University of California, Berkeley
College of Engineering
Computer Science Division – EECS

Spring 2002 Anthony D. Joseph

Midterm Exam *Solutions*

March 13, 2002
CS162 Operating Systems

Your Name:	
SID AND 162 Login:	
TA:	
Discussion Section:	

General Information:
This is a **closed book and notes** examination. You have ninety minutes to answer as many questions as possible. The number in parentheses at the beginning of each question indicates the number of points given to the question; there are 100 points in all. You should read **all** of the questions before starting the exam, as some of the questions are substantially more time consuming.

Write all of your answers directly on this paper. ***Make your answers as concise as possible.*** If there is something in a question that you believe is open to interpretation, then please ask us about it!

Good Luck!!

Problem	Possible	Score
1	20	
2	33	
3	24	
4	10	
5	13	
Total	**100**	

1. (20 points total) Short answer questions:
   a. (8 points) What are the two main functions of an operating system?
      i)

      (a) Coordinator and traffic cop: Resource allocation and control.
      (b) Standard services.
      4 points for each. 2 points for partial answers.

      ii)

   b. (12 points) Which of the following instructions should be allowed only in kernel mode? State whether it is or not (circle one) ***and*** why.
      i) Disable all interrupts.

      Kernel **Kernel and User**

      Why?
      Kernel only, otherwise a user could take control of the machine.
      2 points for each answer and 2 points for reason.

      ii) Read the time-of-day clock.

      Kernel **Kernel and User**

      Why?
      Kernel and User, doesn't affect anything because the operation is read-only. We deducted 2 points if the answer did not mention the read-only nature of the operation

      iii) Set the time-of-day clock.

      Kernel **Kernel and User**

      Why?

      Kernel only, could affect scheduling, other processes, and the kernel. Several students mentioned changing their machine's timezone, however, the access should still be allowed only in the kernel and use a system call.

2. (33 points total) CPU Scheduling.

a. (5 points) The CDC 6600 computers could handle up to 10 I/O processes simultaneously on a single CPU using an interesting form of round-robin scheduling called **processor sharing**. A context switch occurred after each instruction, so instruction 1 came from process 1, instruction 2 came from process 2, etc. The context switching was done by special hardware, and the overhead was zero. If a process needed *T* seconds to complete in the absence of competition, what is the maximum amount of time it would need if processor sharing were used with *N* processes? Assume there are less than 10 processes waiting to run.

It will need NT seconds.

Common minor errors (-2 points) included (N-1)T, and anything else that only changed the answer by a constant factor. All other values were given 0 points.

b. (20 points) Five batch jobs A through E, arrive at a computer center at almost the same time. They have estimated running times of A:10, B:6, C:2, D:4, and E:8 minutes. Their (externally determined) priorities are A:3, B:5, C:2, D:1, and E:4, respectively, with 5 being the highest priority. For each of the following scheduling algorithms, determine the *mean process turnaround time* (average response time). Ignore context switching overhead. **For partial credit, you should list the finishing times.**

i) Round-Robin (the timeslice size is not important, so assume a timeslice of 1 millisecond).

During the first 10 minutes, each job gets 1/5 of the CPU. At the end of 10 minutes, C finishes. During the next 8 minutes, each job gets 1/4 of the CPU, after which time D finishes. Then each of the three remaining jobs gets 1/3 of the CPU for 6 minutes, until B finishes, and so on. The finishing times for the five jobs are 10, 18, 24, 28, and 30, for an average of 22 minutes.

There were many varieties of incorrect answers for this one. The most common mistake was to compute the TAT for round-robin with a time slice of 1 minute, instead of 1 millisecond. This resulted in 3 incorrect finishing times, so 2 points of partial credit were awarded.

General part (b) grading info: If mean turnaround time was incorrect, 1 point of partial credit was awarded for each correct finishing time, up to a total of 4 points (-1 for incorrect TAT).

-1 point for grievous arithmetic errors.

-1 point per section if all finishing times were off by 1.

ii) Priority scheduling.

