

Started on	Friday, 31 March 2023, 6:18 PM
State	Finished
Completed on	Friday, 31 March 2023, 6:59 PM
Time taken	41 mins 4 secs
Grade	6.55 out of 15.00 (43.63%)

Question **1**

Incorrect

Mark 0.00 out of 1.00

Select all the actions taken by ilock()

- ☐ a. Copy the on-disk inode into in-memory inode, if needed
- ☒ b. Get the inode from the inode-cache ❌
- ☒ c. Lock all the buffers of the file in memory ❌
- ☐ d. Mark the in-memory inode as valid, if needed
- ☒ e. Read the inode from disk, if needed ✔️
- ☐ f. Take the sleeplock on the inode, always
- ☐ g. Take the sleeplock on the inode, optionally

Your answer is incorrect.

The correct answers are: Read the inode from disk, if needed, Copy the on-disk inode into in-memory inode, if needed, Take the sleeplock on the inode, always, Mark the in-memory inode as valid, if needed

Question **2**

Correct

Mark 1.00 out of 1.00

Map the function in xv6's file system code, to it's perceived logical layer.

ialloc	inode	✓
namei	pathname lookup	✓
balloc	block allocation on disk	✓
filestat()	file descriptor	✓
skipelem	pathname lookup	✓
stati	inode	✓
sys_chdir()	system call	✓
ideintr	disk driver	✓
dirlookup	directory	✓
bread	buffer cache	✓
commit	logging	✓
bmap	inode	✓

Your answer is correct.

The correct answer is: ialloc → inode, namei → pathname lookup, balloc → block allocation on disk, filestat() → file descriptor, skipelem → pathname lookup, stati → inode, sys_chdir() → system call, ideintr → disk driver, dirlookup → directory, bread → buffer cache, commit → logging, bmap → inode

Question 3

Partially correct

Mark 1.25 out of 2.00

Marks the statements as True/False w.r.t. "struct buf"

True	False		
<input checked="" type="radio"/>	<input type="radio"/>	A buffer can be both on the MRU/LRU list and also on idequeue list.	✓
<input checked="" type="radio"/>	<input type="radio"/>	The buffers are maintained in LRU order, in the function brelse	✗
<input checked="" type="radio"/>	<input type="radio"/>	B_DIRTY flag means the buffer contains modified data	✓
<input type="radio"/>	<input checked="" type="radio"/>	The "next" pointer chain gives the buffers in LRU order	✓ No. MRU order.
<input type="radio"/>	<input checked="" type="radio"/>	A buffer can have both B_VALID and B_DIRTY flags set	✗ only one will be set
<input type="radio"/>	<input checked="" type="radio"/>	B_VALID means the buffer is empty and can be reused	✗ No. it means it contains data, same as the data on disk
<input checked="" type="radio"/>	<input type="radio"/>	Lock on a buffer is acquired in bget, and released in brelse	✓
<input checked="" type="radio"/>	<input type="radio"/>	The reference count (refcnt) in struct buf is = number of processes accessing the buffer	✓

A buffer can be both on the MRU/LRU list and also on idequeue list.: True

The buffers are maintained in LRU order, in the function brelse: True

B_DIRTY flag means the buffer contains modified data: True

The "next" pointer chain gives the buffers in LRU order: False

A buffer can have both B_VALID and B_DIRTY flags set: False

B_VALID means the buffer is empty and can be reused: False

Lock on a buffer is acquired in bget, and released in brelse: True

The reference count (refcnt) in struct buf is = number of processes accessing the buffer: True

Question **4**

Partially correct

Mark 0.50 out of 1.00

Suppose an application on xv6 does the following:

```
int main() {  
    char arr[128];  
    int fd = open("README, O_RDONLY);  
    read(fd, arrr, 100);  
}
```

Assume that the code works.

Which of the following things are true about xv6 kernel code, w.r.t. the above C program.

