CS 111 Midterm exam

NGUYEN; BAOLINH KIANNA

TOTAL POINTS

96 / 100

QUESTION 1

1 Dirty bits 10 / 10

✓ **- 0 pts** **Correct**

- 10 pts No answer

- 5 pts Incorrect in explaining why dirty bit improves performance

- 5 pts Incorrect in explaining how dirty bit improves performance

- 2 pts Did not link performance increase to less disk I/O

QUESTION 2

2 ABI and system call interface 9 / 10

- 0 pts The subset relationship is clearly stated in the answer.

- 10 pts No answer

- 3 pts Wrote down some sentences related to question, but didn't clearly mention system call interface is a subset of ABI.

✓ **- 1 pts** **More close to the answer but still missing clearly mentioning the subset relationship.**

QUESTION 3

3 Shared libraries 10 / 10

✓ **- 0 pts** **Correct**

- 10 pts No answer

- 4 pts Missing details: multiple processes accessing the shared global data at the same time would be a problem.

QUESTION 4

4 System calls and trap instructions 10 / 10

✓ **- 0 pts** **Correct**

- 10 pts No answer

- 2 pts Did not mention transition of processor from unprivileged mode to privileged mode

- 2 pts Did not mention OS runs appropriate code for the system call

- 2 pts Did not explain usage of trap handler

- 2 pts Did not mention OS will determine what trap was caused by trap instruction

- 2 pts Did not mention associated parameters preset by user application are saved

QUESTION 5

5 Working sets and page stealing 10 / 10

✓ **- 0 pts** **Correct**

- 10 pts No answer

- 5 pts Incorrect description of working set.

- 5 pts Incorrect description of page stealing algorithms.

- 3 pts insufficient description of page stealing algorithms.

- 3 pts Insufficient description of working sets.

- 2 pts Important element is that page stealing takes pages from processes whose working sets are too large and gives them to processes whose working sets are too small.

- 3 pts Goal of a working set is not to maximize the number of pages in memory, but to figure out the right number to have there.

- 2 pts Working set has nothing to do with TLB.

- 1 pts Just because a working set is large doesn't mean it isn't using its pages.

- 2 pts working set is not really about preventing thrashing, since that can occur even with properly implemented working sets.

- 3 pts Important to note that working sets are associated with processes and are controlled by their behavior.

- 3 pts Page stealing is used to build proper

working sets, not vice versa.

- 2 pts Processes don't voluntarily release page frames. That's why it's called stealing.

- 2 pts Page table usually bigger than the working set.

QUESTION 6

6 Scheduling algorithm metrics 10 / 10

✓ **- 0 pts** **Correct**

- 10 pts No answer

- 2 pts Maximizing jobs completed does not necessarily translate to maximum throughput, by most definitions.

- 8 pts Fairness not guaranteed by non-preemptive scheduling. Starvation not necessarily avoided, either.

- 1 pts SJF has nothing to do with number of pages.

- 5 pts Insufficient explanation of why metric is maximized.

- 3 pts Not for all non-preemptive algorithms. Turnaround time won't be optimized for non-preemptive FIFO, for example. Question asked about non-preemptive algorithms in general, not just one example of such an algorithm.

- 5 pts Turnaround time is not necessarily optimized. It's time of job arrival till time of job completion. With non-preemptive scheduling, one long-running job can kill the turnaround time of many other jobs.

- 0 pts As stated in the test instructions, nothing on the back of the page is graded.

- 5 pts Won't necessarily optimize time to completion. A long running job will not be interrupted, causing other short jobs to incur long times to completion. If you interrupted the long job for the short ones, average time to completion would improve.

- 10 pts Did not specify a metric.

- 4 pts "Minimizing context switches" isn't a performance metric, though doing so is likely to improve some metrics.

- 3 pts Insufficient explanation of why metric is maximized.

- 10 pts Response time may not be optimized with non-preemptive scheduling, since one long-running job can kill the response time of many other jobs.

- 2 pts Won't also optimize average time to completion, since long-running job can kill time to completion of many other jobs.

- 4 pts Throughput is typically defined as the amount of work produced by a system, not the number of jobs it completes. By the latter definition, non-preemptive scheduling doesn't optimize the metric, since you could finish many short jobs in the time it takes to finish one long job.

- 5 pts Not clear exactly what you mean by "process speed".

