CS140 Operating Systems and Systems Programming
Midterm Exam

October 29th, 2001

(Total time = 50 minutes, Total Points = 50)

Name: (please print)_____

In recognition of and in the spirit of the Stanford University Honor Code, I certify that I will neither give nor receive unpermitted aid on this exam.

Signature:_____

This examination is close notes and close book. You may not collaborate in any manner on this exam. You have 50 minutes to complete the exam. Before starting, please check to make sure that you have all 9 pages.

1	
2	
3	
4	
5	
Total	

Name:_____

1. Page Replacement (12 points)

Assume that you have been given a system with the following attributes:

A. A paged virtual memory system with 6 pages (A,B,C,D,E,F).
B. Two processes (P1,P2)
C. A physical memory system that can hold three pages.

The two processes generate a sequence of memory references to pages. This sequence is listed in the first column of the table on the next page. The memory reference is encoded as <Process>-<Page> where <Process> is the process making the reference (P1 or P2) and <Page> is the virtual memory page being reference (a letter A-F). For example, "P1-A" means process P1 referenced page A.

The rest of the columns of the table are used to show the contents of the 3 page physical memory after each memory reference using three different page replacement policies. Under each replacement policy there are three table entries that describe the contents of the three physical memory frames by either indicating which page is loaded (A-F) or an "*" character if the page is empty. As a guide, the rows for the first two memory references are already filled in for you. Your job is to fill in the rest of the table.

The page replacement algorithms are as follows:

LRU – A global page replacement policy using the LRU replacement algorithm.
MIN – A global page replacement policy using the MIN replacement algorithm.
FIFO – A local page replacement policy using a FIFO replacement algorithm. Under the local replacement policy process P1 has an initial resident set of 2 pages and process P2 has an initial resident set of 1 page.

At the bottom of the table you should include a count of the number of page faults seen by each replacement policy. Include in your count the two initial page faults for the first two memory references already filled in the table.

(Table is on next page…)

Question 1 continued …

Reference	LRU Global			MIN Global			FIFO Local		
P1-A	A	*	*	A	*	*	A	*	*
P1-B	A	B	*	A	B	*	A	B	*
P2-C									
P2-F									
P2-E									
P1-A									
P2-D									
P2-E									
P2-D									
P1-A									
P1-B									
Fault Count									

2. Virtual Memory Paging (10 points)

The MIPS R2000 RISC CPU has the following attributes:

- 32bit virtual address space with a 4096 (2^{12}) byte page size.
- The top half (virtual addresses 0x80000000-0xffffffff) of the 32bit address space is given to the operating system with the architecture encouraging the OS to keep the application's page table in this portion of the virtual address space. The hardware also encourages a linear page table with a 32bit PTE containing:
  a. 20bit Physical Page Number
  b. 6bit PID register
  c. Various protection bits.
- A PID register that works as follows:
  On TLB fills, the entries that are loaded contain a PID field that is equal to the current value in the PID register.
  On TLB lookup operations, TLB entries can only match if their PID field match the current PID register.

Using this information, answer the following questions:

(a) How big is the linear page table for the application's part of the virtual address space (0x0-0x7fffffff)?

(b) How much space is occupied by the block of page table entries (PTEs) in the OS's page table that maps the page table of an application (ie. the page table from part (a))?

Question 2 continued …

(c) How large is the physical address space of the system?

(d) A 6bit PID register seems to imply that only 2^6 (64) unique process ids can exist in the system. Explain what an OS with more than 64 processes would have to do.

3. CPU Scheduling (10 points)

Consider a system with two processes (P1 and P2) that have the following execution behavior over time.

Process	Execution Behavior (time ->)						
P1	CPU 1	DISK 1	CPU 2	DISK 1	CPU 3	DISK 1	CPU 4
P2	CPU 4	DISK 1	CPU 3	DISK 1	CPU 2	DISK 1	CPU 1

Each row represents a process. CPU N means the process executes on the CPU for N units of time and DISK N means that the process blocks on disk for N units of time. For example, process P1 starts by executing for 1 unit on the CPU and then waits for the DISK for 1 unit and then executes 2 units of CPU time before the next disk I/O

In the table below, list the job that would be scheduled at each time unit by a multi-level feedback queue scheduler. You can assume the scheduler overhead and context switches take zero units of time. Assume that process P1 and P2 arrive at the same time and you decide to schedule P1 for the first unit of time (this is shown already in the table).

Time	Process		Time	Process
1	P1		11	
2			12	
3			13	
4			14	
5			15	
6			15	
7			16	
8			17	
9			18	
10			20	

Scratch area for question 3.

4. Synchronization (10 points)

Assume that you have just started working on your Nachos programming assignment #1 and your partner comes with a couple of suggestions. For each of the suggestions below state if you think it is correct or incorrect. Justify your answer.

(a) Your partner claims that he knows an easy way to implement condition variables. His claim is that condition variables are simply semaphores that are always initialized to zero and have the special property that threads that block on the semaphore release the monitor lock when they sleep. He claims he has already implemented a copy of Semaphore class that trivializes the condition variable implementation with signal() being a V() on the semaphore and wait() a P() on the special semaphore. Will this work?

(b) Your partner also claims that he heard that you could get extra credit on the assignment by making an OS that could never deadlock. He suggests that simply disallowing anyone from using locks in your Nachos will always prevent deadlocks. Is this correct?

5. Linking (8 points)

Linking is frequently done as a two pass operation. Errors can be generated from either pass. For each of the linker-detected errors listed below, state when the error will be detected. The possible places are:

1) During the first pass.
2) At the end of the first pass.
3) During the second pass.
4) At the end of the second pass.

Briefly justify your answers.

(a) The duplicate definition of an external variable in multiple object files.

(b) A program that is too big for the virtual address space of the machine.

(c) The definition of an external variable that is never used.

(d) The reference to a non-existent external variable.