

Name:

ID:

Midterm Exam
CS402
26 Oct 1998

You have 1 hr. 20 min. for this exam. The exam has 9 pages. There are 100 possible points. Show all your work for partial credit.

Definitions

Each question includes the number of points. Answer all questions in this section.

1. Give a term that describes the following:
 - a) The four requirements for deadlock (4 pts)
Answer: Mutual exclusion, non-preemption, hold and wait, circular wait
 - b) A piece of hardware that caches virtual page → physical page mappings (1 pt)
Answer: Translation lookaside buffer or associative memory
 - c) The method used by a user program to access operating system services (1 pt)
Answer: system call (I'll take trap or software interrupt, too)
 - d) Two goals of operating systems (2 pts)
Answer: Resource Allocation and Virtual Machine
 - e) Two processor (not process) states (2 pts)
Answer: interrupt, supervisor, user
 - f) The name of an algorithm for deadlock avoidance when you know the maximum resource requirements of all processes (1 pt)
Answer: Dijkstra's Banker's Algorithm
 - g) The first table consulted when looking up an address in a paged segmentation system (1 pt)
Answer: Segment table
 - h) Two synchronization (mutual exclusion) methods that do not involve busy waiting (2 pts)
Answer: Monitors, Semaphores, Message Passing
 - i) Two synchronization (mutual exclusion) methods that do involve busy waiting (2 pts)
Answer: Peterson's algorithm, Dekker's Algorithm, test and set, strict alternation
 - j) What happens when a process references a virtual address that is not currently in physical memory (1 pt)
Answer: page fault
 - k) An OS abstraction of an interrupt that allows processes to set timers or be notified about pending I/O (1 pt)
Answer: signals
 - l) Two semantics for the signal operation on a condition variable (the semantics are named for an inventor or a system that implemented them) (2 pts)

Name:

ID:

Answer: Hansen, Hoare, or Mesa - pick 2.

Short Answer

Each question is worth 10 points. Do all questions in this section.

2. A programmer is using Java on a new uniprocessor system. The application of interest is multi-threaded. On the second system, the application does not perform as well, and the problem is traced to the fact that all of the application's threads stop running whenever one of them makes an I/O request. What difference does this demonstrate between the two systems? (3 pts)

Answer: The first system supported thread-level scheduling and the second supports process-level scheduling. In other words the first system implements threads in the kernel and the second system implements them at user level.

Name another problem that the programmer is likely to encounter because of this difference. (3 pts)

Answer: The first system supported thread-level scheduling and the second supports process-level scheduling. In other words the first system implements threads in the kernel and the second system implements them at user level.

Then the programmer moves the application to a multiprocessor machine (2 processors that run code simultaneously), and race conditions appear that were not evident before. Why are race conditions more likely to cause malfunctions on a true multiprocessor than on a uniprocessor doing time slicing? (4 pts)

Answer: On a uniprocessor, the fact that only one process at a time is running provides some amount of synchronization. Certain possible race conditions are never exercised (or exercised infrequently) because processes are context switched infrequently (compared to the number of instructions they execute per quantum). In a multiprocessor, every instruction is running concurrently with another instruction, which is like context switching every instruction.

Name:

ID:

3. Show the results of the following memory transactions on the buddy system illustrated below. The system has 1MB of memory. Each blank should be filled with the state of the system after one of the given transactions. An allocation for a process is a box with the process identifier (), an unallocated amount of memory is a box with the unallocated size (in KB) in it (). This is the same convention used in the examples in class and in Tannenbaum. (4 pts).

	0	128 K	256 K	384 K	512 K	640 K	768 K	896 K	1 M
initial condition	1024								
A requests 120KB									
B requests 200KB									
C requests 50KB									
A returns its allocation									
D requests 35KB									
C returns its allocation									
D returns its allocation									
B returns its allocation									

In what way would using memory allocations that were a power of 4 rather than a power of 2 make the buddy system more efficient? (3 pts)

In what way would using memory allocations that were a power of 4 rather than a power of 2 make the buddy system less efficient? (3 pts)

Answer:

	0	128 K	256 K	384 K	512 K	640 K	768 K	896 K	1 M
A		128	256			512			
A		128	B			512			
A	C	64	B			512			
128	C	64	B			512			
128	C	D	B			512			
128	64	D	B			512			
	256		B			512			
	1024								

Name:

ID:

Using powers of 4 would mean fewer lists of holes to search (meaning faster look up times), but larger allocations, meaning more internal fragmentation.

Name:

ID:

4. 5 identical jobs are run once on a non-preemptive scheduler, and then again on a preemptive scheduler using a round robin scheduling discipline. The jobs are purely computational – they do almost no I/O. On the non-preemptive scheduler it takes 24 hours for all 5 of them to complete; on the preemptive one it takes 24 hours and 2 minutes for all 5 of them. If the time quantum of the preemptive scheduler is 0.05 sec (50 milliseconds), how long does a context switch take? (Show your work; it's OK to leave the answer as a reduced fraction). (6 pts)

Answer:

Over the 24 hour period there are 24 hrs ~ 60 min/hr ~ 60 sec/min ~ 20 switches/sec = 1728000 context switches. These take 120 seconds. Each context switch takes $120 / 1728000 \text{ sec} = 1/14400 \text{ sec} = 69.4 \mu\text{sec}$.

How can you make the two systems perform equivalently on these jobs? (2 pts)

Answer: Set the time quantum to $24 / 5 = 4.8$ hours. Can another scheduling discipline improve performance (and why or why not)? (2 pts)

Answer: No, the jobs are identical. The total (and the average) times will only get worse than running them sequentially.