- 4 pts That's not the definition of mean response time. It's the average time to get some response from the system, not time to completion.

- 2 pts Not very clear description of chosen metric.

- 4 pts Non-preemptive scheduling is not likely to optimize number of deadlines met, since newly arrived jobs with short deadlines can't preempt a running job with a long deadline.

- 0 pts Not the usual definition of time to completion, but correct as described.

- 10 pts Round robin is not a non-preemptive algorithm

- 10 pts "operations/second->output" makes no sense. Output is not a metric.

- 6 pts Your assumptions are rarely true, and if not true, average time to completion is not optimized, by most definitions of that metric.

- 8 pts Incorrect description of throughput. Throughput is not the same as turnaround time, and turnaround time is not necessarily optimized by non-preemptive scheduling.

- 3 pts Metric you're looking for is throughput, not "turnout" or turnaround time.

- 5 pts Fairness not guaranteed (by any definition) for all types of non-preemptive scheduling, such as non-preemptive shortest job first.

- 2 pts You're thinking of throughput, not total

execution time.

- **10 pts** Round robin is not a metric, it's a scheduling algorithm, and not even a preemptive one.

QUESTION 7

7 Worst fit and fragmentation 10 / 10

✓ **- 0 pts Correct**

- **3 pts** Worst fit algorithms fit allocation requests into the largest free chunk of memory available, assuming no perfect fit is available. The remainder of that chunk will be as large as possible, meaning it will be well suited to match later requests.
- **3 pts** A best fit algorithm will choose the free chunk closest in size to the requested allocation, which implies that the leftover free memory returned to the free list is likely to be a small chunk, poorly suited to matching future requests.
- **4 pts** The definition of external fragmentation is scattering small, useless chunks of free memory throughout the free list, so best fit is more likely to cause external fragmentation than worst fit.
- **10 pts** wrong answer
- **10 pts** No answer

QUESTION 8

8 Page tables for fork vs. shared memory IPC 10 / 10

✓ **- 0 pts Correct**

- **10 pts** No answer
- **5 pts** Major difference is fork results in copy-on-write, while shared memory IPC doesn't.
- **1 pts** new process has its own page table, but its contents are the same.
- **3 pts** No discussion of fork page table issues.
- **1 pts** Stack isn't shared in shared memory IPC.
- **2 pts** Fork need not be followed by exec, leading to COW issues.
- **1 pts** IPC shared memory almost always read/write, at least by one of the processes.
- **2 pts** Data segment also likely to change after fork.
- **0 pts** Not well worded, but I think you have the right idea.
- **10 pts** So what's the difference?
- **10 pts** What is the difference in their page table behavior?
- **3 pts** Difference won't be in the TLB.
- **2 pts** Copy-on-write issue.
- **5 pts** Not a thread issue. Copy on write is the main relevant mechanism.
- **4 pts** Shared memory doesn't share page tables. Just entries in different page tables point to the same page frame.
- **3 pts** IPC is not about libraries, it's about data.

QUESTION 9

9 Condition variables 7 / 10

- **0 pts** Correct
- **10 pts** No answer
- **4 pts** A condition variable is used to determine if some specific pre-defined condition has or has not occurred.
- **3 pts** If the condition does occur, one or more of the blocked processes will be unblocked and permitted to run.

✓ **- 3 pts The condition variable allows a process to wait for a specific condition without requiring the process to use a busy loop to check for the condition's occurrence.**

- **10 pts** wrong answer

QUESTION 10

10 TLB misses 10 / 10

✓ **- 0 pts Correct**

- **10 pts** No answer
- **3 pts** Missing case of invalid entry.
- **4 pts** Missing case of valid entry on disk.
- **3 pts** Missing case of valid entry in RAM.
- **1 pts** Page fault is on non-present, not invalid.
- **3 pts** Case with page on disk is present bit not set. Invalid bit is different.
- **4 pts** Different cases for valid page on disk and valid page in RAM.
- **2 pts** What happens for an invalid entry?

