Entry Number

Name

Question 1

Not yet answered

Marked out of 4.00

When an ext3 transaction is closed, all atomic operations that started before the closing time are a part of the previous transaction (say T1) and all atomic operations that started after the closing time are made a part of the next transaction (say T2). But to maintain serializability, any atomic operation of T2 must wait (before starting) for all atomic operations that are a part of T1 to finish. How long can this waiting period for atomic operations of T2 be?

Select one or more:

- ☐ a. This wait period would include the time required to commit the T1 transaction (on disk) but it would not include the time take for applying the modifications of T1 to the FS tree
- ☐ b. There is no need for such a waiting period. T2's atomic operations can start immediately (without having to wait for T1's atomic operations to finish.
- ☐ c. This wait period would include the time required to commit the T1 transaction (on disk) and applying the modifications of T1 to the FS tree
- ☐ d. This wait period would neither include the time required to commit the T1 transaction (on disk) nor would it include the time take for applying the modifications of T1 to the FS tree. The wait period only includes the time to update the in-memory buffers (for completing T1's atomic operations)

Question 2

Not yet answered

Marked out of 4.00

Which of the following statements are true about the context-switch function in xv6 (that switches the kernel context from one process to another)?

Select one or more:

- ☐ a. If the context-switch function saves all registers (irrespective of whether it is a callee-save register or a caller-save register), then it would be incorrect, i.e., it can cause the kernel to crash. Assume that all the other code in xv6 is unchanged.
- ☐ b. The context-switch function only needs to save the caller-save registers
- ☐ c. The context-switch function only needs to save the callee-save registers
- ☐ d. If the context-switch function saves only one register, e.g., the eax register (but not the other registers), then it would be incorrect, i.e., it can cause the kernel to crash. Assume that all the other code in xv6 is unchanged.

Question 3

Not yet answered

Marked out of 4.00

Which of the following statements are true about Memory-Mapped I/O (MMIO) and Direct Memory Access (DMA)?

Select one or more:

- ☐ a. MMIO helps increases the utilization of the hardware resources by overlapping computation (on the CPU) with data-transfer (between memory and I/O device)
- ☐ b. MMIO cannot be performed without doing DMA
- ☐ c. DMA cannot be perfrormed without doing MMIO
- ☐ d. DMA helps increases the utilization of the hardware resources by overlapping computation (on the CPU) with data-transfer (between memory and I/O device)

Question 4

Not yet answered

Marked out of 4.00

Consider the File Allocation Table (FAT) filesystem and the One-level Indexed filesystem. Which of the following statements are true (mark all that apply).

Select one or more:

- ☐ a. The one-level indexed filesystem limits the maximum size of the disk it can support to 1GB (determined by the number of blocks that can be stored in a single inode)
- ☐ b. The FAT filesystem limits the maximum size of the disk it can support
- ☐ c. The FAT filesysteem places a limit on the size of a single file (which can be over 10x smaller than the size of the disk that can be supported by the filesystem)
- ☐ d. The one-level indexed filesysteem places a limit on the size of a single file (which can be over 10x smaller than the size of the disk that can be supported by the filesystem)

Question 5

Not yet answered

Marked out of 4.00

Consider a thread that acquires two locks, then executes the critical section, and then releases the two locks. Which of the following statements are true about the need to maintain a global lock acquisition order?

Select one or more:

- ☐ a. The locks should be released in the same global order (in which they were acquired)
- ☐ b. The locks may be released in any order, which may not be the same as the acquisition order
- ☐ c. The locks must be released in the reverse order of their acquisition
- ☐ d. The acquisition of the two locks should be performed in a global order to prevent race conditions. i.e., if the locks are acquired in an unordered manner (in the setting described in the question), a race condition could manifest.
- ☐ e. The acquisition of the two locks should be performed in a global order to prevent deadlocks. i.e., if the locks are acquired in an unordered manner (in the setting described in the question), a deadlock could manifest.

Question 6

Not yet answered

Marked out of 4.00

Which of the following statements are true about the implementation of fork() on a UNIX-like kernel like xv6?

