

**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 trap instruction is a privileged operation.

**Part (b)** [1 MARK]

Locks and semaphores are identical structures.

**Part (c)** [1 MARK]

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

**Part (d)** [1 MARK]

Limited direct execution means that once the OS has prepared a process for execution, the process has full control of the CPU.

**Part (e)** [1 MARK]

Indexed-based file systems suffer from external fragmentation because blocks of the file may be scattered across the disk.

**Part (f)** [1 MARK]

Extent-based file systems may require more disk block accesses than indexed-based files systems for random access because it may need to traverse an extent to get to a particular byte in the file.

**Question 2.** [7 MARKS]**Part (a)** [3 MARKS]

Name 3 fields stored in the data structure used to keep track of processes in the kernel.

**Part (b)** [2 MARKS]

Describe what state needs to be saved for a process during a context switch.

**Part (c)** [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.

**Question 3.** [2 MARKS]

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

```
// assume all variable are correctly initialized
// assume writing will only ever hold the values 0 or 1

pthread_mutex_lock(&mutex);
while(writing) {
    pthread_cond_wait(&turn, &mutex);
}
readcount++;
pthread_mutex_unlock(&mutex);
```

Which of the following statements are true?

- ☐ `writing` must be 0 when `pthread_cond_wait` returns.
- ☐ The mutex is held by this thread while it is blocked in `pthread_cond_wait`
- ☐ The mutex is held by this thread whenever it checks the value of `writing`
- ☐ The mutex is held by this thread when the while loop terminates.

**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 64 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("/a/b/c", 0); // open in read-only mode
lseek(fd, 1034*4096, 0);    // seek to position (1034*4096) 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)
2. Directory block(s)
3. Indirect block(s) (include single, double or triple indirect)
4. Other data block(s)

**Question 5.** File system consistency [9 MARKS]

On an ext2 or FFS file system, consider the operation of creating a new directory in an existing directory. Assume the existing directory occupies one block and there is enough space to add a new entry. Assume the existing directory inode and the new directory inode are in different disk blocks.

**Part (a)** [2 MARKS]

Which of the following blocks must be updated? Check all that apply.

- ☐ inode bitmap
- ☐ new directory inode
- ☐ new directory inode
- ☐ new directory data block
- ☐ existing directory inode
- ☐ existing directory data block

**Part (b)** [2 MARKS]

What data is updated in the existing directory inode?

*last modified time, number of links*

**Part (c)** [5 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 and 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)
- ☐ Inconsistent inode data (Some inode field does not match what is stored in data blocks)
- ☐ Multiple file paths may point to same inode
- ☐ Something points to garbage

**New 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

**Inode Bitmap, Data Block Bitmap, Existing Directory data, New Directory inode, and New 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

**Inode Bitmap and New 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

**New Directory inode, Existing Directory inode, and Existing 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