

GEORGE MASON UNIVERSITY
Computer Science Department

CS 571 – Operating Systems

Fall 1995

Mid-term Exam

DUE DATE – Nov 15

1. (a) Write pseudo-code for implementing Solaris mutexes and condition variables **using semaphores**. Specifically, you have to describe (i) what data structures would be used to implement condition variables and mutexes (assuming the existence of a semaphore data structure) and (ii) how the `mutex_lock`, `mutex_unlock`, `cond_wait` and `cond_signal` calls would be implemented. **10**
- (b) Why is a `cond_wait` call in Solaris always embedded in a while loop in which the condition that caused the wait is being tested? How and why is this different from the way `wait` operations on condition variables in a **monitor** are used? Give an example which illustrates the problems that can occur if the `cond_wait` call is not embedded in a while loop. **5**
2. Consider the following snapshot of a system with 6 instances of resource R_1 , 3 instances of resource R_2 , 4 instances of resource R_3 , and 2 instances of resource R_4 .

	<u>Allocation</u>				<u>Need</u>				<u>Available</u>			
	R_1	R_2	R_3	R_4	R_1	R_2	R_3	R_4	R_1	R_2	R_3	R_4
P_0	3	0	1	1	1	1	0	0	1	0	2	0
P_2	0	1	0	0	0	1	1	2				
P_3	1	1	1	0	3	1	0	0				
P_4	1	1	0	1	0	0	1	0				
P_5	0	0	0	0	2	1	1	0				

Here the *Allocation* matrix shows the number of instances of a resource type allocated to a particular process. Similarly, the *Need* matrix shows the number of instances of a resource that could be requested by a particular process (in addition to the resources already allocated to it). The *Available* vector shows the number of instances of each resource type that are currently free.

- (a) The system is currently in a *safe* state. Explain why.
- (b) Suppose process P_5 now requests an instance of resource R_1 . If the system is using a *deadlock avoidance* approach, should this request be granted? Explain why or why not. **(5)**
3. (a) How long does it take to fork off a child process under the following conditions: text size = 100 Kbytes, data size = 20 Kbytes, stack size = 10 Kbytes, process table structure = 1 Kbytes, user structure = 5 Kbytes. Reading and writing a 32-bit word from memory take 200 nanoseconds each, and a kernel trap and return take 1 ms each. Assume that text segments are shared, and that any other operations that are required take negligible time. **4**
- (b) How would the `vfork` call reduce this time? What is the disadvantage of using `vfork`? **3**
- (c) What is *copy-on-write*? How would it reduce the time for forking a process? **3**
4. A pure paging system has a page size of 512 words, a virtual memory of 512 pages numbered 0 through 511, and a physical memory of 10 page frames numbered 0 through 9. The 18-bit virtual

address is split up into two fields: the highest 9 bits are used as an index into the page table, and the remaining 9 bits are used as an offset into a page.

The current content of physical memory is as follows:

0	:
	:
1536	Process A, PAGE 3
2048	Process B, PAGE 10
	:
	:
3072	Process A, PAGE 9
3584	Process B, PAGE 65
	:
	:
4608	Process B, PAGE 9
	:
	:

Assume that the page table for process A is stored in physical memory starting at location 0, while the page table for process B is stored in physical memory starting at location 512. Assume that one of the hardware registers is used as a **page table base register**, and that the process currently being executed is process A.

- ASSUMING THAT PAGE TABLES CONTAIN PAGE FRAME NUMBERS (RATHER THAN PHYSICAL MEMORY ADDRESSES), show the current contents of the page tables for process A and process B.
- Assuming that address translation was done using an inverted page table, what would be the current contents of the inverted page table (show only the entries corresponding to the contents of memory as shown in the figure).
- What are the current contents of the page table base register? What actions would be taken by the operating system on a context switch from process A to process B?

For your information: $1536 = 512 \times 3$; $2048 = 512 \times 4$; $3072 = 512 \times 6$;
 $3584 = 512 \times 7$; $4608 = 512 \times 9$; $5120 = 512 \times 10$; $17408 = 512 \times 34$; $33280 = 512 \times 65$;
 $2^3 = 8$; $2^4 = 16$; $2^5 = 32$; $2^6 = 64$; $2^7 = 128$; $2^8 = 256$; $2^9 = 512$; (10)

5. A process has 4 page frames allocated to it (All the following numbers are decimal, and everything is numbered starting from zero). The time of the last loading of a page into each page frame, the time of the last access to the page in each page frame, the virtual page number in each page frame, and the referenced (R) and modified (M) bits for each page frame are shown below (the times are in clock ticks from the process start at time 0 to the event).

Virtual Page Number	Page Frame	Time Loaded	Time Referenced	R Bit	M Bit
2	0	60	161	0	1
1	1	130	160	0	0
0	2	26	162	1	0
3	3	20	163	1	1

A page fault to virtual page 4 has occurred. Which page frame will have its contents replaced for each of the following memory management policies? Explain why in each case. (a) FIFO (b) LRU (c) Clock. **(5)**

6. (a) Why does UNIX use the Clock replacement policy instead of LRU? **2**
- (b) The clock policy examines the reference bit corresponding to a page in order to determine if the page should be given a “second chance”. Explain in not more than 3 sentences how an operating system can keep track of the pages that are referenced by a process on a computer that does not have hardware support for a reference bit. In other words, how can the clock policy be implemented on a computer that does not have a reference bit per page. **3**
7. (a) The mechanism used by user programs to invoke a service provided by the operating system is called a *system call*. How does a *system call* differ from an ordinary procedure call? Briefly describe the sequence of events that occurs when a user program makes a system call. **3**
- (b) What are the advantages and disadvantages of user-level threads as opposed to kernel-level threads? **3**
- (c) Consider a Sparc system with 4 processors. Suppose an application that creates 100 user level threads begins execution. How many kernel level threads (or LWPs) should the Solaris multithread library create for this application? Should the number of LWPs created be fixed for the duration of this application’s execution or should it vary? Explain. **4**
8. (a) Explain all the steps taken by the UNIX file system on a request to open the file */faculty/setia/exam.tex*. Assume that the inode for the root directory is already in memory and that all directories are one block long. However, no other inodes are in memory before the open. How many disk I/O operations will be needed for this request? **4**
- (b) Assume that an inode contains pointers to the first 10 data blocks of the file, to a single-indirect block, and to a double indirect block. Each indirect block contains pointers to 100 disk blocks. Once the file */faculty/setia/exam.tex* has been opened describe the steps taken by the file system on a request to read the 20th block of the file. How many disk I/O operations are required? **3**
- (c) What are the pros and cons of using a delayed write policy for disk buffer caches? Should this policy be applied to data blocks? inodes? directory blocks? Explain. **3**