Name:
Login (UPI):

COMPSCI 340SC & SOFTENG 370SC 2011

Operating Systems Test

Monday 22nd August, 12:10pm – 12:55pm

- Answer all questions in the spaces provided.
- The test is out of 45 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	*/6*
Question 2	*/5*
Question 3	*/10*
Question 4	*/8*
Question 5	*/4*
Question 6	*/12*
Total	

Name:
Login (UPI):

Question 1 – History and Development of Operating Systems (6 marks)

a) Very early computers did not provide interrupt handling capabilities. What advantage does an interrupt handling system provide?

The ability to continue processing while waiting for some event, usually IO, to occur.

This also opens up the possibility of preemptive multi-tasking because a clock can

provide interrupts.

2 marks

b) Most computers use a stack to store return information while handling interrupts. Some early computers used a single memory location to store the value of the saved program counter. What advantage is there in the use of a stack for this information?

Another interrupt can be handled during the handling of the first interrupt.

This is particularly important if there are multiple interrupt sources.

2 marks

c) The first personal computer operating systems (including the Mac) were effectively resident monitors. Describe one way in which early PC operating systems were like resident monitors.

No memory protection.

Single task execution.

Standard IO routines.

2 marks

Question 2 - Design and Implementation (5 marks)

a) Give reasons why a language such as Java is seldom used to implement operating systems?

Runs on a Java Virtual Machine rather than directly on the hardware.

Does not allow programmers access to memory locations.

Large run-time requirements etc.

Some people will say speed. That is not entirely true, but I will accept it.

2 marks

b) Describe *application virtualization* and how it differs from traditional virtual machine technology such as VMWare player and IBM's VM operating system.

An application runs on a layer which provides the resources the application needs to run, even though it may be running on a different OS.

It differs from a traditional VM in that it does not allow (or require) the running of a guest operating system. It merely provides the functions required by the application.

3 marks

Question 3 – Processes (10 marks)

a) Explain the difference between a program and a process?

A process is a running program. There can be multiple processes created from the same program.

2 marks

b) Explain the difference between a thread and a process?

A thread is a sequence of instructions. There can be multiple threads within a process. Every process needs to have at least one thread in order to run. A process is sometimes thought of as the resources associated with a running program. A thread is one stream of instructions being performed by the process.

2 marks

This C program runs on Unix.

```
#include <stdio.h>

int main(int argc, char** argv) {
    int i;
    for (i = 0; i < 1; i++) {
        printf("One\n");
        fork();
        printf("Two\n");
        if (fork() == 0)
            printf("Three\n");
    }
}
```

Name:
Login (UPI):

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

1

2 marks

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

2

2 marks

e) How many times would this program print "`Three`" to the display?

2

2 marks

Question 4 – Scheduling (8 marks)

a) Here are the burst times (in milliseconds) for a number of processes:

Process	Burst time
A	7
B	5
C	4
D	1
E	2

From this table draw a Gantt chart showing a round-robin schedule with a time slice of 4 milliseconds and calculate the average waiting time. All of the processes are ready to run at the start of the schedule and they are originally scheduled in alphabetical order.

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

(11 + (4 + 10) + 8 + 12 + 13) / 5 = 58/5 = 11.6

4 marks

Name:
Login (UPI):

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 8 both Process B and Process C have the same deadline and neither process was running at time 7 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)

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

4 marks

Question 5 – Low-level C (4 marks)

a) What error do you commonly get if you try to access a pointer which has not been initialised correctly in Linux?

Segmentation fault

1 mark

b) When the following C program was run in Linux it produced the output shown:

```
#include <stdlib.h>
#include <stdio.h>

int *local_pointer(void) {
    int x = 6;
    return &x;
}

void add(void) {
    int a;
    a = 4;
    a = a + 1;
}

int main(int argc, char** argv) {
    int *result;
    result = local_pointer();
    printf("int is %d\n", *result);
    add();
    printf("int is %d\n", *result);
    return EXIT_SUCCESS;
}
```

Output:

```
int is 6
int is 5
```

Explain why this happened.

The “x” int is allocated on the stack. The returned pointer points to this position on the stack. The second function call then reuses the same position.

3 marks

Question 6 – Assignment 1 (12 marks)

a) Given the task and dispatch queue functions from Assignment 1 what would be the normally expected output from the following program?

```
#include "dispatchQueue.h"
#include <stdio.h>
#include <stdlib.h>

void run() {
    sleep(1);
    printf("The task is running.\n");
}

int main(int argc, char** argv) {
    dispatch_queue_t *concurrent_dispatch_queue;
    task_t *task;
    concurrent_dispatch_queue = dispatch_queue_create(CONCURRENT);
    task = task_create(run, NULL, "run");
    dispatch_async(concurrent_dispatch_queue, task);
    printf("Dispatched the task\n");
    dispatch_queue_destroy(concurrent_dispatch_queue);
    return EXIT_SUCCESS;
}
```

Dispatched the task.

2 marks

b) It is possible, though unlikely, that the output could be different. What else could the output be and how is this possible?

A variation of: Dispatched the task/The task is running.

If the main thread gets delayed for at least a second, the run task could get to the print statement before the program finishes.

3 marks

c) Concurrent dispatch queues were created with a number of threads to execute tasks. Give a reason why the number of threads was chosen to be the same as the number of processors or cores on the machine the program was executed on?

In order to allow as many tasks as possible from one concurrent dispatch queue to execute simultaneously. For this to happen there has to be at least as many worker threads as there are processors.

Having no more threads than processors reduces scheduling costs. But of course we really would need information about all threads currently active in the system.

3 marks

d) A serial dispatch queue only requires one thread to execute its tasks. When would using a serial dispatch queue be preferable to using a concurrent one?

Whenever the tasks modify a shared data structure. By putting all of these tasks in a serial dispatch queue there is no need to lock the data structure as only one task will modify the data at a time.

2 marks

e) On a machine which reports it has 4 cores, if a program uses two serial dispatch queues and two concurrent dispatch queues how many threads would be created to execute tasks? And how many tasks could actually be running simultaneously?

Name:
Login (UPI):

10 threads = 2 x 1 + 2 x 4
4 tasks can run simultaneously.

2 marks

Name:
Login (UPI):

Overflow space for answers.

Name:
Login (UPI):

Overflow space for answers.

This page may be used for working.