University of California, Berkeley
College of Engineering
Computer Science Division – EECS

Fall 1999 Anthony D. Joseph

Midterm Exam #2 *Solutions*

November 10, 1999
CS162 Operating Systems

Your Name:	
SID:	
Circle the letters of your CS162 login (1 per line):	a b c d e f g h i j k l m n o p q r s t u v w x y z a b c d e f g h i j k l m n o p q r s t u v w x y z
TA Name / Section:	

This is a **closed book** examination. You have two hours to answer as many questions as possible. The number in parentheses at the beginning of each question indicates the number of points given to the question; there are 100 points in all. You should read **all** of the questions before starting the exam, as some of the questions are substantially more time consuming.

Write all of your answers directly on this paper. *Make your answers as concise as possible.* If there is something in a question that you believe is open to interpretation, then please ask us about it!

Good Luck!!

Problem	Possible	Score
1	12	
2	28	
3	15	
4	12	
5	15	
6	18	
Total	**100**	

1. (12 points total) True / False and ***Why***. Circle the correct answer and provide a ***short*** reason for your answer:
   a. Doubling the block size in the UNIX 4.2 BSD file system will exactly double the maximum file size.

      True / False

      Why:
      False. The size will more than double because the size of indirect, doubly indirect, and triply indirect blocks also doubles.

      The correct answer was worth 2 points. Good reasoning was worth 2 points.

   b. A direct mapped cache can sometimes have a higher hit rate than a fully associative cache with an LRU replacement policy (on the same reference pattern).

      True / False

      Why:
      True. A direct mapped cache will do better than LRU on a pattern like ABCDEABCDE… if the cache size is one entry smaller than the total number of items in the pattern (e.g., four cache entries). LRU will miss on every access, while a direct mapped cache will only miss on the two entries that map to the same cache entry.

   c. Virtual memory address translation is useful even if the total size of virtual memory (summed over all programs) is guaranteed to be smaller than physical memory.

      True / False

      Why:
      True. It avoids the need to relocate programs when they are loaded and allows memory to be dynamically allocated. Also, provides protection and doesn't require that each program's addresses be contiguous.

2. (28 points total) File system buffer cache management.

a. (12 points) Unix systems use a 30 second write-behind policy for file system data. List the advantages and disadvantages

i) Name ***two*** advantages of this policy.

(1) Delaying writes allows multiple small writes to be batched together into a larger, more efficient large write (also improves allocation)
(2) Temporary files that are created, read, then deleted may never be written to disk.
(3) Write-behind doesn't delay writes like write-through.
(4) Periodically writing data back is better than write-back (WB could lose more data if the machine crashes).

Each answer was worth 4 points. Weak answers received 2 points.

ii) Name ***one*** disadvantage of this policy.

(1) A crash may cause data to be lost, leading to an inconsistent filesystem or loss of user data.

Several people listed the disadvantage that temporary files that live for more than 30 seconds are written to disk (versus write-back). This answer isn't really a big disadvantage (compared to the potential for filesystem corruption after a crash).

b. (14 points) Specify how you implement a write-behind policy. Assume a write-behind occurs every 30 seconds. Assume that a file is only opened by one process at a time. You do *not* have to show how you would modify the read operation.

i) List the data structures, methods, or classes you would either use or modify:

Use a global `SyncList`, `writesPending`, (or alternatively, per-process `SyncLists`) of `ToWrite` structures, containing : the ID of the open file, a buffer for holding the data to be written, and an offset into the file (where the data is to be written).
We graded solutions to (i) and (ii) by subtracting points for errors:

- *Major errors subtracted 4 points each, up to –12 total. Examples: using a thread per request (has a severe performance impact and could result in out-of-order writes), missing locks/synchronization, explicitly using interrupts to perform write-behind (can't do I/O in an interrupt handler).*
- *Minor errors cost 2 points each, up to -6 total. Example, not defining data structures, or failing to say how the kernel procedure for write-behind was executed (many solutions relied on the procedure being called by an interrupt handler). If interrupts were not explicitly used, we only counted this as a minor error.*
- *For solutions slightly over the length limit, we subtracted 2 points; for major overruns, 4 points; for extremely long solutions, 8 points.*

ii) Provide the algorithm for the policy in pseudocode form. Your solution should take less than *twenty* lines of pseudocode. No optimizations are necessary, however your solution should handle any synchronization issues. You can use any Nachos functions that you would need (if they're not standard, or are from your own project phases, explain their operation).