True	False		
<input checked="" type="radio"/>	<input type="radio"/>	The loop in readi() will always read a different block using bread()	<input checked="" type="radio"/> No. the offsets can overlap over the same block
<input checked="" type="radio"/>	<input type="radio"/>	value of fd will be 3	<input checked="" type="radio"/>
<input checked="" type="radio"/>	<input type="radio"/>	The data is transferred from disk to kernel buffers first, and then address of arr is mapped to the kernel buffers	<input checked="" type="radio"/> No. data is copied into arr.
<input checked="" type="radio"/>	<input type="radio"/>	The ONLY function that gets called on return devsw[ip->major].read(ip, dst, n); is consoleread	<input checked="" type="radio"/> There is no other device file in xv6
<input checked="" type="radio"/>	<input type="radio"/>	The "memmove(dst, bp->data + off%BSIZE, m);" in readi() will copy the data from the disk to the kernel buffers	<input checked="" type="radio"/> It will transfer from kernel buffer to user memory (arr)
<input checked="" type="radio"/>	<input type="radio"/>	The process will be made to sleep only once	<input checked="" type="radio"/> Yes. Reading 100 bytes means reading only one disk block. So bread()->iderw() is called only once.

The loop in readi() will always read a different block using bread(): False

value of fd will be 3: True

The data is transferred from disk to kernel buffers first, and then address of arr is mapped to the kernel buffers: False

The ONLY function that gets called on return devsw[ip->major].read(ip, dst, n); is consoleread: True

The "memmove(dst, bp->data + off%BSIZE, m);" in readi() will copy the data from the disk to the kernel buffers: False

The process will be made to sleep only once: True

Question **5**

Partially correct

Mark 0.50 out of 1.00

Match function with it's functionality

dirlookup	Lookup (search) for a given directory	✗
dirlink	Link a directory with another directory	✗
namex	return in-memory inode for a given pathname	✓
nameiparent	return in-memory inode for parent directory of a given pathname	✓

Your answer is partially correct.

You have correctly selected 2.

The correct answer is: dirlookup → Search a given name in a given directory, dirlink → Write a new entry in a given directory, namex → return in-memory inode for a given pathname, nameiparent → return in-memory inode for parent directory of a given pathname

Question **6**

Correct

Mark 1.00 out of 1.00

An inode is read from disk as a part of this function

- ☐ a. readi
- ☒ b. ilock ✓
- ☐ c. iget
- ☐ d. iread
- ☐ e. sys_read

Your answer is correct.

The correct answer is: ilock

Question **7**

Incorrect

Mark 0.00 out of 1.00

Maximum size of a file on xv6 in **bytes** is

(just write a numeric answer)

Answer: 512 ✗

The correct answer is: 71680

Question **8**

Incorrect


Mark 0.00 out of 1.00

The lines

```
if(ip->type != T_DIR){  
    iunlockput(ip);  
    return 0;  
}
```

in namex() function

mean

- ☐ a. No directory entry was found for the file to be opened, hence an error
- ☐ b. One of the sub-components on the given path name, was a directory, but it was not supposed to be a directory, hence an error
- ☒ c. The last path component (which is a file, and not a directory) has been resolved, so release the lock (using iunlockput) and return 
- ☐ d. ilock is held on the inode, and hence it's an error if it is a directory
- ☐ e. One of the sub-components on the given path name, did not exist, hence it's an error
- ☐ f. One of the sub-components on the given path name, was not a directory, hence it's an error
- ☐ g. There was a syntax error in the pathname specified

Your answer is incorrect.

The correct answer is: One of the sub-components on the given path name, was not a directory, hence it's an error


Question **9**

Incorrect

Mark 0.00 out of 1.00

Note: for this question you get full marks if you select all and only correct options, you get ZERO if at least one option is wrong or not selected.

Select all the correct statements about log structured file systems.

- ☐ a. log structured file systems considerably improve the recovery time
- ☐ b. ext2 is by default a log structured file system
- ☒ c. xv6 has a log structured file system 
- ☐ d. ext4 is a log structured file system
- ☐ e. file system recovery recovers all the lost data

Your answer is incorrect.


