CS 537: Introduction to Operating Systems
Fall 2015: Midterm Exam #3
PRACTICE QUESTIONS

This exam is closed book, closed notes. All cell phones must be turned off. No calculators may be used.

You have two hours to complete this exam.

Write all of your answers on the accu-scan form with a #2 pencil.

These exam questions must be returned at the end of the exam, but we will not grade anything in this booklet.

Good luck!

Part 1: Straight-forward True/False from Virtualization [1 point each?]
Designate if the statement is True (a) or False (b).
Expect approximately 10 questions similar to questions on previous exams this semester.

Part 2: Straight-forward True/False from Concurrency [2 points each?]
Designate if the statement is True (a) or False (b).
Expect approximately 20 questions similar to questions on previous exams this semester.

Part 3: True/False from Persistence [3 points each?]
Designate if the statement is True (a) or False (b).
Expect approximately 30 questions similar in style to the following.

1) Peripheral devices are typically on a bus further away from the CPU than RAM is.
2) When interacting with a device, it is always better to use interrupts instead of spin-waiting.
3) DMA stands for Direct Memory Addressing.
4) Special instructions are required for the CPU to control and interact with a peripheral device.
5) If a disk contains 10240 sectors per track and there are 2 double-sided platters, there are 20480 sectors per cylinder.
6) Seek cost is a function of the cylinder distance.
7) With a 7200 RPM disk, the expected rotation time for a random access is more than 5 ms.
8) Transfer time for disk sectors is significantly longer for random accesses than for sequential accesses.
9) SPTF scheduling is easier to implement inside of a disk than within the OS.
10) A disadvantage of the SCAN scheduling algorithm is that it ignores the influence of seek time on positioning cost.
11) A work conserving scheduler always performs work if there is work to be done.
12) JBOD and RAID-0 are identical.
13) RAID-1 delivers half the useful capacity of RAID-0.
14) With RAID-1, the storage system can continue executing correctly as long as half the disks do not fail.
15) RAID-1 improves the latency of random read operations compared to RAID-0.
16) With RAID-1, the steady-state throughput of random reads and random writes are identical.
17) RAID-4 has better capacity than RAID-1 and better reliability than RAID-0.
18) The disadvantage of RAID-4 is that it cannot continue operating if the single parity disk fails.
19) For random read operation, RAID-1 delivers better throughput than RAID-5.
20) Given the metrics of capacity, reliability, and performance, RAID-0 is strictly better than RAID-1.
21) In an FFS-like file system, multiple file descriptors may point to the same inode.
22) An advantage of hard links over soft links is that hard links can be used to refer to directories.
23) When a file is moved into a different directory in the same file system, the amount of time for the operation is a function of the amount of data in that file.
24) Contiguous allocation of files to blocks on disk requires very little overhead for meta-data.
25) A File-Allocation Table suffers from external fragmentation.
26) With multi-level indexing, 4KB blocks, 4 byte pointers, 10 direct pointers, 1 indirect block, and 1 double indirect block, the maximum file size that can be supported is XYZ.
27) An inode structure typically contains a field indicating the owner of this file.
28) FFS sacrificed disk capacity in order to achieve better bandwidth for small files (< 4KB).
29) FFS attempts to put small files from the same directory in the same cylinder group.
30) When placing a directory inode in a “new” group, FFS looks for a group with more than the average number of free inodes.

Part 4: RAID Mapping [2 points each?]

The next questions ask you to translate logical read and write operations performed on top of a RAID system to the physical read and operations that will be required to the underlying disks. Specifically, for each RAID configuration, translate the logical requests to the physical operations performed on the correct disk number and physical offset. These questions are all similar to those available from the homework simulations.

