Name:
Login (UPI):

COMPSCI 340SC & SOFTENG 370SC 2009

Operating Systems Test

Wednesday 26th August, 9:05am – 9:55am

- Answer all questions in the spaces provided.
- The test is out of 60 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	*/5*		
Question 2	*/6*		
Question 3	*/6*		
Question 4	*/10*		
Question 5	*/14*		
Question 6	*/6*		
Question 7	*/13*	*Total*	

Name:
Login (UPI):

Question 1 – History and Development of Operating Systems (5 marks)

a) The earliest identifiable operating systems were resident monitors. Describe one aspect of current desktop operating systems (e.g. Linux or Windows) which **was** present in resident monitor systems.

Any one of: job control language, device drivers, separation of OS from user program etc.

2 marks

b) Describe one aspect of current desktop operating systems which **was not** present in resident monitor systems and explain why it wasn't present.

One of protected memory, virtual memory, concurrency control, etc.
Protected memory or virtual memory were not present because to do them efficiently it must be in
the hardware.
Concurrency control wasn't necessary because only one program was running at a time.
etc.

3 marks

Question 2 - Structure and Design (6 marks)

a) Windows and Linux have monolithic kernels. What is a monolithic kernel?

A kernel which has most of the different components compiled together. This means it is possible
for one part of the kernel to talk to another part directly by modifying kernel data structures.
No message passing is required.

3 marks

Name:
Login (UPI):

b) In a virtual machine environment what are virtual user and virtual kernel modes?

In a virtual machine the user mode of a process is called virtual user mode because it is running
in a guest virtual machine on the real host machine. The virtual kernel mode is actually running
at user mode level on the host but must be made to run as though it were running at kernel mode.
In reality both virtual user and virtual kernel modes are real user mode on the host.

3 marks

Question 3 – Processes (6 marks)

a) When a Unix program calls `fork()` what values get returned?

0 in the child (new) process. The process id of the child in the parent (original) process

2 marks

This C program runs on Unix.

```
int main(int argc, char** argv) {
    int i;
    for (i = 0; i < 2; i++) {
        fork();
        printf("One\n");
    }
    printf("Two\n");
}
```

b) How many times would this program print "`One`" to the display?

6

2 marks

c) How many times would this program print "`Two`" to the display?

4

2 marks

Name:
Login (UPI):

Question 4 – Scheduling (10 marks)

a) Here are the arrival and burst times for a number of processes:

Process	Arrival time	Burst time
A	0	7
B	2	4
C	3	1
D	7	4
E	8	2

From this table draw a Gantt chart showing a Shortest Job First schedule and calculate the average waiting time. If the times are the same do NOT pre-empt the running process.

```
A A B C B B B D E E D D D A A A A A
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7
```

Average waiting time = (11 + 1 + 0 + 2 + 0) / 5 = 2.8

5 marks

b) Given the following real-time processes calculate a cyclic schedule using Earliest Deadline First. If the deadlines are the same, do NOT unnecessarily pre-empt the running process. If the deadlines are the same for a number of non-running processes, choose the alphabetically lowest. e.g. If at time 9 both Process B and Process D have the same earliest deadline and neither process was running at time 8 then choose B. Show the schedule as a Gantt chart. The three numbers are Compute time, Period, and Deadline.

Process A (3, 8, 8) Process B (2, 6, 6) Process C(3, 12, 12) Process D(1, 24, 24)

```
B B A A A C C C B B A A A B B C C C A A A B B D
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3
```

5 marks

Name:
Login (UPI):

Question 5 – Concurrency (14 marks)

a) What is a spin lock?

A lock which repeatedly checks the lock variable until it becomes free.

2 marks

b) Why do we prefer not to use spin locks if possible?

They unnecessarily use processor cycles. If a resource is currently not free we should sleep until
it is available.

2 marks

c) Why do most operating systems include spin locks? Describe a possible case for when a spin lock is necessary?

Many kernel data structures are only held for very short periods of time and spin locks,
especially on multiprocessors are the most efficient way of controlling access to these.
On most multiprocessor hardware they are necessary to implement normal waiting locks.

4 marks

CONTINUED

Name:
Login (UPI):

You are given an **atomic** function called `Exchange` which works like this:

```
exchange(variableOne, variableTwo)
```

This **atomically** swaps the values of `variableOne` with `variableTwo`. So that afer calling the function `variableOne` holds the value that was in `variableTwo` and `variableTwo` holds the value that was in `variableOne`.

d) Complete the following pseudocode spin lock using the `exchange` atomic operation.

```
boolean lockVariable  // this is the lock variable
                      // if it is true it means the lock is busy

spinlock(lockVariable):    // write your spinlock here
     myLockVariable = true;
     while myLockVariable
          Exchange(lockVariable, myLockVariable)
     end
```

4 marks

e) Also write the equivalent unlock.

```
lockVariable = false;

Also acceptable
     mylockVariable = false;
     Exchange(lockVariable, myLockVariable)
```

2 marks

Name:
Login (UPI):

Question 6 - Dining Philosophers (6 marks)

Here is some code from an attempted C solution to the Dining Philosophers' problem.

```
typedef struct philos {
    char *status;
    pthread_mutex_t *right, *left;
} Philosopher;

// This code is run in a separate thread for each philosopher.
void *philosopherRun(void *phil) {
    Philosopher *philosopher = phil;
    while (true) {
        philosopher->status = "thinking";
        philosopher->status = "waiting";
        bool both = false;
        do {
            if (pthread_mutex_lock(philosopher->left) != 0)
                fprintf(stderr, "Left lock error.\n");
            int right_lock_result = pthread_mutex_trylock(philosopher->right);
            switch (right_lock_result) {
                case 0:
                    both = true;
                    break;
                case EBUSY:
                    pthread_mutex_unlock(philosopher->left); // try again
                    both = false;
                    break;
                default:
                    fprintf(stderr, "Right lock error\n.");
            }
        } while (!both);
        printf("."); // this won't appear if blocked
        philosopher->status = "eating";
        sleep(1);
        pthread_mutex_unlock(philosopher->left);
        pthread_mutex_unlock(philosopher->right);
    }
}
```

a) What does the constant `EBUSY` represent?

