CS 111 Midterm

Erynn-Marie Phan

TOTAL POINTS

88 / 100

QUESTION 1

1 1 8 / 8

✓ **- 0 pts Correct**

- 8 pts No answer

- 7 pts Wrong answer

- 4 pts Answer on right track but not correct

- 3 pts Answer needs more detail

QUESTION 2

2 2 10 / 10

✓ **- 0 pts Correct**

- 10 pts No answer

- 9 pts Wrong answer

- 3 pts Incorrect answers for RR

- 3 pts Incorrect answers for FCFS

- 3 pts Incorrect answers for SJF

- 3 pts Answer of which has the largest overhead is incorrect or not present

QUESTION 3

3 3 10 / 10

✓ **- 0 pts Correct**

- 10 pts No answer

- 9 pts Wrong answer

- 5 pts Answer on the right track but not correct OR missing part

- 3 pts Answer needs a little more detail OR is slightly off

QUESTION 4

4 4 8 / 8

✓ **- 0 pts Correct**

- 2 pts Miss some details or some sentences are not accurate/correct enough.

- 5 pts Wrote down something, but far from correct/enough.

- 7 pts Wrong answer.

- 7 pts Cannot fully understand/recognize your answer. Please type down your answer using regrading request. Thanks.

- 8 pts No answer.

QUESTION 5

5 5 8 / 8

✓ **- 0 pts Correct**

- 3 pts Didn't explain for shared memory IPC, different processes refer to the exact same page frames or need synchronization.

- 3 pts Didn't explain the copy-on-write property for fork.

- 6 pts Wrong answer or not what we want.

- 7 pts Cannot fully understand/recognize your answer. Please type down your answer using regrading request. Thanks.

- 8 pts No answer.

- 3 pts Missing details.

QUESTION 6

6 6 10 / 10

✓ **- 0 pts Correct**

- 3 pts Didn't consider the case where the page is in RAM.

- 3 pts Didn't consider the case where the page is not in RAM but in disk (page fault).

- 6 pts Wrote down something that makes sense, but didn't cover the main points that we are looking for. For example, didn't answer what operations are required (page table lookup) and didn't cover all outcomes.

- 9 pts Cannot fully understand/recognize your answer. Please type down your answer using

regrading request. Thanks.

- **10 pts** No answer.
- **3 pts** Missing details.

QUESTION 7

7 7 **15 / 15**

✓ **- 0 pts Correct**

- **5 pts** The first 4 iterations are page faults
- **2 pts** Missing last page fault
- **15 pts** Incorrect
- **10 pts** All squares were not filled out
- **5 pts** Incorrect use of the algorithm

QUESTION 8

8 8 **8 / 15**

- **0 pts** Correct
- **15 pts** Incorrect/ Not Done

✓ **- 5 pts Used bit should be set on load**

- **5 pts** Page fault on startup
- **5 pts** Incorrect use of the algorithm

✓ **- 2 pts Missing page fault**

QUESTION 9

9 16 pts

9.1 a **2 / 4**

- **3 pts** Prolematic
- **4 pts** Incorrect
- **0 pts** Correct

✓ **- 2 pts Partially correct**

We also need to emulate system calls.

9.2 b **3 / 3**

- **3 pts** Incorrect
- **2 pts** Problematic

✓ **- 0 pts Click here to replace this description.**

- **1 pts** Partially correct

9.3 c **3 / 3**

- **1 pts** Partially correct.
- **2 pts** Problematic
- **3 pts** Incorrect

✓ **- 0 pts Correct**

9.4 d **0 / 3**

- **2 pts** Problematic
- **0 pts** Correct

✓ **- 3 pts Incorrect**

- **1 pts** Partially correct

Architecture is the same. Why would we need to simulate virtual memory?

9.5 e **3 / 3**

✓ **- 0 pts Correct**

- **3 pts** Incorrect
- **2 pts** Click here to replace this description.

Midterm Examination
CS 111, Spring 2019
5/1/2019, 4 – 5:50pm

Name: Erynn-Marie Pham Student ID: 504 757 459

This is a closed book, closed notes test. One single-sided cheat sheet is allowed.

1. What is the benefit of using the copy-on-right optimization when performing a *fork* in the Linux system?
A fork creates a child process whose address space is almost identical to its parent. It would be expensive to create an entire duplicate copy of the process's memory, and it is unnecessary if neither process will modify a segment.
Copy-on-write means the parent and child share data until one of them writes to the data segment. At that time, the writing process gets its own copy of the data. This is less expensive than copying the data segment at fork time, because it may never be copied at all.