31) RAID Level: 0; Block size: 4KB; Chunk size: 4KB; Number of Disks: 4
Random Read from logical address 8444
   a. Read from Disk 0, offset 2111
   b. Read from Disk 1, offset 2111
   c. Read from Disk 2, offset 2111
   d. Read from Disk 3, offset 2111
   e. None of the above

32) RAID Level: 1; Block size: 4KB; Chunk size: 4KB; Number of Disks: 4
Random Write to logical address 4205
   a. Write to Disk 2, offset 2102
   b. Write to Disk 3, offset 2102
   c. Write to Disk 0 and Disk 1 at offset 2102
   d. Write to Disk 2 and Disk 3 at offset 2102
   e. None of the above

33) RAID Level: 4; Block size: 4KB; Chunk size: 4KB; Number of Disks: 4
Random Write to logical address 5112
   a) Write to disk 0, offset 1704
   b) Write to disk 3, offset 1704
   c) Write to disk 0 and disk 3, offset 1704
   d) Read from disk 0 and disk 3, offset 1704; write to disk 0 and disk 3, offset 1704
   e) None of the above

34) RAID Level: 5 (Left symmetric); Block size: 4KB; Chunk size: 4KB; Number of Disks: 4
Random Read from logical addr:4765
   a) Read from disk 0, offset 1588
   b) Read from disk 1, offset 1588
   c) Read from disk 2, offset 1588
   d) Read from disk 3, offset 1588
   e) None of the above

35) RAID Level: 5 (Left symmetric); Block size: 4KB; Chunk size: 4KB; Number of Disks: 4
Sequential Write to logical addr: 5112 of 4 blocks
   a. Write to disk 0, 1, 2 and 3 at offset 1704
   b. Write to disk 0 at offset 1704, 1705, 1706, 1707
   c. Read from disk 0, 1, 2, and 3 at offset 1704; Write to disk 0, 1, 2, and 3 at offset 1704
   d. Read from disk 0, 1, and 2 at offset 1704; Write to disk 0, 1, and 2 at offset 1704
   e. None of the above

Part 5: Basic File System Operations and Data Structures [2 points each?]
These questions ask you to understand how different file system operations lead to different file system data structures being modified on disk. You do not need to consider journaling or crash consistency in these questions. This part is based on the available homework simulations.

This file system supports 7 operations:
- mkdir() - creates a new directory
- creat() - creates a new (empty) file
- open(), write(), close() - appends a block to a file
- link() - creates a hard link to a file
- unlink() - unlinks a file (removing it if linkcnt==0)

The state of the file system is shown with the contents of four different data structures:

inode bitmap	- indicates which inodes are allocated
inodes	- table of inodes and their contents
data bitmap	- indicates which data blocks are allocated
data	- indicates contents of data blocks

With the bitmaps, 1 indicates that the corresponding inode or data block is allocated; 0 indicates it is free.

The inodes each have three fields: the first field indicates the type of file (f for a regular file, d for a directory); the second indicates which data block belongs to a file (here, files can only be empty, which have the address of the data block set to -1, or one block in size, which would have a non-negative address); the third shows the reference count for the file or directory. For example, the following inode is a regular file, which is empty (address field set to -1), and has just one link in the file system: [f a:-1 r:1]. If the same file had a block allocated to it (say block 10), it would be shown as follows: [f a:10 r:1]. If someone then created a hard link to this inode, it would then become [f a:10 r:2].

Finally, data blocks can either retain user data or directory data. If filled with directory data, each entry within the block is of the form (name, inumber), where "name" is the name of the file or directory, and "inumber" is the inode number of the file. Thus, an empty root directory looks like this, assuming the root inode is 0: [(.,0) (..,0)]. If we add a single file "f" to the root directory, which has been allocated inode number 1, the root directory contents would then become: [(.,0) (..,0) (f,1)]

If a data block contains user data, it is shown as just a single character within the block, e.g., "h". If it is empty and unallocated, just a pair of empty brackets ([]) are shown.

An entire file system is thus depicted as follows:
```
inode bitmap 11110000
inodes       [d a:0 r:6] [f a:1 r:1] [f a:-1 r:1] [d a:2 r:2] [] ...
data bitmap  11100000
data         [(.,0) (..,0) (y,1) (z,2) (f,3)] [u] [(.,3) (..,0)] [] ...
```

This file system has eight inodes and eight data blocks. The root directory contains three entries (other than "." and ".."), to "y", "z", and "f". By looking up inode 1, we can see that "y" is a regular file (type f), with a single data block allocated to it (address 1). In that data block 1 are the contents of the file "y": namely, "u". We can also see that "z" is an empty regular file (address field set to -1), and that "f" (inode number 3) is a directory, also empty. You can also see from the bitmaps that the first four inode bitmap entries are marked as allocated, as well as the first three data bitmap entries.

