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

CMPSCI 377: Operating Systems

Final Exam

May 19, 2003

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

Problem	Topic	Max	Grade
0	-	4	
1	Memory	30	
2	Disk Support	35	
3	Segmented Memory	20	
4	I/O Systems	10	
5	IPC	20	
6	Synchronization and Deadlock	28	
7	Process Management	23	
8	File Systems	20	
Total		190	

1. **Virtual Memory, Paging, and Segmentation** (30 pts)

   (a) (5 pts) True or False (and explain)? When a page is swapped into main memory, it is always brought in from the swap area of the disk.

   False. Text segments may be read directly from other sections of the file system (since they are generally constant).

   (b) (5 pts) True or False (and explain)? The use of a translation look-aside buffer (TLB) for a paging memory system eliminates the need for keeping a page table in memory.

   False. A TLB will only maintain a subset of the entries stored in the full memory-based page table. When there is a TLB miss, the system will need to access the full page table.

   (c) (10 pts) List the three key pieces of information that are stored in a single page table entry. Assume pure paging and virtual memory.

   A frame number, a bit indicating whether the page is in physical memory or on the disk, and a reference for the disk block that stores the page.

   (d) (10 pts) List two advantages that segmented paging has over pure segmentation.

   1. No external fragmentation

   2. A segment does not have to exist in a contiguous range of memory (so growing the segment becomes easier and the segment does not have to fit in physical memory).

   3. From the hardware perspective, addition of offset and base is simpler with segmented paging (it is an append operation as opposed to a full addition).

2. **Disk Support and Demand Paging** (35 pts)

(a) (10 pts) Given the probability p of a page fault, what is the equation for the effective memory access time in a virtual memory system? Why do we believe that virtual memory could be efficient?

$$T_{effective} = p * T_{page\_swap} + (1 - p) * T_{memory\_access}$$

Virtual memory works because of the typical locality of memory accesses made by a set of processes (which implies that p is very small most of the time).

(b) (5 pts) What hardware support is **required** for a static memory relocation scheme?

None. Static relocation is performed and compile or load time.

(c) (10 pts) Consider the situation in which the disk read/write head is currently located at track 45 (of tracks 0-255) and moving in the positive direction. Assume that the following track requests have been made in this order: 40, 67, 11, 240, 87. What is the order in which *optimized C-SCAN* would service these requests and what is the total seek distance?

Service order: 67, 87, 240, 11, 40.
Total seek distance: $(240 - 45) + (240 - 11) + (40 - 11) = 453$.

(d) (10 pts) Consider a demand paging system with three frames and the given page reference sequence. How many page faults do each of the LRU and MIN page replacement algorithms generate? (show your work)

LRU:

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

Total page faults: 10

MIN:

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

Total page faults: 7. (Note that this is one of two possible sequences.)

3. **Segmented Memory** (20 pts)

(20 pts) Consider the following segmented paging memory system. There are 4 segments for the given process, and a total of 5 page tables in the entire system. Each page table has a total of 8 entries. The physical memory requires 12 bits to address it; there are a total of 128 frames.

Segment Table

0	0x3
1	0x1
2	0x0
3	0x4

Page Tables

	0	1	2	3	4
0	0x73	0x25	0x85	0x0F	0x17
1	0x2C	0x2D	0x31	0x3D	0x00
2	0x05	0x1E	0x01	0x5D	0x0D
3	0x17	0x5A	0x1F	0x1E	0x66
4	0x57	0x0F	0x09	0x6C	0x62
5	0x1A	0x7A	0x0A	0x2F	0x50
6	0x4B	0x2B	0x1A	0x78	0x32
7	0x11	0x6C	0x32	0x7B	0x11

physical memory; address = 12 bits

(a) (5 pts) How many bytes are contained within the physical memory?
$2^{12} = 4096$ bytes

(b) (5 pts) How large is the virtual address?
$2 + 3 + 5 = 10$ bits

(c) (5 pts) What is the physical address that corresponds to virtual address 0x312?
0x2F2

(d) (5 pts) What is the physical address that corresponds to virtual address 0x1E9?
0xD89

4. **I/O Systems** (10 pts)

Define Direct Memory Access. Under what conditions is it useful?

