CS 537: Introduction to Operating Systems
Fall 2023: Midterm Exam #3

This exam is closed book, closed notes.

No calculators may be used. All cell phones must be turned off and put away.

You have 1 hour 15 minutes to complete this exam.

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

- CANVAS LAST NAME - fill in your last (family) name starting at the leftmost column
- CANVAS FIRST NAME - fill in the first five letters of your first (given) name
- IDENTIFICATION NUMBER - This is your UW Student WisCard ID number
- ABC of SPECIAL CODES - Write your lecture number as a three digit value:
    - 001 (Morning Lecture)
    - 002 (Afternoon Lecture)
    - 010 (Epic Lecture)

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

You may separate the pages of this exam if it helps you.

Unless stated (or implied) otherwise, you should make the following assumptions:

- The OS manages a single uniprocessor (single core)
- All memory is byte addressable
- The terminology lg means $\log_2$
- 2^{10} bytes = 1KB
- 2^{20} bytes = 1MB
- Data is allocated with optimal alignment, starting at the beginning of a page
- Assume leading zeros can be removed from numbers (e.g., 0x06 == 0x6).
- Hex numbers are represented with a proceeding “0x”

This exam has 60 questions. Each question has the same number of points.

Good luck!

Part 1. Designate if the statement is True (A) or False (B).

1. Device driver code is software that executes on the microcontroller of a peripheral device.

2. Using Programmed IO lowers CPU overhead.

3. Device drivers have come to represent a huge portion of kernel code.

4. With a 15,000 RPM disk, the expected rotation time for a random access is 4ms.

5. Transfer time is significantly higher for random accesses compared to sequential accesses.

6. One difference between the SCAN and C-SCAN algorithms is that SCAN reads/writes data when the head is traveling in either direction.

7. A non-work conserving scheduler may not schedule available requests even when the disk is idle.

8. A disk cylinder is composed of all tracks on both surfaces of a platter.

9. The Shortest Seek Time First algorithm suffers from starvation.

10. The SPTF algorithm is usually performed inside a drive.

11. A RAID-1 system has a capacity that is the same as using the disks without being in a RAID.

12. Striping in a RAID is a way to design the system to extract the most parallelism when requests are made to contiguous chunks.

13. RAID-4 has better capacity that RAID-1 and better reliability than RAID-0.

14. One disadvantage of RAID-4 is that it cannot reconstruct the necessary data if the parity disk fails.

15. The capacity of RAID-4 and RAID-5 are identical.

16. The throughput rate for random writes in a RAID-5 system is double that of a RAID-4 system.

17. The latency of a write in a RAID-5 system is twice that of a single disk because it requires doing both a read and a write for each logical write.

18. In an FFS-like file system, multiple inodes may point to the same file descriptor.

19. An inode structure typically contains a field indicating the time at which this file was last accessed.

20. An inode structure can contain pointers to directory data.

21. FFS attempts to put inodes from files in the same directory in the same cylinder group.

22. After a crash, FSCK fixes the on-disk state of the file system to match its state before the last update to disk began.

23. After the journal commit block is successfully written to disk, a journaling file system can then checkpoint the relevant blocks to their final in-place positions.

24. The difference between data journaling and metadata (order) journaling is the order that blocks are copied from the journal to their final in-place positions.

25. When LFS writes a new copy of an inode to a segment, it also writes a new copy of all data that the inode points to.

26. LFS periodically checkpoints imaps to a known location on disk (alternating between two locations to withstand crashes).

27. When performing garbage collection, LFS determines that a data block is valid by scanning all valid inodes from the imap and verifying that one of the valid inodes points to this location.

28. SSDs erase at the page level but read and program at the block level.

29. Latent sector errors are more common than block corruption in cheap disks.

30. One way of detecting lost writes is to place the checksum elsewhere than with the data block.

Part 2. Disks -- select the one best answer, A - E.

