CS 415 (Fall 2024)

Prof. Allen D. Malony

Final – December 9, 2024

NAME: ________________________________

Section	**True / False**	**Multiple Choice**	**Short Answer**	**Problems**	**Total**	**Score**
1. Memory management / VM	15	12	15	30	72	
2. File systems	15	12	15	15	57	
3. I/O systems	15	12	—	—	27	
Total	45	36	30	45	156	

Comments:

1. Congratulations!!! The hard part (studying) is over! Whoo hoo!
2. Take a deep breath. You did your best to prepare. Keep calm.
3. Write your name on the front page now.
4. You have 120 minutes to do the final exam. There are 3 sections. Do all 3 sections. Section 1 combines memory management and virtual memory. It will take longer. Sections 2 has more work to do than Section 3.
5. Concept questions (true/false, multiple choice, short answer) are 71% of the total points. DO THESE FIRST!!! If you are spending more than 5 minutes on ANY concept questions, move on!
6. Problem questions appear in Sections 1 (2 problems) and 2 (1 problem). Try to leave yourself at least 45 minutes for the problems.
7. If you have a question, raise your hand.

Exams should be challenging learning experiences! Enjoy it!!!

1 Memory Management

1.1 True/False (15)

Although segments are a non-contiguous memory management approach, external fragmentation is still possible.

_____ TRUE _____ FALSE

The translation lookaside buffer (TLB) is stored in main memory.

_____ TRUE _____ FALSE

The # entries in an inverted page table is equal to the number of frames in logical memory.

_____ TRUE _____ FALSE

The amount of virtual memory used by a process can exceed physical memory.

_____ TRUE _____ FALSE

The working set model is an approach used to help manage the number of frames a process needs in order to execute smoothly and avoid thrashing.

_____ TRUE _____ FALSE

1.2 Multiple Choice (12)

Using a hierarchical page table helps to solve what problems?

_____ Thrashing

_____ Large page table sizes

_____ Belady's anomaly

_____ Memory sharing

_____ External fragmentation

Which of the following steps are NOT involved in servicing a page fault?

_____ Flush the TLB

_____ Find a free frame

_____ Set the dirty bit to TRUE

_____ Terminate the executing process that caused the page fault

_____ Update the page table entry for the referenced page

_____ Write the referenced page to backing store

Which of the following could be beneficial to improving the efficiency of a virtual memory system.

_____ Smaller page sizes

_____ Larger TLB

_____ Larger backing store

_____ Faster backing store

_____ Page prefetching (pre-paging)

_____ Fewer active processes

1.3 Short Answer (15)

!!! CHOOSE ONLY 3 OF THE FOLLOWING 4 CONCEPT QUESTIONS TO ANSWER !!!
NOTE: Each question is worth the same. You can choose to answer more than 3 questions, but only 3 will be counted towards your score. Please mark the questions you want graded with *.

What is thrashing (from both a system-wide and per process perspective)? What should you do when the system is thrashing? What should be done when a process is thrashing?

One big advantage of paging is that it eliminates external fragmentation. However, internal fragmentation can occur, especially as the page size increases. This would argue for making page sizes smaller. Do you see anything wrong with this argument? Typical page sizes are between 512 bytes and 4K bytes. Why not make page sizes 128 bytes or even less? Name one advantage, if any, of having page sizes 4K?

What does "virtual memory" mean (try to give a concise definition)? Describe how it works? If there is only 1 process executing, do we really need virtual memory?

Explain the difference between local and global page replacement. in a virtual memory system? Why might using only 1 strategy be sub-optimal?

1.4 Problems (30)

!!! DO BOTH PROBLEMS !!!

1.4.1 Running on Air!

Now that you have survived the CS 415 final exam and received your B.S. degree in Computer Science at the University of Oregon, you take a job at a stealth Eugene AI startup, *AI Run, Incorporated*, funded by a mysterious angel investor from Nike. On your first day, you attend a briefing and learn that the company is designing a wearable computer system for real-time AI inferencing that will be put in the sole of a shoes, called the AIR-1. You are excited that this could be your career path to fame and fortune, but then you see your new boss running madly down the hallway. "Our best people have been trying to fix the poor performance on the AIR-0 prototype. You are our last hope. Prof. Malony recommends you highly. Our last hire from Oregon State fizzled under the pressure. You better not disappoint me hotshot or you're fired!!"