DMA is the direct transfer of data between memory and a device controller (circumventing the CPU). This mode of device/system interaction is most useful when large volumes of data are being transferred very quickly (e.g., for disk or video operations).

5. **Interprocess Communication** *(20 pts)*

(a) *(10 pts) Give an example application in which resource sharing is accomplished by computation migration.*

Computation migration is used in querying remotely located databases.

(b) *(10 pts) In TCP/IP, what does a client process minimally need to know in order to establish a connection with the server process?*

The client must know the server's hostname and listener port number.

6. **Synchronization and Deadlock** *(28 pts)*

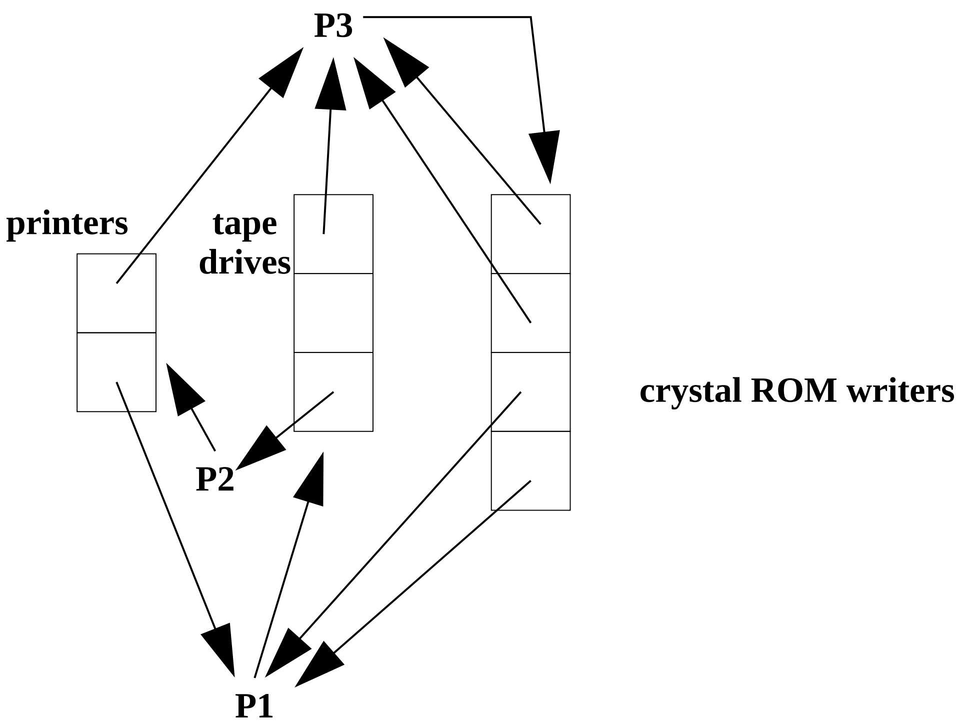
(a) *(14 pts) Implement using monitors a buffer class that is to be accessed by an arbitrary number of producers and consumers. The putItem(object, elem1, elem2) method will place a new object in one of two buffer locations (elem1 or elem2), and will block if both locations are already full. The getItem(elem1, elem2) method will return an object stored in one of these two locations and free the corresponding buffer entry, but will block if no object is available.*

