CSci 402 - Operating Systems

Final Exam (DEN Section)

Fall 2021

(9:00:00am - 9:40:00am, Monday, December 13)

Instructor: Bill Cheng

Teaching Assistant: Ben Yan

(This exam is open book and open notes.
Remember what you have promised when you signed your
Academic Integrity Honor Code Pledge.)

Time: 40 minutes

_____-
Name (please print)

Total: 38 points

_____-
Signature

Instructions

1. This is the first page of your exam. The previous page is a title page and does not have a page number. Since this is a take-home exam, no need to sign above since you won't submit this file.
2. Read problem descriptions carefully. You may not receive any credit if you answer the wrong question. Furthermore, if a problem says *"in N words or less"*, use that as a hint that N words or less are expected in the answer (your answer can be longer if you want). Please note that points may get *deducted* if you put in wrong stuff in your answer.
3. If a question doesn't say `weenix`, please do not give `weenix`-specific answers.
4. Write answers to all problems in the **answers text file**.
5. For non-multiple-choice and non-fill-in-the blank questions, please show all work (if applicable and appropriate). If you cannot finish a problem, your written work may help us to give you partial credit. We may not give full credit for answers only (i.e., for answers that do not show any work). Grading can only be based on what you wrote and cannot be based on what's on your mind when you wrote your answers.
6. Please do *not* just draw pictures to answer questions (unless you are specifically asked to draw pictures). Pictures will not be considered for grading unless they are clearly explained with words, equations, and/or formulas. It's very difficult to draw pictures in a text file and you are not permitted to submit additional files other than the answers text file.
7. For problems that have multiple parts, please clearly *label* which part you are providing answers for.
8. Please ignore minor spelling and grammatical errors. They do not make an answer invalid or incorrect.
9. During the exam, please only ask questions to *clarify* problems. Questions such as "would it be okay if I answer it this way" will not be answered (unless it can be answered to the whole class). Also, you are suppose to know the definitions and abbreviations/acronyms of *all technical terms*. We cannot "clarify" them for you. We also will **not** answer any clarification-type question for multiple choice problems since that would often give answers away.
10. Unless otherwise specified and stated explicitly, multiple choice questions have one or more correct answers. You will get points for selecting correct ones and you will lose points for selecting wrong ones.
11. When we grade your exam, we must assume that you wrote what you meant and you meant what you wrote. So, please write your answers accordingly.

(Q1) (3 points) Let's say that you have four threads A, B, C, and D and you are using the basic **round-robin (RR) / time-slicing** scheduler with a very small time slice. At time zero, all four threads are in the run queue and their processing times are shown in the table below. Assuming that there are no future arrivals into the run queue, please complete the table below with the "waiting time" of all four threads and the "average waiting time" (AWT) of these four threads and write the results on your answer sheet. Please make it very clear which waiting time is for which thread and which one is the AWT. For non-integer answers, you can use fractions or decimals with two digits after the decimal point. Your answer must not contain plus or multiplication symbols. You must use the definition of "waiting time" given in lectures.

	A	B	C	D	AWT (1 pt)
T (hrs)	6	7	9	7	-
wt (hrs)					

(Q2) (2 points) The first procedure of an **idle thread** is shown here:

```
void idle_thread() {
  while (1) {
    euqueue(runqueue, CurrentThread);
    thread_switch();
  }
}
```

Which of the following statements are correct about such an **idle thread**?

(1) an idle thread is often used in user space when there are multiple CPUs
(2) an idle thread is a kernel thread that never gives up the CPU
(3) an idle thread does not need a thread control block because it never needs to wait for mutex or I/O
(4) an idle thread can never be in the zombie state since it does not self-terminate
(5) an idle thread can never sleep in a mutex queue or an I/O queue

Answer (just give numbers): _____

(Q3) (2 points) Let's say that you are using **extensible hashing** to speed up directory lookup. If $h_3(''proc.c'') = 1$, which of the following are possible values of $h_5(''proc.c'')$?

(1) 41
(2) 17
(3) 21
(4) 25
(5) 33

Answer (just give numbers): _____

(Q4) (2 points) Which of the following are maintained in a **S5FS superblock**?

