EXAM ON FILE SYSTEMS AND I/O – SPRING 2005
CSC 262 – OPERATING SYSTEMS
NICHOLAS R. HOWE

This is a closed-book exam. You may use one double-sided 8.5x11 sheet of notes.

All answers to this exam should be written in your exam booklet(s). Start with the questions that you know how to do, and try not to spend too long on any one question. Show your work to be eligible for partial credit. You will have two hours and twenty minutes. Good luck!

1. **Efficiency Measures**. (12 points) For the scenario below, compute the mean latency, mean cpu utilization, and mean throughput.

> *Job A: Assigned @ 0 ms, finished @ 10 ms, consuming 4 ms of cpu time in all.*
> *Job B: Assigned @ 4 ms, finished @ 7 ms, consuming 2 ms of cpu time in all.*
> *Job C: Assigned @ 2 ms, finished @ 10 ms, consuming 3 ms of cpu time.*
>
> *Job latencies: A = 10, B = 3, C = 8; mean = 7.*
> *CPU Utilization: 9 ms/10 ms = 90%.*
> *Throughput: 3 jobs / 0.01 s = 300 jobs/s.*

2. **Disk Head Scheduling**. (10 points) Below are several questions listing two sets of scheduling algorithms. In each case, some key difference(s) separate(s) all the algorithms in one set from all the ones in the other. Explain the shortcomings of one group, and how the other group attempts to address those shortcomings.

a. SCAN vs. LOOK
SCAN always travels out to the edge of the disk, even if there are no requests for the outermost tracks. LOOK examines the track request queue and only travels as far as the most distant outstanding request.

b. F-SCAN and F-LOOK vs. SCAN and LOOK
SCAN and LOOK can get "stuck" on a particular track if processes continually generate requests for items there. F-SCAN and F-LOOK address the problem by using two queues. While requests are serviced off one queue, new requests are placed in the other queue. Thus all tracks will be serviced, even if there is a continuous stream of requests for one.

c. C-SCAN and C-LOOK vs. SCAN and LOOK
SCAN and LOOK can have high latency for requests on high or low-numbered tracks, because they do not return to these areas until they have scanned the disk in two directions. C-SCAN and C-LOOK reduce the latency by scanning in only one direction.

d. FIFO vs. SSTF

FIFO can lead to unnecessary movement of the disk head, because requests on nearby tracks are not necessarily serviced all at once. SSTF requires that nearby tracks be serviced first.

e. SCAN vs. SSTF

SSTF can lead to starvation if there is a stream of requests in one area of the disk. SCAN addresses this by requiring the disk head to move in only one direction at a time, so that it keeps making progress.

3. **RAID**. (10 points) Explain the motivation behind the technology known as the Redundant Array of Independent Disks. Cite two areas in which a RAID device may achieve better performance than an ordinary disk, and give at least one specific example explaining how such improved performance may be achieved in each area. (You may wish to use diagrams to make your answer more clear.)

Disk drives are slow compared to main memory. Furthermore, disk drives sometimes fail, and the more disk drives are used, the more likely it is that at least one will fail. RAID is designed to address both of these shortcomings. A RAID disk can achieve faster data transfers rates than a traditional disk, and can also provide greater data security, because lost data can be recreated in the event of a crash.

Improved transfer time may come from two different factors. The first involves simple data replication. If the desired data is stored on two separate physical drives, then the latency of a read operation will be the lesser of the latencies of the two drives. Data striping is a more complicated way to speed up transfer times. File data are distributed across multiple physical disks, and are read and written in parallel to increase speed.

Improved data security may also come from two different techniques. Simple data replication provides a backup copy of every piece of information on a separate physical drive. A more space-efficient approach uses parity information to provide backup information for an arbitrary number of drives, should one of them fail. A similar but independent set of backup data can make up for the failure of two drives at once, if used in conjunction with parity.

4. **Protection and Security**. (12 points) A hypothetical system has three privilege domains, {D_1, D_2, D_3}, and five resources of interest, { F_1, F_2, F_3, F_4, F_5}. Below is a set of capabilities lists for three hypothetical domains, in *(resource,rights)* format. Convert this to an access control list describing the same security policy, using a *(domain,rights)* format.