Complete the implementation on the next page by filling in the necessary monitor code. You must use both condition variables. You may assume that all buffer indices are valid (you don't need to check their bounds).

```
class buffer {
    condition A, B;     // Condition variables
    object buffer[];    // Buffer of objects
    boolean full[];     // Boolean array that indicates state of each
                        //    buffer item

    buffer(int n) {     // Constructor
       int i;
       buffer = new object[n];  // Create the buffer
       full = new boolean[n];   // Create the boolean array
       for(i=0; i < n; ++i) {   // Default: all entries are empty
          full[i] = false;
       }
    }
```

```
    // Place <obj> in buffer element <elem1> or <elem2>
    synchronized void putItem(object obj, int elem1, int elem2) {
        // Wait until one of the two entries is free
        while(full[elem1] && full[elem2]) {
            A.wait();
        }

        // Actually place the object (at this point, we guarantee that
        //  at least one of the buffer entries is free)
        if(!full[elem1]) {
            full[elem1] = true;
            buffer[elem1] = obj;
        }else{
            full[elem2] = true;
            buffer[elem2] = obj;
        }

        // Let the consumers know that an object is available
        B.notifyAll();
    }

    // Get <obj> from buffer element <elem1> or <elem2>
    synchronized object getItem(int elem1, int elem2) {
        object obj;

        // Wait until there is at least one entry to be removed
        while(!full[elem1] && !full[elem2]) {
            B.wait();
        }

        // Actually remove the object (at this point, we guarantee that
        //  at least one of the buffer entries is full)
        if(full[elem1]) {
            full[elem1] = false;
            obj = buffer[elem1];
        }else{
            full[elem2] = false;
            obj = buffer[elem2];
        }

        // Let the producers know that an entry has become free
        A.notifyAll();
        return(obj);
    }
}
```

(b) *(14 pts) Suppose a computer lab contains two printers, three tape drives, and four terabyte crystal ROM writers. Assume program 1 is currently using a printer and two crystal ROM writers, program 2 is using a tape drive, and program 3 is using two crystal ROM writers, a tape drive, and a printer. Suppose now that program 1 requires a tape drive, program 2 requires a printer, and program 3 requires one more crystal ROM writer. Draw the corresponding resource allocation graph. Is the system in a deadlocked state? Why or why not?*



No. P1's request is satisfied immediately. Assuming that no more resource requests are made, P1 will be able to eventually complete, freeing the resources required by the other processes.

7. **Process/Thread Management** *(23 pts)*

(a) *(10 pts) Suppose that a process scheduling algorithm favors those processes that have used the least processor time in the recent past. Why will this algorithm favor I/O-bound processes, but not starve CPU-bound processes?*

I/O bound processes generally spend most of their time in the wait queue while an I/O operation is being performed. When their I/O finally completes, they will generally not execute for a long time before they initiate the next I/O operation. Therefore, they will be favored in scheduling relative to long-running processes. However, if CPU-bound processes are blocked long enough by I/O-bound processes, they too will eventually not have run for a long time and they will be scheduled.

(b) *(13 pts) Suppose two processes enter the ready 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). Assume that there is no work to be done following the last I/O operation.*

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

Show the resulting schedule for the shortest-job-first (preemptive) and the round-robin algorithms. Assume a time slice of 4 units for RR. What is the completion time of each process under each algorithm?

SJF:

Start Time	0	2	3	5	6	8	9	11	28
Process	P1	P2	P1	P2	P1	P2	P1	P2	

P1 completes (with I/O) at time unit 12. P2 completes at 28.

RR:

Start Time	0	2	6	8	12	14	18	20	28
Process	P1	P2	P1	P2	P1	P2	P1	P2	

P1 completes (with I/O) at time unit 21. P2 completes at 28.

8. **File Systems** *(20 pts)*

(a) *(10 pts) List one advantage and one disadvantage of having large block size.*

Advantage: On average, fewer disk seeks for the same amount of data read from the disk.

Disadvantage: More loss due to internal fragmentation.

(b) *(10 pts) List one advantage and one disadvantage of representing the list of blocks containing file data as we did in lab 4 (i.e., as a linked list).*

Advantages: A file can be as large as the number of blocks on the disk. Deleting a file is inexpensive.

Disadvantages: More complex to implement than indexed arrays. We pay a heavy price with randomly accessed files because we must scan through the list of blocks (loading each block from disk) until we reach the point in the file that is of interest.

(c) *(2 pts: bonus question) Are all operating systems destined to be free? Why/why not?*

Any well-reasoned answer will do here.