Create a new kernel thread for write-behind that does the following:

```
while (1) {
     WaitUntil(30 seconds);
     while (writesPending is not empty) {
          dequeue ToWrite element
          write buffer to file
          free ToWrite element
     }
}
```

Add to write procedure:

```
     create ToWrite element
     copy user data to buffer
     fill in element fields
     append to writesPending
```

(This page intentionally left blank)

3. (15 points total) One of your friends has a little extra money to spare and has decided to upgrade their HAL Junior HAL-Z4u computer. The HAL Junior has a simple paged virtual memory system (with no segments). First you measure the system to find out why it is running so slow. You discover the following values:

Measurement	Value
Pt = probability of a TLB miss	0.1
Pp = probability of a page fault when a TLB miss occurs	0.0002
Tt = time to access TLB	0
Tm = time to access memory	1 microsecond
Td = time to transfer a page to/from disk	10 milliseconds = 10000 microseconds
Pd = probability page is dirty when replaced	0.5

The TLB is refilled automatically by the hardware on a miss (the TLB is only accessed once per reference). The page tables are kept in physical memory, so looking up a page table entry incurs one memory access. Assume that the costs of the page replacement algorithm and updates to the page table are included in the Td measurement.

a. (10 points) What is the average memory access time (the time for an application program to do one memory reference) on the HAL Junior? Express your answer symbolically and compute the result to two significant digits. *Show all steps of the computation.*

```
Time = Tt                                   TLB access time
     + Tm                                   Memory read time
     + Pt * (Tm                             TLB fault (read entry)
            + Pp * (Td                      Page fault (read page)
                   + Pd * Td))              Write dirty page
```

*Time = 0 + 1us + 0.1 * (1us + 0.0002 * (10,000 + 0.5 * 10,000))*
*= 1 + 0.1 * (1 + 0.0002 * 15,000)*
*= 1 + 0.1 *4*
= 1.4 us

Major errors (-3 points): Not including Tm, missing or extra terms (for more than two terms, we only took off –1 point per term.
Minor errors (-2 points): Arithmetic errors, missing the Tt term.

If the solution did not include a symbolic expression of the time and was incorrect, we did not award partial credit.

b. (5 points) The HAL Junior price sheet looks like this:

Item	Specs	Price
Larger TLB	Reduces the probability of a TLB miss to 0.05. Assume that there is sufficient memory that the page fault rate is unaffected.	$500
Hard Disk-Drive II	Transfers a page in 8 milliseconds	$500
8 MByte more memory	Makes probability of a page fault, given a TLB miss, 0.0001	$500

Suppose your friend has $1000. Which components should they buy if they want to maximize the performance of their HAL Junior? State the reasoning behind your choices.

*TLB => 1 + 0.05 * 4 => 1.2*
*HDII => 1 + 0.1 * 3.4 => 1.34*
*Mem => 1 + 0.1 * (1 + 0.0001 * 15,000) => 1+ 0.1 * 2.5 => 1.25*
TLB + Mem => 1.12
TLB + HDII => 1.17
HDII + Mem => 1.22

All are good choices, but the best choice is to spend $500 on TLB and $500 on memory.

Each correct choice was worth 2 points. A good argument was worth one point.

No Credit – Problem X: (000000000000 points)

The news is not very good for Bill Gates. On Friday, the judge in the antitrust trial of Microsoft ruled that the software company has monopoly power in PC operating systems and that it has used that power to crush potential threats from competitors.

The Top 10 Things on Bill Gates's To-Do List

10. Change menacing cackle to more of a charming titter.

9. Stop payment on Satan's check.

8. Search JobOptions.com:
   Field = "Technology"
   Salary > $25 Billion

7. Put somebody else in charge temporarily; take the winter off and find Rosebud.

6. Push own "Start" button. At prompt, choose "Shut Down" and then "Re-start Ego."

5. Create new corporate division in charge of sending flowers and candy to Sandy O'Connor.

4. Send message to mother ship: "My job here is done."

3. Dedicate my life to finding the *real* monopolists.

2. See how quickly the government can prepare for the "11/10/99 Bug."

1. Halt global economy by taking all my money and going home.

4. (12 points total) Consider a demand paging system, where a dedicated disk is used for paging, and file system activity uses other disks. The measured utilizations of the various system components, in terms of **time**, not space, are as follows:

