University of Illinois at Urbana-Champaign
Department of Computer Science

Midterm Makeup Examination 2

CS 323 Operating System Design
Spring, 2003

5:30-6:30pm Monday, Apr 23rd
DCL 2501

Print your name and ID number neatly in the space provided below; print your name at the upper right corner of every page.

Name:
Net ID:

This is a **closed book, closed notes, No calculator** exam.

Write your answers CLEARLY. The answer is wrong if the instructor and the TAs cannot read it.

Do all parts of all four problems in this booklet. This booklet should include this title page, 9 additional pages and 1 empty page. Do your work inside this booklet, using the empty pages and backs of pages if needed. Problems are of various difficulty degree, hence if you don't know the answer immediately, progress to the next problem and come back to the unsolved problem later.

Problem	Score	Grader
1		
2		
3		
4		
Total		

1. Problem (20 Points) (each question 2 points)

1. Ideally, what criteria should we use to replace pages?
   a. Choose the victims to achieve the highest memory utilization.
   b. Choose the victims to achieve the lowest page-fault rate.
   c. Choose the victims to achieve the highest disk utilization.

   Your answer: (b)

2. Which page replacement algorithm suffers from Belady's anomaly?
   a. Least Recently Used
   b. FIFO
   c. Optimal

   Your answer: (b)

3. Which of the following tells the system whether not a page needs to write to the disk when it is replaced?
   a. Base register
   b. Dirty bit
   c. Page table
   d. Limit register

   Your answer: (b)

4. In a UNIX file-system the block-size has been set to 4K. Given that the inode blocks are already allocated on disk, how many free blocks need to be found to store a file of size 64K?
   a. 16
   b. 17
   c. 64
   d. 65
   e. none of the above

   Your answer: (b)

5. 17. The CPU detects an interrupt
   a. using busy bit
   b. using interrupt handler
   c. using interrupt request line

   Your answer: (c)

6. Given a file of 100 blocks, what is the minimum number of disk I/O operations needed to insert a block in the middle of the file if linked list allocation is used (assume the block to be inserted is already in memory)?
   a. 2
   b. 52
   c. 50
   d. 151

   Your answer: (b)

7. The file-reference count is used for
   a. counting number of bytes read from the file
   b. counting number of open files
   c. counting number of links pointing to a file

   Your answer: (c)

8. Difference between a non-blocking and asynchronous read system calls is that
   a. asynchronous call returns immediately and non-blocking does not return immediately
   b. asynchronous call returns immediately and gets complete results in the future, where non-blocking call returns immediately with whatever data available,
   c. asynchronous call returns immediately with whatever data available and the non-blocking call returns later with complete results

   Your answer: (b)

9. The reason for buffering in Input/Output is
   a. to cope with speed mismatch only
   b. to cope with different data transfer sizes only
   c. to cope with context switching overhead and avoid copy semantics for application I/O.
   d. (a) and (b)
   e. (a) and (b) and (c)

   Your answer: (d)

10. The disk scheduling algorithm that may cause starvation is
   a. FCFS
   b. SSTF
   c. C-Scan
   d. Scan

   Your answer: (b)

2. Problem (25 points)

(1) (8 points) In the Unix operating system, suppose the root directory "/" inode is in memory, and everything else is not in memory, assuming that all directories fit in one disk block, how many disk accesses are needed to read the first file block of "/home/alice/cs323/homework/report.txt" ? Please describe the purpose of each disk access

1. **Load table(content) for / root directory.**
2. **Load inode for home directory.**
3. **Load table(content) for home directory.**
4. **Load inode for alice directory.**
5. **Load table(content) for alice directory.**
6. **Load inode for cs323 directory.**
7. **Load table(content) for cs323 directory.**
8. **load inode for homework directory**
9. **load table for homework directory**
10. **Load inode for report.txt.**
11. **Load first file block for report.txt.**

11 disk accesses.

(2) (5 points) Describe what is incremental backup? How can it be implemented in Unix?

Incremental backup only writes to tapes those file updates made ever since the previous backup.

In Unix, it can use the modified_time attribute in each file inode. If the modified_time is later than the last backup time, this file needs to copy to tapes. Otherwise, this file does not need to backup to tapes.

(8 points) In many Unix systems, the inodes are kept at the start of the disk. An alternative design is to allocate an inode when a file is created and put the inode at the start of the first block of the file. Discuss at least 2 advantages and 2 disadvantages of this alternative.

Advantages
-No need to reserve space to store inodes
-Minimizes disk arm movement for retrieving the first file block.
-No limit is placed on the number of i-nodes in the file system.

Disadvantages
-Increases disk arm movement for listing the contents of a directory including file names, sizes, permissions, etc.
-Waste 1 block for files that fit exactly into n blocks, whereas in the Unix system, multiple inodes are pact together.
-The Unix system can fetch multiple inodes at one disk access while this alternative cannot.

Other answers will be considered if they subjectively make sense.

(3) (4 points) Name one advantage of hard links over symbolic links and one advantage of symbolic links over hard links

Advantage of symbolic links (soft links) over hard links:
-Deleting the file which is soft linked has predictable behavior (the file is deleted).
-Soft links can span partitions and networks (hard links cannot).

Advantage of hard links over symbolic links:
-Accessing hard links are faster than accessing soft links on average.
-Hard links can create deadlocks (via loops) for system calls such as rename.
-Deleting a file which is hard linked does not leave dangling references.

Other answers will be considered if they subjectively make sense.

3. Problem (30 points)

Consider the following page reference string:

A B A D B C E B F B E A C D E

How many page faults would occur for the following replacement algorithms, assume **four page frames** in memory. All frames are initially empty.