5. Name 2 things that need to be saved on a context switch. (2 pts)

Answer: CPU registers, MMU state, FPU state, OS software state

Name something that has to be saved when creating a checkpoint (of an arbitrary process) that does not need to be saved on a context switch. (2 pts)

Why does your item need to be saved for a checkpoint but not for a context switch? (2 pts)

Answer: The entire memory contents of the process needs to be saved for a checkpoint. A checkpoint is enough information to restart the process from scratch, unlike a context switch, which assumes that state about the process (the PCB, the address space, etc.) will still be around.

Explain 2 uses of checkpoints. (4 pts)

Answer: Checkpoints are primarily used in deadlock recovery. A checkpointed process is killed and restarted (perhaps after other deadlocked processes have completed) in the hopes that it will not deadlock again. They can also enhance system reliability because long computations can be restarted if other elements of the system fail (for example a 3 week simulation that encounters a power failure). They can also be used for load balancing. The checkpointed process is killed on one machine and restarted on another.

Answer:

Virtual Address	Physical Address
0x0315	4315
0xC2F3	32F3
0x521F	721F
0xA319	1319

The third reference causes virtual page 0xA to be evicted because page 0x5 is not present and 0xA has the lowest timestamp. This makes 0x5's virtual page 0x7 and evicts 0xA. New pages have a timestamp of 1000. Then page 0xA is referenced causing a second fault, evicting 0x8 and resulting in this table:

Page Table Final State		
Virtual Page	Physical Page	Time stamp
0000	100	1000
0001	-	-
0010	101	0010
0011	000	1100
0100	-	-
0101	111	1000
0110	010	1101
0111	-	-
1000	-	-
1001	-	-
1010	001	1000
1011	-	-
1100	011	1011
1101	-	-
1110	110	0011
1111	-	-

What is the reference string generated by this sequence of accesses? (4 pts)

Answer: The reference string is just the virtual page numbers in order, so for this problem: 0 12 5 10 or in hex 0 C 5 A

What is Belady's Anomaly? (1 pt)

Answer: Is the page replacement algorithm in this problem affected by it? (1 pt)

Answer: Why or why not? (2 pts)

Answer: Belady's Anomaly is the property of certain paging algorithms that they can exhibit more page faults in a larger memory. LRU is not affected by it, because it's a stack algorithm. That's enough for credit, but the reason that LRU is a stack algorithm is that the timestamps totally order the pages in a manner independent of the size of the memory. (Actually, the approximation could be subject to it because pages with the same timestamp are not totally ordered, but there are ways to make it safe. One easy one is to say that pages with the same timestamp are ordered by virtual address.) That's a lot of answer for 2 points, but "stack algorithm" is sufficient.

7. Consider a computer system composed of the standard CPU and 9 smart peripherals. Each smart peripheral has its own processor and can communicate with the other smart peripherals via a shared bus. The bus has infinite bandwidth (that is, it can support simultaneous full-speed transfers between all the smart peripherals). However each peripheral is only half duplex, which means that it can either be sending information or receiving it, but not both. A peripheral that is sending and receiving simultaneously silently corrupts both transactions. Peripherals can also suspend processing, which means that their CPU is stopped. Peripherals can receive information any time that they are not sending it, including when they are suspended.

One of the smart peripherals is a lock manager that takes a request for a lock from a peripheral (or the CPU). After the peripheral makes the request, it suspends itself until the lock manager sends an interrupt to the peripheral indicating that it holds the lock. Peripherals can similarly release locks. Peripherals can send requests to the lock manager at any time without disrupting the reception of data. Locks are fair – peripherals never starve.

a)

Describe an algorithm that each smart peripheral can use to communicate with another peripheral if the lock manager can only support one lock. (All the peripherals will run the same algorithm for sending data.) (4 pts)

Answer: Assign a lock to the bus and require each peripheral to acquire the lock before it sends and release it when the transmission is complete.

Explain how your algorithm prevents data corruption. (2 pts)

Answer: Because only one peripheral can hold the lock at a time, at most one can be transmitting. Disrupting a transmission requires 2 transmitters.

Explain how your algorithm prevents deadlock. (2 pts)

Answer: There is only one lockable resource in the network, so a circular wait is impossible.

Explain how your algorithm affects the efficiency of the system (for better or worse). (2 pts)

Answer: Only one pair of peripherals can communicate at a time, which wastes bus bandwidth.

This question continues on the next page

Name:

ID:

b)

Describe an algorithm that each smart peripheral can use to communicate with another peripheral if the lock manager can support as many locks as you need. (4 pts)

Answer: Assign a lock to each peripheral, and number the peripherals. A peripheral must acquire both the lock for the source and destination peripheral in the numerical order before sending, and releases them when the transmission is complete.

Explain how your algorithm prevents data corruption. (2 pts)

Answer: Only one end of the connection can hold both locks, so only one end can be transmitting at once. Data corruption requires that both ends of the connection be transmitting.

Explain how your algorithm prevents deadlock. (2 pts)

Answer: Because the locks are acquired in numerical order, a circular wait is impossible. (A peripheral can only be trying for a high numbered lock if it has the lower lock. There can be no part of the resource graph that “doubles back” to form the loop).

Explain how your algorithm affects the efficiency of the system (for better or worse). (2 pts)

Answer: The system allows all the peripherals to be involved in transactions simultaneously. A system that actually scheduled transactions could probably do better, but it's questionable by how much.