Name:
Login (UPI):

COMPSCI340SC & SOFTENG370SC 2006

Operating Systems Test

Tuesday 22nd 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	*/14*		
Question 2	*/6*		
Question 3	*/16*		
Question 4	*/9*		
Question 5	*/7*		
Question 6	*/18*		
		Total	

Name:
Login (UPI):

Question 1 – History and development of Operating Systems (14 marks)

a) Resident monitors were the first operating systems. Describe the main services they provided.

They read and carried out job control instructions, e.g., cleared memory, loaded programs and data, jumped to the program start.

They also incorporated basic device driver code for use by all programs.

2 marks

b) Describe two changes to the hardware that were necessary before multiprogramming could be used. In both cases explain why the change was required.

More memory because several programs had to be in memory simultaneously.

Memory protection otherwise one program could clobber another.

Possibly interrupts to stop one program monopolising the processor.

4 marks

c) In systems designed to maximize the use of computer resources, the scheduler needs to know the resources a job requires before it is run. Describe two ways this can be done.

Explicitly by including lists of resources with the job itself, e.g., as part of the job control language statements.

Implicitly by putting the job into a specific queue. Different queues have different resource limits.

2 marks

Name:
Login (UPI):

d) Early microcomputer or personal computer operating systems did not require user authentication and security was non-existent. Explain why and then describe the changes that occurred in the use of these systems which required security to be taken seriously.

They were single user systems. The only information that could be tampered with belonged to the user.

One change was implementing multi-user systems on PCs; another was networking. As soon as the information for several people could be accessed by the system, authentication had to be provided and used as the basis for security.

This was made even more essential with anonymous networks, such as the Internet.

3 marks

e) Virtual machines are currently very popular, especially as servers. Give reasons for this popularity.

Each virtual machine is effectively isolated from other virtual machines. This usually means that problems occurring in one won't affect the others (or the real machine).

Many servers are designed to have control of a whole machine. The use of virtual machines reduces the number of real servers needed, hence saving money.

If some programs only run in a specific OS, you can run them from another OS via a virtual machine.

Development work on operating systems can be done more easily and safely via virtual machines.

3 marks

Question 2 – Processes and threads (6 marks)

a) What is a Process Control Block and what is its purpose?

The data structure that holds the information the operating system needs to know about the process.

Its main purpose is to keep track of the resources allocated to or used by a process. It is also used to hold the state of the process and information about the process such as its privileges and limits.

3 marks

Name:
Login (UPI):

b) What is a zombie process in Unix?

A zombie is a process that has finished but its parent process hasn't collected its completion status (using wait).

2 marks

c) How are zombies avoided?

Parent processes can wait for their children, can detach them or finish before them.

1 mark

Question 3 – Scheduling (16 marks)

a) Real-time processes are commonly referred to by a triple (c, p, d). What do the "c", "p" and "d" stand for?

"c" the compute time.

"p" the period. How long between repetitions of the process.

"d" the deadline. When the process has to be completed after it is scheduled.

3 marks

b) Given the following real-time processes calculate a cyclic executive schedule using Earliest Deadline First. If the deadlines are the same, do NOT unnecessarily pre-empt the running process. Show the schedule as a Gantt chart.
Process A (4, 8, 8) Process B (3, 6, 6)

B B B A A A | A B B B A A | A A B B B A | A A A B B B

3 marks

c) Given the following real-time processes with fixed priorities (bigger means better), calculate a cyclic executive schedule. Better priority processes can pre-empt processes with worse priorities.
Process A (4, 8, 8) – priority 10 Process B (3, 6, 6) – priority 5
Show the schedule and explain what goes wrong.

A A A A B B

By this time B has already failed to meet its deadline. The schedule is not feasible.

3 marks

Name:
Login (UPI):

d) What is priority inversion? Give an example of how it happens.

A process with a worse priority blocks a process with a higher priority. e.g., Three processes A (best priority), B (middle priority), C (worse priority). C has locked resource X. A wants X and is waiting for it, but C cannot release it because B is running.

3 marks

e) Many operating systems have a method for recalculating each task's timeslice when they have all reached zero. This is implemented as a loop over each task, such as:

```
for (each task on the system) {
    recalculate priority
    recalculate timeslice
}
```

Describe one potential problem with this approach.

As the number of tasks increases the amount of time to calculate the priorities increases. This can have unfortunate consequences for real-time processes.

2 marks

f) When a thread has been waiting for a resource in Windows and becomes runnable it usually has its priority improved. The amount of improvement varies depending on the resource the thread was waiting for. Give one example where the priority improvement is high. Explain why the priority improvement is high in this case.

Waiting on keyboard input.