B is run first. After 6 minutes, it is finished. The other jobs finish at 14, 24, 26, and 30, for an average of 20 minutes.

iii) First-come, First-served (run in order 10, 6, 2, 4, 8).
If the jobs run in the order A through E, they finish at 10, 16, 18, 22, and 30, for an average of 19.2 minutes.

iv) Shortest Job First.
Finish times are 2, 6, 12, 20, and 30, for an average of 14 minutes.

c. (8 points) Five jobs are waiting to be run. Their expected running times are 9, 6, 3, 5, and X. In what order should they be run to minimize average response time? State the scheduling algorithm that should be used AND the order in which the jobs should be run. (Your answer will depend on X).

Shortest job first is the way to minimize average response time (we also accepted Shorted Remaining Time to Completion First).
0 < X <= 3: X, 3, 5, 6, 9.
3 < X <= 5: 3, X, 5, 6, 9.
5 < X <= 6: 3, 5, X, 6, 9.
6 < X <= 9: 3, 5, 6, X, 9.
X> 9: X, 3, 5, 6, 9, X.

3 points for correct scheduling algorithm.
1 points for each correct range.
For the job orderings, accepted both an explicit list of the possible orderings, as well as a general description of where to put X.
-1 point for descriptions that were judged too vague.
-1 point for each ordering omitted. This included not placing X between 5 and 6, since we don't specify that the running times must be integer values.

Answers stating that the scheduling algorithm used would change based on the value of X were graded on a case-by-case basis. In general, if a solution mentioned SJF, it received 2 points. Otherwise, 0 points were awarded.

No Credit – Problem X: (000000000000 points)

The following is an excerpt from a Wall Street Journal article:

1. A man called the Canon help desk with a problem with his printer. The tech asked him if he was running it under "Windows." The man responded, "No, my desk is next to the door. But that is a good point. The woman sitting in the cubicle next to me is under a window and her printer is working fine."
2. An AST customer was asked to send a copy of her defective diskettes. A few days later a letter arrived from the customer along with photocopies of the floppies.
3. A Dell customer called to say he couldn't get his computer to fax anything. After 40 minutes of troubleshooting, the technician discovered the man was trying to fax a piece of paper by holding it in front of the monitor screen and hitting the "send" key.
4. A Dell technician received a call from a customer who was enraged because his computer had told him he was "bad and an invalid." The tech explained that the computer's "bad command" and "invalid" response's shouldn't be taken personally.
5. A confused caller to IBM was having troubles printing documents. He told the technician that the computer had said it "couldn't find printer." The user had also tried turning the computer screen to face the printer but that his computer still couldn't "see" the printer.
6. An exasperated caller to Dell Computer Tech Support couldn't get her new Dell Computer to turn on. After ensuring the computer was plugged in, the technician asked her what happened when she pushed the power button. Her response, "I pushed and pushed on this foot pedal and nothing happens." The "foot pedal" turned out to be the computer's mouse.
7. Another customer called Compaq tech support to say his brand-new computer wouldn't work. He said he unpacked the unit, plugged it in and sat there for 20 minutes waiting for something to happen. When asked what happened when he pressed the power switch, he asked, "What power switch?"
8. True story from a Novell NetWire Sysop:
   Tech: "Hello, this Tech Support. How may I help you?"
   Caller: "The cup holder on my PC is broken and I am within my warranty period. How do I go about getting that fixed?"
   Tech: "I'm sorry, but did you say a cup holder?"
   Caller: "Yes, it's attached to the front of my computer."
   Tech: "Please excuse me. If I seem a bit stumped, it's because I am. Did you receive this as part of a promotional at a trade show? How did you get this cup holder? Does it have any trademark on it?"
   Caller: It came with my computer. I don't know anything about a promotional. It just has '4X' on it."
   At this point, the Tech Rep had to mute the caller because he couldn't stand it. He was laughing too hard. The caller had been using the load drawer of the CD-ROM drive as a cup holder and snapped it off the drive.