CPU utilization	20%
Paging disk	99.7%
Other I/O devices	5%

For each of the following changes, say what its most likely impact will be on CPU utilization: (**+**)increase, (**0**) no effect, or (**–**)decrease, and ***why***.

a. Get a larger capacity paging disk

0*: No effect. The system has insufficient physical memory, causing to thrash (saturating the paging disk in terms of I/O bandwidth).*

Each answer was worth four points, 2 points for the correct choice and 2 points for a good argument.

b. Increase the degree of multiprogramming

–: Decrease. The system has insufficient physical memory, causing to thrash (saturating the paging disk). Adding more programs will increase the amount of paging, making the problem worse.

c. Get more physical memory

+*: Increase. Providing more physical memory will reduce the amount of paging.*

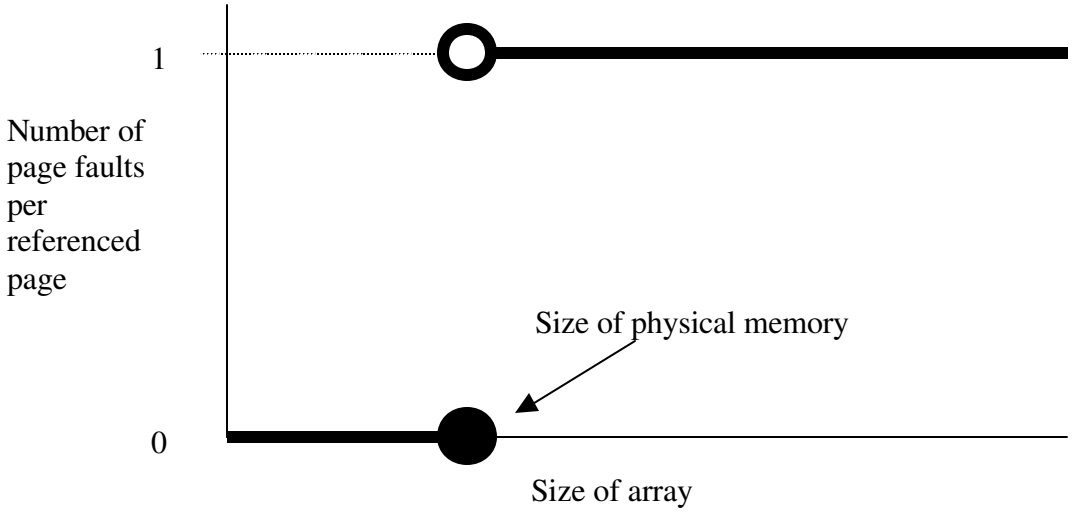
5. (15 points total) Virtual memory page replacement policies.

You are given a program that **repeatedly** scans through the elements of a very large array in virtual memory. For example, if the array is 4 pages long, then its page reference pattern is ABCDABCDABCD...

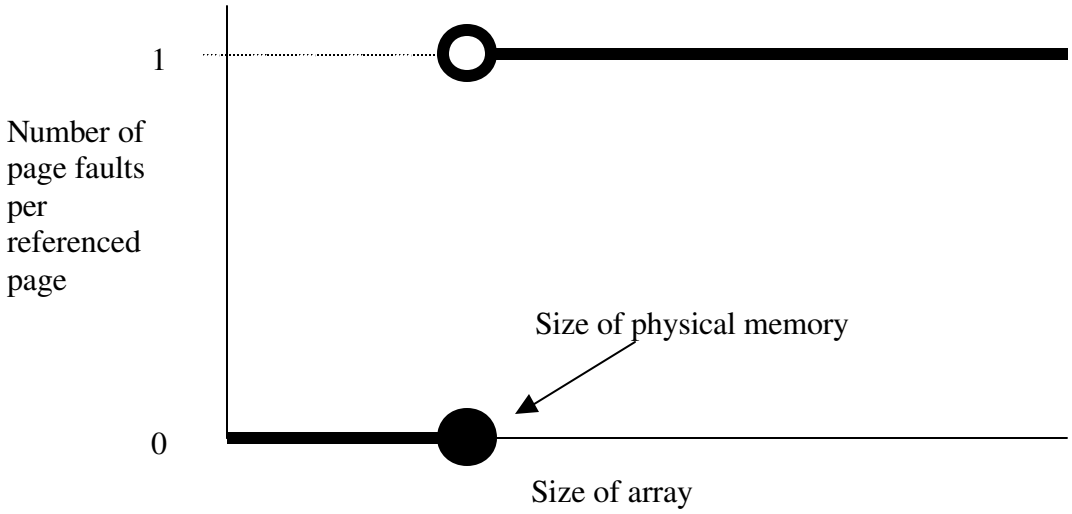
For each of the following page replacement algorithms, sketch a graph showing the paging behavior. Assume that the program has been running for several iterations through the array.

