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

Spring 2003 Anthony D. Joseph

Midterm Exam #2 *Solutions*

April 29, 2003
CS162 Operating Systems

Your Name:	
SID AND 162 Login:	
TA:	
Discussion Section:	

General Information:
This is a **closed book and notes** 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	16	
2	25	
3	24	
4	10	
5	25	
Total	**100**	

1. (16 points total) Short answer questions:

a. (2 points) What UNIX structure is used to keep track of the sectors allocated to a given file?

i-node, file header, or index block

b. (2 points) What is the smallest addressable piece of data on a disk drive?

Disk sector or block

c. (2 points) What is a persistent, named collection of data?

File or directory

d. (2 points) What is a piece of hardware that caches virtual page → physical page mappings?

Translation lookaside buffer or associative memory

e. (8 points) Consider two processes P and Q that are communicating using mailboxes. From P → Q they use mailbox A, and from Q → P they use mailbox B. Assume both mailboxes are currently empty and the communication link between P and Q is reliable. If P now wants to determine that Q has terminated, what is the sequence of instructions that P should execute?

You may only use the following blocking instructions: `send`, `reply`, `re-send`, `receive`, `deleteMailbox`, `createMailbox`, and `setAlarm`. Note that an alarm will interrupt a blocking instruction.

```
P
send(A, msg);
setAlarm(timeout);
receive(B, msg);
if timeout then Q has terminated
```

We awarded 2 points for each part of the solution. We took off 2 points if you had an infinite loop or placed the `setAlarm` after the `receive` statement.

2. (25 points total) Disk Drives and File Systems.
   a. (6 points) Itsy Bitsy Machines Corporation develops a new disk drive that has a separate read/write head for each track and thus its disk heads don't move. Which of the following file-system features are no longer important?
   State whether the feature is useful or not useful and ***in 1 – 2 sentences*** explain why *or* why not.
   i) Cylinder groups (e.g., Unix 4.2 BSD):
   Not useful. The purpose of cylinder groups is to group data together so as to reduce disk-head movement – minimize seeks. Since the disk heads don't move, cylinder groups aren't useful.
   The answer was worth 2 points and the explanation was worth 1 point.

   ii) A bitmap free list (e.g., Unix 4.2 BSD)
   Useful. This structure allows us to allocate space for contiguous regions of a file in contiguous disk locations. This type of allocation then makes it possible to read and write such regions in a single operation instead of multiple operations (with seek and rotation gaps between them). Thus a bitmap free list is still useful with our modified disk drive.
   The answer was worth 2 points and the explanation was worth 1 point.

   b. (10 points) List the set of disk blocks that must be read into memory in order to read the file `/home/cs162/test.doc` in its entirety from a UNIX BSD 4.2 filesystem. Assume the file is 15,234 bytes long and that disk blocks are 1,024 bytes long. Assume that the directories in question all fit into a single disk block each. *Note that this is not always true in reality.*
   1. *Read in file header for root (always at fixed spot on disk).*
   2. *Read in first data block for root (`/`).*
   3. *Read in file header for* `home`.
   4. *Read in first data block for* `home`.
   5. *Read in file header for* `cs162`.
   6. *Read in first data block for* `cs162`.
   7. *Read in file header for* `test.doc`.
   8. *Read in first data block for* `test.doc`.
   9. *– 17. Read in second through 10th data blocks for* `test.doc`.
   18. *Read in indirect block pointed to by the 11th entry in* `test.doc`*'s file header.*
   19. *– 23. Read in 11th – 15th* `test.doc` *data blocks. The 15th data block is partially full. An incorrect number of blocks cost 1 point, missing all data blocks cost 3 points.*

   Missing the read of an individual component cost 1 point, missing directories cost 5 points.

Problem 2 (cont'd)

c. (9 Points) Suppose a file system can have three disk allocation strategies, contiguous, linked, and indexed. We have just read the information for a file from its parent directory. For contiguous and linked allocation, this gives the address of the first block, and for indexed allocation this gives the address of the index block. Now we want to read the 10^{th} data block into the memory. How many disk blocks (*R*) do we have to read for each of the allocation strategies? *For partial credit, explicitly list which block(s) you have to read.*

Contiguous allocation:

R =

1 block

We took off one point if you included reading a file header.
For major errors, we deducted two points.