(1) inode cache
(2) disk map
(3) first node of the singly-linked free list
(4) inode numbers of all the free inodes
(5) root node of the B+ tree for locating all the inodes

Answer (just give numbers): _____

(Q5) (3 points) Let's say that you have four threads A, B, C, and D and you are using **stride scheduling**. You have decided to give thread A 7 ticket, thread B 7 tickets, thread C 6 tickets, and thread D 9 tickets. The initial pass values that **you must used** for the four threads are shown below along with the "winner" of the iteration 1. Please run **stride scheduling** to fill out all the entries (pass values) in the table and keep track of the "winner" in each round. For **iterations 2 through 7**, please write on your answer sheet the "winner" and the winning pass value of that iteration. (For example, you would write "A:2" for iteration 1 since A is the "winner" of iteration 1 and the winning pass value is 2.) You must use the **smallest possible integer stride values** when calculating all the pass values. If you get the stride values wrong, you will not get any partial credit for this problem.

itr	A	B	C	D
1	(2)	23	34	15
2				
3				
4				
5				
6				
7				

(Q6) (2 points) Which of the following statements are correct about **real-time** systems and threads?

(1) a real-time thread in the kernel is a thread that uses timer-related system calls
(2) a real-time thread in the kernel is a thread that must start running inside the CPU before a deadline
(3) there is not much difference between a soft real-time system and a hard real-time system
(4) a real-time thread in the kernel is a thread that can schedule itself to run at the time it wants without the help of the scheduler
(5) none of the above is a correct answer

Answer (just give numbers): _____

(Q7) (2 points) On a uniprocessor, which of the following steps are **required** when a **pageout daemon** wants to **free up** a **dirty/modified page** that belongs to a user process on an "inactive page list"?

(1) unmap this page from the page tables of all the processes that share this page
(2) flush the entire translation lookaside buffer
(3) "unpin" the page when it is being written to the disk
(4) look at the page table of this process to locate the corresponding backing store
(5) none of the above is a correct answer

Answer (just give numbers): _____

(Q8) (2 points) For a terminal, input characters may need to be processed/edited in some way before they reach the application. Which of the following **data structures** are used to solve this problem?

(1) a B tree and a hash table
(2) a memory map and a page table
(3) a partial-line queue and a completed-line queue
(4) a translation lookaside buffer
(5) none of the above is a correct answer

Answer (just give numbers): _____

(Q9) (2 points) Which of the following statements are correct about using the **multi-level feedback queue** to schedule both interactive and non-interactive jobs?

(1) if a highest priority thread gives up the CPU voluntarily before using up a full time slice, you should decrease its priority
(2) if a highest priority thread uses a full time slice, you should add a new higher priority queue and increase its priority
(3) if a highest priority thread uses a full time slice, you should decrease its priority
(4) if a highest priority thread gives up the CPU voluntarily before using up a full time slice, you should add a new higher priority queue and increase its priority
(5) none of the above is a correct answer

Answer (just give numbers): _____

(Q10) (2 points) Let's say that you are using a **rate-monitonic scheduler** to schedule 4 periodic tasks with $T_1 = 0.5$, $P_1 = 4$, $T_2 = 1$, $P_2 = 4.5$, $T_3 = 0.5$, $P_3 = 5$, and $T_4 = 1$, $P_4 = 6.5$. Let's say that you schedule all 4 period tasts to start a time = 0. Since the total utilization is too large to guarantee that all jobs will meet their deadlines, the only way to know is to simulate the **rate-monitonic scheduler**. How many seconds into the simulation would be the first time all 4 jobs would start executing at exactly the same time again? Please just give a numeric answer (no partial credit for this problem).

(Q11) (2 points) Which of the following statements are correct about using **base and bounds** registers in a **segmented virtual memory** scheme?

(1) by adding some hardware in MMU, the basic base and bounds scheme can be extended to speed up directory lookup
(2) by adding some hardware in MMU, the basic base and bounds scheme can be extended to reduce the size of page tables
(3) by adding some hardware in MMU, the basic base and bounds scheme can be extended to provide crash resiliency
(4) by adding some hardware in MMU, the basic base and bounds scheme can be extended to make a memory segment read-only
(5) none of the above is a correct answer

Answer (just give numbers): _____

(Q12) (2 points) The following code can be used to lock a mutex in a **straight-threads** (i.e., no interrupt) mutex implementation on a **single CPU**:

```
if (!m->locked) {
  m->locked = 1;
}
```

Which of the following statements are correct about using the above code in a **multiple CPU** system to lock mutex `m` correctly and safely?