Keeping your cool (as Prof. Malony taught you to do), you check out the AIR-0 specifications and see that the machine is designed to use paged-based memory management. Benchmark tests report that AIR-0 is running really slow and when presented a picture of a duck, it says that it is a beaver. That is not good. The question is what can be done to improve performance and stay under very tight budget constraints. Turning to ChapGPT, two possible approaches are suggested to you:

1. Add a small, fast translation lookaside buffer (TLB) for holding recently referenced page table entries.
2. Increase the speed of the memory.

The company can not afford to do both options before the AIR-1 product launch.
The specifications further indicate that M is the the time to access memory of the AIR-0. Consulting ChatGPT again, it suggest the following improvements:

- Case 1 (TLB): Use a TLB with an access time of $\frac{1}{4} * M$.
- Case 2 (Faster memory): Use a new memory (M') that is 20% faster (i.e., $M' = 0.8 * M$ is the time to access the new memory).

You decide to run a few benchmarks on the AIR-0 prototype and determine:

- All of the pages for a process will fit in main memory. There are no pages faults. That is good.
- The AIR-0 operating system allocates additional memory frames to hold the page table and the entire page table will always fits in main memory. That is good too.
- The hit ratio h for finding a page table entry in the TLB (if used) would be 80%.

Just then the paranoid AI Run, Inc. CEO shuts down internet access and you lose access to ChatGPT. You are on your own. Good luck!

a) Write an expression for the *effective access time* (EAT) for the two cases. Case 1 is where a TLB is used. Case 2 is where the faster memory is used. Show your work. [Hint: In calculating EAT, make sure to take into account the time to find the frame number AND the time to access the memory for reading/writing data.]

b) Now suppose the (original) memory speed is 100 ns. Calculate EAT for the two cases. If the TLB hit ratio was 60%, would your answer change?

c) Based on your analysis is part a), if Case 1 (TLB, $h = 80\%$) is better, how much faster would the memory need to be to make Case 2 (faster memory) the better solution. If Case 2 (Faster memory) is better, how much does the hit ratio need to improve to make Case 1 (TLB) the better solution?

1.4.2 It Depends on Your Frame of Reference

Consider a physical memory with 4 frames. For the three page-replacement algorithms below, show the contents of the frames (i.e., the page #) for the given page reference sequence; the reference sequence is the same in all cases. Indicate if a page fault occurs for a page reference by putting an 'x' in the "Fault" row for that reference. Determine how many page faults occur in each case. The frames are initially empty. The first page fault is shown.

FIFO

Page #	6	5	3	6	2	4	3	7	6	2	4	2	1	3	5	4	6	0	2	3	3	2	4	1
Frame 0	6																							
Frame 1	-																							
Frame 2	-																							
Frame 3	-																							
Faults	x																							

page faults =

Optimal

Page #	6	5	3	6	2	4	3	7	6	2	4	2	1	3	5	4	6	0	2	3	3	2	4	1
Frame 0	6																							
Frame 1	-																							
Frame 2	-																							
Frame 3	-																							
Faults	x																							

page faults =

LRU

Page #	6	5	3	6	2	4	3	7	6	2	4	2	1	3	5	4	6	0	2	3	3	2	4	1
Frame 0	6																							
Frame 1	-																							
Frame 2	-																							
Frame 3	-																							
Faults	x																							

page faults =

2 File Systems

2.1 True/False (15)

A file system must be mounted before it can be used by an program.

_____ TRUE _____ FALSE

A file control block contains the directory of a file system.

_____ TRUE _____ FALSE

It is possible for different file systems to be mounted simultaneously and each file system can use a different block size for file allocation.

_____ TRUE _____ FALSE

The open file table for a process is allocated in the process memory.

_____ TRUE _____ FALSE

Data in a file are referenced by byte with a *logical byte address* starting at position 0 (byte address 0) in the file and going to the file size minus 1.

_____ TRUE _____ FALSE

2.2 Multiple Choice (12)

Which of the following are considered as a main part of a file system?

______ Logical file system

______ I/O buses

______ File system organization (i.e., directory)

______ Storage devices

______ Memory-mapped files

______ Vectored interrupts

What file system structures are in kernel memory:

_____ System-wide open file table

_____ Unmounted file system directory

_____ Per-process open file table

_____ File control block of unopened file

_____ User file data blocks

_____ Page cache

Which of the following could be beneficial to improving the efficiency of a file system implementation and file operation?

