

CSci 402 - Operating Systems
Final Exam
Summer 2021

(10:00:00am - 10:40:00am, Tuesday, August 3)

Instructor: Bill Cheng

Teaching Assistant: (N/A)

*(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) (2 points) Let's say that the address space of a user space in **weenix** looks like the following:

VADDR RANGE	PROT	FLAGS	MMOBJ	OFFSET	VFN RANGE
0x08039000-0x08048000	rw-	PRIVATE	0xcfe0c034	0x0000d	0x08039-0x08048
0x08048000-0x0804e000	r-x	PRIVATE	0xcfe0c004	0x00009	0x08048-0x0804e
0x0804e000-0x0805f000	rw-	PRIVATE	0xcfe0c064	0x00003	0x0804e-0x0805f

If you get a page fault with vaddr = 0x0805d668, what **pagenum** would you use to lookup a page frame when you are handling a page fault? Please just give an integer value answer (no partial credit for this problem).

(Q2) (2 points) Which of the following statements are correct about the **FIFO** scheduler?

- (1) it appears to be a fair scheduling policy
- (2) compared with some other schedulers, this scheduler can have a large average waiting time for the jobs that are in run queue
- (3) it's a scheduler that's inherently unfair to long jobs
- (4) it has the largest variance in waiting time among all scheduling disciplines
- (5) "starvation" at the scheduler is a common problem for this scheduling policy

Answer (just give numbers): 1, 2

(Q3) (2 points) Which of the following statements are correct about the **NOR** vs. **NAND** flash memory technologies?

- (1) for a NAND flash, the smallest addressable unit for reading is a block
- (2) a NOR flash is byte-addressable
- (3) a NAND flash is byte-addressable
- (4) a NAND flash is more suitable to be used in a file system than a NOR flash
- (5) for writing, a NAND flash is page-erasable but not block-erasable

Answer (just give numbers): 2, 4

NAND: block erasable but not page erasable

(Q4) (2 points) Which of the following statements are correct about the **popf machine instruction** in a **traditional Intel x86 processor**, i.e., non-Vanderpool?

- (1) `popf` behaves differently when it is executed in the user mode and the privilege mode
- (2) `popf` is considered to be a “sensitive instruction”
- (3) executing `popf` in either the privileged mode or the user mode will cause a trap
- (4) executing `popf` in the user mode causes a trap while executing it in the privileged mode does not
- (5) executing `popf` in the user mode will not cause a trap while executing it in the privileged mode will cause a trap

Answer (just give numbers): 1, 2

(Q5) (2 points) Which of the following statements are **correct** about **I/O virtualization**?

- (1) I/O virtualization in building virtual machines for desktop machines is challenging because it's virtually impossible for virtual machine vendors to support all devices
- (2) in VMware's solution to I/O virtualization, most device drivers in the guest OS must be rewritten so that they can be supported
- (3) in Xen's solution to I/O virtualization, only a few device drivers in the guest OS has to be rewritten in order for Xen to use them
- (4) I/O virtualization is not as big of a problem in building virtual machines for high performance servers because only a small number of devices need to be supported
- (5) VMware's I/O virtualization solution performs better than Xen's I/O virtualization solution

Answer (just give numbers): 1, 3


(Q6) (2 points) What are the **OS design approaches** to fix the **reliability** problem of a **monolithic kernel** to reduce kernel crashes?

- (1) use dynamically loaded kernel modules so that unreliable kernel module can be unloaded while the rest of the kernel is running
- (2) shrink the size of the kernel code that must run in the privileged mode
- (3) never release a kernel unless it's completely bug-free
- (4) run unreliable part of the kernel in user space
- (5) none of the above is a correct answer

Answer (just give numbers): 2

(Q7) (2 points) Let's say that you are using a **rate-monitonic scheduler** to schedule 4 periodic tasks with $T_1 = 0.5$, $P_1 = 3$, $T_2 = 1$, $P_2 = 3.5$, $T_3 = 0.5$, $P_3 = 4$, and $T_4 = 1$, $P_4 = 5.5$. Let's say that you schedule all 4 periodic tasks to start at 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).

(Q8) (2 points) Which of the following statements are correct about **futex**?

- (1) if a futex is currently **locked** and not being released, a thread calling `futex_lock()` must enter the kernel to wait for the lock to be released
-  (2) the best place to use a futex is inside the kernel
- (3) in order for a futex to function correctly when there are multiple CPUs, the kernel is required to be atomic with respect to that futex
- (4) futex is not fast at all because it uses **CAS()** operations
- (5) futex is designed to work only in multi-CPU systems and will not work in single-CPU systems

Answer (just give numbers): 1, 3

(Q9) (2 points) Which of the following statements are correct about **pseudo-terminal driver** vs. **terminal driver**?