Assume the initial state of the file system is as follows:

```
inode bitmap  10000000
inodes        [d a:0 r:2] [] [] [] [] [] [] []
data bitmap   10000000
data          [(.,0) (..,0)] [] [] [] [] [] [] []
```

If the file system transitions into each of the following states (the state of the file system and the questions are cumulative), what operation must have been performed?

36) File System State:

```
inode bitmap  11000000
inodes        [d a:0 r:3] [f a:-1 r:1] [] [] [] [] [] []
data bitmap   10000000
data          [(.,0) (..,0) (y,1)] [] [] [] [] [] [] []
```

a. creat("/y");
b. mkdir("/y");
c. creat("/f");
d. mkdir("/f");
e. None of the above

37) File System State:

```
inode bitmap  11000000
inodes        [d a:0 r:3] [f a:1 r:1] [] [] [] [] [] []
data bitmap   11000000
data          [(.,0) (..,0) (y,1)] [u] [] [] [] [] [] []
```

a. creat("/y");
b. fd=open("/y", O_WRONLY|O_APPEND); write(fd, buf, BLOCKSIZE); close(fd);
c. link("/y", "/u");
d. creat("/y/u");
e. None of the above

38) File System State:

```
inode bitmap  11000000
inodes        [d a:0 r:4] [f a:1 r:2] [] [] [] [] [] []
data bitmap   11000000
data          [(.,0) (..,0) (y,1) (m,1)] [u] [] [] [] [] [] []
```

a. creat("/m");
b. mkdir("/m");
c. link("/y", "/m");
d. creat("/y/m");
e. None of the above

39) File System State:

```
inode bitmap  11000000
inodes        [d a:0 r:3] [f a:1 r:1] [] [] [] [] [] []
data bitmap   11000000
data          [(.,0) (..,0) (y,1)] [u] [] [] [] [] [] []
```

a. unlink("/m");
b. unlink("/y");
c. unlink("/u");
d. unlink("/y/m");
e. None of the above

40) File System State:
```
inode bitmap  11100000
inodes        [d a:0 r:4] [f a:1 r:1] [f a:-1 r:1] [] [] [] [] []
data bitmap   11000000
data          [(.,0) (..,0) (y,1) (z,2)] [u] [] [] [] [] [] []
```
a. mkdir("/z");
b. creat("/z");
c. creat("/z"); fd=open("/z", O_WRONLY|O_APPEND); write(fd, buf, BLOCKSIZE); close(fd);
d. link("/y", "/z");
e. None of the above

41) File System State:
```
inode bitmap  11110000
inodes        [d a:0 r:5] [f a:1 r:1] [f a:-1 r:1] [d a:2 r:2] [] [] []
data bitmap   11100000
data          [(.,0) (..,0) (y,1) (z,2) (f,3)] [u] [(.,3) (..,0)] [] []
```
a. mkdir("/f");
b. creat("/f");
c. creat("/f"); fd=open("/z", O_WRONLY|O_APPEND); write(fd, buf, BLOCKSIZE); close(fd);
d. link("/y", "/f");
e. None of the above

Part 6. Disks.

Your test will still be multiple choice; imagine there are some good options to pick from…

Consider a disk with the following characteristics:

- Number of surfaces: 2
- Number of tracks / surface: 128 K
- Number of bytes / track: 2 MB
- Number of sectors / track: 1 K

42) How many heads does this disk have?
43) What is the size of each sector?
44) How many bytes per cylinder?
45) What is the total capacity of this disk?

Now assume a stream of requests arrives for this disk; the requests are for cylinder numbers: 20, 5, 8, 9, 2, 30, 35, 40, and 3. Assume the initial position of the disk head is at cylinder 12 and moving towards higher cylinders.

