University of Waterloo
Midterm Examination Model Solution
CS350 Operating Systems
Fall, 2003

1. (10 total marks)

Suppose that two processes, P_a and P_b, are running in a uniprocessor system. P_a has three threads. P_b has two threads. All threads in both processes are CPU-intensive, i.e., they never block for I/O. The operating system uses simple round-robin scheduling.

a. (5 marks)

Suppose that all of the threads are user-level threads, and that user-level threads are implemented using a single kernel thread per process. What percentage of the processor's time will be spent running P_a's threads? For full credit, briefly justify your answer.

50% of the processor's time will be spent running P_a's threads.
Each process has a single kernel thread. Threads never block, so they will always consume their entire scheduling quantum. One a thread has consumed its quantum, it will be preempted in to allow the thread from the other process to run.

b. (5 marks)

Suppose instead that all of the threads are kernel threads. What percentage of the processor's time will be spent running P_a's threads? For full credit, briefly justify your answer.

60% of the process's time will be spent running P_a's threads.
In this case there are five kernel threads, each of which will receive 20% of the processor's time. Three of these are associated with process P_a.

2. (8 marks)

In a system that uses simple paging and no TLB, each page table entry includes a use (reference) bit, a dirty (modified) bit, a valid bit, and some protection bits. Complete the grid below to indicate how these bits are used by the operating system and by the MMU. In each grid cell, write "-" if the bit is neither read nor written, write "R" if the bit is read but not changed, and write "W" if the bit may be changed. For example, if you believe that the MMU does not use or change the protection bits, you should write "-" in the fourth row of the MMU column.

	Operating System	Memory Management Unit (MMU)
use bit	W	W
dirty bit	W	W
valid bit	W	R
protection bits	W	R

3. (5 marks)

Under what circumstances do page faults occur? Describe the actions taken by the operating system when a page fault occurs.

A page fault occurs when a process accesses a page whose page table entry is marked as invalid.

The OS takes the following actions:

- *If necessary, choose a page to be replaced (to obtain a frame).*
- *If the chosen page is dirty (modified), write it to secondary storage.*
- *Bring the faulted page into the selected frame.*
- *Update the page table entries for the faulted and replaced pages, and return from the exception.*

4. (10 total marks)

Consider a variation of the bounded buffer producer/consumer problem in which there are many concurrent producers and the items produced and consumed vary in size. The total capacity of the buffer is N. These processes use two semaphores to synchronize access to the buffer: `full` is a counting semaphore with initial value 0 and `empty` is a counting semaphore with initial value N. Each producer executes the following code when it wishes to insert an item of size k ($1 \leq k \leq N$) into the buffer:

```
for i from 1 to k do {
     P(empty)
}
insert item of size k into buffer
V(full)
```

The consumer executes the following code when it wishes to remove an item from the buffer:

```
P(full)
remove item of size k from the buffer
for i from 1 to k do {
     V(empty)
}
```

In each of the code fragments shown above, i is an integer variable that is local to each process. Answer each of the following questions. In each case, you must (briefly) justify your answer to receive full credit.

a. (2 marks)
Does the code above ensure that producers do not insert items into the buffer unless the buffer has room to hold them?

It does. Each producer claims k units of space from the buffer (via `P(empty)`*) before inserting an item of size k.*

b. (2 marks)
Does the code above ensure that the consumer does not remove items from the buffer unless the buffer contains items to remove.

It does. The value of the `full` *semaphore equals the number of items in the buffer. The consumer does* `P(full)` *before removing an item.*

c. (2 marks)
Does the code above ensure only one process (producer or consumer) will use (insert or remove items from) the buffer at a time?

It does not. As long as their is sufficient empty space in the buffer for both of their items, two producers may insert concurrently. Similarly, if the buffer contains items but is not full, a producer and a consumer may use the buffer concurrently.

d. (4 marks)
Does the code above ensure that processes using the buffer will not deadlock?