(1) it's safe to use the code as-is to lock mutex `m` because it's impossible for multiple CPUs to write to the same memory location at the same time
(2) it's safe to use the code as-is to lock mutex `m` because reading the value of m->locked is an atomic operation
(3) it's safe to use the code as-is to lock mutex `m` because setting the value of m->locked is an atomic operation
(4) it's safe to use the code as-is to lock mutex `m` because it's impossible for multiple CPUs to read from the same memory location at the same time
(5) it's unsafe to use the code as-is to lock mutex `m` because multiple threads may all think that they own the same mutex at the same time

Answer (just give numbers): _____

(Q13) (2 points) Which of the following statements are correct about **user processes and threads in weenix**?

(1) a user stack is explicitly represented in the kernel as a kernel stack
(2) a user process and its corresponding kernel process share the same "process control block"
(3) to "kill" a user thread, you just need to "kill" the corresponding kernel thread since they are exactly the same thread
(4) there is no kernel data structure to represent a kernel stack
(5) none of the above is a correct answer

Answer (just give numbers): _____

(Q14) (2 points) A correct implementation of **straight-threads** (i.e., no interrupt) **thread switching** on a **single CPU** is shown here (assuming that the run queue is never empty):

```
void thread_switch() {
  thread_t NextThread, OldCurrent;

  NextThread = dequeue(RunQueue);
  OldCurrent = CurrentThread;
  CurrentThread = NextThread;
  swapcontext(&OldCurrent->context, &NextThread->context);
}
```

Which of the following statements are correct about using the above code in a **multiple-CPU** system?

(1) cannot use the code as-is because **swapcontext()** must include an argument to specify which CPU to use for context swapping
(2) cannot use the code as-is because **RunQueue** must be an array since we have multiple CPUs
(3) cannot use the code as-is because `thread_switch()` is missing an argument that specifies which CPU to use
(4) cannot use the code as-is because **CurrentThread** must be an array since we have multiple CPUs
(5) none of the above is a correct answer

Answer (just give numbers): _____

(Q15) (2 points) Which of the following statements are correct about **backing store**?

(1) none of the pages in a read-only mapping of a file needs a backing store in swap space
(2) pages in a shadow object should have its backing store in swap space
(3) none of the pages in a read-write shared mapping of a file needs a backing store in swap space
(4) none of the pages in a read-write private mapping of a file needs a backing store in swap space
(5) none of the above is a correct answer

Answer (just give numbers): _____

(Q16) (2 points) Which of the following statements are correct about a **SJF (shortest job first)** scheduler?

(1) it is possible that short jobs may "starve" if long jobs keep arriving
(2) this scheduler generally has a smaller variance in waiting time than other schedulers
(3) compared with some other schedulers, this scheduler can have a large average waiting time
(4) this scheduler is inherently unfair to short jobs
(5) none of the above is a correct answer

Answer (just give numbers): _____

(Q17) (2 points) Considering only **clustered hash page table** schemes and **(non-clustered) hashed page table** schemes, which of the following statements are correct?

(1) non-clustered hash page tables typically performs better than clustered page tables
(2) clustered hash page tables would perform better if address space is truly sparsely allocated
(3) the performance of clustered hash page tables depends on how address space is allocated
(4) the performance of non-clustered hash page tables depends on the lengths of the hash conflict/collision resolution chains
(5) none of the above is a correct answer

Answer (just give numbers): _____

(Q18) (2 points) Which of the following statements are correct about approaches to deal with the problem caused by the **popf** instruction so that a virtual machine can be built for **x86 processors**?

(1) with paravirtualization, sensitive instructions are replaced with hypercalls at the time the kernel is compiled
(2) in Intel's solution, a new "processor mode" was added
(3) in VMware's solution is a compile-time solution, i.e., sensitive instructions are replaced with hypercalls when kernel is compiled
(4) in Intel's solution, the `popf` instruction is disabled so that it won't cause any problem
(5) none of the above is a correct answer

Answer (just give numbers): _____