Université d'Ottawa
Faculté de genie

École d'ingénierie et
de technologie de l'information

uOttawa
L'Université canadienne
Canada's university

University of Ottawa
Faculty of Engineering

School of Information
Technology and Engineering

CSI 3131
Operating Systems

FINAL EXAM

Length of Examination: 3 hours **April 29, 2010, 14:00**

Professor: Gilbert Arbez

Name: _____

Student Number: _____

Closed book, no scientific calculators or other electronic devices (except simple calculators) are allowed.

If you do not understand a question, clearly state an assumption and proceed.
At the end of the exam, when time is up:

- Stop working and turn your exam upside down.
- Remain silent.
- Do not move or speak until all exams have been picked up, and a TA or the Professor gives the go-ahead to leave.

The exam consists of three (3) parts

Part 1	**Short Answer Questions**	**10 points**
Part 2	**Theory Questions**	**20 points**
Part 3	**Problem Solving Questions**	**30 points**
Total		**60 points**

Number of pages: 14

Short Answer Questions – (5 X 2 points = 10 points total)

Answer the following questions in the provided space. Answer with a few words or a single phrase.

1) What type of variable are used in "monitors" to support synchronization of processes/threads?

2) What file allocation method is used in the FAT file system?

3) Name at least 2 places in the OS where FCFS (first come first served) algorithm can be used.

4) Give at least one advantage of virtual memory management over the first memory management schemes?

5) How does a UNIX operating system allow communication between two processes running on different physical computers?

Part 2 – Theory Questions (20 points)

1. (5 points) Complex disk scheduling algorithms have better performance than simpler algorithms, but performance difference varies with different request loads. Associate each of the following algorithms to one (and only one) the following load conditions: FCFS (First-Come First Serve), SSTF (Shortest Seek Time First), C-LOOK (look version of the circular scan). Explain your answer.

   Lightly loaded disk (0-3 requests in the scheduling queue):

   Medium loaded disk (0-20 requests in the scheduling queue):

   Heavily loaded disk (30-50 requests in the scheduling queue):

2. (5 points) The page fault rate (page faults per second) can be used as part of a resident set management policy. Explain the circumstances under which such a policy will suspended/swapped out a process?

3. (5 points) Explain how the TLB can be used to increase the performance of the OS?

4. (5 points) Paging memory management can be improved with page buffering; explain how page buffering is applied and how it improves the memory management?

Part 3 – Problem solving questions (30 points)

Question 1 - Deadlock Avoidance (5 points)

The following shows the state of 5 processes (P0 to P4) sharing 3 resources (R0 to R2).

$$\text{Max} = \begin{array}{c} \\ P0 \\ P1 \\ P2 \\ P3 \\ P4 \end{array} \begin{array}{c} \begin{array}{ccc} R0 & R1 & R2 \end{array} \\ \begin{pmatrix} 0 & 0 & 1 \\ 2 & 7 & 5 \\ 6 & 6 & 5 \\ 4 & 3 & 5 \\ 0 & 6 & 5 \end{pmatrix} \end{array} \qquad \text{Allocated} = \begin{pmatrix} 0 & 0 & 1 \\ 2 & 0 & 0 \\ 0 & 0 & 3 \\ 2 & 3 & 5 \\ 0 & 3 & 3 \end{pmatrix} \qquad \text{Available} = \{ 2, 1, 0\}$$

a. What is the total number of instances for each type of resource?

b. Complete the Need matrix (i.e. show the maximum amount of resources each process might request).

$$\text{Need} = \begin{pmatrix} & & \\ & & \\ & & \\ & & \\ & & \end{pmatrix}$$

c. Show that the system is in a safe state by completing the following table (i.e. apply the safety algorithm).

Process	Need	Allocated	Work	Finished

d. Can a request from P2 for the resources (0, 1, 0) be granted? Show how you arrived at your answer by applying the Banker's algorithm (please show all your steps). Indicate how the OS will treat the requesting process P2.

Available = { , , }

Max = $\begin{array}{c} \\ P0 \\ P1 \\ P2 \\ P3 \\ P4 \end{array} \begin{array}{c} \begin{array}{ccc} R0 & R1 & R2 \end{array} \\ \left\{\begin{array}{ccc} 0 & 0 & 1 \\ 2 & 7 & 5 \\ 6 & 6 & 5 \\ 4 & 3 & 5 \\ 0 & 6 & 5 \end{array}\right. \end{array}$ Allocated = $\left\{\begin{array}{c} \\ \\ \\ \\ \\ \end{array}\right.$ Need = $\left\{\begin{array}{c} \\ \\ \\ \\ \\ \end{array}\right.$

Process	Need	Allocated	Work	Finished

Question 2 – Memory Management (10 points total)

(a) Segmentation with Paging (5 points)

There is one process in a system that currently uses 3 segments; they contain the following (this is a simplified example; the strings represent variable size content, i.e. variable size segments):

Segment	Content
0	Pointers_are_great_but_quite_a_challenge.
1	Data_can_exist_on_the_stack_or_heap_or_data_segment.
2	Stack_and_heap_are_important.