The thread is interactive and we want interactive threads to be responsive.

2 marks

Name:
Login (UPI):

Question 4 – Concurrency (9 marks)

a) Brinch Hansen and Hoare’s Monitors have been implemented in many languages to provide mutually exclusive access to a resourse. Describe what a monitor is and the main benefit it has over simple locks.

A monitor is an object which only allows one thread at a time to execute its methods. Its main benefit over simple locks is that is provides the locking and unlocking automatically. It also maintaines a queue of waiting threads.

3 marks

b) Most monitor implementations provide condition variables. What are condition variables and why are they useful?

Queues for threads which must wait within a monitor.

Rather than busy wait within monitor code, the condition variable allows a thread to wait safely within the monitor. It must allow other threads to enter the monitor as they will be the ones that release the waiting thread.

3 marks

c) Signalling a condition variable can cause a waiting thread to run. What problem does this lead to? Give one solution to this problem.

There may be two threads runnable within the monitor. This is exactly what monitors must prevent.

Solutions – block the thread which signalled the monitor until the woken thread either waits or exits the monitor.

- don’t allow the signalled thread to run until the signaller has left the monitor or waited.

3 marks

Name:
Login (UPI):

Question 5 – Interprocess communication (7 marks)

a) What are Unix pipes? What are they used for.

Single direction communication channels. Sending data from one process to another.

2 marks

b) What is the major limitation of standard pipes and what causes the limitation?

Only related processes can communicate via normal pipes.

When the pipe call is made to create the pipe the pipe ends are referred to via file descriptors into the calling process' open file table. Processes forked from here then have access to the same open file table information.

3 marks

c) Given the following Ruby code, the first call to `write_end.write` succeeds but the second does not. Explain why.

```
read_end, write_end = IO.pipe
write_end.write("z")
read_end.close
write_end.write("z")
```

Writing to a pipe that can't be read from causes an exception.

2 marks

Name:
Login (UPI):

Question 6 – Assignment 1 (18 marks)

Here is some code from the `linuxThreads.c` program from assignment 1.

```
const int SWITCHES = 1000;

void non_switcher(int *counter) {
        int i;
        for (i = 0; i < SWITCHES; i++)
                *counter += 1;
}

void switcher(int *counter) {
        int i;
        for (i = 0; i < SWITCHES; i++) {
                int error;
                if ((error = sched_yield()))
                        perror("switcher :Thread switching failed.");
                *counter += 1;
        }
}
```

When these functions were called inside 1000 Pthreads from the `timer.rb` program the following data was collected. (This was on a single processor machine without hyper-threading.)

```
pthread non-switch 1000
        real:0.062 system:0.053 user:0.0 switches:219 waits:8
pthread switch 1000
        real:1.487 system:1.274 user:0.203 switches:1000624 waits:2
```

a) Explain what "real", "system", "user", "switches" and "waits" mean.

"real": The actual elapsed time from the start of the process until it completed.

"system": The time spent running in kernel mode by all threads of the process (and its children).

"user": The time spent running in user mode by all threads of the process (and its children).

"switches": The number of context switches on threads in this process caused by external factors such as clock interrupts (involuntary switches), also from sched_yield calls.

"waits": The number of context switches on threads in this process caused by a thread waiting (voluntary switches).

6 marks

Name:
Login (UPI):

b) Compare the “real”, “user”, “system” and “switches” values from the *non-switch* data with those from the *switch* data. Explain the differences.

“real”: The switch version is over 1.4 seconds slower. This is because each thread is calling “sched_yield” 1000 times.

“system”: Most of real time is taken up by running in kernel mode. The “sched_yield” causes a system call to invoke the thread scheduler. System calls execute in kernel mode.

“user”: There is a little more user time for the switch version. This can be explained by the code to make the call to “sched_yield” and the corresponding if statement.

“switches”: Over 1,000,000 more of these in the switch version. These are invoked directly by “sched_yield”. The 1000 threads calling “sched_yield” 1000 times each.

8 marks

c) Using the same operating system and the same computer but running the operating system within a virtual machine led to these results.

```
pthread non-switch 1000
        real:0.149 system:0.145 user:0.0 switches:221 waits:3
pthread switch 1000
        real:3.03 system:3.038 user:0.072 switches:1000723 waits:2
```

Compare these results with the results from the real machine and comment on the largest differences.

The only significant differences are with the system and real times. The system times are bigger because any privileged instructions cause a trap to the real operating system which then have to be handled by the virtual machine monitor.

Real times are bigger by the increase in the system times.

Name:
Login (UPI):

4 marks

Name:
Login (UPI):

Overflow space for answers.

Name:
Login (UPI):

Overflow space for answers.

This page may be used for working.