CS414 SP 2007 Assignment 3

Due Mar. 5 at 11:59pm Submit your assignment using CMS

1. A dental clinic consists of a waiting room with N chairs and a treatment room. If there are no patient to be treated, the dentist plays solitaire on his computer (which can be considered as a sleeping state). If a patient enters the dental clinic, the dentist will stop playing and start treating the patient in the treatment room(think of this as the dentist being waken from his sleeping state). If the patient comes in and dentist is busy (treating another patient in the treatment room), but there are available chairs in the waiting room, the patient sits in one of the chairs and waits. Otherwise, if all chairs in the waiting room are occupied, then the patient leaves the dental clinic. Write a program to coordinate the dentist and patients using monitor.

2.

(a) Consider the following three scenarios. One of them is best described as deadlock, one as livelock and one as starvation. Explain which is which.

> (i) Philosophers A and B want to pick up a chopstick. They try to be nice by waiting for one second if the other one is about to take the chopstick. They both try to pick up the chopstick, wait for a second, try again, wait again, ad infinitum.
>
> (ii) Process A is waiting for the result of a computation performed by process B. However, A has higher priority than B, and so the OS prefers A and refuses to give CPU time to B. Therefore, both A and B are stuck.
>
> (iii) Processes A and B communicate through a buffer. Process A is waiting for the buffer to be more full before it reads it. Process B is waiting for the buffer to be less full before it writes more data to it. Therefore, both A and B are stuck.

(b) A system has 4 processes and 5 allocatable resource. The current allocation and maximum needs are as follows:

	Allocated	Maximum	Available
Process A	1 0 2 1 1	1 1 2 1 2	0 0 *x* 1 1
Process B	2 0 1 1 0	2 2 2 1 0	
Process C	1 1 0 1 0	2 1 3 1 0	
Process D	1 1 1 1 0	1 1 2 2 1	

What is the smallest value of *x* for which this a safe state. Explain.

3. CPUs all have on-chip caches for instructions and data. One design is to have caches *physically indexed and physically tagged*, which means both the cache index and tag are physical addresses. Alternatively, the CPU can index the cache and uses tags with virtual address (*virtually index and virtually tagged*) (there are other two type of designs which we omit for the sake of this question)

(a) Discuss the advantage having *virtually index and virtually tagged* cache over *physically indexed and physically tagged.*

(b) You are asked to consult on the design of Lunix again. (Recall from hw1, that Lunix is a simple kernel designed to support multiprogramming on a single-processor machine). The designers tell you that they have implemented and tested (successfully) the Lunix's virtual memory management component. However after they added the support for *memory-mapped files,* one test case failed. The relevant C code snippet of the test case looks like this:

```
fd = open("TestFile", O_RDWR);
ptr1 = (int*)mmap(some_addr, some_length, PROT_WRITE, MAP_SHARED, fd, some_offset);
ptr2 = (int*)mmap(some_addr, some_length, PROT_WRITE, MAP_SHARED, fd, some_offset);
*ptr1 = 1;
*ptr2 = (*ptr2)+1;
close(fd);
```

Essentially what the code does is that it opens a file, memory mapped the same segment of the file twice, and modify the mapped memory (hence the file). The designers find out that incorrect value gets written to the file "TestFile" if the code is running on system with *virtually index and virtually tagged* caches; however, on a *physically indexed and physically tagged* machine, the outcome is correct. What do you think is the problem? How would you change Lunix to address this problem.

4. In a 32-bit machine we subdivide the virtual address into 4 segments as follows:

We use a 3-level page table, such that the first 8 bits are for the first level and so on. In the following questions, sizes are in bytes.

(a) What is the page size in such a system?

(b) What is the size of the page tables for a process that has 256K of memory starting at address 0? Assume each page-table entry is 4 bytes.

(c) Also assume that each page-table entry is 4 bytes. What is the size of the page tables for a process that has a code segment of 48KB starting at address 0x01000000, a data segment of 600KB starting at address 0x80000000 and a stack segment of 64KB starting at address 0xF0000000 and growing upward (address of top of stack increases)?

Note: for (b) and (c), your answer should be size of the page table, rather than the number of PTE.

5. Suppose that a large file is organized as a binary tree of nodes and that you are using a procedure that searches the tree. Each node is the size of one page. Every operation on the tree involves a search that starts at the root. Suppose that this tree is the only paged data segment in your program.

(a) Give a brief definition of the term "working set"

(b) Which pages of the tree are likely to belong to the working set?

(c) Suppose that the tree has depth 500, that most searches descend all the way to a leaf node, and that the operating system can assign only 250 page frames of physical memory to your program. What problem might arise if the LRU page replacement algorithms is used to manage this memory?

(d) Suppose the conditions are as in part 3(c), except that now 750 pages of the physical memory are available for your program. Now would LRU work significantly better as a page-replacement policy? Explain briefly

(e) Suppose that you had the option of implementing your own page-replacement policy. What policy would you propose? would it significantly out-perform LRU?

6.
(a) Why are shared libraries highly desirable?

(b) Why are shared libraries sometimes undesirable?

(c) For the following statement indicate True or False, briefly explain: One problem with heavy use of threads for parallelism and dynamically linked files for sharing is that an application can end up with a large, fragmented, and sparse virtual address space.

(d) For the following statement indicate True or False, briefly explain: Large fragmented and sparse virtual address spaces diminish performance by reducing spatial locality and consequently lowering cache hit ratios.