2. Round Robin, First come First Serve, and Shortest Job First are three scheduling algorithms that can be used to schedule a CPU. What are their advantages and disadvantages? Which one is likely to have the largest overhead? Why?
RR (time-division multiplexing) pros - good response time and fairness because a lor job doesn't delay shorter jobs. cons - most overhead due to context switches

FCFS (in order of arrival, run to completion) pros - non-preemptive → low overhead from context switches. cons - a long job can delay response to other, shorter jobs. non-preemptive → no overlap. not good utilization of CPU

SJF (like FCFS, but goes to the shortest job) pros - low all turnaround time, low overhead from context switches. cons - a long job that arrives earlier can still delay others, shorter jobs. non-preemptive → no overlap not good utilization of CP

SJF
- preemptive (when a new job arrived, compare its completion time with the current process and switch if it's shorter) pros: minimal turnaround time. long jobs do not delay shorter jobs. other jobs can run when the running job is blocked. cons: some overhead from context switches.
- not preemptive

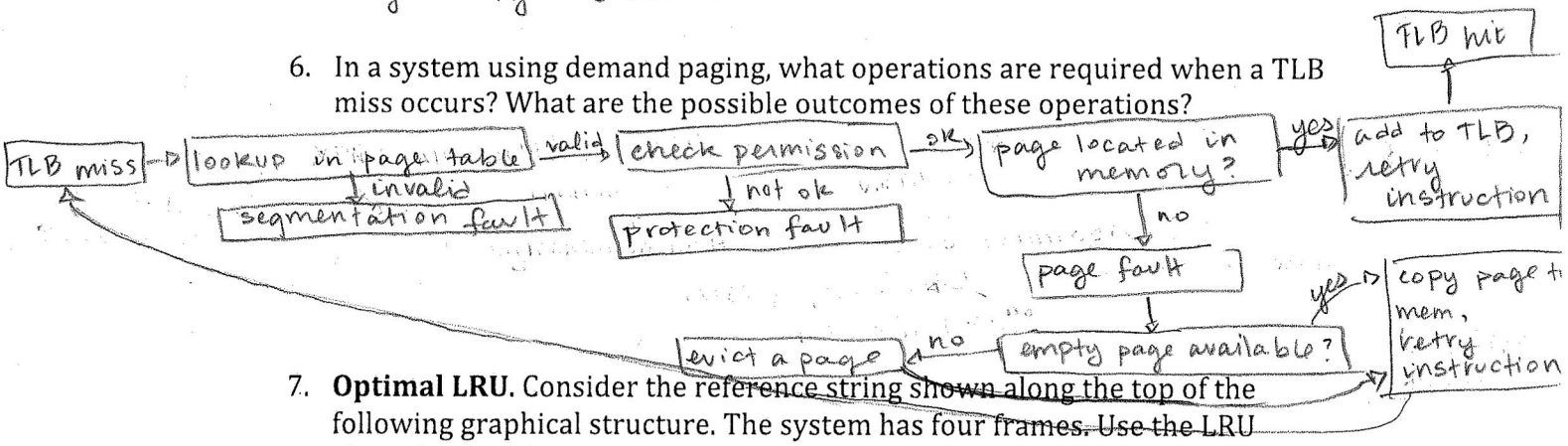
3. In a virtual memory system, why is it beneficial to have a dirty bit associated with a page? What are the techniques we can use to reduce the I/O involved in evicting dirty pages?
The dirty bit is useful when there are 2 copies of the page - 1 in memory, 1 on disk The dirty bit=1 if the process writes to the page. This means the copy on disk is needs to be overwritten with the one in memory if the page gets evicted from memory.
We can store up many writes and do them at the same time, because one large write to disk is more efficient than many small writes.
We can update the disk copy in a background process, before the page is evicted, so the page replacement takes less time later.

4. What is the relationship between the concept of working sets and page stealing algorithms?
Working set sizes are dynamically adjusted by page stealing algorithms. A process that needs a larger working set will get it by stealing pages from other processes that seem to need less. A process that can use a smaller working set will gradually find its working set reduced as pages are stolen from it.

5. 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 IPC, both processes could write to the shared data section. After a fork, if a process tried to write to the shared data section, the OS would give it its own copy of the data section. The other process's page table would still point to the unmodified, original frame. Because of this, a parent and child cannot communicate by writing to shared data.

