CS 111 Midterm exam

VIVEKANANDH; ASHWIN

TOTAL POINTS

87 / 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 7 / 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 8 / 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.

- 0 pts Click here to replace this description.

QUESTION 6

6 Scheduling algorithm metrics 5 / 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.
- **0 pts** Click here to replace this description.

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 and larger 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?

- 2 pts Not present pages are in the page table. They're just marked as "not present."

Midterm Exam
CS 111, Principles of Operating Systems
Winter 2018

Name: Ashwin Vivekanandh

Student ID Number: 204705339

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.

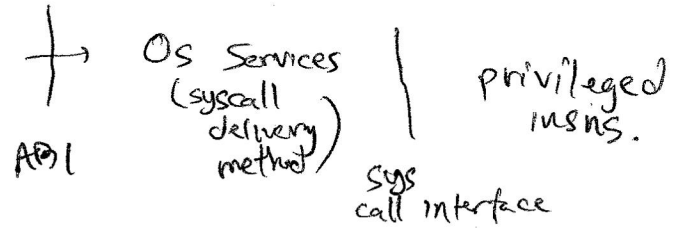
API
services
libraries

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

when a page is brought in from disk into memory, it is an identical cop of the page in disk. This ~~is~~ page is in a 'clean' state. When a proce writes to the page or modifies data in the page, it is different than the copy on disk, and is in a 'dirty' state. The dirty bit is beneficial as it helps us implement page swapping more efficiently ~~because it is no~~. If the dirty bit is set, ~~the~~ a ~~the swap~~ and the page must be copied back to disk, the disk has to be written to, to capture the page's current data. However if the dirty bit is off, we can ~~copy~~ remove the page from RAM knowing the copy in disk is correct. Page swapping will prefer to swap clean pages since it is more efficient, and thus the dirty bit is beneficial in the formulation of swapping algorithm

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

The ABI is an interface that sits between a user application and the OS services. The system call interface ~~sits between~~ is an interface that allows a service to access certain privileged instructions / functions in the OS kernel. The ABI specifies the binding of a program to a hardware architecture to get accesses to ~~the~~ services. One delivery method of an OS service is through system calls, which will use the system call interface to access the kernel and perform the privileged instructions. Thus, the two are directly related.



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

→A shared library is a collection of object modules that is available to all processes. ~~These libraries are loaded at~~

→Global data is not included in a shared library since shared libraries are implementation independent, and written, compiled, and loaded without the knowledge of the processes that are going to use the library. ~~The shar.~~

→ Shared libraries will not be able to access the address space containing global data, since accessing global data is only allowed if the process has the permissions (given by the protection bit), ~~and this would pose a security risk since writing to global data~~. If it was included, any process could access global data, ~~is~~ even without the necessary ~~read/write~~ permissions

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

(*) this level is first handler

when ~~a~~ user code issues a system call, the OS will perform a trap instruction that will access the trap table, managed by the hardware. This trap table is initialized at boot time, ~~in trap~~ The system call number, which identifies the system call, will act as an index into the trap table, ~~that~~ (*) The entry at that index ~~is~~ stores a program counter and processor status word, which are loaded into a kernel stack unique to the calling process, along with registers, program counter, etc. Then, the trap table calls a ~~the~~ second level - trap handler to actually perform the privileged instruction in the kernel using the data in the kernel stack and return to the first level handler, which will ~~return to the instruction just resume~~ return-from-trap and resume normal execution.

this description...
page stealing is when a process unloads a page if it is not using it and another process (in a queue of some sort) will take that page for its own use

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

The working set is the optimal number of pages required in memory for a process to run efficiently. Page stealing occurs when one process takes an unused page from another process for its own use. In a typical working set - page replacement algorithm, the process will let go of a page when it reaches some 'time of not being referenced by the owning process' and is deemed unnecessary for the process. This process is then available on some queue for another process to ~~steal~~ perform page stealing, and bring that page into its working set.

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?

In non-preemptive scheduling, a scheduled job is always run to completion. One metric that is maximized, (under the right conditions) is throughput or number of jobs completed per unit time. This is because when a job is scheduled there is a ~~level of confidence~~ certainty that the job will run to completion in the fastest time possible. When the job mix is relatively uniform, preemption is not useful, and non-preemptive scheduling will maximize throughput, compromising response time.

→ As opposed to preemption where multiple jobs are done in pieces, and it takes longer for the first job to complete, decreasing throughput.

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

The worst fit algorithm will search for the largest chunk in the free list that is bigger than the request, and carve it
The best fit algorithm will search for the smallest chunk that is bigger than what is requested and carve the amount requested

~~Worst fit handles external fragmentation better, as it delays it from happening since the~~

in worst fit, the leftover chunk from the carving is bigger than the leftover chunk in best fit, ~~which delays the~~ This delays the formation of smaller fragments which become unusable. In short, best fit is likely to cause external fragments faster than worst fit since ~~it will cause~~ the splitting naturally results in smaller leftover chunks, which accumulate over time.

8. Both shared memory IPC and the processes' data areas after a Linux fork() 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)?

In a Linux fork(), the two processes initially point to the same page frame until either of them writes to the shared data area, in which case the OS will copy-on-write, and duplicate the data area for the child process and allocate a new page frame. Now the two processes point to different pages.

In shared memory IPC, it is assumed that both processes want to write to the same data area to communicate with each other, so copy-on-write is not a valid policy. This will lead to synchronization issues if the two processes are trying to write to the same resource at the same time, so we need auxiliary locking of blocking of preemption/ atomic instructions to access and write to the resource so that we don't run into race conditions or synchronization errors

outcomes
- valid bit is 1
- valid bit is 0
-

9. What is the purpose of a condition variable?

The condition variable is a synchronization object that will wake up the requestor when some condition is met or some event occurs. Condition variables are used in waiting lists for locks, where all the threads waiting for a lock are asleep, and will wake up when the condition variable posts to the waiting list. This indicates the lock is free and the first thread in the waiting list queue will wake up, obtain the lock, and have access to the critical section. Condition variables can also be used for processes that are waiting for a file / shared resource to be open. In short, condition variables facilitate synchronization

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

PTE = page table entry

Demand paging is when the process requests a page when it wants to use it at that very moment. When a TLB miss occurs, the process will consult the page table and search for the page table entry with that specific virtual page number. There are a few outcomes:

TLB miss = translation not in TLB

1) Valid bit of PTE = 1: Then the OS will consult the physical address pointed to by the PTE and If it is present in memory, use the data as expected. If the present bit is off, the memory is in disk and will need to be swapped in to RAM before the process can use it, which will cause the process to run slower. In both scenarios, the page is confirmed to be in physical memory after the operation and the translation will be inserted into the TLB, in hopes the the program uses locality

2) Valid bit of PTE = 0: This means the process is attempting to access a page outside its address space / it doesn't have permissions to. This will result in an exception that traps to the OS and the OS will deal with the exception accordingly (Segfault, etc) In this case, the PTE will not be inserted into the TLB since the process has exited most probably exited, and it would not be useful to future processes (maybe)