

Module Name: Operating Systems
Module Number: CE01000-3
Title of Assessment: Portfolio of exercises

Module Learning Outcomes for This Assessment

3. Apply to the solution of a range of problems, the fundamental concepts, principles and algorithms employed in the operation of a multi-user/multi-tasking operating systems.	

Hand in deadline: Friday 26th November, 2010.

Assessment description

Selected exercises from the weekly tutorial/practical class exercises are to be included in a portfolio of exercises to be submitted Faculty Office in a folder by the end of week 12 (Friday 26th November 2010).

The weekly tutorial/practical exercises that are to be included in the portfolio are specified below.

What you are required to do.

Write up the selected weekly exercises in Word or other suitable word processor.

The answers you submit should be your own work and not the work of any other person.

Note the University policy on **Plagiarism and Academic Dishonesty** - see *Breaches in Assessment Regulations: Academic Dishonesty* - at <http://www.staffs.ac.uk/current/regulations/academic/index.php>

Marks

The marks associated with each selected exercise will be indicated when you are told which exercises are being selected.

Week 1.

Tutorial questions:

6. How does the distinction between supervisor mode and user mode function as a rudimentary form of protection (security) system? (3 marks)
7. What are the differences between a software interrupt (trap) and a hardware interrupt? What is the use of each? (3 marks)

Week 2.

Tutorial question:

4. What is the purpose of system calls? Why are they necessary? (3 marks)

Week 3.

Virtual machine exercise:

16) Writing a short Shell script

Use man to find out what the head command does.

Using an editor (can be the simple editor you used in 1), write a short (two or four line) shell script which applies sort and head to a test file and will give a sorted list of the first five lines in the file.

It is a good idea to begin your script with a comment that explains what your script does. Comment lines begin with a hash (#). (You might have to experiment with the keyboard to find out which key produces the # on screen. For example, it might be the '£' key if you have got a British keyboard.)

Remember that you will have to make your script executable, using the chmod command (come out of the editor first).

Run your script by typing its name at the prompt. If it doesn't work, go back into the editor and modify it until it does.

(4 marks)

Tutorial question:

- 12) The ability of a process to create a new process is an important capability, but it has dangers. Consider the consequences of allowing a user to run the process below. Assume that fork() is a system call that creates a new child process.

```
int main() {  
  
    while (true) {
```

```

        fork();
    }
}

```

- a) What would the consequences of allowing such a process to run? (2 marks)
- b) Suppose that you are an operating systems designer and have been asked to build in safeguards against such processes. Assume that you have decided that it is inappropriate to reject certain processes, and that the best approach is to place certain runtime controls on them. What controls might the operating system use to detect processes like the above at runtime? (4 marks)

Week 4.

Tutorial questions:

- 6) Consider the following set of processes, with the length of the CPU-burst time given in milliseconds:

Process	Burst Time	Priority
P1	4	2
P2	1	1
P3	3	3
P4	1	4
P5	7	3

At time 0, the processes are queued in the order of arrival P1, P2, P3, P4, P5.

- a) Draw four Gantt charts illustrating the execution of these processes using
- FCFS
 - SJF (nonpreemptive)
 - nonpreemptive Priority (with a smaller priority number implying a higher priority)
 - RR (quantum = 1) scheduling
- (8 marks)
- b) What is the turnaround time of each process for each of the scheduling algorithms in part a)?
- (8 marks)
- c) What is the waiting time of each process for each of the scheduling algorithms in part a)?
- (8 marks)
- d) Which of the schedules in part a results in the minimal average waiting time (over all processes)?
- (2 marks)

9) Determining the quantum is a complex and critical task. Assume that the average context-switching time between processes is s , and the average amount of time an I/O bound process uses before generating an I/O request is t ($t \gg s$). Discuss the effect of each of the following quantum settings, q .

- a) q is slightly greater than zero
- b) $q = s$
- c) $s < q < t$
- d) $q = t$
- e) $q > t$
- f) q is an extremely large number

(12 marks)

Week 5.

Tutorial question:

8) Give an example of a simple resource deadlock involving 3 processes and 3 resources. Draw the Resource Allocation Graph that illustrates this.

(7 marks)

Virtual machine exercises with Linux processes.