The correct answers are: xv6 has a log structured file system, log structured file systems considerably improve the recovery time

Question **10**

Incorrect

Mark 0.00 out of 1.00

Select all the actions taken by `iget()`

- ☐ a. Returns an inode with given dev+inode-number from cache, if it exists in cache
- ☐ b. Returns the inode with reference count incremented
- ☒ c. Returns the inode with inode-cache lock held 
- ☐ d. Returns a valid inode if not found in cache
- ☐ e. Returns the inode locked
- ☐ f. Returns a free-inode , with dev+inode-number set, if not found in cache
- ☐ g. Panics if inode does not exist in cache

Your answer is incorrect.

The correct answers are: Returns an inode with given dev+inode-number from cache, if it exists in cache, Returns the inode with reference count incremented, Returns a free-inode , with dev+inode-number set, if not found in cache

Question 11

Partially correct

Mark 0.67 out of 1.00

Compare XV6 and EXT2 file systems.

Select True/False for each point.

True	False		
<input checked="" type="radio"/>	<input type="radio"/>	Ext2 contains group descriptors but xv6 does not	✓
<input type="radio"/>	<input checked="" type="radio"/>	In both ext2 and xv6, the superblock gives location of first inode block	✗
<input type="radio"/>	<input checked="" type="radio"/>	Ext2 contains superblock but xv6 does not.	✓
<input type="radio"/>	<input checked="" type="radio"/>	Both xv6 and ext2 contain magic number	✗
<input checked="" type="radio"/>	<input type="radio"/>	xv6 contains journal, ext2 does not	✓
<input type="radio"/>	<input checked="" type="radio"/>	xv6 contains inode bitmap, but ext2 does not	✓

Ext2 contains group descriptors but xv6 does not: True

In both ext2 and xv6, the superblock gives location of first inode block: False

Ext2 contains superblock but xv6 does not.: False

Both xv6 and ext2 contain magic number: False

xv6 contains journal, ext2 does not: True

xv6 contains inode bitmap, but ext2 does not: False

Question **12**

Partially correct

Mark 0.20 out of 1.00

Arrange the following in their typical order of use in xv6.

1. ☒ iget
2. ☐ ilock
3. ☐ iput
4. ☐ iunlock
5. ☐ use inode

Your answer is partially correct.

Grading type: Relative to the next item (including last)

Grade details: 1 / 5 = 20%

Here are the scores for each item in this response:

1. 1 / 1 = 100%
2. 0 / 1 = 0%
3. 0 / 1 = 0%
4. 0 / 1 = 0%
5. 0 / 1 = 0%

The correct order for these items is as follows:

1. iget
2. ilock
3. use inode
4. iunlock
5. iput

Question **13**

Partially correct

Mark 1.43 out of 2.00

Select T/F w.r.t physical disk handling in xv6 code

True	False		
<input checked="" type="radio"/>	<input type="radio"/>	log is kept on the same device as the file system	✓
<input checked="" type="radio"/>	<input type="radio"/>	only 2 disks are handled by default	✓
<input type="radio"/>	<input checked="" type="radio"/>	only direct blocks are supported	✗
<input checked="" type="radio"/>	<input type="radio"/>	The code supports IDE, and not SATA/SCSI	✓
<input checked="" type="radio"/>	<input type="radio"/>	the superblock does not contain number of free blocks	✓
<input checked="" type="radio"/>	<input type="radio"/>	disk driver handles only one buffer at a time	✓
<input type="radio"/>	<input checked="" type="radio"/>	device files are not supported	✗

log is kept on the same device as the file system: True

only 2 disks are handled by default: True

only direct blocks are supported: False

The code supports IDE, and not SATA/SCSI: True

the superblock does not contain number of free blocks: True

disk driver handles only one buffer at a time: True

device files are not supported: False

◀ Random Quiz - 5: xv6 make, bootloader, interrupt handling, memory management

Jump to...



(Random Quiz - 7) Pre-Endsem Quiz ▶