Question 1. [6 MARKS]

Each of the following statements is false. In one sentence, explain why. (If you disagree that the statement is false, provide your reasoning.)

Part (a) [1 MARK]

A system call is invoked in the same manner as a user function.

A system call traps into the kernel so that privileged instructions can be run. A function call just jumps to another instruction in user-mode.

(Might say, switch to kernel-mode, software interrupt, syscall instruction.)

Part (b) [1 MARK]

If two threads access a shared variable at the same time, there will be a concurrency error.

It is fine for two threads to load a shared variable at the same time. They will both see the right value. It may also be fine if only one thread is guaranteed to be updating the shared variable, and the timing of when the second thread looks at the value doesn't matter.

Part (c) [1 MARK]

Multiple threads in the same process share all code and data resources.

Each thread has its own stack.

Part (d) [1 MARK]

A context switch from one process to another can be accomplished without executing OS code in kernel mode.

The kernel needs to select the next process to run.

Part (e) [1 MARK]

External fragmentation occurs when a file is broken up into many pieces on a disk.

No external fragmentation

Part (f) [1 MARK]

It is important to store blocks of a file close together on a Solid State Drive to speed up sequential access.

One of the benefits of SSDs is uniform access time for all blocks on the drive.

Question 2. [7 MARKS]

Part (a) [1 MARK]

Briefly explain what causes a thread to move from the ready state to running state. *The kernel dispatches the process.*

Part (b) [1 MARK]

Briefly explain what causes a thread to move from the running state to the blocked state. *A system call or signal my cause the running process to block.*

Part (c) [1 MARK]

Briefly explain what causes a thread to move from the blocked state to the ready state. *The event that caused the process to block occurs. E.g. Data is now available for a read call, a thread blocked on a condition variable is awakened.*

Part (d) [2 MARKS]

Identify and briefly explain two types of operations that result in a mode switch from user to kernel mode.

timer interrupt, system call

Part (e) [2 MARKS]

Give one concrete example of how the operating system virtualizes the hardware resources that we have seen so far in the course. Identify which resource is virtualized and how the operating system achieves this virtualization.

There are several possibilities here. The OS virtualizes the CPU. A process runs without knowing about context switches. The file API virtualizes the disk. Some explanation is necessary for full marks.

Question 3. [2 MARKS]

Consider the following excerpt from the solution to the reader/writer problem using semaphores:

```
// assume all variables are correctly initialized

sem_wait(mutex);
readcount += 1;

if(readcount == 1)
    sem_wait(w_or_r);

sem_signal(mutex);
```

Which of the following statements are true? *Note: This question wasn't entirely clear because the rest of the code for the reader/write problem was not given.*

- ☐ `readcount` must be 1 when `sem_wait(w_or_r)` returns. *Assuming knowledge of the reader/writer problem, this is true because readcount is only changed when mutex is held. If the assumption is that other threads might call sem_signal(mutex), then this coud be false*
- ☐ `readcount` cannot be 0 when `sem_wait(w_or_r)` returns. *Same reasoning as above*
- ☒ If thread A is blocked in `sem_wait(w_or_r)` other threads will block on `sem_wait(mutex)` because thread A "holds" the mutex. *This is true under normal interpretations of wait and signal and the expectation that the rest of the code is written correctly.*
- ☒ Multiple threads might pass through this "critical section" while a thread is blocked in `sem_wait(w_or_r)` *False under normal interpretations of semaphore wait and signal.*

Question 4. [6 MARKS]

A program issues the following system calls on the original Fast File System (FFS) file system. The file system block size is 4KB and the size of an inode is 128 bytes. Assume that the file system is already mounted when the program starts and that all system calls succeed.

```
char buf[4096];
int fd = open("/cs/dw.txt", 0); // open in read-only mode
lseek(fd, (13*4096) + 2048, 0);    // seek to position (13*4096)+2048 from start of file
read(fd, buf, 4096);        // read 4k of data from file
```

Assume that the file buffer cache is empty. State the *minimum* number of the following types of blocks that will be read when the program above is run. Explain your answer for each block type.

1. Inode block(s) *1 - we need 4 inodes (for root dir, dir a, dir b, and file c) but at 64 bytes per inode, they can all fit in 1 block if we are lucky.*

2. Directory block(s) *2 - each dir has its own data blocks, so we need 1 for each dir (root, a and b)*

3. Indirect block(s) (include single, double or triple indirect) *1 - Original Unix FS had 12 direct pointers, 4 byte block addresses, so indirect block holds 1024 block ptrs, so blocks 12-1035 can be read with 1 indirect block. We are reading block 13.*

4. Other data block(s) *2 - read crosses a block boundary*

Question 5. File system consistency [9 MARKS]

On an ext2 or FFS (Fast File System) file system, consider the operation of removing a file of size 1024 bytes. Assume the directory that contains the file occupies one block. Assume the directory inode and the file inode are in different disk blocks.

Part (a) [4 MARKS]

Which of the following blocks must be updated? Check all that apply. For each block that must be updated, explain briefly how the values in the block change.

- ☒ inode bitmap
- ☒ data block bitmap
- ☐ file inode
- ☐ file data block
- ☒ directory inode
- ☒ directory data block

Part (b) [1 MARK]

What data is updated in the directory inode?

last modified time

Part (c) [4 MARKS]

In each of the remaining questions, check all of the boxes that most closely explain what happens if a crash occurs after updating only the block(s) specified.

Inode Bitmap

- ☐ No inconsistency (it simply appears that the operation was not performed)
- ☐ Data leak (data block is lost for any future use)
- ☐ Inode leak (Inode is lost for any future use)
- ☒ Multiple file paths may point to same inode
- ☐ Inconsistent inode data (Some inode field does not match what is stored in data blocks)
- ☐ Something points to garbage

Data Block Bitmap

- ☐ No inconsistency (it simply appears that the operation was not performed)
- ☐ Data leak (data block is lost for any future use)
- ☐ Inode leak (Inode is lost for any future use)
- ☐ Multiple file paths may point to same inode
- ☐ Inconsistent inode data (Some inode field does not match what is stored in data blocks)
- ☒ Something points to garbage

Directory Inode

- ☐ No inconsistency (it simply appears that the operation was not performed)
- ☐ Data leak (data block is lost for any future use)
- ☐ Inode leak (Inode is lost for any future use)
- ☐ Multiple file paths may point to same inode
- ☒ Inconsistent inode data (Some inode field does not match what is stored in data blocks)
- ☐ Something points to garbage

Directory inode and Directory data

- ☐ No inconsistency (it simply appears that the operation was not performed)
- ☒ Data leak (data block is lost for any future use)
- ☒ Inode leak (Inode is lost for any future use)
- ☐ Multiple file paths may point to same inode
- ☐ Inconsistent inode data (Some inode field does not match what is stored in data blocks)
- ☐ Something points to garbage