(2 pts) Name: **(2 pts) User Id:**

CMPSCI 377: Operating Systems
Exam 3
December 18, 2002

Answer pieces in *italics* were not required to receive full points.

---

1. **Virtual Memory, Paging, and Segmentation** (35 pts)

   (a) (5 pts) Why are translation look-aside buffers (TLBs) important? (i.e., what do they generally do?)

   TLBs are a hardware implementation of an associative memory and are used to cache a subset of the entries stored in some table. Each TLB table entry consists of a key and some associated data. A query of the table involves the presentation of the key. In parallel, all table entries compare their keys with the input; the table entry that matches the key outputs its associated data. TLBs are particularly important because one is able to look up a table entry in constant time (i.e., lookup time does not depend of the size of the table).

   (b) (5 pts) In a simple paging system, what information is stored in a typical TLB table entry?

   The key is the page number. With this key, we will store the physical frame number for this page and a bit that indicates whether this TLB entry is valid.

   (c) (10 pts) In a virtual memory system, does a TLB miss imply a disk operation will follow? Why or why not?

   No. A TLB miss implies that the full page table must be accessed (which is most likely stored in memory). We must have a miss of the page table (more specifically, a page fault) in order to require a disk operation.

(d) (10 pts) Why is segmented paging important (as compared to a paging system)? What are the different pieces of the virtual address in a segmented paging system?

Segmented paging makes it easier to share logical segments between different processes. In addition, growing the size of a logical segment requires less overhead than growing the size of a process that is located within a monolithic segment.

Segment number, page number, and offset.

(e) (5 pts) True or False? (and explain): Segmented paging incurs less internal fragmentation than pure, process-level paging.

False. On average, we pay one half of a page for every segment. So - a process that is partitioned into N segments will on average waste N/2 pages of memory to internal fragmentation.

2. **Disk Support and Demand Paging** (34 pts)

(a) (10 pts) Define Belady's anomaly. Under what conditions will an algorithm not exhibit this behavior?

Belady's anomaly is the situation in which adding more physical memory to a system results in a higher number of page faults.

An algorithm will exhibit this behavior if it cannot guarantee that it will keep a page that is kept when a smaller number of frames are available. (Stack-based algorithms do satisfy this criterion.)

(b) (14 pts) Consider a demand paging system with three frames and the given page reference sequence. How many page faults do each of the LRU, FIFO, and MIN page replacement algorithms generate? (show your work)

LRU:

	A	D	B	E	A	E	F	G	A	G	E	F
F1	A	A	A	E	E		E	E	A		A	F
F2		D	D	D	A		A	G	G		G	G
F3			B	B	B		F	F	F		E	E

FIFO:

	A	D	B	E	A	E	F	G	A	G	E	F
F1	A	A	A	E	E		E	G			G	
F2		D	D	D	A		A	A			E	
F3			B	B	B		F	F			F	

MIN:

	A	D	B	E	A	E	F	G	A	G	E	F
F1	A	A	A	A			A	A				?
F2		D	D	E			E	E				?
F3			B	B			F	G				?

(Note that this is one of two possible answers.)

(c) (10 pts) Suppose that a new disk technology is developed that makes read/write head movement infinitely fast (zero movement time for any head movement requests). It should be clear that First-Come-First-Served disk scheduling essentially stays the same. However, pure Shortest-Seek-Time-First becomes pointless. Outline a new disk scheduling algorithm that attempts to greedily minimize the amount of time that a process waits for its disk requests to complete. For simplicity, assume only one, single-sided platter (hence only one read/write head). HINT: what would shortest-job-first scheduling look like under these conditions?

At any one time, there may be many different track/sector read/write requests. Each of these requests is tied to some process, and we keep track of the number of such queued requests that an individual process has made. At any instant, only a subset of these may be serviced immediately (due to the current orientation of the platter), but only one may actually be serviced (since the scheduling algorithm must select the position of the head). Our new "Shortest-Job-First" algorithm selects the one sector whose process has the fewest number of requests in the queue.

