primitives to implement semaphore primitives. You will receive no credit if you use any primitives other than channels and if your implementation has busy waiting, deadlock, and race conditions

Use this and the next blank page for your answer

Use this blank page for your answer

- (b) [20 points] Suppose we have a system of linear equations in the following form, where $a_{i,j}$'s are known coefficients, b_i 's are known constant terms, and x_i 's are unknowns to be solved:

$$\begin{array}{cccccccc}
 a_{11}x_1 & + & a_{12}x_2 & + & \cdots & + & a_{1,n-1}x_{n-1} & + & a_{1n}x_n & = & b_1 \\
 & & a_{22}x_2 & + & \cdots & + & a_{2,n-1}x_{n-1} & + & a_{2n}x_n & = & b_2 \\
 & & & & \vdots & & & & & & \vdots \\
 & & & & & & a_{n-1,n-1}x_{n-1} & + & a_{n-1,n}x_n & = & b_{n-1} \\
 & & & & & & & & a_{nn}x_n & = & b_n
 \end{array}$$

We may solve the x_i 's in a backward order as follows:

$$\begin{aligned}
 x_n^* &= \frac{b_n}{a_{nn}} \\
 x_{n-1}^* &= \frac{1}{a_{n-1,n-1}} (b_{n-1} - a_{n-1,n}x_n^*) \\
 &\vdots \\
 x_2^* &= \frac{1}{a_{22}} [b_2 - (a_{23}x_3^* + a_{24}x_4^* + \cdots + a_{2n}x_n^*)] \\
 x_1^* &= \frac{1}{a_{11}} [b_1 - (a_{12}x_2^* + a_{13}x_3^* + \cdots + a_{1n}x_n^*)]
 \end{aligned}$$

In other words, we can solve for x_n from the first equation, substitute the obtained value x_n^* into the next equation to solve for x_{n-1} , from x_n^* and x_{n-1}^* solve for x_{n-2} , and so on. This process continues until x_1 is computed. In general, if $x_n^*, x_{n-1}^*, \dots, x_{i+1}^*$ are the available solutions, the following computes x_i^* :

$$x_i^* = \frac{1}{a_{ii}} \left(b_i - \sum_{k=i+1}^n a_{i,k}x_k^* \right)$$

Now suppose we have n threads T_1, T_2, \dots, T_n , and thread T_i has the values of $a_{i,i}, a_{i,i+1}, \dots, a_{i,n}$ and b_i . Suppose further that there is a channel $C_{i+1,i}$ between T_{i+1} and T_i , where $1 \leq i \leq n-1$. Thread T_i receives $x_{i+1}^*, \dots, x_{n-1}^*, x_n^*$ from T_{i+1} via channel $C_{i+1,i}$, computes x_i^* , and sends $x_i^*, x_{i+1}^*, \dots, x_n^*$ to thread T_{i-1} if $i > 1$.

Create the channels and design threads T_i 's to perform the above equation solving task. You may assume that the channels send and receive messages in the FIFO order. Your solution must meet the following requirements: **(1)** maximum parallelism, **(2)** an elaboration and convincing argument for each channel parameter and each thread, **(3)** using only channels, and **(4)** no busy waiting, no race condition, and no deadlock. and **You only need to declare the channel(s) and write the thread function(s). You do not have to write a complete program. However, violating any of the above stated requirement will cause you to receive zero or very low grade.**

Use this and the next blank page for your answer

Use this blank page for your answer

Grade Report

<i>Problem</i>		<i>Possible</i>	<i>You Received</i>
1	a	8	
	b	8	
2	a	8	
	b	8	
3	a	8	
	b	8	
4	a	8	
	b	10	
5	a	10	
	b	10	
	c	10	
	d	8	
	e	8	
	f	8	
6	a	8	
	b	16	
	c	16	
7	a	20	
	b	20	
Total		200	