_____ File block buffering

_____ Memory-mapped files with page caching

_____ Limiting the number of files per volume

_____ Block prefetching

_____ Inode caching

_____ High-speed disk drives

2.3 Short Answer (15)

!!! CHOOSE ONLY 3 OF THE FOLLOWING 4 CONCEPT QUESTIONS TO ANSWER !!!
NOTE: Each question is worth the same. You can choose to answer more than 3 questions, but only 3 will be counted towards your score. Please mark the questions you want graded with *.

If an application could alert the operating system that it will access a particular file's data in a sequential manner, how might the OS exploit this information to improve performance?

Could you implement the equivalent of a multi-level directory structure with a single-level directory structure in which arbitrarily long names can be used? Explain how you can do so. What problems, if any, do you see with such an approach?

What is the purpose of the virtual file system layer in the OS? How does it interact with the physical file system layer?

Discuss the differences and merits of linked file allocation versus using a file allocation table (FAT).

2.4 File System Problems (15)

!!! THERE IS ONLY 1 PROBLEM TO ANSWER !!!

2.4.1 "Oh Grandma! What big files you have!"

The Berkeley Unix Fast File System (FFS) was created for 2 purposes: to store big files efficiently and to improve performance of file access. (FFS was the forerunner to the Unix Inode.) Figure 1 shows how FFS is organized.

