NATIONAL UNIVERSITY OF SINGAPORE

CS2106 – Operating Systems

(Semester 2: AY2016/17)

Time Allowed: 2 Hours

INSTRUCTIONS TO CANDIDATES

1. This assessment paper consists of **FOUR (4)** questions and comprises **EIGHTEEN (18)** printed pages.
2. This is an **OPEN BOOK** assessment. All printed materials are allowed.
3. Answer all questions and write your answers in this assessment paper.
4. Fill in your Student Number with a pen clearly on the front cover and at the top of each page.
5. You may use pen or pencil to write your answers.
6. You are to submit only the **THIS ASSESSMENT PAPER** and no other document.

STUDENT NUMBER (fill in with a pen):

For examiner's use only		
Question	***Total***	***Marks***
Q1	35	
Q2	20	
Q3	25	
Q4	20	
Total	**100**	

Question 1 (35 MARKS)

We have a simplified version of the LINUX non real-time round-robin queues, with 5 priority levels. The table below shows the priority levels, and the time quantum and processes queued for execution at each priority level. As per the lecture the processes are picked off the active set, run for the time quantum given, and moved to the expired set, etc.

Priority Level	Time Quantum	Processes in Queue
0	75ms	0, 1
1	60ms	
2	40ms	2
3	30ms	3, 4,5
4	15ms	6

Our system has just started up, and process 0 is about to be executed. In addition, every process is not expected to terminate at any point in time, and processes are executed in a round-robin fashion. We assume that all processes are not blocked and are able to run when their turn comes. No new processes will be added, and each process will use its allotted quantum fully.

a. Calculate the expected waiting times (i.e. time a process has to wait before starting execution) for each of the processes in the diagram above, assuming that Process 0 is about to start execution. (7 marks)

Process ID	Expected waiting time
0	
1	
2	
3	
4	
5	
6	

b. Suppose Process ID 5 has its priority level changed by the following renice command (i.e. change process ID's priority by -2 levels). Note: "sudo" means to execute the renice as superuser, which allows you to promote a process's priority level.

```
sudo renice -2 -p 5
```

Recalculate the expected waiting times for each of the processes, assuming once again that Process 0 is about to start. (7 marks)

Process ID	Expected waiting time
0	
1	
2	
3	
4	
5	
6	

c. Continuing on with our discussion about expected waiting times of processes in our non real-time round-robin queues:

Suppose that each priority level p_i has n_i processes in the queue and each process has time quantum q_i. Smaller priority numbers have higher priorities, and thus priority levels 0 to *i-1* have higher priority (and larger time quanta) than processes at priority level *i*.

Find a general expression to estimate the waiting time $D_{i,j}$ of a process *j* at priority level *i*, assuming that every higher priority process is able to run.

(12 marks)

d. Your esteemed colleague Bob Sacamano said that $D_{i,j}$ can be used to reliably test whether a particular process j with priority level i will meet its deadline, given by a fixed interval $d_{i,j}$ ms. I.e. this process must complete a particular objective every $d_{i,j}$ ms. Your other colleague Jackie Chiles on the other hand calls the idea "Outrageous, egregious, PREPOSTEROUS!" (i.e., he thinks that Bob is an idiot). Who do you agree with, and why? (9 marks)

Question 2 (20 MARKS)

"Throughput", which is the rate of data transfer in (mega)bytes per second, is an important measure of drive performance.

Suppose we have a disk drive that's connected to a 12 MBPS (megabyte per second) data bus.

The drive has the following characteristics:

Rotation Speed	7200 rpm
Number of Platters	3
Tracks per Side	16
Blocks per Track	16
Bytes per Block	4096
Track to Track Seek Time	1.5 ms
Average Seek Time	12 ms
Full Sweep Seek Time	22 ms
Block Numbering Sequence	Continuous across tracks in a cylinder. Continuous across cylinders.

a. What is the peak throughput achievable on this drive/bus combination? (2 marks)

In all of the sub-parts below, assume that the drive head is always already positioned over the first block to be read, and that there is no seek nor rotational latency to read the first block.