- (1) typically, pseudo-terminal driver and terminal driver are both user space drivers
- (2) for a pseudo-terminal, the input and output (on the device end) comes from and goes to an actual device
- (3) for a pseudo-terminal, the input and output (on the device end) comes from and goes to a special kernel process
- (4) a terminal driver typically runs in kernel while a pseudo-terminal driver typically runs in user space
- (5) none of the above is a correct answer

Answer (just give numbers): 5

(Q10) (2 points) Which of the following statements are correct about a **B+ tree of order $m = 9$** ?

- (1) since m is 9, it's okay for an intermediate node (i.e., neither a root node nor a leaf node) to have 5 child nodes
- (2) since m is 9, the root node must have at least 4 child nodes
- (3) since m is 9, the height of the B+ tree must be strictly less than 9
- (4) since m is 9, it's okay for an intermediate node (i.e., neither a root node nor a leaf node) to have 4 child nodes
- (5) since m is 9, the height of the B+ tree must be strictly greater than 5

Answer (just give numbers): 1

(Q11) (2 points) Which of the following statements are correct about **physical vs. virtual addresses** on a 32-bit machine?

- (1) a process uses physical addresses to execute code when it is created in the kernel and switch to use virtual addresses when it runs in the user space for the first time
- (2) a device driver uses physical addresses to execute code but use virtual address to access data on a device
- (3) there is a system call a user process can call to convert a user space virtual address into a corresponding kernel space virtual address
- (4) a user process uses physical addresses when it's running in the kernel and uses virtual addresses when it's running in user space
- (5) none of the above is a correct answer

Answer (just give numbers): 2, 3

(Q12) (2 points) Which of the following statements are correct about **terminal device drivers** vs. **network device drivers**?

- (1) network drivers deals with binary data while terminal drivers only deals with non-binary data
- (2) in network communication, data needs to be passed from one kernel module to another without copying to achieve acceptable performance
- (3) for a terminal, data needs to be passed from one kernel module to another without copying to achieve acceptable performance
- (4) their abstractions to applications are completely different
- (5) most of the work performed by a terminal device driver is done as “deferred processing”

Answer (just give numbers): 2

(Q13) (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 a thread in user space that never gives up the CPU
- (2) an idle thread can never be in the zombie state since it does not self-terminate
- (3) an idle thread is often used in the kernel even when there is only one CPU
- (4) an idle thread does not need a thread control block because it never needs to wait for I/O
- (5) an idle thread can never sleep in a mutex queue or an I/O queue

Answer (just give numbers): 3

(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 `thread_switch()` is missing an argument that specifies which CPU to use
- (2) cannot use the code as-is because `swapcontext()` must include an argument to specify which CPU to use for context swapping
- (3) cannot use the code as-is because **CurrentThread** must be an array since we have multiple CPUs
- (4) cannot use the code as-is because **RunQueue** must be an array since we have multiple CPUs
- (5) cannot use the code as-is because if a single **RunQueue** is used, accessing **RunQueue** must be synchronized across multiple CPUs

Answer (just give numbers): 5

- (Q15) (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)	3	7	5	5	-
wt (hrs)					

- (Q16) (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 5 ticket, thread B 4 tickets, thread C 4 tickets, and thread D 7 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:5" for iteration 1 since A is the "winner" of iteration 1 and the winning pass value is 5.) 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	11	10	7	(5)
2				
3				
4				
5				
6				
7				

(Q17) (2 points) Which of the following statements are correct about a **SJF (shortest job first)** scheduler?

- (1) it generally has a smaller variance in waiting time than other schedulers
- (2) it appears to have a high throughput, although it cannot achieve a throughput higher than one job per second
- (3) compared with some other schedulers, this scheduler can have a large average waiting time for the jobs that are in run queue
- (4) it is possible that long jobs may “starve” if short jobs keep arriving
- (5) it’s a scheduler that’s inherently unfair to long jobs

Answer (just give numbers): 4,5

(Q18) (2 points) Which of the following statements are correct about thread implementation strategies?

- (1) one problem with the $M \times N$ model is priority inversion
- (2) $N \times 1$ model is preferred over 1×1 model because $N \times 1$ model can achieve higher parallelism
- (3) one problem with the $N \times 1$ model is priority inversion
- (4) one problem with the 1×1 model is that it’s slow because system calls are slow
- (5) the scheduler activations model is a variation on the one-level model

Answer (just give numbers): 1,4