46) With a FCFS scheduling policy, in what order will the requests be serviced?
47) With a FCFS scheduling policy, what is the total seek distance?
48) With a SSTF scheduling policy, in what order will the requests be serviced?
49) With a SSTF scheduling policy, what is the total seek distance?
50) With a SCAN scheduling policy, in what order will the requests be serviced?
51) With a SCAN scheduling policy, what is the total seek distance?
52) With a C-SCAN scheduling policy, in what order will the requests be serviced?
53) With a C-SCAN scheduling policy, what is the total seek distance?

Part 7. FFS-Based File System
Your test will still be multiple choice; imagine there are some good options to pick from…

Consider a UNIX file system based on FFS. Assume that the inode contains 10 direct pointers, 1 indirect pointer, 1 double-indirect pointer, and 1 triple-indirect pointer. Assume the block size is 4~KB and that each pointer requires 4 bytes. Assume that each directory entry requires 32 bytes.

54) What is the largest size file that can be supported with this design?
55) How many files can be placed in a single directory with this design?
56) How many disk reads will be required to read data block 1 of a file named /a/b? Assume that there are only a small number of files in each specified directory. Assume that nothing relevant is in the file cache to begin with, but that blocks are placed in the cache after being read from the disk.
57) Imagine that after reading block 1 of file /a/b for the previous question, block 33 is read next; how many disk reads are required for this second operation, assuming that blocks stay in the file cache after they are read?
58) What are all of the disk blocks that must be **read** when a user creates an empty new c in directory a?
59) What are all of the disk blocks that must be **written** when a user creates an empty new c in directory a?

Part 8. Motivation for Journaling.
Imagine you have an FFS-like file system that is appending to a file and must update an inode, the data bitmap, and a data block. Assume this initial system does not perform any journaling. What happens if a crash occurs after only updating the following block(s)?

60) Bitmap
   a. No inconsistency
   b. Data leak
   c. Another file may re-use data block
   d. Point to garbage
   e. Multiple problems listed above

61) Data
   a. No inconsistency
   b. Data leak
   c. Another file may re-use data block
   d. Point to garbage
   e. Multiple problems listed above

62) inode:
   a. No inconsistency
   b. Data leak
   c. Another file may re-use data block
   d. Point to garbage
   e. Multiple problems listed above

63) bitmap and data:
   a. No inconsistency
   b. Data leak
   c. Another file may re-use data block
   d. Point to garbage
   e. Multiple problems listed above

64) bitmap and inode:
   a. No inconsistency
   b. Data leak
   c. Another file may re-use data block
   d. Point to garbage
   e. Multiple problems listed above

65) data and inode:
   a. No inconsistency
   b. Data leak
   c. Another file may re-use data block
   d. Point to garbage
   e. Multiple problems listed above

Assume a basic implementation of full-data journaling, with a transaction header block and a transaction commit block. Assume the system crashes after the following number of blocks have been written to disk; what will be the state of the file system when it reboots and AFTER recovery is run (if specified by the protocol)?

66) 1 disk write (just transaction header)
   a. No recovery is run; file system in old state
   b. No recovery is run; file system in new state
   c. Recover is run; file system in old state

    d. Recovery is run; file system in new state
    e. Recovery is run; file system may be in old or new state

67) 4 disk writes (transaction header, plus 3 blocks to journal)
    a. No recovery is run; file system in old state
    b. No recovery is run; file system in new state
    c. Recover is run; file system in old state
    d. Recovery is run; file system in new state
    e. Recovery is run; file system may be in old or new state

68) 5 disk writes (transaction header, plus 3 blocks to journal, plus transaction commit)
    a. No recovery is run; file system in old state
    b. No recovery is run; file system in new state
    c. Recover is run; file system in old state
    d. Recovery is run; file system in new state
    e. Recovery is run; file system may be in old or new state

69) 6 disk writes(5 above plus 1 checkpoint block)
    a. No recovery is run; file system in old state
    b. No recovery is run; file system in new state
    c. Recover is run; file system in old state
    d. Recovery is run; file system in new state
    e. Recovery is run; file system may be in old or new state

70) How many disk writes must occur for the system to not need to perform recovery after the crash and for the file system to reflect the new state?
    a. 6
    b. 7
    c. 8
    d. 9
    e. None of the above