Consider a disk with the following characteristics:

• Number of surfaces: 8 (=2^3)
• Number of tracks / surface: 2 M (=2^21)
• Number of bytes / track: 4 MB (=2^22 bytes)
• Number of sectors / track: 4 K (=2^12)

31. How many read/write heads does this disk have?
A. 1
B. 2
C. 4
D. 8
E. Not enough information or none of the above

32. What is the size of each sector?
A. 512 bytes
B. 1 KB
C. 2 KB
D. 4 KB
E. Not enough information or none of the above

33. How many bytes per cylinder?
A. 1 MB
B. 8 MB
C. 32 MB
D. 64 MB
E. Not enough information or none of the above

34. What is the total capacity of this disk?
A. 2^42 bytes
B. 2^43 bytes
C. 2^45 bytes
D. 2^46 bytes
E. Not enough information or none of the above

Assume that the disk head is at cylinder 12 and moving towards higher cylinders. Assume a stream of requests arrives for the following cylinders: 7, 22, 3, 62, 1, 4, 90, 2, 20, 40, 5, 70.

35. With a FCFS scheduling policy, what is the total seek distance to service all the requests?
A. 586
B. 534
C. 507
D. 474
E. Not enough information or none of the above

36. With a SSTF scheduling policy, what is the total seek distance?
A. 100
B. 237
C. 493
D. 501
E. Not enough information or none of the above

37. With a SCAN scheduling policy, what is the total seek distance?
A. 123
B. 165
C. 217
D. 435
E. Not enough information or none of the above

38. With a C-SCAN scheduling policy (servicing only when head is moving towards higher cylinders), what is the total seek distance?
A. 167
B. 173
C. 223
D. 312
E. Not enough information or none of the above

Part 3. Basic File System Operations and Data Structures -- select the one best answer, A - E.

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 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 indicated by the contents of four different data structures:

inode bitmap - indicates which inodes are allocated (not shown because not needed for questions)
inodes - table of inodes and their contents
data bitmap - indicates which data blocks are allocated (not shown)
data - indicates contents of data blocks

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].

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.

Empty inodes and empty data blocks may not all be 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 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:

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

If the file system transitions into each of the following states, what operation or operations must have been performed? The state of the file system is cumulative across questions.

39. File System State:

```
inodes [d a:0 r:2] [f a:-1 r:1] [] [] [] []
data [(.,0) (..,0) (c,1)] [] [] []
```

A. mkdir("/c");
B. creat("/c");
C. link("/","/c");
D. mkdir("/c/d");
E. None of the above

40. File System State:

```
inodes [d a:0 r:3] [f a:-1 r:1] [d a:1 r:2] [] [] []
data [(.,0) (..,0) (c,1) (q,2)] [(.,2) (..,0)] [] []
```

A. mkdir("/q");
B. creat("/c/q");
C. link("/","/q");
D. link("/c","/q");
E. None of the above

41. File System State:

```
inodes [d a:0 r:3] [f a:-1 r:1] [d a:1 r:2] [f a:-1 r:1] [] []
data [(.,0) (..,0) (c,1) (q,2)] [(.,2) (..,0) (c,3)] [] []
```

A. creat("/c/q");
B. link("/c","/q");
C. mkdir("/q/c");
D. unlink("/q");
E. None of the above

42. File System State after **TWO** operations:

```
inodes [d a:0 r:3] [f a:-1 r:1] [d a:1 r:2] [f a:2 r:2] [] []
data [(.,0) (..,0) (c,1) (q,2) (r,3)] [(.,2) (..,0) (c,3)] [v] []
```

A. creat("/r");
fd=open("/q/c", O_WRONLY|O_APPEND); write(fd,buf,BLOCKSIZE);

B. mkdir("/r");
link("/q/c","/r");

C. link("/q/c","/r");
fd=open("/q/c", O_WRONLY|O_APPEND); write(fd,buf,BLOCKSIZE); close(fd);

D. creat("/r");
creat("/q/c");

E. None of the above

43. File System State after **TWO** operations:

```
inodes [d a:0 r:3] [d a:3 r:2] [d a:1 r:2] [f a:2 r:2] [] []
data [(.,0) (..,0) (q,2) (r,3) (s,1)] [(.,2) (..,0) (c,3)] [v] [(.,1),(..,0)]
```

A. link("/c","/s");
mkdir("/v");

B. creat("/s");
unlink("/c");

C. unlink("/c");
mkdir("/s");

D. mkdir("/s");
creat("/q/c");

E. None of the above

Part 4. Crash Consistency and Journaling -- select the one best answer, A - E.

Imagine you have an FFS-like file system that is creating a new empty file in an existing directory and must update 4 blocks: the directory inode, the directory data block, the file inode, and the inode bitmap. Assume the directory inode and the file inode are in different on-disk blocks. Assume this initial system does not perform any journaling and FSCK is not run. What happens if a crash occurs after only updating the following block(s)?