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



7. **Optimal LRU.** Consider the reference string shown along the top of the following graphical structure. The system has four frames. Use the LRU algorithm to select pages for replacement. Place the page number in the proper frame. Mark when page faults occur in the bottom line of boxes. State how many page faults occur. The numbers across the top indicate the reference string.

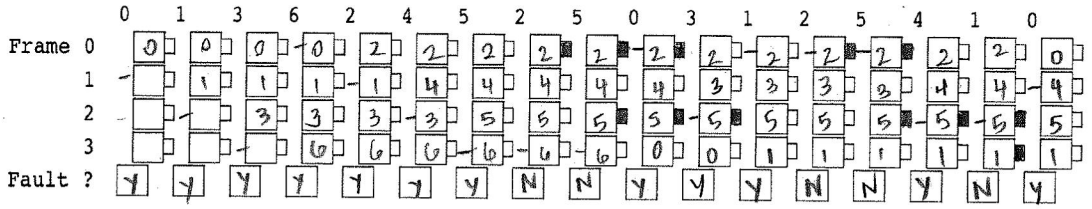
	0	1	3	6	2	4	5	2	5	0	3	1	2	5	4	1	0
Frame 0	0	0	0	0	2	2	2	2	2	2	2	1	1	1	1	1	1
1		1	1	1	1	4	4	4	4	4	3	3	3	3	4	4	4
2			3	3	3	3	5	5	5	5	5	5	2	2	2	2	0
3				6	6	6	6	6	6	0	0	0	0	5	5	5	5
Fault ?	Y	Y	Y	Y	Y	Y	Y	N	N	Y	Y	Y	Y	Y	Y	N	Y

14 page faults total

8. **Clock Algorithm**. The clock algorithm is an approximation of LRU based on using one use bit for each page. When a page is used its use bit is set to 1. We also use a pointer to the next victim, which is initialized to the first page/frame. When a page is loaded, it is

set to point to the next frame. The list of pages is considered as a circular queue. When a page is considered for replacement, the use bit for the next victim page is examined. If it is zero [that page is replaced] otherwise [the use bit is set to zero, the next victim pointer is advanced, and the process repeated until a page is found with a zero use bit].

Consider the reference string shown along the top of the following graphical structure. The system has four frames. Use the clock algorithm described in the previous paragraph. The narrow boxes to the right of the page number boxes can be used to keep up with use bits. Place the page number in the proper frame. Mark when page faults occur in the bottom line of boxes. State how many page faults occur.



12 page faults total

9. In the early 1990s, SUN Microsystems, the maker of the Solaris Operating System, wanted to move from the engineering desktop, where it was well established, to a broader market for personal productivity tools. The best personal productivity tools were all being written for Windows platforms, and SUN was on the wrong side of the applications/demand/volume cycle, which made getting those applications ported to Solaris a non-option.

One approach to their problem was to modify the version of Solaris that ran on x86 processors (the popular hardware platform for Windows) to be able to run Windows binaries without any alterations to those binaries. This would allow Sun to automatically offer all of the great applications that were available for Windows.

(a) What would have to be done to permit Windows binaries to be loaded into memory and executed on a Solaris/x86 system?

The SUN operating system would have to recognize Windows load modules as executable code. The Windows load modules may have had a different format.

(b) What would have to be done to correctly execute the system calls that the Windows programs requested?

SUN would have to implement 2nd-level system call handlers matching the specifications of the Windows system call handlers, because when the user processes regain control after a system call, they expect that certain events have occurred. Also, SUN would have to make their system call numbers match Windows', so that the right handler is called when the trap table is indexed.

depending on the limitatio and capabilitie of the hard war

(c) How good might the performance of such a system be? Justify your answer.

The applications might perform as well on Solaris as they do on Windows. Once they are loaded into memory, they should run directly on the CPU with minimal intervention from either OS (limited direct execution).

(d) List another critical thing, besides supporting a new load module format and the basic system calls, that the system would have to be prepared to simulate? How might that be done?

The system needs to simulate virtual memory in the way the applications expect. Since the programs are already load modules, their virtual addresses are already filled in. The virtual address space provided by the Solaris OS should be the same size as the one provided by Windows, with the code, data, and stack segments in the same spaces, so that the OS can properly check for virtual addresses being out of bounds.

(e) Could a similar approach work on a Solaris/PowerPC or Solaris/SPARC system? Why or why not?

It would not work because the ISAs don't match. The Windows ABI is only compatible with the x86 ISA. The load modules consist of instructions specific to this ISA.