Select one or more:

- ☐ a. The entire kernel-side of the virtual address space is shared between the parent and child processes (even if copy-on-write is not implemented)
- ☐ b. The entire kernel-side of the virtual address space is copied (duplicated) in physical memory during fork
- ☐ c. The entire kstack of the child process has identical contents to the entire kstack of the parent process (just after fork has finished executing)
- ☐ d. The entire user-side of the virtual address space is shared between the parent and child processes (even if copy-on-write is not implemented)

Question 7

Not yet answered

Marked out of 4.00

Which of the following statements are true about a logging-based filesystem (also called journaling filesystem) like xv6's filesystem and ext3? Mark all that apply.

Select one or more:

- ☐ a. For a 1TB disk, it may take hours to recover a consistent filesystem state after a crash (for logging-based filesystems)
- ☐ b. For logging-based filesystems, within a single transaction, the writes to the buffer cache must be made in a certain order, e.g., during file creation, the file inode (child) must be allocated before creating a pointer in the parent directory
- ☐ c. For logging-based filesystems, within a single transaction, the writes to the buffer cache may be made in any order, and the flushes of these buffer cache writes to the log may also be performed in any order
- ☐ d. For logging-based filesystems, within a single transaction, the writes to the buffer cache may be made in any order, but the flushes of these buffer cache writes to the log must be performed in a certain order, e.g., during file creation, the file inode (child) must be allocated before creating a pointer in the parent directory

Question 8

Not yet answered

Marked out of 4.00

Which of the following statements are true about the elevator scheduling algorithm for the magnetic disk?

Select one or more:

- ☐ a. For the elevator algorithm to yield benefits, the OS must try and ensure that there are multiple in-flight (outstanding) I/Os to the disk. If there is only a single outstanding I/O (to the disk) at all times, then the elevator algorithm does not yield any benefit (and any other scheduling algorithm would have performed just as well).
- ☐ b. The elevator algorithm ensures starvation freedom
- ☐ c. The elevator algorithm minimizes the response latency for all the requests
- ☐ d. The elevator algorithm maximizes the disk throughput

Question 9

Not yet answered

Marked out of 4.00

In an Ordering-based Filesystem (without logging/journaling), a single operation (e.g., file create) may involve write to multiple disk blocks. Further, each disk block may be represented by a buffer in the buffer cache. Just like xv6, assume that the OS uses fine-grained locking, with one lock per buffer. Which of the following statements are true about the order of lock acquisition and the order of writes (mark all that apply)?

Select one or more:

- ☐ a. Both the write of disk blocks and acquisition of buffer locks must be carefully ordered. The order in which the disk blocks are written to disk can be different from the order in which the locks for the corresponding buffers are acquired
- ☐ b. The write of disk blocks can be arbitrarily ordered but the acquisition of buffer locks must be carefully ordered.
- ☐ c. Both the write of disk blocks and acquisition of buffer locks must be carefully ordered. The order in which the disk blocks are written to disk must be the same as the order in which the locks for the corresponding buffers are acquired
- ☐ d. The write of disk blocks must be carefully ordered but the acquisition of buffer locks may be arbitrarily ordered.

Question 10

Not yet answered

Marked out of 4.00

Take a look at the sleep() implementation of xv6 shown below, and select the correct choices (mark all that apply).

```
2552 void
2553 sleep(void *chan, struct spinlock *lk)
2554 {
2555   if(proc == 0)
2556     panic("sleep");
2557
2558   if(lk == 0)
2559     panic("sleep without lk");
2560
2561   // Must acquire ptable.lock in order to
2562   // change p->state and then call sched.
2563   // Once we hold ptable.lock, we can be
2564   // guaranteed that we won't miss any wakeup
2565   // (wakeup runs with ptable.lock locked),
2566   // so it's okay to release lk.
2567   if(lk != &ptable.lock){
2568     acquire(&ptable.lock);
2569     release(lk);
2570   }
2571
2572   // Go to sleep.
2573   proc->chan = chan;
2574   proc->state = SLEEPING;
2575   sched();
2576
2577   // Tidy up.
2578   proc->chan = 0;
2579
2580   // Reacquire original lock.
2581   if(lk != &ptable.lock){
2582     release(&ptable.lock);
2583     acquire(lk);
2584   }
2585 }
```

Select one or more:

☐ a. It is not ok to swap Lines 2568 and 2569, because if we release the "lk" before acquiring the "ptable.lock", then we could potentially have a deadlock.

☐ b. This implementation of sleep() is incorrect. The correct implementation must release "lk" before acquiring "ptable.lock".