The fact that the lock is currently being used. i.e. it is locked.

1 mark

b) If the right fork of a philosopher is being used when this code runs describe what happens.

The philospher tries to lock the right fork and finds it is already locked so the philosopher
releases the left fork and goes back to try again.

3 marks

Name:
Login (UPI):

c) Theoretically this solution is not a perfect solution to the Dining Philosophers' problem. Explain why not.

It is possible that everytime a philosopher tries to collect the right fork it is being used again.
In that case it might consistently be released and then acquired while the philosopher is
waiting for the right fork.

2 marks

Question 7 – Assignment 1 (13 marks)

a) In the Windows Research Kernel (WRK) what is a `KPROCESS` structure, and what does the `K` in `KPROCESS` stand for?

It holds information about a process, similar to a PCB. The K stands for kernel.

2 marks

b) In assignment 1 you had to follow the state transitions of one thread. What process was that thread created by? i.e. The name of the process the followed thread was part of.

explorer.exe

1 mark

c) Briefly explain how you could find the process identified in part b) as the WRK was running.

Each process has a file name associated with it. This is the ImageFileName field. Each new
process was checked until the one with the ImageFileName "explorer.exe" was created.
This was the explorer process.

3 marks

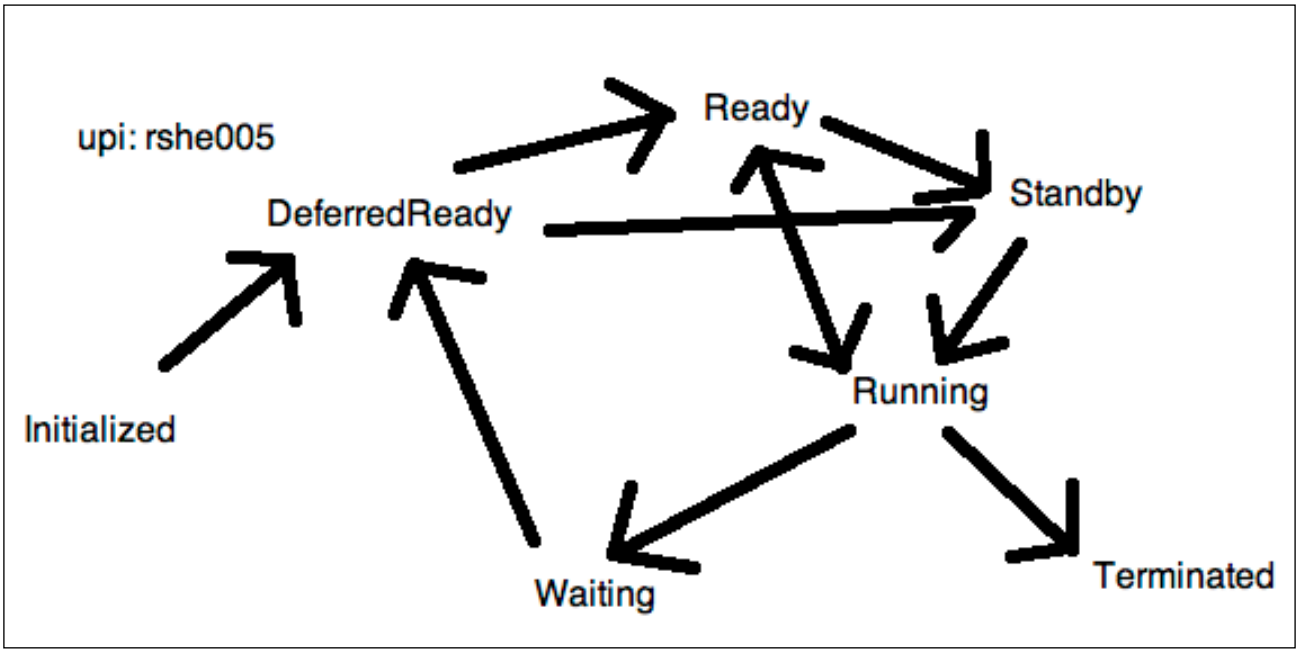
d) Briefly explain how you could find a thread associated with the process identified in part b) as the WRK was running. This thread was referred to as `ChosenPKThread` in the assignment.

When a thread was given the Initialized state we compared the creating process against the
ExplorerProcess. If ChosenPKThread was still null and the creating process was the
ExplorerProcess we had the required thread.

2 marks

Name:
Login (UPI):

e) Given the following Windows NT thread states draw the state transition diagram as in the assignment: Initialized, DeferredReady, Ready, Standby, Running, Waiting, Terminated.



5 marks

Name:
Login (UPI):

Overflow space for answers.

Name:
Login (UPI):

Overflow space for answers.

This page may be used for working.