3. **Real-Time Operating Systems** (19 pts)

(a) (5 pts) What requirement(s) characterize a real-time operating system?

A RTOS must be able to guarantee that a process (or task) is completed in full or in part by a certain deadline.

In RTOS's, we characterize processes in terms of their deadline(s), and the amount of CPU time that they require (usually, this is done with respect to their worst case CPU utilization).

(b) (14 pts) Determine when the first five instances of the following two tasks will complete under the Rate Monotonic Algorithm. Assume tasks share the same start time, the relative deadlines are equal to the respective periods, and the processes are preemptive.

Task	Period	Computation-time
1	5	1
2	20	5

Task	Instance1	Instance2	Instance3	Instance4	Instance5
1	1	6	11	16	21
2	7	27	47	67	87

4. **Distributed File Systems** (10 pts)

(a) (10 pts) Consider the file system that we implemented in lab 4. What would we have to add in order to implement a distributed file system that provides a form of location transparency? In particular, address what we could minimally change in the interface to the file system and in the individual inodes (both are short answers).

The file system interface would have to support file access requests from remote sites. Minimally, one would need to provide a socket connection on a known port through which these requests could be received and serviced.

The inodes would need to have the ability to reference files/directories on other systems. To support this, we could add 1. a flag that indicates whether a file is local or remote, 2. a host name for remote files, and 3. a file/directory reference on the remote host.

Alternatively, one could include an answer in which mount points are discussed.

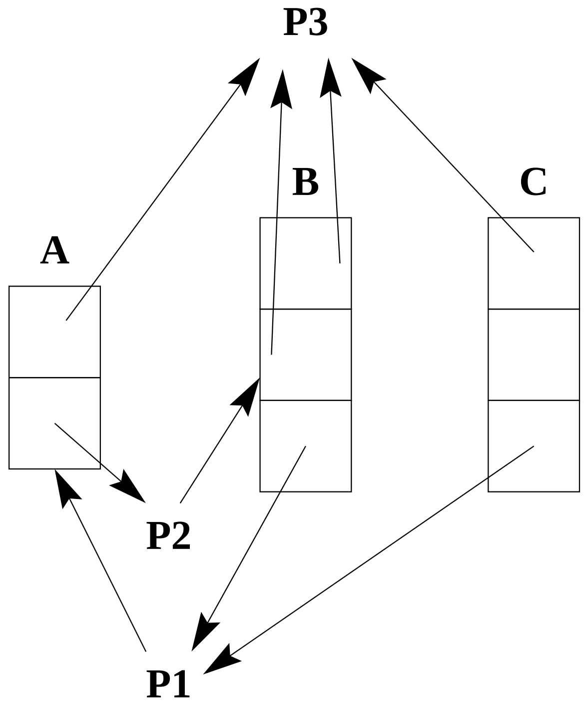
5. **Synchronization and Deadlock** (29 pts)

(a) (19 pts) Give a pseudo-Java solution to the readers/writers problem in which writers are favored (specifically, if a writer is waiting to start reading, then no readers are allowed to start reading). Use a monitor as your synchronization primitive. Specifically, provide the following methods: start_read(), end_read(), start_write(), end_write(). Also, provide any monitor-related variables and initialize them properly. HINT: If you need more than one page for your answer, then you are probably doing something wrong.

```
int num_readers = 0;  // current # of readers
int num_writers = 0;  // current # of writers (can only be 0 or 1)
int num_writers_waiting = 0;

public synchronized void start_read() {
    while(num_writers_waiting > 0 || num_writers > 0) {
        wait();
    }
    ++num_readers;
}

public synchronized void start_write() {
    ++num_writers_waiting;
    while(num_readers > 0 || num_writers > 0) {
        wait();
    }
    --num_writers_waiting;
    ++num_writers;
}

public synchronized void end_read() {
    --num_readers;
    notifyAll();
}

public synchronized void end_write() {
    --num_writers;
    notifyAll();
}
```

Alternatively, one could use two condition variables: one for each of the readers and writers.

