ADVANCED OPERATING SYSTEM (CS-505)

Final Exam, Spring 2016

Date: May 20, 2016 **Marks:**100 **Time:**180 mins

Question 1 (20 points): We have a distributed system with centralized mutual exclusion system. The processes in the distributed system are arriving at following times.

Process Name	Arrival Time
P_1	3
P_2	9
P_3	11
P_4	12

The distributed system uses, wait and die algorithm for deadlock prevention. Following are different requests from the processes. You have to calculate the status of each resource, process and the resource queue.

Event No.	Requesting Process	Resource	Event	Time of Event
1	P_1	R_1	Request	5
2	P_2	R_2	Request	10
3	P_1	R_2	Request	11
4	P_1	R_1	Release	13
5	P_3	R_1	Request	14
6	P_4	R_3	Request	15
7	P_2	R_1	Request	17
8	P_4	R_1	Request	19
9	P_1	R_3	Request	20

While answering the question, before anything else, you should enumerate all the messages a centralized coordinator can send and receive, along with their meanings. Then you should create a heading for each event by mentioning its number showed in the leftmost column, and explain the status of the respective **Resource**, the **Process** and the centralized **Resource Queue**. Keep in mind that in our implementation one process can ask for other resources, while waiting for some. You may need to adjust messages from centralized coordinator to accommodate provisioning of multiple resource requests.

Question 2 (40 points): Following are three processes running in parallel, on three different machines m_1, m_2 and m_3. The variable a is on page p_1, the variable b is on page p_2 and the variable c is on page p_3. Initially, all pages are residing on machine m_1. Afterwards the pages have to be moved as needed. The semaphores (barriers in Munin language) sem_1, sem_2 and sem_3 are handled as Munin handles them. You have to tell the status of all pages at the execution of each statement.

Process 1	Process 2	Process 3
while *true* **do**	**while** *true* **do**	**while** *true* **do**
$wait(sem_1)$	$wait(sem_2)$	$wait(sem_3)$
if $a > 2$ **then**	**if** $b > 2$ **then**	**if** $c > 2$ **then**
break	**break**	**break**
end if	**end if**	**end if**
$b = b + 1$	$c = c + 1$	$a = a + 1$
$print(a)$	$print(b)$	$print(c)$
$increment(sem_2)$	$increment(sem_3)$	$increment(sem_1)$
end while	**end while**	**end while**

To understand the execution, you must note that all semaphores (barriers) are initially 0, except sem_1. The semaphore

sem_1 is initially equal to 1. The variables a, b and c are initially 0. Luckily, although the processes are running in parallel, due to the placement of semaphores, there is only one possible execution sequence. First, you have to write that execution sequence by numbering each statement. Then elaborate clearly, instruction by instruction, the process of locating the page, changing its ownership, managing directories and maintaining copy-sets.

Please do not write the general rules how Munin manages distributed shared memory, including its constructs like copy-sets and directories etc., doing so will deduct your marks. Rather just focus on the given example, and tell what will happen in result of execution of each line of code. While writing your answer, you should write the line number as a heading and then briefly explain, what is happening in the result of its execution.

Question 3 (30 points): Suppose, we have a system just like Memnet, with the only difference that with a Memnet hardware we have associated a theoretically infinite capacity hard disk. The pages which are not needed any more by a machine are not pushed towards their home machine, instead they are stored on the backing store, located on a hard disk.

We want to implement a protocol for distributed shared memory with this modified Memnet hardware. Consider the fact that the only operations done on any page are **Read** and **Write**. Secondly, the page can be on a local machine or on a remote one. Third, a read or write operation could be concurrent or exclusive. Concurrent operation means that there are more than one processes trying to perform an operation on the same page. In total, these are $2^3 = 8$ cases.

Design a protocol which handles all the eight cases, separately. In your answer, you should first enumerate all the eight cases, then you should put a heading of each case and list the commands a process should follow. Do not explain in detail, your protocol should only contain commands, or combination of commands, like "Do read", or "Copy from source" etc., like what we have seen in lectures.

Question 4 (10 points): In real time scheduling, we use "earliest deadline first" algorithm. There is another version of it called "earliest deadline first with unforced idle time". What is the difference between these two? Give a simple example.