D_1: {(F_1,rw),(F_3,rx)}
D_2: {(F_3,r), (F_4,w),(F_5,rx)}
D_3: {(F_4,r)}*

F_1: {(D_1,rw)}
F_2: ∅
F_3: {(D_1,rx),(D_2,r)}
F_4: {(D_2,w),(D_3,r)}*
F_5: {(D_2,rx)}

5. **File System Organization**. (12 points) Shown below are the inode of a file and the first twelve data blocks of a disk on which the file is stored. The data blocks belonging to the file are labeled A through K in the diagram below. Your job is to fill in the data block pointers so that the data blocks for the file can be assembled in the proper (alphabetical) order. If necessary and appropriate, you may use additional data blocks (13+) for pointers. Assume that the inode has only two direct pointers, one single indirect, one double indirect, and zero triple indirect, and that a data block can hold exactly four pointers. Fill in unused pointers with an appropriate value.

inode	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
4		*12*											*14*	*11*	*3*
1	B	*8*	K	A	E	F	H	D	J	I	G	C	*15*	*7*	*0*
2		*5*											*0*	*10*	*0*
13		*6*											*0*	*9*	*0*

6. **Vocabulary**. (16 points) Identify the following terms or acronyms with a short definition (no more than one or two sentences).

a. Security mechanism
The means by which a security policy can be carried out. Mechanisms should be as flexible as possible.

b. Multiprogramming
Refers to the ability of an operating system to keep multiple processes running at once, alternately executing each one in turn.

c. Line printer
A printer controlled directly by the CPU (as opposed to an off-line printer, which is not controlled by the CPU).

d. MBR

Master Boot Record. Contains the partition table and boot loader program.

e. Fetch-execute cycle

The continuous cycle in which the CPU loads a machine instruction from the memory address specified in the IR into the PC, and then executes the instruction.

f. Bus (the computer-related meaning, not the vehicle)

A piece of hardware that carries data between components of the computer. An example is the system bus, connecting the CPU to main memory, and the device hardware controllers.

g. Asynchronous I/O

A style of I/O in which the process executing the I/O operation keeps executing while the operation is completed. Stands opposed to synchronous I/O, where the process executing the operation must pause until it is complete..

h. ELF

Executable and Linking Format. A newer file format used for binary executable files on Linux machines.

7. **OS Organization**. (12 points) You are working on an experimental operating system. Diagnose the following symptoms, based upon what you know about the organization of a modern operating system, by explaining what level of the operating system is most likely at fault and why you think that. Be as specific as possible. [You do not need to actually propose a solution to the problem.]

a. You can see all the files on your hard drive and Zip disks. However, you can't read anything from the floppy drive. In fact, the computer doesn't even seem to know that the floppy drive is there at all. You know it isn't a hardware problem, because when you remove the floppy drive from this computer and put it into another one, it works fine.

This sounds like the device driver level. The file system is fine, since you can see files on other disks. The kernel doesn't deal with I/O. Thus is must be the driver level. (Probably the driver has a bug or is installed incorrectly.)

b. Two people, both connecting to your computer remotely and running identical jobs, get very different performance. One job finishes almost immediately; the other runs very slowly, and seems to freeze up for long periods of time. You have taken steps to rule out the network connection as the source of trouble. The same thing also happens to a single user running

multiple jobs at once: some finish right away, and some run very slowly. It happens regardless of what jobs are being run.

Time sharing between processes is the kernel's job. Since the problem is independent of the job being run, it must not be at the user level. There is no indication that the driver or service levels are involved. Therefore it must be the kernel.

c. Files on your system have both a character-string name and a numeric file id. For some reason, you can't get a complete listing of all the files on your hard drive. Some of them are missing when you list the names of all the files. Other files appear normally and you can access them normally. However, you know that the non-appearing files are still there and intact because you can still access them normally using the file id.

This sounds like a problem with the file system (service layer). Since some of the files work fine, and the missing files can be accessed by their file id, the driver is probably ok. Administration of file names is the file system's job.

8. **Ext2 File System**. (16 points) Suppose a floppy disk is formatted with an Ext2 file system, using 2KB blocks, and 176 inodes of 128 B size. Compute the offset from the beginning of the disk for the start of the following items, showing all your work:

a. The superblock.
The superblock always appears right after the boot block, which is always 1024 bytes.

b. The inode bitmap.
The inode bitmap appears after the boot block, super block, group descriptors, and block bitmap, i.e., block 4.
*Offset = 1024+(4-1)*2048 = 7168*

c. The root directory inode.
The root directory inode is always inode number 2 in the inode table, which is after the inode bitmap (i.e., block 5).
*Offset = 1024+(5-1)*2048+(2-1)*128 = 9344*

d. The tenth data block (i.e., the tenth block after all the header blocks)
The header information takes up 15 blocks on this disk (1 each for the superblock, group descriptors, block bitmap and inode bitmaps, plus 176/(2048/128) = 11 for the inode table), so the tenth data block is block 25.
*Offset = 1024+(25-1)*2048 = 50176*