

**CS 574**  
**Final Exam**  
**December 15, 2006**

The following exam is open book and open notes. You may feel free to use whatever additional reference material you wish, but **no electronic aids** are allowed. Please note the following instructions. There will be a ten point deduction for failure to comply with them:

- start each problem on a new sheet of paper
- write your social security number, but not your name, on each sheet of paper you turn in

Also, please note the following:

- show your work whenever appropriate. There can be no partial credit unless you show how you derived your answers
- be succinct. You may lose points for facts that, while true, are not relevant to the question at hand

You have until 12:30 to finish the exam.

1. (10 points) A unique feature of the Intel x86 architecture is the “IO bitmap”: each user-level task can be allowed to read and write particular IO ports, on a port-by-port basis.

**Question:**

**How could a microkernel operating system take advantage of this? Note: Minix V3 does not do this. So the Minix V3 approach would be completely irrelevant to this question.**

2. (20 points) Some computer system uses a hashed page table. The system uses 32 bit virtual and physical addresses, and a 4K page size. In order to make the problem tractable we’re going to assume only eight entries in the HPT; the hash function will just be to use the low-order three bits of the page number. In order to make this a problem in operating systems instead of arithmetic, I’ll put the full VPN in the table instead of just the most significant 17 bits. A collision chain is being used for collision resolution. Here’s the HPT:

Index	VPN	PFN	Next
0	98765	10134	3
1	2468a	1101b	5
2	1234a	41424	0
3	7a45c	10257	6
4	1234a	98acd	7
5	3579b	3468a	
6	2468b	893ac	
7	abcde	13502	

An empty “Next” field is used to indicate the end of a collision chain.

**Question: for each of the following virtual addresses, is there a translation in the table? If so, what is the translated physical address?**

- (a) **1234abcd**
- (b) **2468bdf0**
- (c) **abcd1234**

3. (25 points) You wish to run a mix of realtime, multimedia, and batch processes on some system. The scheduling rules are:

- All of the realtime tasks must meet their deadlines. If they can be scheduled in such a way as to improve the ability of the multimedia tasks to meet their goals so much the better, but the realtime task deadlines must take precedence.
- Each of the multimedia tasks want to receive some number of the time slices from each quantum. If one of them cannot receive sufficient time during a quantum, don't give it any time during that quantum.
- Any time left over will be given to the batch job.

Realtime tasks: the following table gives the start time, required processing time, and deadline for each of the realtime tasks.

Task	Start Time	Processing Time	Deadline
$R_1$	0	5	25
$R_2$	5	10	20
$R_3$	20	5	30

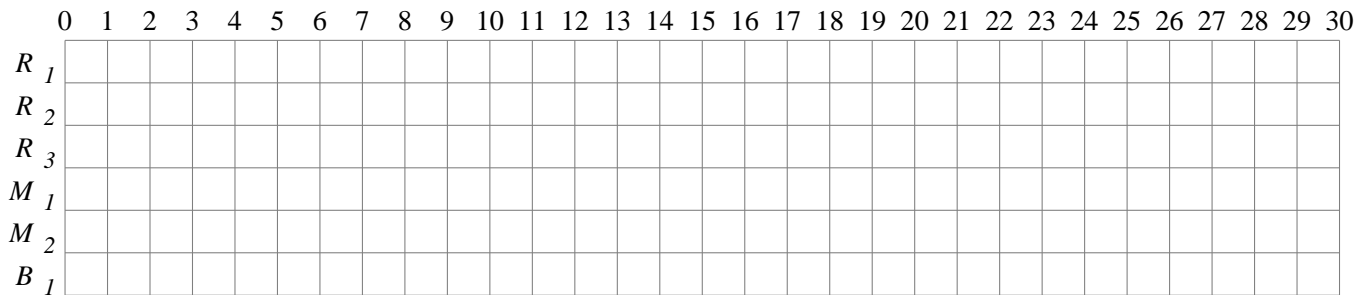
Multimedia Tasks: the following table gives the time requirements of the multimedia tasks.  $M_1$ 's first quantum runs from time 0 to time 10, its second from time 10 to time 20, and its third from time 20 to time 30.  $M_2$ 's first quantum runs from time 0 to time 5, its second from time 5 to time 10, and so forth.

Time	Quantum Length	Time per Quantum
$M_1$	10	1
$M_2$	5	1

Batch tasks: there is a single batch job,  $B_1$ .

**Question:**

Schedule these tasks. You may use the following figure if you'd like. There is space for your SSN at the top right of this page.



4. (20 points) In a computer system using the DASH consistency protocol, variable  $x$  is homed (has its directory) on cluster  $C_1$ , while  $y$  is homed on  $C_2$ . Initially, they are both in state *uncached-remote* in their respective directory controllers. Processor  $P_2$  is located in cluster  $C_2$ , while  $P_3$  is located in cluster  $C_3$  (the lack of a processor  $P_1$  is not an error. I wanted to have some particular interactions happen in this question, which required the layout of processors and clusters I've selected). The two processors execute the following program:

$P_2$	$P_3$
$S;$ $y = x + 1;$	$x = 1;$ $S;$

The  $S;$  statements represent synchronization between the programs and are only there to assure the ordering of the statements; for purposes of this exam you should ignore any messages they might cause to be sent.

**Questions:**

- (a) **What messages need to be transferred between the directory controllers as a result of these statements?**
  - (b) **What are the final states of the variables in the directory controllers?**
5. (15 points) Assume we have two CPU-bound processes running on a node of a system that supports process migration. They will each require CPU time  $T_c$  to complete. Since they are on a single node, they will each only get 50% of the CPU, and they will take time a total time of  $2T_c$  time as a result.

To migrate one of the processes to an unused node will take some time  $T_m$ . The process will need to be frozen for the duration of the transfer.

Migrating a process to another node is advantageous if *both* processes will complete more quickly as a result.

**Question:**

**Develop an expression in terms of  $T_c$  and  $T_m$  that determines whether it is advantageous to migrate one process.**

6. (10 points) All modern operating systems permit disk reads and writes to occur out of order: several processes make unrelated read and write requests, and instead of executing them in the specified order the operating system will reorder them so the amount of disk head movement can be minimized and the overall performance can be improved. In a distributed filesystem, there's no real way for a client to know anything about the disk geometry of the fileserver, and so there is no real way for the client to optimize the order of reads and writes.

**Question:**

**With this in mind, would it make more sense to build a distributed filesystem on top of RPC or a lower-level UDP or TCP implementation?**