It does not. For example, deadlock can occur if two producers try to insert items of size greater than $\frac{N}{2}$. Each producer may "claim" half of the empty space, at which point neither producer can proceed.

5. (9 total marks)

a. (3 marks)
Briefly explain the difference between the user mode and the privileged mode of a CPU. Why is that difference important to an operating system?

User mode is used by application programs, privileged mode is used by the kernel. Certain operations, such as accessing protected memory and executing privileged instructions, can be performed only in privileged mode. The kernel takes advantage of this to retain control over system hardware and other resources, and to isolate itself from application programs.

b. (3 marks)
Briefly explain the difference between a system call and an interrupt.

A system call occurs because of an action (usually, execution of a system call instruction) taken by an application program. An interrupt is generated by system hardware, such as a timer or a disk controller.

c. (3 marks)
Briefly explain the difference between a system call and a subroutine (or procedure or function or method or whatever you like to call them) call?

- *A system call transfers control from the application to a fixed location in the OS. A procedure call transfers control within an application (or within the OS).*
- *A system call causes the CPU to change from user to privileged mode. A procedure call does not.*

6. (9 total marks)

a. (3 marks)
There are two types of internal events (events generated by the process itself) that may cause control to transfer from a running user program to the kernel. Name these types of events and briefly describe each one.

system call: *explicitly programmed by an application to request a service from the operating system*

exception: *an error condition resulting from the execution of an instruction by a application program*

b. (2 marks)
There is one type of external event that may cause control to transfer from a running user program to the kernel. Name and describe that type of event.

An interrupt is an event caused by a hardware device. It causes control to transfer to an interrupt handler inside the operating system.

c. (4 marks)

Operating systems must track various types of information about the processes and threads that exist in the system. For each type of information listed in the following table, write "P" if the OS tracks that type of information on a per-process basis. Write "T" if the OS tracks that type of information on a per-thread basis. For example, if you believe that the operating system records a page table base register value for each thread, write "T" on that line. If you believe instead that such a value is stored for each process, write "P".

	"T" or "P"
saved value of program counter register	T
saved value of stack pointer register	T
open file table	P
scheduling priority	T
page table base register value	P
page table limit register value	P
saved values of general CPU registers	T
name of executable file	P

7. (16 marks)

For both parts of this question, you should assume that there are three page frames available, and that the page reference string is as follows:

1 4 5 3 2 1 5 3 3 4 5 1

Each number in the reference string is a page number. For both parts of the question, you should assume that all three frames are initially empty.

a. (8 marks)

Complete the table below under the assumption that the optimal page replacement algorithm (OPT) is used. Your table should show what the content of each frame would be just after each reference. Use a "-" to indicate that a frame is empty. In the last row of the table, mark an "X" for each page reference that results in a page fault.

	1	4	5	3	2	1	5	3	3	4	5	1
Frame 1	1	1	1	1	1	1	1	1	1	1	1	1
Frame 2	-	4	4	3	2	2	2	3	3	4	4	4
Frame 3	-	-	5	5	5	5	5	5	5	5	5	5
Fault?	x	x	x	x	x			x		x		

b. (8 marks)

Complete the table below under the assumption that the least recently used page (LRU) replacement algorithm is used. Your table should show what the content of each frame would be just after each reference. Use a "-" to indicate that a frame is empty. In the last row of the table, mark an "X" for each page reference that results in a page fault.