- **1 pts** TLB is a cache of page table entries, not pages.
- **1 pts** Per test instructions, text on the back of the page is not graded.
- **2 pts** First step is to consult in-RAM page table.
- **1 pts** Disk isn't searched, since page table contains disk location of non-present pages.
- **2 pts** More details on not present case.
- **3 pts** Spacial locality does not play into TLB miss handling.
- **3 pts** Memory won't be searched. Either the page is present, not present, or not valid. Present pages have their PTE loaded, not present pages are fetched from disk, invalid pages cause an exception.
- **2 pts** Page table entry itself will indicate if page is on disk. No need to invoke clock algorithm.
- **1 pts** Dirty bit doesn't indicate whether a page is in memory or not. Present bit does. Dirty bit indicates if an in-memory page has been written.
- **1 pts** Invalid case is not that the page cannot be found, but that its PTE is marked invalid.
- **1 pts** How is it determined if a segmentation fault should occur?

Midterm Exam
CS 111, Principles of Operating Systems
Winter 2018

Name: Baolinh Nguyen

Student ID Number: [redacted]

This is a closed book, closed note test. Answer all questions.

Each question should be answered in 2-5 sentences. DO NOT simply write everything you remember about the topic of the question. Answer the question that was asked. Extraneous information not related to the answer to the question will not improve your grade and may make it difficult to determine if the pertinent part of your answer is correct. Confine your answers to the space directly below each question. Only text in this space will be graded. No question requires a longer answer than the space provided.

1. In a virtual memory system, why is it beneficial to have a dirty bit associated with a page?

It is beneficial to have a dirty bit to indicate whether a page has been modified in RAM. This helps in the event that you have to choose a page to page back out to disk. Paging out to disk requires that if a page has been modified, the disk copy must also reflect these modifications. Using a dirty bit helps indicate whether a page has been modified. Thus, when choosing a page to swap out, the system can look for pages whose dirty bits have not been set - these pages have not been modified - and swap out those pages (the "clean" pages) more efficiently than if the system were to choose the dirty pages - which means they would have to write back to disk, which is expensive. Moreover, these dirty bits can be used by the system in laundering the pages in the background to make for more efficient page replacement later by creating more clean pages.

2. What is the relationship between a system's Application Binary Interface and its system call interface?

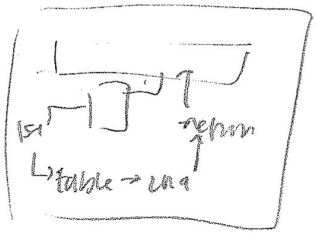
A system's Application Binary Interface binds an API (Application Programming Interfaces) to the particular Instruction Set Architecture. The system call interface consists of hardware traps that tell the OS a process is requesting a privileged service. Both of these interact with the hardware. A system's ABI configures APIs to the specific hardware instruction set architecture. The system call interface reveals specific instructions that require OS permission because they interact directly with the hardware.

3. Why can't shared libraries include global data?

A shared library cannot include global data because the library is used by multiple programs and is placed in a specific location of memory when the first program requests services from it. It cannot include global data because they are not linkage edited with the program. Thus, external references are not resolved on a program-by-program basis, including static and global data. Areas like the BSS are not yet initialized at the time when the shared library is placed into memory. Moreover, the library may be in memory before some programs are even beginning and thus, the library cannot have any knowledge of these programs. Furthermore, these libraries can only include read-only code segments. Because they are shared they cannot be written to and thus, cannot include global data.

4. Describe how a trap instruction is used to implement a system call in a typical operating system.

A trap instruction is used by a program to signal to the OS that it wants the OS to execute privileged instructions for it. The trap occurs in the hardware & causes the first level trap handler to activate, saving the process' information (registers, stack, program counter) onto the kernel stack. Then, the first level handler selects the appropriate entry from the trap table, indexed by system call numbers, and uses the entries in the trap table to figure out which location the specific instructions to execute are. The specific system call implementation is run by the second level trap handler. Once complete, control goes to the return-from-trap handler, which pops off the process' context information from the kernel stack, modifies the CPSR and returns control to the program, executing again from the instruction after the trap.



5. What is the relationship between the concept of working sets and page stealing algorithms?

The working set of a process is the number of pages it is given to run the process. The working set, thus, is the amount of virtual memory pages given to a process. When a process needs more pages in its working set, because there usually are not many free pages lying around, it must then steal a page from another process. Page stealing algorithms help dynamically manage the working set of processes. Processes that do not use as many pages will thus get their pages stolen and leave them with a smaller working set. Processes that require a lot of pages will, as a result, steal more pages and thus, increase the size of their working set. If working sets are too small, more page can get stolen. Page stealing algorithms manage the pages that are chosen to be stolen. Usually these algorithms implement a LRU (least recently used) page stealing policy. However, if the memory of the system cannot support all of the working pages of the processes, thrashing can occur, meaning pages are stolen unnecessarily.