IMPORTANT:

"Sustained throughput" is the rate of data transfer in one second including all latencies. For example if it takes 0.5 seconds to transfer 3KB of data, the sustained throughput is 3KB/0.5 = 6KB per second.

b. What is the sustained throughput for reading 32 consecutive blocks, with no track skewing? (3 marks)

c. What is the sustained throughput for reading 120 consecutive blocks, with no track skewing? (6 marks)

d. What is the sustained throughput for reading 120 consecutive blocks, with track skewing? (4 marks)

e. Assuming that the 120 blocks are scattered all over the disk (and that the drive head is already positioned over the first block to be read), what is the sustained throughput to read these blocks? (5 marks)

Question 3 (25 MARKS)

We have a FAT16 file system, and FAT entries may have special values of 65535 to indicate a free cluster, 65534 to indicate EOF, and 65533 to indicate a bad cluster.

a. Find the largest partition size possible for cluster sizes of 1KB, 8KB, 16KB and 32KB. (3 marks)

Cluster Size	Maximum Partition Size
1KB	
8KB	
16KB	
32KB	

b. Name and explain one advantage and one disadvantage of large cluster sizes. (2 marks)

The table below shows the ranges of free clusters in this FAT system (note: this table is NOT a FAT, but only shows which FAT entries have values of 65535)

Free Clusters	3 to 4, 92 to 94, 475 to 478, 54000 to 65532

c. MS-DOS allocates free clusters on a "first available" basis; I.e. it will allocate the first free cluster, then the next free cluster, and the free cluster after that, etc. In such a scheme, what clusters would be allocated if we created a new file that uses 10 clusters? (4 marks)

d. Since MS-DOS allocates on a "first available" basis, there should not be small gaps of free clusters at the start of the disk. Give one explanation why such gaps still exist. (2 marks)

In the subsequent parts we will assume the same disk system in Question 2, spinning at 7200 rpm and a track-to-track seek time of 1.5 ms, and average seek time of 22 ms. However we will assume that the data transfer rate is 10 megabytes / second. We also assume that the drive head is initially at a random location, and that clusters 3 to 4 are adjacent to each other, clusters 92 to 94 are adjacent to each other, and clusters 475 to 478 are also adjacent to each other.

e. Assume that we have 8KB clusters, what is the total transfer time if we wanted to read the 10 clusters in part c? (5 marks)

f. We now modify MS-DOS's free-cluster allocation to instead allocate from the clusters from the part of the disk that is completely free (54000 onwards in our case). Which clusters will be allocated to our new 10 cluster file? (2 marks)

g. Repeat your calculation for part e, again assuming 8KB clusters. Assume that cluster 54000 occurs right at the start of a cylinder. (5 marks)

h. Based on your results, explain why fragmentation is bad, and give one possible explanation for how "fragmentation resistant" file systems work. (2 marks)

Question 4 (20 MARKS)

a. Disk blocks that are allocated to a file do not have to be contiguous, but memory units allocated in a single request must be contiguous. Explain why. (3 marks)

The base and limit register values (in decimal) for processes 0 to 3 are shown below:

Process ID	Base Register Value	Limit Register Value
0	4096	3128
1	8192	4524
2	16384	1176
3	32768	2516

b. Find the physical addresses (in decimal) for each of the memory accesses below, or indicate "N/A" if the access will result in a violation. (5 marks)

Process ID	Memory Operation	Address of jump/branch target, or address of load/store, or N/A
2	// Jump to location given by // (PC+4) + (4*56). PC is currently // 16404 jmp 56 ;	

1	// Load word from address given by
	// Contents of register $4 + (4 * 128)
	// $4 = 12216
	lw $1, 128($4)	

0	// Branch to (PC+4) +(4* -128) if
	// $1 and $2 are equal
	// PC is currently 4612
	beq $1, $2, -128	

3	// Store word in $1 to address given by
	// contents of $4 + (4*64). $4 is
	// currently 2100
	sw $1, 64($4)	

2	// Store word in $2 to address given
	// contents of $4 + (4 * -128). $4 is
	// 16888
	lw $2, -128($4)	

The "fork" call taught in lectures does not actually make a new copy of a process; instead both parent and child will use the SAME copy of the process, with one exception: When a parent or a child tries to modify a variable, the section of memory containing that variable is duplicated, and the duplicate is modified, allowing the other process to retain the original value. This is called "copy-on-write" or COW.

Note also that in general each process has its own page table.

c. Explain, with diagrams, how virtual memory makes it possible to "duplicate" a parent's memory space without actually having to use new areas of memory, nor actually copying anything. (6 marks)

d. Explain again, with diagrams, how COW can be implemented using virtual memory. (6 marks)

~ END OF PAPER ~