Name:
Login (UPI):

COMPSCI340SC & SOFTENG370SC 2005

Operating Systems Test

Tuesday 23rd August, 6:30pm – 7:30pm

- Answer all questions in the spaces provided.
- The test is out of 70 marks. Please allocate your time accordingly.
- Make sure your name is on every piece of paper that you hand in.
- When you are asked to explain something or to give reasons for something you can give your answers as a series of points. Be brief.
- The last page of the booklet may be removed and used for working.

Name:
Login (UPI):

For markers only:

Question 1	*/15*
Question 2	*/12*
Question 3	*/12*
Question 4	*/9*
Question 5	*/8*
Question 6	*/14*
Total	

Name:
Login (UPI):

Question 1 – History and development of Operating Systems (15 marks)

a) What hardware advances were necessary in order to allow spooling of data for input and output?

Interrupts, fast io e.g. disk drives.

2 marks

b) Describe two different ways in which memory can be protected, so that a memory access error in one program cannot affect the memory in another program.

Any two of: base and limit registers, relocation registers, memory partitions, virtual memory, pages.

2 marks

c) It was common for batch systems to provide the user with several queues to submit jobs to the system. What purpose did the queues have? Give an example.

Different queues signified different resource limitations. e.g. The amount of CPU time allocated to the job, the number of lines of printer output, the number of tapes required to be loaded to run ...

2 marks

d) Early versions of PDAs (Personal Digital Assistants) or pocket computers did not have memory protection. Give a reason why.

Either lack of hardware support, or a desire to keep things simple. Little memory available, so it didn't make sense to add extra features to the OS.

1 mark

Name:
Login (UPI):

e) At least two processor (CPU) execution modes are necessary for an operating system that protects memory and hardware. What are the modes called and what is the difference between them. (There are several different names for the modes, any version of the names is acceptable.)

User/program mode and

Kernel/supervisor/protected/privileged/system mode

Kernel mode provides access to all processor instructions. User mode cannot perform privileged instructions. If a privileged instruction is executed in user mode it causes an exception.

3 marks

f) Briefly describe why batch systems generally keep CPUs busier than interactive time-sharing systems do.

Batch systems are designed to use the hardware efficiently. The scheduler selects jobs according to their resource requirements and the currently available resources in the system. The idea is to always have a job ready to run. Interactive systems must keep a certain number of resource available to handle new requests from users. They cannot foretell the resource requirements of new jobs. Especially necessary for good response times.

5 marks

Name:
Login (UPI):

Question 2 – Processes and threads (12 marks)

a) Draw a typical process state diagram. Include the states: being created, ready to run, running, waiting, finishing. Label the transitions between the states and briefly describe what causes a process to move from one state to another. Each description of a transition should be no longer than two sentences.

being created -> ready to run. The process creation has been completed.

ready to run -> running. The process is selected by the scheduler to run.

running -> ready to run. The process yields or the processor is pre-empted.

running -> waiting. The process requests a resource that is currently unavailable.

running -> finishing. The process has completed (or caused some fatal error).

waiting and ready to run -> finishing. The process has been terminated, either by another process or by the operating system.

10 marks

b) Describe one way in which the suspended state is different from the waiting state.

Any of (and anything similar to):

A suspended process is waiting to be resumed. It is not waiting for some other resource.

A process is normally suspended by some other process or the user or the OS. A waiting process has requested some resource.

A suspended process may be treated differently, e.g., it may lose pages quicker than a waiting process.

2 marks

Name:
Login (UPI):

Question 3 – Scheduling (12 marks)

a) Given the following processes, arrival times and burst times, draw the Gantt chart showing schedules using the specified algorithms. In each case also calculate the average waiting time.

Process	***arrival time***	***burst time***
P1	0	5
P2	1	3
P3	5	2
P4	6	1

First come, first served, no pre-emption.

0 – p1 – 5, 5 – p2 – 8, 8 – p3 – 10, 10 – p4 – 11

average waiting time = (0+4+3+4)/4 = 11/4 = 2.75

3 marks

Shortest job first, no pre-emption.

0 – p1 – 5, 5 – p3 – 7, 7 – p4 – 8, 8 – p2 – 11

average waiting time = (0+7+0+1)/4 = 8/4 = 2

3 marks

Shortest job first, with pre-emption (if the current process has the shortest equal time then do not pre-empt it).