(1) (**5 points**) The Second Chance Algorithm. Please use the following table to show steps.

Referenced Pages	**Page Fault? (Y/N)**	**Frame1**	**Frame2**	**Frame3**	**Frame4**
A	Y	A*			
B	Y	A*	B*		
A	N	A*	B*		
D	Y	A*	B*	D*	
B	N	A*	B*	D*	
C	Y	A*	B*	D*	C*
E	Y	B	D	C	E*
B	N	B*	D	C	E*
F	Y	C	E*	B*	F*
B	N	C	E*	B*	F*
E	N	C	E*	B*	F*
A	Y	E*	B*	F*	A*
C	Y	B	F	A	C
D	Y	F	A	C	D*
E	Y	A	C	D*	E*

Total 10 faults

(5 Points) The LRU replacement algorithm Please use the following table to show your steps

Referenced Pages	**Page Fault (Y/N)**	**Frame1**	**Frame2**	**Frame3**	**Frame4**	LRU stack	Reference distance
A	**Y**	**A**				\$A	Infinity
B	**Y**	**A**	**B**			\$AB	Infinity
A	**N**	**A**	**B**			\$BA	2
D	**Y**	**A**	**B**	**D**		\$BAD	Infinity
B	**N**	**A**	**B**	**D**		\$ADB	3
C	**Y**	**A**	**B**	**D**	**C**	\$ADBC	Infinity
E	**Y**	**E**	**B**	**D**	**C**	A\$DBCE	Infinity
B	**N**	**E**	**B**	**D**	**C**	A\$DCEB	3
F	**Y**	**E**	**B**	**F**	**C**	AD\$CEBF	Infinity
B	**N**	**E**	**B**	**F**	**C**	AD\$CEFB	2
E	**N**	**E**	**B**	**F**	**C**	AD\$CFBE	3
A	**Y**	**E**	**B**	**F**	**A**	DC\$FBEA	6
C	**Y**	**E**	**B**	**C**	**A**	DF\$BEAC	5
D	**Y**	**E**	**D**	**C**	**A**	FB\$EACD	6
E	**N**	**E**	**D**	**C**	**A**	FB\$ACDE	4

Total faults: 9

(2) **(10 Points).** Using distance strings, predict what is the number of page faults with the above reference string if LRU has available 2,3,5,6,7 page frames. Show steps in order to get full credits

See above table for reference distance.

C_i is the number of times the value *i* appears in the last column in the table above. F_i is the sum of all C_j for j > i (and inf > i always).

$C_1 = 0$	$F_1 = 15$
$C_2 = 2$	$\mathbf{F_s = 13}$
$C_3 = 3$	$\mathbf{F_3 = 10}$
$C_4 = 1$	$F_4 = 9$
$C_5 = 1$	$\mathbf{F_5 = 8}$
$C_6 = 2$	$\mathbf{F_6 = 6}$
$C_{inf} = 6$	$\mathbf{F_7 = 6}$

The values of F_2, F_3, F_5, F_6 and F_7 give the number of faults for 2, 3, 5, 6 and 7 page frames, respectively.

(5 points) Provide two advantages and two disadvantages of using **LARGE** page sizes in the paging systems.

Disadvantages	Advantages
More internal fragmentation.	**Smaller page table.**
For random accesses, each page miss takes longer to read the missing page into memory.	**Read a page from disk has an effect of prefetching more data into memory. So if memory are accessed sequentially, there will be fewer page misses.**

Other answers will be considered if they subjectively make sense.

(3) **(5 Points)** Does Second Chance algorithm need a hardware support? Explain in detail your answer.

Yes, it requires hardware support for the reference bit.

or

No. It is possible to be implemented in software by trapping each memory reference or instrument each memory reference.

(The following answer: "No, it can be implemented in software because we did it in a MP" is not a correct answer. This indicates that you don't understand the Nacho system. The Nacho system emulates the hardware for the OS and in Nacho there is a hardware reference bit.

4. Problem (25 points)

Let us consider a disk with 100 cylinders. Disk requests come in to the disk driver for cylinders 9, 21, 3, 38, 15, 7, 80, 96, and 50 in that order.

A seek takes **1 msec per cylinder** moved. Let us assume the disk driver has a buffer queue **holding four disk requests at a time** and the requests arrive so that the queue is always full. How much **seek time** is needed for disk scheduling policies such as

(1) (10 points) Shortest Seek Time First

(2) (15 points) Look algorithm (initially moving **upwards** from lower cylinder towards higher cylinders)

In all cases, **the arm is initially at cylinder 20**.

The following working table is provided for your convenience. Extended the table if you need more rows

Shortest Seek Time First

Buffer Queue Content				Request Being Serviced	Total Time elapsed
9	21	3	38	21	1
9	3	38	15	15	+6 =7
9	3	38	7	9	+6=13
80	3	38	7	7	+2=15
80	3	38	96	3	+4=19
80	38	96		38	+35=54
80	96			80	+42=96
96				96	+16=112

Total seek time: 112 msec

9, 21, 3, 38, 15, 7, 80, 96, and 50 in that order.

Look Algorithm

Buffer Queue Content				Request Being Serviced	Total Time elapsed
9	21	3	38	21	1
9	3	38	15	38	+17=18
9	3	15	7	15	+13=31
9	3	7	80	9	+6=37
3	7	80	96	7	+2=39
3	80	96	50	3	+4=43
80	96	50		50	+47=90
80	96			80	+30=120
96				96	+16=136

Total seek time: 136 msec