Linked allocation:

R =

10 blocks

We took off one point if you included reading a file header.

Indexed allocation:

R =

2 blocks

We took off one point if you did not read the index block. We took off two points if you assumed the index block was already in memory.

3. (24 points total) Adding Links to a File System.
This design question asks you to consider adding links to a file system. **This is a design question; you should not write code.**

The first set of questions is about adding **hard** links to the file system.

a. (4 points) What changes would you make to the internals (directory, fileheader, free map, etc.) of the filesystem to support hard links?
You would have to add a link count to the file header. No partial credit.

b. (5 points) How do the semantics of the `Remove` system call change, and how do you implement that change to `Remove` and any other affected system calls?
`Remove` must decrement the link count (2 points) and check the on disk link count (3 points). No other system calls are affected and we deducted up to two points if you included other system calls.

c. (3 points) What new system calls, if any, must be added to the system? Give the C language-style signature of any new calls, e.g., `int Open(char *file)`. List the signature and give a one sentence definition of each argument and return value.
Add a `int Link(char *src, char *dest)` or `int Link(int inodeNumber, char *dest)` to create a new link to an existing file.

Problem 3 (cont'd).

This set of questions is about adding **soft** links to the file system. Soft links are also called symbolic links.

d. (4 points) What changes would you make to the internals (directory, fileheader, free map, etc.) of the filesystem to support soft links?

Add a new type of file to the directory or the file header (2 points), and store the soft link's pathname in a file (2 points).

e. (5 points) What existing system calls have to be modified and how to support soft links?

`Open` *needs to be changed to implement the recursive lookup implied by soft links (1 point). If the file being opened resolves to a soft link,* `Open` *must open the file pointed to by the soft link (3 points). The dereferencing operation should count the number of dereference operations to prevent an error from loops (1 point).* `Remove` *must be modified in a minor way to remove soft links correctly. If you removed the target of a link (instead of the link), we deducted 2 points. If you said that system calls use file descriptors instead of names, we deducted 2 points.*

f. (3 points) What new system calls, if any, must be added to the system? Give the C language-style signature of any new calls, e.g., `int Open(char *file)`. List the signature and give a one sentence definition of each argument and return value.

Add a `int SymLink(char *src, char *dest)` *to create a new symbolic link to an existing file. No partial credit.*

4. (10 points total) Interactions between Operating System Components.

Generally we've talked about each operating system component in isolation. This question asks you to think about ways in which they interoperate. For each pair of systems below, ***in four sentences or less***, give a specific way that they interact (or that they could interact).

Writing that the file system and I/O system interact because they both use the disk is not worth more than a point, and may be worth none. Writing that the file system and I/O system interact when they determine the mapping from logical blocks → physical blocks which impacts the size of file system structures, and the efficiency of the disk usage because larger logical blocks imply more internal fragmentation on the disk is a more complete answer.

a. (6 points) How does a demand paged, lazy-loaded virtual memory system and the file system interact?

The VM system may page executables directly from the file system, which requires coordination to make sure that users cannot write to files being used as backing store and that deletion of such files is disallowed or handled properly. The advantage is that pages swapped from the file system do not need to be copied to swap space.

Swap files can generally be put in the file system, but if so, the VM and file systems need to coordinate such use (maybe with a quota system) to prevent the swap files from running the file system out of room. Using the file system interface simplifies the VM system, but adds per-swap file overhead.

In a system that allows memory mapped files, the two systems have to interact closely to guarantee consistency between the file cache and the VM allocated to a memory mapped file.

We subtracted 2 points for overly long answers. We gave credit for answers that provided a higher level analysis of how you might change the block size or use a different allocation policy.

b. (4 points) How do the security system (e.g., file permissions) and the virtual memory system?