44. Bitmap
A. No inconsistency (it simply appears that the operation was not performed)
B. Data/inode leak
C. Multiple file paths may point to same inode
D. Point to garbage
E. Multiple problems listed above

45. File inode
A. No inconsistency (it simply appears that the operation was not performed)
B. Data/inode leak
C. Multiple file paths may point to same inode
D. Point to garbage
E. Multiple problems listed above

46. Directory inode and Directory data
A. No inconsistency (it simply appears that the operation was not performed)
B. Data/inode leak
C. Multiple file paths may point to same inode
D. Point to garbage
E. Multiple problems listed above

47. Bitmap and File inode
A. No inconsistency (it simply appears that the operation was not performed)
B. Data/inode leak
C. Multiple file paths may point to same inode
D. Point to garbage
E. Multiple problems listed above

48. Bitmap and Directory inode and Directory data
A. No inconsistency (it simply appears that the operation was not performed)
B. Data/inode leak
C. Multiple file paths may point to same inode
D. Point to garbage
E. Multiple problems listed above

49. File inode and Directory inode and Directory data
A. No inconsistency (it simply appears that the operation was not performed)
B. Data/inode leak
C. Multiple file paths may point to same inode
D. Point to garbage
E. Multiple problems listed above

Now assume a basic implementation of metadata journaling has been added to the file system and the same file create operation is performed. Assume a transaction header block and a transaction commit block are used. Assume each block is written in its entirety (no crashes in the middle of writing a block) and only one block is written at a time and completes before the next block is written. If the system crashes after the following number of blocks have been written to disk, what will happen after the system reboots? (If the number of disk writes exceeds those needed, assume they are unrelated.)

50. 1 disk write (hint: just the transaction header block is written to disk)
A. No transactions replayed during recover; file system in old state
B. No transactions replayed during recover; file system in new state
C. Transaction replayed during recover; file system in old state
D. Transaction replayed during recover; file system in new state
E. Transaction replayed during recover; file system in unknown state

51. 4 disk writes (hint: transaction header, plus 3 additional blocks to journal)
A. No transactions replayed during recover; file system in old state
B. No transactions replayed during recover; file system in new state
C. Transaction replayed during recover; file system in old state
D. Transaction replayed during recover; file system in new state
E. Transaction replayed during recover; file system in unknown state

52. 6 disk writes
A. No transactions replayed during recover; file system in old state
B. No transactions replayed during recover; file system in new state
C. Transaction replayed during recover; file system in old state
D. Transaction replayed during recover; file system in new state
E. Transaction replayed during recover; file system in unknown state

53. 10 disk writes
A. No transactions replayed during recover; file system in old state
B. No transactions replayed during recover; file system in new state
C. Transaction replayed during recover; file system in old state
D. Transaction replayed during recover; file system in new state
E. Transaction replayed during recover; file system in unknown state

54. 11 disk writes
A. No transactions replayed during recover; file system in old state
B. No transactions replayed during recover; file system in new state
C. Transaction replayed during recover; file system in old state
D. Transaction replayed during recover; file system in new state
E. Transaction replayed during recover; file system in unknown state

Part 5. Data integrity, LFS, and SSDs -- select the one best answer, A - E.

A 4 byte block is written to disk with the data 0xB5C11103 (binary 1011 0101 1100 0001 0001 0001 0000 0011) and its checksum value.

55. What is the 1 byte XOR checksum value?
A. 0x8A (binary 1000 1010)
B. 0x66 (binary 0110 0110)
C. 0x8B, 0x41 (binary 1000 1011, 0100 0001)
D. 0x1B, 0x7F (binary 0001 1011 0111 1111)
E. None of the above

56. What is the 1 byte addition (ignoring overflow) checksum value?
A. 0x8A (binary 1000 1010)
B. 0x66 (binary 0110 0110)
C. 0x8B, 0x41 (binary 1000 1011, 0100 0001)
D. 0x1B, 0x7F (binary 0001 1011 0111 1111)
E. None of the above

57. Storing a physical identifier along with the checksum can be used to detect what type of errors?
A. Latent Sector Error
B. Misdirected Write
C. Block corruption
D. Lost write
E. None of the above

An LFS system contains the following initial state:

INITIAL file system contents:
```
[   0 ] live checkpoint: 3 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
[   1 ] live [.,0] [..,0] -- -- -- -- -- --
[   2 ] live type:dir size:1 refs:2 ptrs: 1 -- -- -- -- -- -- --
[   3 ] live chunk(imap): 2 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
```

Notice that the checkpoint region lives in block 0 and initially contains the 0th entry pointing to the 3rd data block. This data block holds the first 16 entries (index 0 through 15) of the imap, with inode 0 referring to block 2. Block 2 is the inode of the root directory of the file system and this inode's data pointers refer to block 1. Block 1 is the data for the directory, which initially has directory entries for "." and ".." which both refer back to inode 0. also notice that all data blocks are marked live because they are reachable (i.e. connected into the file system).

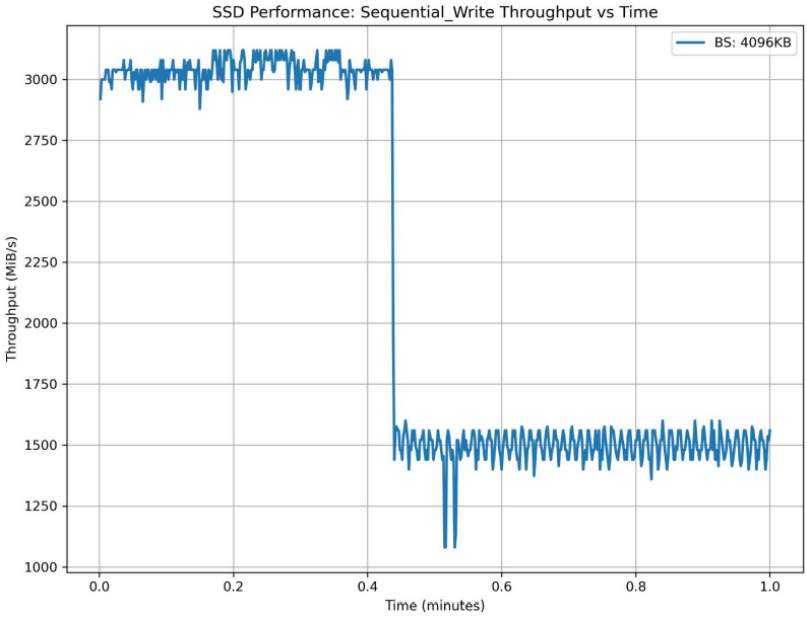
An I/O operation is performed on the LFS which modifies the system contents to the final system state:

FINAL file system contents:
```
[   0 ]  ?   checkpoint: 7 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
[   1 ]  ?   [.,0] [..,0] -- -- -- -- -- --
[   2 ]  ?   type:dir size:1 refs:2 ptrs: 1 -- -- -- -- -- -- --
[   3 ]  ?   chunk(imap): 2 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
[   4 ]  ?   [.,0] [..,0] [ku3,1] -- -- -- -- --
[   5 ]  ?   type:dir size:1 refs:2 ptrs: 4 -- -- -- -- -- -- --
[   6 ]  ?   type:reg size:0 refs:1 ptrs: -- -- -- -- -- -- -- --
[   7 ]  ?   chunk(imap): 5 6 -- -- -- -- -- -- -- -- -- -- -- -- -- --
```

58. What I/O operation was performed?
A. creat("/ku3");
B. mkdir("/ku3");
C. link("/","/ku3");
D. unlink("/ku3");
E. None of the above

59. Which data blocks are live?
A. 0, 3, 4, 5, 6, 7
B. 0, 1, 2, 3, 4, 5, 6, 7
C. 0, 3, 5, 6, 7
D. 0, 4, 5, 6, 7
E. None of the above

60. A 2TB Samsung 970 EVO Plus SSD contains ~1.9TB of TLC with 78 GB of SLC Cache. It is benchmarked by doing sequential writes and observing the throughput rate. The plot below is the result of this benchmarking.



What is the most likely reason for the dramatic drop in throughput rate just afer 0.4 minutes?

A. The FTL's mapping table is full and must resort to a backup mapping table
B. A block is fully programmed and a new block must be erased before it can be programmed
C. The SLC is filled and now data must be transferred at the slower TLC rate
D. A page is fully programmed and a new page must be erased before it can be programmed
E. The disk is full and garbage collection starts running

That's it. Have a great break!