	1	4	5	3	2	1	5	3	3	4	5	1
Frame 1	1	1	1	3	3	3	5	5	5	5	5	5
Frame 2	-	4	4	4	2	2	2	3	3	3	3	1
Frame 3	-	-	5	5	5	1	1	1	1	4	4	4
Fault?	x	x	x	x	x	x	x	x		x		x

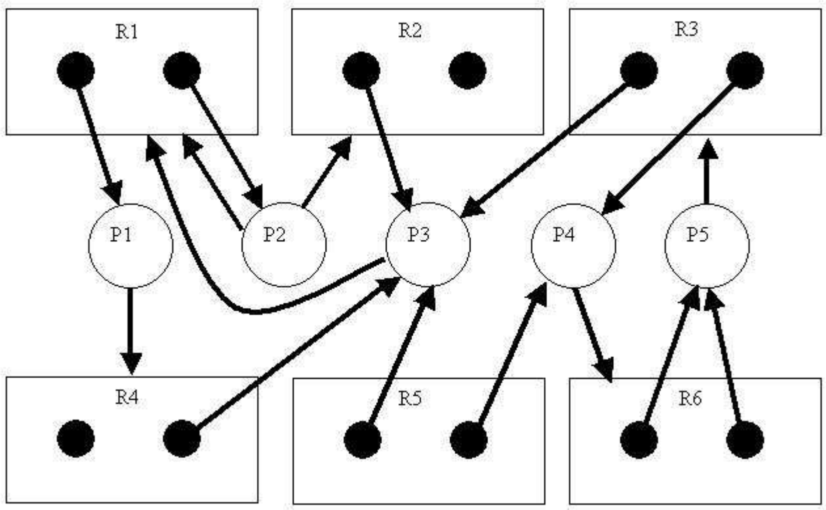
8. (11 total marks)

Consider the following information about resources in a system:

- There are six classes of allocatable resource labelled R1 through R6
- there are five processes labelled P1 through P5
- there are two instances of each resource
- there are some resource instances already allocated to processes, as follows:
  - one instance of R1 held by P1, another held by P2
  - one instance of R2 held by P3
  - one instance of R3 held by P3, another held by P4
  - one instance of R4 held by P3
  - one instance of R5 held by P3, another held by P4
  - two instances of R6 held by P5
- some processes have requested additional resources, as follows:
  - P1 wants one instance of R4
  - P2 wants one instance of R1 and one instance of R2
  - P3 wants one instance of R1
  - P4 wants one instance of R6
  - P5 wants one instance of R3

a. (5 marks)

Draw the resource allocation graph for this system. Use the style of diagram from the lecture notes.



b. (2 marks)

What is the state (runnable, waiting) of each process? For each process that is waiting, indicate what it is waiting for.

- *P1 is runnable and can acquire R4*
- *P2 is waiting for R1 and can acquire R2*
- *P3 is waiting for R1*
- *P4 is waiting for R6*
- *P5 is waiting for R3*

c. (4 marks)
Is this system deadlocked? If so, state which processes are involved. If not, give an execution sequence that eventually ends, showing resource acquisition and release at each step.

There is no deadlock. A possible sequence:

- *P1 acquires R4, runs to completion and releases R1 and R4*
- *P2 acquires R1 and R2, runs to completion and releases R1 (two instances) and R2*
- *P3 acquires R1, runs to completion and releases R1, R2, R3, R4 and R5*
- *P5 acquires R3, runs to completion and releases R3 and R6 (two instances)*
- *P4 acquires R6, runs to completion and releases R3, R5 and R6*

9. (12 total marks)
Consider an MMU which uses segmentation and paging combined, with the following characteristics:

- 4 GByte (2^{32} byte) physical memory
- 8 KByte (2^{13} byte) frame size
- maximum of 16 segments per process
- maximum segment size of 64 MByte (2^{26} byte)

a. (3 marks)
What is size (in bits) of a virtual address in this system? What parts does it have, and how big (in bits) is each part?

30 bits: 4-bit segment number, 13-bit page number and 13-bit offset

b. (3 marks)
What is size (in bits) of a physical address in this system? What parts does it have, and how big is each part?

32 bits: 19-bit frame number and 13-bit offset

c. (3 marks)
What are the two address-translation components of a segment table entry, and how big (in bits) are they?

- *page-table length: 13 bits*
- *physical address of page-table: 32 bits*

d. (3 marks)
What is the address-translation component of a page table entry, and how big (in bits) is it?

19 bits for a frame number