This one's deliberately tricky. If the OS allows memory mapping of files, the memory mapping interface has to respect file system permissions - potentially every time that the file is read or written. This is a potentially very complex interaction. Answers that discussed caching files and the need to keep permission bits correct were worth one point.

No Credit – Problem X: (000000000000 points)

The Story of Modern Math

Found on the Internet:

Mathematics in 1950:

Question: A logger sells a truckload of lumber for $100. His cost of production is 4/5 of the price. What is his profit?

Mathematics in 1960:

Question: A logger sells a truckload of lumber for $100. His cost of production is 4/5 of the price, or $80. What is his profit?

Mathematics in 1970:

Question: A logger exchanges a set "L" of lumber for a set "M" of money. The cardinality of set "M" is 100. Each element is worth one dollar. Make 100 dots representing the elements of the set "M." Set "C," the cost of production, contains 20 fewer points than set "M." Represent set "C" as a subset of set "M" and answer the following question: What is the cardinality of the set "P" of profits?

Mathematics in 1980:

Question: A logger sells a truckload of lumber for $100. His cost of production is $80 and his profit is $20. Your assignment: <u>Underline the number 20.</u>

Mathematics in 1990:

Question: By cutting down beautiful forest trees, the logger makes $20. What do you think of this way of making a living? Topic for class participation after answering the question: How did the forest's birds and squirrels feel as the logger cut down the trees? There are no wrong answers.

Mathematics in 2002:

Question: A logger sells a truckload of lumber for $100. His cost of production is $120. How does Arthur Andersen determine that the logger's profit margin is $60?

Bonus points: Explain how an Enron accountant can double this profit by selling the lumber to an out-of-state friend, repurchasing it, and selling it to the State of California.

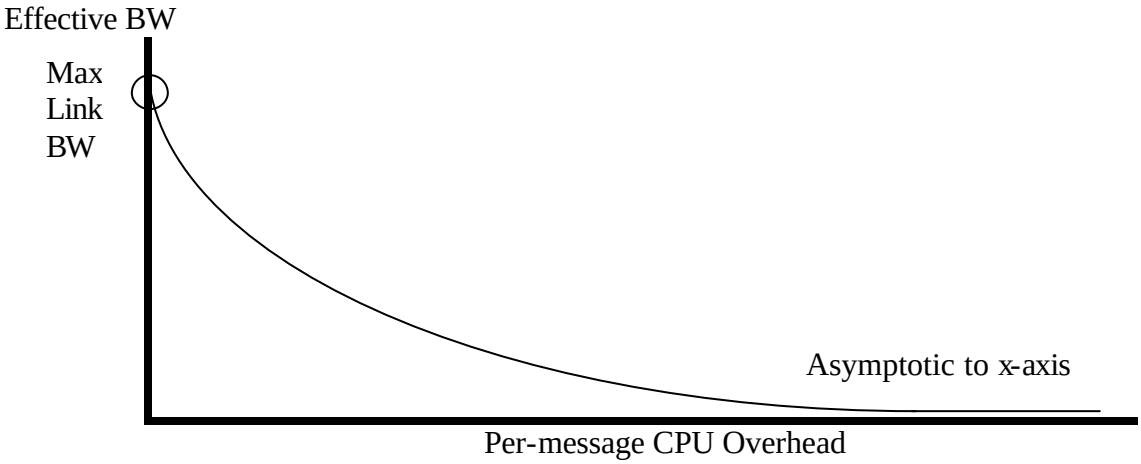
5. (25 points total) Network Performance.
   a. (9 points) Consider a TCP network connection with a current window size for unacknowledged bytes of 1,000 bytes, over a cross-country link with a one-way latency of 50 milliseconds, and a link bandwidth of 100 Mbit/second. You may assume that no packets are lost for this particular problem, and that the size of an acknowledgement is essentially 0 bytes long.

      How long does it take TCP to transmit 100,000 bytes across the link? That is, how much time elapses from when the first byte is sent by the sender to when the sender *knows* that the receiver has received the last byte?