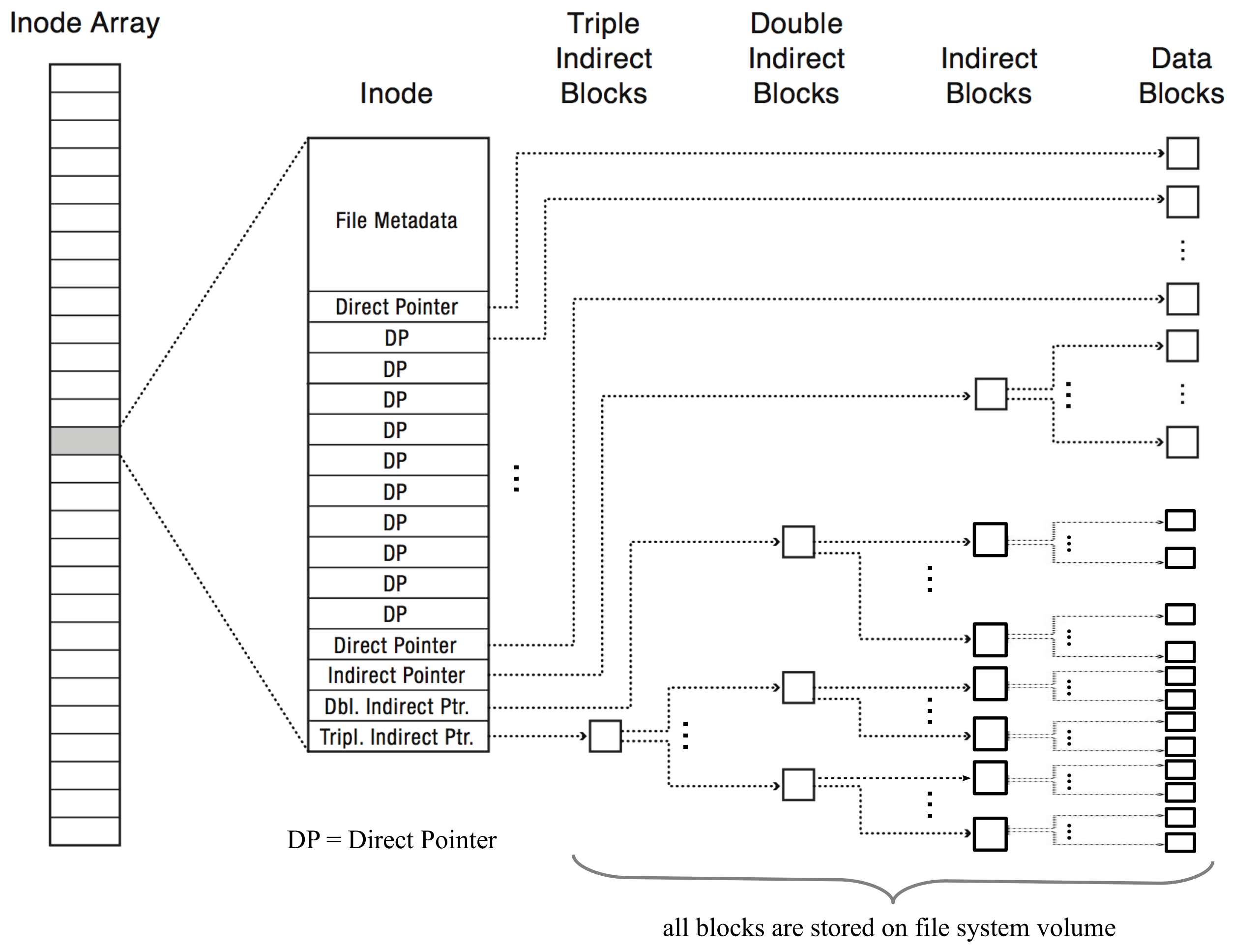

Figure 1: Berkeley FFS

a. From what you can see, explain how FFS works. Describe how a logical reference to some location in a file is processed using FFS to find the data block containing that location. Do you notice any relationship between how FFS works and a multi-level page table? (HINT: All "blocks" in the picture (shown as rectangles ☐) are storage blocks and each storage block is of the same size (in bytes), whether it is used for indirection or data. Think about what is in an "indirect" block.) (Use space below and at the top of the next page.)

b. In an inode-based file system implementation like FFS, the inode typically stores 12 direct block (DP) pointers, one 1-level indirect block pointer, one 2-level indirect block pointer, and one 3-level indirect block pointer. Suppose the file system is configured to use a block size of 2^{12} (4096) bytes and all pointers (direct, indirect, double indirect, or triple indirect) takes 4 bytes. What is the maximum file size that can be supported in the FFS file system shown above? Explain your calculation. (HINT: All of the file's data is stored in the data blocks. You need to figure out how many data blocks can be ultimately pointed to by all FFS pointers.)

c. Thinking more about the relationship of FFS to page tables, you suddenly have a brilliant idea for speeding up file access based on locality of block reference. Suppose that the inode data structure was extended by a small set of the MOST RECENTLY REFERENCED data block pointers (e.g., 5 entries). The idea is that if the data block containing the referenced file byte is found in these most recently referenced blocks, the OS could use the pointer to go directly to the block. (This is sort of like a TLB used in paging, except here it is for file blocks.) What exactly needs to be in the "file TLB" to make this scheme work? Provide an argument for or against the merits of this idea.

3 I/O Systems

3.1 True/False (15)

Direct memory access (DMA) controllers must be able to do transfers on both the memory bus AND the I/O bus.

_____ TRUE _____ FALSE

Programmed I/O moves data to/from I/O devices using DMA.

_____ TRUE _____ FALSE

The OS maintains a I/O device status table to store file metadata and directory information.

_____ TRUE _____ FALSE

Memory-mapped I/O uses I/O request signals from I/O instructions executed in the CPU together with specific port addresses to identify devices.

_____ TRUE _____ FALSE

Polling could potentially be a more efficient way for the OS to interface with an I/O device (versus interrupts) depending on the device type.

_____ TRUE _____ FALSE

3.2 Multiple Choice (12)

Which of the following actions take place in the OS when a process makes an I/O request (assume interrupts are used by a device)?

_____ OS checks to see if the request can be handled immediately.
_____ If the I/O device must be contacted, OS blocks the process.
_____ The device driver issues commands to device controller.
_____ The device driver polls the device, transfers data byte by byte, and generates an interrupt at the end.
_____ If the process is reading data, the OS wakes up the processes after the "input" transfer is done and then the process reads from the device controller's buffer.

Which of the following statements are true about synchronous and asynchronous I/O?

_____ Asynchronous I/O is always more efficient than synchronous I/O?
_____ If a process requires an I/O operation to be complete before doing anything else, synchronous I/O should be used?
_____ Asynchronous I/O blocks the process until an interrupt occurs.
_____ A process must check for completion of asynchronous I/O.
_____ None of the above.

For the following devices, indicate which could benefit from using DMA.

_____ Mouse
_____ 10 gigbit network interface
_____ Keyboard
_____ Video camera
_____ Backing store
_____ Bluetooth headphones