(b) (10 pts) Suppose there are 2 copies of resource A, 3 copies of resource B, and 3 copies of resource C. Suppose further that process 1 holds one unit of resources B and C and is waiting for a unit of A; that process 2 is holding a unit of A and waiting on a unit of B; and that process 3 is holding one unit of A, two units of B, and one unit of C. Draw the resource allocation graph. Is the system in a deadlocked state? Why or why not?



No, the system is not in a deadlocked state. Once P3 completes its processing, it will release its resources. At that stage, both P1 and P2 may obtain the resources that they need.

6. **Process Management** (29 pts)

(a) (10 pts) In a system with paging memory management, list and explain the actions taken by the operating system during a context switch.

The OS must store the following information for the outgoing process (and restore this for the incoming process):

i. Program counter
ii. Stack pointer
iii. General purpose registers
iv. The page table associated with the process (or a reference to a process-specific page table contained in memory).

(b) (5 pts) True or False? (and explain): the Multi-Level Feedback Queue (MLFQ) scheduling algorithm is the same as Shortest-Job-First.

False. MLFQ attempts to approximate SJF by using recent behavior of a process to predict what the CPU needs of the process will be in the near future. Also - SJF suffers from the problem of starvation; in some implementations of MLFQ, this problem is addressed.

(c) (14 pts) Consider a preemptive, MLFQ scheduler with 3 queue levels, corresponding to 1, 2, and 4 units of time. Suppose two processes enter the queue with the following properties:

Process 1 has a total of 8 units of work to perform, but after every 2 units of work, it must perform 1 unit of I/O (so the minimum completion time of this process is 12 units).

Process 2 has a total of 20 units of work to perform. This process arrives just behind P1.

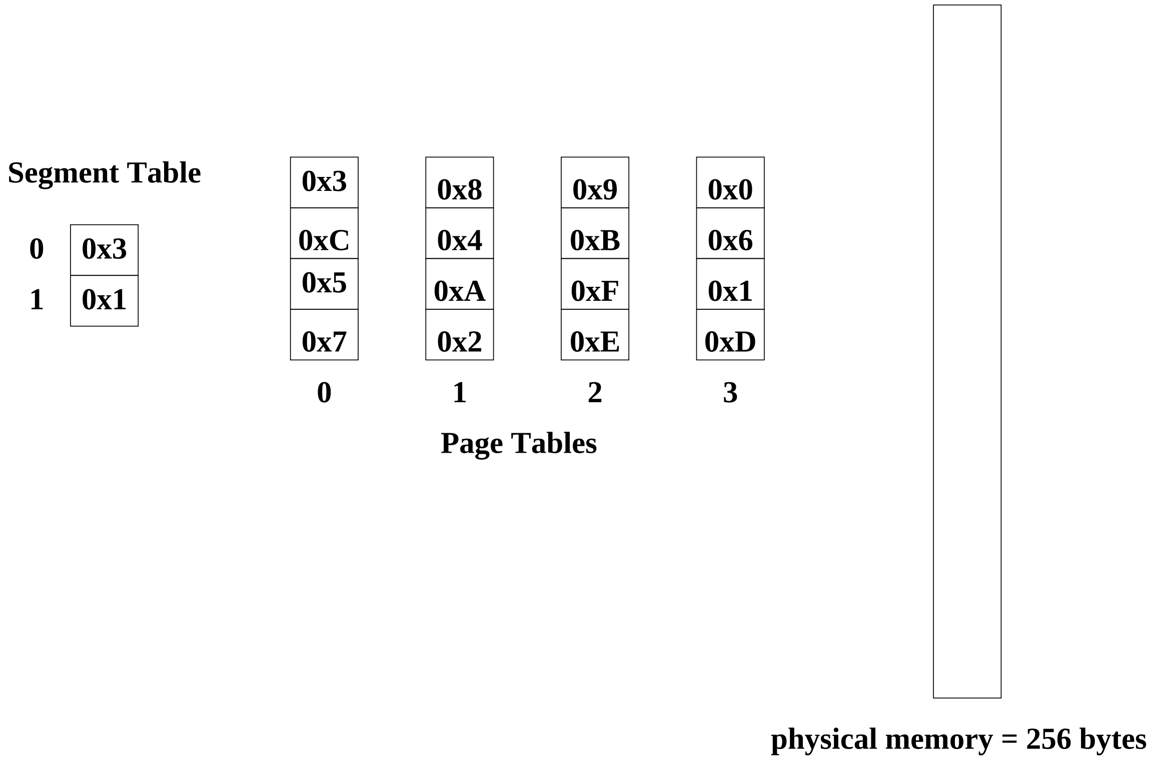
Show the resulting schedule. What is the completion time of each? (assume that a process' priority is not raised if it is preempted prematurely.)