      A TCP transmission window size of 1000 implies that the sender can send 1000 bytes before having to wait for an ACK message from the receiver that will allow it to continue sending again.
      Then the sequence of messages sent is:
      1. *1000 bytes from sender to receiver: requires 50 ms for first byte to get there and another 1000/(100 Mbps / 8 bytes/bit) secs for the rest of the 1000 bytes to get there after that.*
      2. *ACK msg from receiver to sender: requires 50 ms to get there.*
      3. *To send 100,000 bytes will require 100 round trips of this kind.*

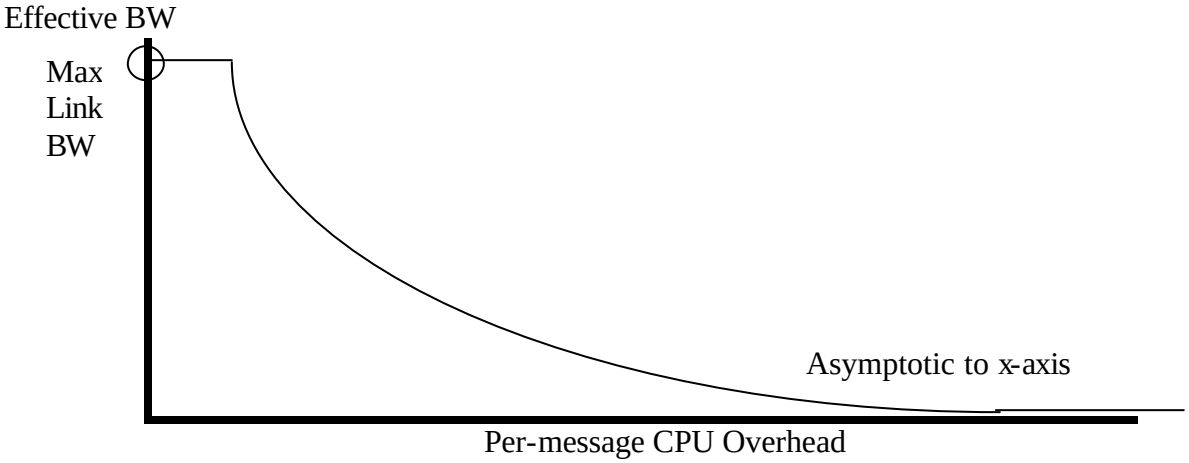
      The total time required is:
      *100 * (2 * 50 ms + 1000/(100,000,000/8)) = 100 * (100 ms + 0.08 ms)*
      = 10008 ms
      = 10.008 seconds
      We subtracted 2 points for math errors and subtracted 3 points for each missing component of the total time.

b. (6 points) Sketch a rough graph ("back of the envelope") illustrating *effective* network bandwidth as a function of per-message CPU overhead. Indicate any important relative values.



We subtracted 2 points if you didn't indicate the maximum link bandwidth. If your curve was close to the correct shape, we subtracted 2 points. If your curve was partially incorrect (e.g., intersected the x-axis), we subtracted 4 points.

Depending on the x-axis timescale, your graph could look like this:



Problem 5 (cont'd)

c. (6 points) Assume that the receiver can process incoming data at greater than 100 Mbit/s, what is the optimal window size that the receiver should advertise?

The optimal window size will keep the "pipe" full during the time it takes for the data to arrive at the receiver and the ACK to arrive back at the sender. Thus, the receiver should advertise the Bandwidth Delay Product:

(2 x 50 milliseconds x100 Mbit/s) = 10 Mbits or 1.25 Mbytes.

We subtracted 3 points for answers that used only the one way latency, and we subtracted 3 points for extra terms. For minor math errors, we subtracted 2 points. Note that saying the window size should be infinite or 100 Mbits is incorrect, as that size would yield a sending rate of 1 Gbit/s (100 Mbits sent in 0.1 seconds) leading to packets being dropped by the network or being queued by the sender's network driver and then dropped when the queue overflow).

d. (4 points) If the link is shared by N pairs of senders and receivers, does your answer for part (c) change? If so, how? If not, why?

To fairly share the link, the window size for each pair would be 1/N times the answer from (c). We subtracted 2 points if you said the answer changed but didn't indicate the amount of change.