6. Name a performance metric that is likely to be maximizable using non-preemptive scheduling. Why is this form of scheduling useful to maximize this metric?

Throughput can easily be maximizable using a non-preemptive scheduling discipline. This is useful because the non-preemptive scheduling discipline ensures that jobs run to completion as they will not preemptively be interrupted by the OS to run another job. Jobs that are ensured to be completed - raising the throughput of the system but in turn, lowering the fairness. For instance, a job that enters in a non-preemptive First-time, First served scheduling discipline will enter a queue of jobs. The jobs will be computed in the order the enter the queue. Because the OS cannot interrupt the currently running job, all jobs will run to completion unless they yield or block for I/O, thus, the computation or amount of work done per job is maximized.

7. Why does a worst fit algorithm for managing a free list handle external fragmentation better than a best fit algorithm?

A worst fit algorithm handles external fragmentation better than a best fit algorithm because it delays the fragmentation. It does so by choosing the block that fits the worst for a requested memory size. As a result, the list fragments into LARGE chunks unlike best fit, which tries to choose blocks that best fit a request. Best fit causes quick fragmentation of memory as many small, unusable chunks will appear as a result. Worst fit does better than best-fit because it delays this fragmentation by creating large chunks, that still are usable. However, eventually worst fit will perform, and thus fragment, similarly to best fit after some time.

8. Both shared memory IPC and the processes' data areas after a Linux fork() (copy on write) operation would require the page tables of two processes to point to the same physical page frames. What would be different about the two cases (other than being caused by IPC vs. forking)?

Shared memory IPC and forking both result in two processes with pointers to the same physical page frames. However, the difference is apparent after the process attempts to write to the page frame. In IPC, a shared memory is intended to allow two processes to communicate quickly and efficiently. Thus, when a process writes, the other process can immediately see the results. However, when two processes share the same page frame due to a fork, this is because the OS is trying to optimize and is using a copy-on-write policy. This means that the memory that the child process inherits from the parent is shared until either process tries to write to it. Unlike in IPC where a write to the shared memory is seen by both processes, a write to the shared memory after a fork() initiates a copy, as per the copy on write policy. The process attempting to write will get a copy of the memory to write on, rather than writing on the original shared memory itself. This ensures that neither process sees the write and also prevents the expensive copy in a case where neither process writes.

9. What is the purpose of a condition variable?

A condition variable is used in situations where a process/thread is waiting for some condition to be fulfilled or some process to be completed before it can continue execution. The condition variable is waited upon by these threads. Once the condition is fulfilled, the thread completing the condition being waited upon can signal to the threads waiting, or post, in order to signal to the threads that they can now continue execution. Otherwise, the other threads must wait. These threads wait in a queue for other threads to signal to their condition variables indicating that the condition they have been waiting on has been fulfilled and allowing them to continue execution.

bring pages in when you ask for them.

10. In a system using demand paging, what operations are required when a TLB miss occurs? What are the possible outcomes of those operations?

- page not in TLB
- check page table
- either invalid or valid
- either in RAM
- or in disk

When a TLB miss occurs, this means that the page requested is not in the TLB. Thus, since the TLB is a cache for memory, the OS must now consult the page table to see whether the page can be accessed. In the page table there are two possibilities, the page can either be valid or invalid. This is determined by the valid bit.

In the case that the page is invalid, the program is terminated because it had tried to access an invalid address and thus, has caused a segmentation fault.

In the case that the page is valid, then the OS must consult the present bit to see if the page is present in memory or if the page is currently in disk. If the page is present in memory, then the page is fetched. The program is blocked and the page is brought into the TLB. The program must try the instruction again and then it will get a TLB hit. In the case that the present bit indicates that the page is in disk, the OS can then consult the area of the page table where now, instead of the physical page frame number, the disk address is stored. Because the page being accessed is on disk, this is a page fault. The page must then be brought in from disk, and the page table updated to indicate that the page is in RAM now and is valid. The program must fault again to run the page