Queue	time units												
Q1	1	$1^1_1 2^1_2$		1^3_5			1^5_{10}			1^7_{13}			
Q2	2		$1^2_3 2^2_4$		$2^4_7\ 1^4_8$			1^6_{11}			1^8_{14}		
Q3	4					2^5_9			2^6_{12}				$2^{10,14,18,20}_{18,22,26,28}$
P1 I/O			*			*			*			*	

P1 completes (with I/O) at time unit 15. P2 completes at 28.

7. **Memory Management** (15 pts)

(15 pts) Consider the following segmented paging memory system. There are 2 segments and 4 page tables. Each page table has a total of 4 entries. The physical memory is 256 bytes, with frames of 16 bytes each.



(a) (5 pts) How large is the virtual address?

$1 + 2 + 4 = 7$ bits

(b) (5 pts) What is the physical address that corresponds to virtual address 0x2D?

0x1D (decimal 29)

If only answering the paging question: 0x5D (decimal 93)

(c) (5 pts) What is the physical address that corresponds to virtual address 0x75?

0x25 (decimal 37)

If only answering the paging question: 0x75 (decimal 117)

NOTE: for partial credit, you may pretend that this is just a paging question. Use page table 0 and drop the segment part of the virtual address.

8. **Interprocess Communication** (10 pts)

(10 pts) Briefly describe the sequence of actions performed by the operating system (or library) in executing a remote procedure call.

On the client side:

(a) The parameters are marshalled (bundled) into a transmittable form (i.e., a serialized form).

(b) The client stub then identifies the location of the remote server and sends a message to this server.

(c) The stub then waits for a response

On the server side:

(a) Upon receiving a message, the server unpacks the parameters

(b) The server then calls the appropriate procedure (typically, a new thread is generated to handle this request)

(c) The server then marshals the return objects and transmits them to the client.

Back on the client side:

(a) The client stub unpacks the returned values.

(b) The values (and control) is returned to the calling program.

9. **File Systems** (15 pts)

(a) (7 pts) What problem do hard links create in terms of referential naming? How does one solve it?

When using hard links, we effectively create multiple names for the same data contained within a file. The problem in managing these structures comes down to knowing when the data within the file should actually be deleted. In particular, the data should be deleted only when the last name (or reference) is removed from the file system. This is typically detected through the use of reference counters.

It is also possible to create cycles in the directory structure, which can lead to files and/or directories being left on the disk without any way to get to the from the root directory. This may be solved by either not allowing hardlinks to directories or by explicitly checking to make sure that a cycle is not created when a link is made to a directory.

(b) (8 pts) What are the advantages and disadvantages of representing the list of blocks containing file data as we did in our blockPtr array in lab 4 (i.e., as a single-level, indexed representation)?

Advantages: simple to implement, only need one disk access to obtain the full set of pointers.

Disadvantages: Limits the size of the file, or requires very large index tables.

(c) (2 pts: bonus question) What does twelve.c do?

Generates the lyrics to the "Twelve Days of Christmas"