CS 111 Midterm exam

AGI; KUBILAY YUCEL

TOTAL POINTS

93 / 100

QUESTION 1

1 Dirty bits **8 / 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
- **+ 3 Point adjustment**

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 **8 / 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.

- 0 pts Click here to replace this description.

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.
- **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 10 / 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: Kubilay Agi

Student ID Number: 304784519

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?

The dirty bit tells us whether a page of memory has been modified or not. Having the dirty bit allows us to save time because we won't have to write the page back to disk (which is very slow) if the page is not dirty. This is during page swapping. we can simply evict the page from RAM. It also allows us to change a page without updating it everytime (which would also be slow). Instead we can delay writing, and take advantage of the fact that bulk writes on disk are faster than many small writes. Additionally it provides safety because we know we have changed the file and need to save its changes

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

The Application Binary (ABI) Interface is related to the system call interface because the system call interface is lower level than the ABI in the interface hierarchy. This means that for tasks that require system calls, the ABI needs to go through the system call interface so that the OS can handle these operations. This allows for more protection for the system because the operations of an ABI that was distributed by an outside source can be mediated by the operating system.

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

The shared libraries are shared between different processes. If they have global data and one process changes that data, the data will also be changed for the other process. This leads to issues of correctness b/c the second process will lose the value that it was previously using. It also causes issues of security b/c processes could access others' data.

As a result, it breaks the isolation that is required by the virtual memory abstraction, and it could lead to security issues

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

Each system call (when called by the user) has a trap number associated with it. After the call is made, a trap is recognized, which takes the system to a trap table and first level trap handler. The first level handler looks at which trap number is associated with the trap (the system call number) and it sets up the registers with the appropriate arguments before giving this information to the second level handler. The second level handler takes the information in the registers and will execute the system call. After execution, it will return control to the user and the user will continue as normal.(Assuming no errors occurred)

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

Each process has a certain size working set to begin with. However, if a process keeps getting page faults, the system will steal pages and give them to this process from other processes who only use a subset of their working set and page fault very infrequently. The general relationship is that page stealing algorithms grow and shrink the working set size of the running processes depending on which processes need more pages and which ones can live without them.

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 is maximizable with non preemptive Scheduling. This is because non preemption means we do not interrupt running processses and instead wait for them to finish and yield the CPU for the next process. Because we only perform context switches when a process finishes, relatively few cycles are spent switching and more cycles are spent doing actual computation.

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

External fragmentation is when there are several interspersed blocks of memory in RAM that are too small to be allocated for any process. Worst fit handles this better than best fit because worst fit uses the biggest block possible to shave off chunks for processes so that the left over free block is still (generally) of usable size. Since they are still usable, this delays/handles external fragmentation. This is in contrast to best fit which ends up leaving very small chunks which leads to external fragmentation very quickly.

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)?

For shared memory IPC, this memory is used solely for communication. Whenever a process writes to this memory, the change is available to ~~[illegible]~~ read or write to by the other process. On the other hand forked processes only point to the same data area while neither process writes/changes the data section. As soon as one of them does write to ~~the~~ this shared region, the system makes an individual copy for each one to reflect the changes made (or lack thereof). This system is called Copy-on-write which is not used with shared memory IPC because it would defeat the purpose of IPC.

9. What is the purpose of a condition variable?

The purpose of a condition variable is to save time and can be used instead of spinning locks. After [illegible] a thread or process ~~to~~ goes to sleep because it is waiting for something to finish before it can run, the condition variable signals for them to wake up. Threads that are waiting are generally put into a queue and sleep; once the process is woken up they are popped of one by one to continue execution. This saves CPU cycles because the threads won't keep spinning to check for a lock to be open or I/O to complete. They can rely on the condition variable to tell them when to run.

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

After a TLB miss, the OS needs to look at the page table for the running process to find the page it is asking for. Once it goes to the page table, there are several possible scenarios. One outcome could be that the page is known by the page table, ~~so the~~ and the page is in RAM, so then the TLB is updated and the instruction is run again, this time getting a TLB hit.

Another outcome is that the system gets a page fault because the page is valid but it is not in RAM. In this case, the system brings the page into RAM from disk, updates the TLB and then runs the instruction again. The third outcome is that the system goes to the page table and sees that the desired page hasn't been allocated and is therefore invalid. This case gives us a segmentation fault and the process is killed.