17) Use your `forktest.c` program as the basis, but modify it by deleting the `printf`s that occur under the `if` and the `else` that output the PIDs and replace them with code for the parent to output the series of odd (in French 'impair') numbers in an infinite loop (e.g. 1, 3, 5, 7, 9,...) and for the child process to output the series of even (in French 'pair') numbers in an infinite loop (e.g. 2, 4, 6, 8, 10,...). You could do this by using a `for` loop under the relevant branch of the `if..else` statement.

(6 marks)

23) Up till now the parent process has continued to execute while the child process has executed. We can use the `wait()` system call to make the parent process suspend itself and wait until it is woken up by the termination of the child process.

So modify `forktest4.c` to make the parent process wait for the child. You will need to use the `wait()` system call.

You will need to add to the `#include` list

```
#include <sys/wait.h>
```

The wait() system call takes a pointer to an int as a parameter. It is the location where the exit status of the child will be placed when it terminates. So you will need to add an int declaration,

```
int status;
```

to your code. Then you can call wait() as

```
wait(&status);
```

This needs to go in the code that is to be executed by the parent process.

The & in the parameter gives the address of the variable status to wait() rather than the value.

(5 marks)

Week 6.

Tutorial question:

2. One method of deadlock prevention by denying the ‘wait for’ condition requires that processes must request all of the resources they will need before the system will allow them to compete for resources. The system grants resources on an ‘all or none’ basis. Discuss the advantages and disadvantages of this method.

(4 marks)

Virtual machine exercises – simple windows system programming.

14) Now we are in a position to modify the program above. We are going to change it so that now instead of copying a file from one file to another, the code will copy a file but output it to the console window. So it is a little like the DOS Type command.

To do this you need to know about one further function.

GetStdHandle – this returns a HANDLE and takes as an argument nStdHandle a DWORD that is the numerical value of the standard input/output stream. The permissible values are:

STD_INPUT_HANDLE – for keyboard input

STD_OUTPUT_HANDLE – for console output

STD_ERROR_HANDLE – error output (normally the console)

These identify which of the standard I/O values you need to use. You can now modify the code to send the file contents to the console screen.

a) Use WriteFile with the appropriate HANDLE you now get from GetStdHandle.

b) NOTE – YOU NEED TO DELETE (OR COMMENT OUT) THE CODE THAT CLOSES THE hOut handle. This is because it will close the console for output and so you won't get ANY further output to the console output.

c) You will have to change the number of arguments in the if statement that guards the usage prompt as the program will now be invoked with only one parameter, as

FileCopy <file to be copied to screen>.

[10 marks for code]

Week 7.

Tutorial questions

4) Given memory partitions of 200K, 450K, 100K, 700K, and 300K (in order), how would each of the First-fit, Best-fit, and Worst-fit algorithms place processes of 212K, 417K, 112K, and 426K (in order)? Which algorithm makes the most efficient use of memory?

Remember that space left over after a process has been placed in a partition will then become part of the free space available for later allocations.

(12 marks)

Virtual machine exercise:

Linux interprocess communication.

14) Use pipes to communicate between the parent and child process.

Remember the parent process repeatedly writes the alphabet to the pipe using an infinite loop and the child process repeatedly reads from the pipe and actually outputting the characters to the screen.

[10 marks for code]

Week 8.

Tutorial question.

4) Consider the following page reference string:

1, 2, 3, 2, 4, 2, 5, 2, 3, 6, 2, 5, 1, 2, 3

How many page faults would occur for the following replacement algorithms, assuming three and four frames? Remember all frames are initially empty, so your first unique pages will all cost one fault each.

- LRU replacement
- FIFO replacement
- Optimal replacement

(6 marks)

Week 9.

Tutorial questions.

3. Assume a system uses 2K blocks and 16 bit (2 byte) addresses. What is the largest file size that such a system can support using indexed block file allocation with a single level index. Comment on the resulting maximum size in terms of its adequacy for a modern filing system.

(2 marks)

9. What problems might arise when a file is deleted, if it is shared?

(3 marks)

10. How can we solve this problem?

(2 marks)

Week 10.

1) Suppose that a disk drive has 5000 cylinders, numbered 0 to 4999. The drive is currently serving a request at cylinder 163, and the previous request was at cylinder 115. The queue of pending requests, in FIFO order, is 186, 2460, 513, 1764, 942, 509, 1122, 1250, 1260.

Starting from the current head position, what is the total distance (in cylinders) that the disk arm moves to satisfy all the pending requests, for each of the following disk scheduling algorithms?

(6 marks)

- a. FCFS
- b. SCAN