The "simplified" system uses an 8-bit logical address with paged segments. Each process can have at most 4 segments, each paged with up to 4 pages; a page contains 16 bytes. Thus segments can be at most 64 bytes long (4 pages x 16 bytes) with the process at most 256 bytes long (4 segments x 64 bytes). The content of the physical memory is as follows (ignore the fact that there is no space for the OS itself and other processes):

Frame #	Frame content	Frame Description
0 (0000)	Pointers_are_gre	Data/code
1 (0001)	< , , , >	Segment table
2 (0010)	< , , , >	Page table of segment 0
3 (0011)	Old_data_here	Empty
4 (0100)	< , , , >	Page table of segment 2
5 (0101)	Data_can_exist_o	Data/code
6 (0110)	< , , , >	Page table of segment 1
7 (0111)	eap_or_data_segm	Data/code
8 (1000)	at_but_quite_a_c	Data/code
9 (1001)	hallenge.	Data/code
10 (1010)	n_the_stack_or_h	Data/code
11 (1011)	ent.	Data/code
12 (1100)	xxxx_xxxx_xxxx	Empty
13 (1101)	Stack_and_heap_a	Data/code
14 (1110)	aaaa_bbbb_cccc	Empty
15 (1111)	re_important.	Data/code

i. **(2 points)** In the table above, fill in the empty fields in the segment table and in the page tables of the segments. Use "-1" to indicate an invalid segment/page (only 4 entries exist in each table).

ii. **(1 point)** The logical address is 8 bits long as is the physical address. Give the number of bits for the following:

Offset into the frame/page:

Frame number:

Segment number:

Page number:

iii. **(2 points)** Fill in the last two columns of the following table:

Logical address	**Corresponding physical address (or access violation)**	**Data/code byte at this address**
00000101		
01110001		
10111110		
01101101		

(b) Virtual Memory Management (5 points)

The following table shows how 4 physical frames allocated to a process have been loaded with its logical pages (at time 164). All frame/page values are numbered using decimal values and starting at 0. All timing values correspond to the number of clock ticks from the start of the process execution. The U bit is the use bit that is set whenever the loaded page is referenced.

Page Frame Number	Virtual Page Number	Time loaded	Time Referenced	U Bit
0	2	60	161	0
1	1	130	160	1
2	0	26	162	1
3	3	20	163	1

For each page replacement algorithm below, circle the virtual page numbers that cause a page fault in the given sequence of virtual page accesses. Under each page number that causes a page fault, show the page replacement by giving the new list of loaded virtual pages (i.e. the new contents of the second column of the above table). It is NOT necessary to show the contents of the list for the virtual pages that do not cause a page fault.

Optimal 4 0 0 0 2 4 2 1 0 3 2

Clock 4 0 0 0 2 4 2 1 0 3 2

Assume that a pointer is initially set to Page Frame 2 at time 164.

LRU 4 0 0 0 2 4 2 1 0 3 2

Question 3 - Semaphores (5 points)

Develop the synchronization logic of a unisex washroom simulation according to the following instructions:

1. Men and women cannot be in the washroom at the same time.
2. At most 3 people can be in the washroom (either 3 men or 3 women).
3. The method `useWashroom()`simulates a person using the washroom and called when the person has the right to enter and use the washroom according to instructions 1 and 2.
4. The method `maleUse()` is called by male "threads" to use the washroom; complete **this** method. (The method `femaleUse()` will be used by female "threads" and has the same logic.)
5. In the method, use the following semaphores and variable; give initial values for them in the declarations below.
    a. Variable `maleNum` to denote the number of males that are allowed to use the washroom (either waiting for it because it is full, or already inside).
    b. Semaphore `maleAccess` to control access to the variable (male threads will block on this Semaphore when women are in the washroom).
    c. Semaphore `maleEnter` to control the number of males in the washroom (male threads will block on this Semaphore when 3 men are in the washroom).
    d. Semaphore `empty` for gaining access to the washroom when the washroom is empty (this semaphore is used by both male and female threads)

    (Note, the method `femaleUse()` would used the variable `femaleNum`, the emaphores `femaleAccess` and `femaleEnter`; and also the semaphore `empty`).
6. Your solution must not deadlock, but do not worry about starvation.

```
// Semaphores/variable declarations:  Show initial values
int maleNum = ___
Semaphore maleAccess=___, maleEnter=___, empty=___

maleUse()
{




      useWashroom();




}
```

Question 5 - File Systems (10 points)

The exam annex provides a description of the contents of the Quantum Mechanics[1] Minix File System including:

- A diagram that illustrates the organization of the directories and files in the Minix file system; directory names are in bold font and file names in regular font.
- A block allocation table that gives the numbers of the blocks (Minix zones) where the content of each directory and file is located; the order of the numbers reflect the order of the logical blocks.
- The listing produced by "ls –lR" executed from the root of the mounted Minix file system.

Use the available information in the annex to complete the Minix file system structures on the following pages: the inodes, directory tables and index blocks. It is NOT necessary to fill in the fields that are greyed out.

[1] The directory names are names of scientists who contributed to the development of quantum mechanics and the file names are related to subjects they studied.

MINIX File System

MINIX Inode Table							
inode #	**i_mode**	**i_uid**	**i_size**	**i_time**	**i_gid**	**i_nlinks**	**i_zone[0..8]**
1 (root)	0100 000 101101101		128			2	524,0,0,0,0,0,0,0,0
2							
3							
4							
5							
6							
7							
8							
9							
10							
11							
12							
13							
14							
15							
16							
17							
18							
19							
20							

MINIX Directory Tables

Directory: *root*	
Ino	**Name**

Directory: bohr	
Ino	**Name**

Directory: plank	
Ino	**Name**

Directory: deBroglie	
Ino	**Name**

Directory: einstein	
Ino	**Name**

Directory: schrodinger	
Ino	**Name**

Directory: born	
Ino	**Name**

The following table can be used to define index blocks. You may use any data block (i.e. block number) not used in the Block Allocation Table of the annex (i.e. block allocated to content of directories and files. Data blocks start at block number 524.

Index Blocks	
Index Block Number	**Index Block Content**