The y-axis of each graph is the number of page faults *per referenced page*, varying from 0 to 1; the x-axis is the size of the array being scanned, varying from smaller than physical memory to much larger than physical memory. ***Label any interesting points on the graph on both the x and y axes.***

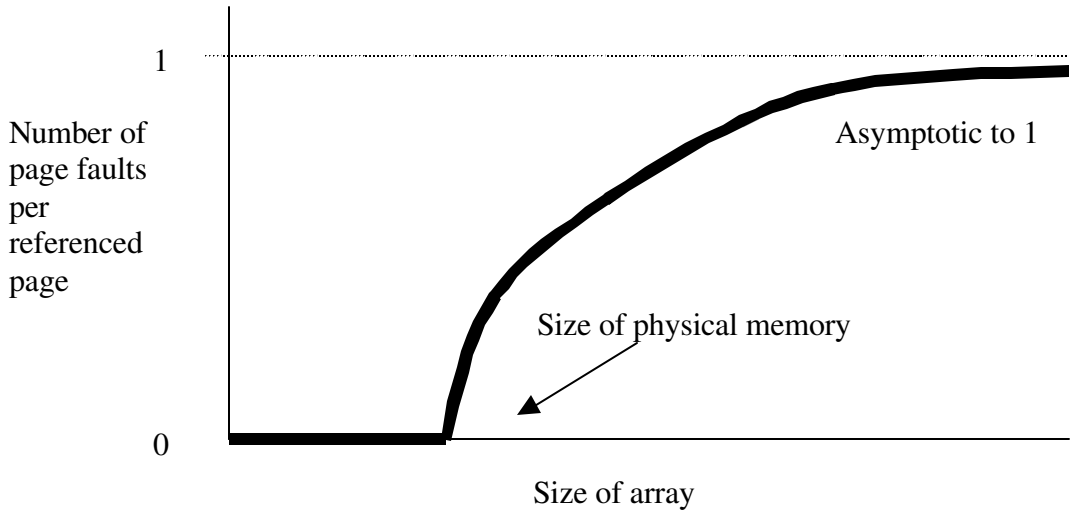
a. **FIFO**



b. **LRU**



c. **MIN**



For FIFO and LRU, we subtracted two points if there was a significant slope to transition from 0 to 1. We also subtracted two points if the graph did not have a zero fault rate before the size of physical memory was reached (we specifically state that the program has been running for some time).

For MIN, we subtracted two points if the graph was not asymptotic to 1 or if it did not have a zero fault rate before the size of physical memory was reached.

6. (18 points total) Consider the following processes, arrival times, and CPU processing requirements:

Process Name	Arrival Time	Processing Time
A	0	3
B	1	3
C	4	3
D	6	2

For each of the following scheduling algorithms, fill in the table with the process that is running on the CPU (for timeslice-based algorithms, assume a 1 unit timeslice). For RR, assume that an arriving thread is run at the beginning of its arrival time. All processes produce output *only at the end of their execution*.

Time	**FIFO**	**RR**	**SRTCF**
0	**A**	**A**	**A**
1	*A*	*B*	*A*
2	*A*	*A*	*A*
3	*B*	*B*	*B*
4	*B*	*C*	*B*
5	*B*	*A*	*B*
6	*C*	*D*	*D*
7	*C*	*B*	*D*
8	*C*	*C*	*C*
9	*D*	*D*	*C*
10	*D*	*C*	*C*
Average response time	*(3+5+5+5)/ 4 = 4.5*	*(6+7+7+4)/ 4= 6*	*(3+5+7+2)/4= 4.25*

Each column is worth 6 points: 4 for correctness of the schedule (we deducted 1/2/3 points if you made minor/intermediate/major mistakes), and 2 for the average response time (1 point was deducted for minor errors).