3. (24 points total) Concurrency problem: Dining Philosophers.
The goal of this exercise is to implement a solution to the Dining Philosophers problem using monitors instead of semaphores. Create a method `Dine()`, which waits until a diner has two chopsticks and can eat, then calls `Eat()`, and then releases the chopsticks before returning. Your solution should allow multiple philosophers to eat at the same time (as long as there are sufficient chopsticks in a pile in the middle of the table).

a. (4 points) Specify the correctness constraints. Be succinct and explicit in your answer.

1) A diner waits for two chopsticks.
2) Only one thread accesses shared state at a time
2 points for each correct constraint.

b. (20 points) Implement the `Dine()` method. Specify any state variables that you use:

```
State variables:
chopsticks     Number of chopsticks left     Initial value N

Monitor: waitingP

Lock: lock

    Dine() {
      lock.acquire();
      while (chopsticks < 2) {
        waitingP.wait()
      }
      chopsticks -= 2;
      lock.release();

      Eat();

      lock.acquire();
      chopsticks +=2;
      waitingP.broadcast(); or signal
      lock.release();
    }
```

This page intentionally left blank as scratch space.

We gave zero credit for solutions based on semaphores or that did not use condition variables.

We subtracted points for various mistakes as follows:

-3 points for unnecessary wait operations
-2 points for not releasing the lock
-8 points for code that may deadlock
-2 points for unreachable code
-4 points for too many copies of the same code
-5 points for busy waiting
-4 points for unclear usage of state variables
-6 points for not using chopsticks
-5 points for using chopsticks outside a lock
-2 points for holding the lock in Eat()
-3 points for incorrect while
-2 points for grabbing or releasing chopsticks
-2 points for signaling more than once
-3 points for signaling or waiting without holding lock
-5 points for no call to Eat()
-2 points for no signal
-3 points for signaling at the wrong time
-3 points for unnecessary lock.acquire() calls
-3 points for infinite loops
-4 points for mis-matched lock.acquire()
s
each answer and 2 points for reason. of two reasons.

4. (10 points) Deadlock:

Consider a system that starts with a total of 150 units of memory, which is then allocated to three processes as shown in the following table of processes, their maximum resource requirements, and their current allocations:

Process	Max Demand	Currently Holds
P1	70	45
P2	60	40
P3	60	15
P4	60	

Determine whether it would be safe to grant each of the following requests. If YES, give an execution order that could be guaranteed possible. If NO, show the resulting allocation table.

a. (5 points) A fourth process arrives, with a maximum memory need of 60 and an initial request for 25 units.

YES, safe. 25 units will be left after the allocation for P4, so there is sufficient memory to guarantee the termination of either P1 or P2. After that, the remaining three jobs can be completed in any order.
2 points for correct answer, 3 points for correct logic and order

b. (5 points) Using the original table above, a fourth process arrives, with a maximum memory need of 60 and an initial request for 35 units.

NO, unsafe. 15 units will be left after the allocation for P4, so there is insufficient memory to guarantee the termination of any process.
2 points for correct answer, 3 points for correct logic.
For logic, we accepted the resulting allocation table AND saying that the available resource is 15 or another CLEAR logical explanation.
-1 point if the table is present, but explanation is missing or available resources = 15 statement is missing.

5. (13 points) Deadlock:

a. (6 points) Three processes share four resource units that can be reserved and released only one at a time. Each process needs a maximum of two units. Show that a deadlock cannot occur.

Deadlock occurs if all resource units are reserved while one or more processes are waiting indefinitely for more units. But, if all four units are reserved, at least one process has acquired two units. Therefore that process will be able to complete its work and release both units, thus enabling another process to continue.

b. (7 points) Evaluate the Banker's algorithm for its usefulness in real life. Give at least two reasons to justify your choice.

It is unrealistic: don't know max demands in advance, number of processes can change over time, number of resources can change over time (something can break). Most OS's ignore deadlock.
3 points for correct answer
2 points for each reason

This page intentionally left blank as scratch space.

Do not write answers on this page

Do not write answers on this page