0 – p1 – 1, 1 – p2 – 4, 4 – p1 – 5, 5 – p3 – 7, 7 – p4 – 8, 8 – p1 – 11

average waiting time = ((0+3+3)+0+0+1)/4 = 7/4 = 1.75

3 marks

b) What major problem can be caused by using fixed priorities to schedule processes? Describe a solution to this problem.

Starvation or indefinite postponement of lower priority processes.

Increase priorities of waiting processes over time. (Known as process aging.)

3 marks

Name:
Login (UPI):

Question 4 – Concurrency (9 marks)

a) What is busy waiting and why is it not generally a good thing?

The process repeatedly checks to see if the resource is available. It consumes the CPU unnecessarily.

2 marks

b) What is a test and set instruction and what is it useful for?

It is an indivisible (atomic) instruction that retrieves the value of a variable at the same time as setting it.

It is used to provide simple locks.

2 marks

c) The following code is an attempt to get rid of busy waiting.

```
1:  lock:
2:   if (testAndSet(locked))
3:    put process onto a sleep queue;

4:  unlock:
5:   if at least one process on the sleep queue
6:    wake up first process in queue;
7:   else
8:    locked = false;
```

Describe what could go wrong with this code. Briefly explain how it could be fixed. You may refer to the line numbers.

A process could be about to put itself into the sleep queue when a context switch occurs and the process holding the resource calls unlock. In this case if there is nothing waiting on the queue, the first process will end up stuck on the queue unnecessarily.

It can be fixed by applying a lock around lines 2 and 3. This same lock is used in the unlock lines 5 and 6. This lock would be implemented as a spin lock.

5 marks

Name:
Login (UPI):

Question 5 – Interprocess communication (8 marks)

a) What is a Unix (or Linux) signal handler, what does it mean to "handle" a signal, and how does a process inform the operating system how it will "handle" a signal?

A signal handler is some code that will be executed when a process receives a signal from the operating system.

Handling a signal means executing the signal handling code.

The process makes a signal system call (or one of its derivatives) to inform the operating system how it wants to handle the signal.

3 marks

b) Suppose a Unix process is executing in user mode, and attempts to divide by zero. Describe the sequence of events (including actions by the hardware and the operating system kernel) that would be likely to occur between the attempt to divide by zero, and the signal being "handled".

The divide instruction causes an exception.

The processor jumps to the exception handling code in the kernel.

The address of the signal handler for the process is retrieved. The processor jumps to the signal handling code for the process. If the handler does not terminate the process, control returns to the process.

5 marks

Name:
Login (UPI):

Question 6 – Assignment 1 (14 marks)

a) Here is some code from a producer/consumer system using the message passing library from assignment 1.

```
class Buffer

  def initialize
    @buffer = []
  end

  def go
    loop do
      receive(
        Message.new("put") do |data|
          @buffer << data
        end,
        Message.new("get", lambda { !@buffer.empty? }) do |consumer|
          consumer.give("data", @buffer.shift)
        end,
        Message.new("stop", lambda { @buffer.empty? }) do
          Thread.exit
        end
      )
    end
  end

end
```

Describe the three message types the Buffer object can receive. In particular say whether the message would be sent by a producer or a consumer and mention what else has to be true before the message would be received. Also describe what the Buffer does when it receives the message.

"put". From a producer. Nothing else has to be true. The Buffer puts the data into the @buffer array.

"get". From a consumer. Only active if the @buffer array is not empty. In this case sends a "data" message to the consumer consisting of the first element in the @buffer array. Removes the element.

"stop". From a producer. Only active if the @buffer array is empty. Exits the Buffer thread.

more space over the page ...

Name:
Login (UPI):

6 a) continued ...

10 marks

b) Here is the sample solution `give` method for Assignment 1.

```
# Sends a message to this thread.
# Returns the number of messages waiting.

def give(*message)
  Thread.critical = true
  @received ||= []
  result = (@received << message).length
  Thread.critical = false
  self.run # start the thread up if not running
  result
end
```

Explain what the lines with Thread.critical do and explain why they are necessary.

Thread.critical = true, stops the thread scheduler.

Thread.critical = false, restarts the thread scheduler.

They are necessary to stop race conditions. Another thread could access the @received array and leave it in an inconsistent state.

4 marks

Name:
Login (UPI):

Overflow space for answers.

Name:
Login (UPI):

Overflow space for answers.

This page may be used for working.