☐ c. It is ok to swap Lines 2568 and 2569 --- in other words, it is ok to release "lk" before acquiring "ptable.lock"

☐ d. It is not ok to swap Lines 2568 and 2569, because if we release the "lk" before acquiring the "ptable.lock", then we could potentially have a race condition that may violate the program invariants

Question 11

Not yet answered

Marked out of 4.00

Take a look at the sleep() implementation of xv6 shown below, and select the correct choices (mark all that apply).

```
2552 void
2553 sleep(void *chan, struct spinlock *lk)
2554 {
2555   if(proc == 0)
2556     panic("sleep");
2557
2558   if(lk == 0)
2559     panic("sleep without lk");
2560
2561   // Must acquire ptable.lock in order to
2562   // change p->state and then call sched.
2563   // Once we hold ptable.lock, we can be
2564   // guaranteed that we won't miss any wakeup
2565   // (wakeup runs with ptable.lock locked),
2566   // so it's okay to release lk.
2567   if(lk != &ptable.lock){
2568     acquire(&ptable.lock);
2569     release(lk);
2570   }
2571
2572   // Go to sleep.
2573   proc->chan = chan;
2574   proc->state = SLEEPING;
2575   sched();
2576
2577   // Tidy up.
2578   proc->chan = 0;
2579
2580   // Reacquire original lock.
2581   if(lk != &ptable.lock){
2582     release(&ptable.lock);
2583     acquire(lk);
2584   }
2585 }
```

Select one or more:

☐ a. This implementation of sleep() is incorrect. The correct implementation must acquire "lk" before releasing "ptable.lock".

☐ b. It is not ok to swap Lines 2582 and 2583, because if we release the "lk" before acquiring the "ptable.lock", then we could potentially have a race condition that may violate the program invariants

☐ c. It is not ok to swap Lines 2582 and 2583, because if we release the "lk" before acquiring the "ptable.lock", then we could potentially have a deadlock.

☐ d. It is ok to swap Lines 2582 and 2583 --- in other words, it is ok to acquire "lk" before releasing "ptable.lock"

Question 12

Not yet answered

Marked out of 4.00

Which of the following statements are true about the considerations for choosing a channel identifier for xv6's sleep() and wakeup() functions?

Select one or more:

☐ a. We should use a global counter and increment it on each call to sleep() and use the incremented counter as the channel to sleep on.

☐ b. If all sleep() calls are enclosed within a while loop (that checks the sleeping condition), then it is incorrect to use a global channel (e.g., 0) for all sleep and wakeup calls. i.e., it would cause logical errors such as kernel crashes

☐ c. A wakeup() call must be enclosed within a while loop that checks the condition for wakeup.

☐ d. If all sleep() calls are enclosed within a while loop (that checks the sleeping condition), then it is correct to use a global channel (e.g., 0) for all sleep and wakeup calls. Even though this is correct, this would have suboptimal performance.

Question 13

Not yet answered

Marked out of 4.00

Consider a set of 10 long-running processes, and a machine with 8 processors and 512MB RAM. Assume that the processes are memory hungry, and the sum total of their working set sizes does not fit in the available RAM (physical memory). The OS scheduler needs to choose which processes to run on which CPUs and for how long (before context-switching). Assume that the OS scheduler is interested in minimizing the total time taken by all the processes to complete. Which of the following statements are true in this setting?

Select one or more:

☐ a. The optimal scheduling scheme must keep all the available processors busy (i.e., running one of the processes) at all times. (Here optimality is with respect to the total running time of all processes)

☐ b. In some situations, it may be more profitable to keep some of the processors idle, so that the total working set of the running processes fit within the availabile physical memory (RAM).

☐ c. In the optimal solution, each process should get to run on all the eight processors (in round robin fashion).

☐ d. There should never be any context-switching on any of the processors (in the optimal solution).

Question 14

Not yet answered

Marked out of 4.00

Which of the following statements are true about trap handling in xv6?

Select one or more:

- ☐ a. On trap entry, only caller save registers need to be saved
- ☐ b. A trap handler is called only for system calls and not for external interrupts.
- ☐ c. On trap entry, only callee save registers need to be saved
- ☐ d. The trapret label is used as one of the return addresses on the kernel stack (kstack) while constructing a fresh kernel stack (kstack) for a newly-forked child.

Question 15

Not yet answered

Marked out of 4.00

In xv6, in "struct proc" (which represents the PCB), there are two stored pointers, namely "context" and "tf" (trapframe). Which of the following statements are true about these pointers?

Select one or more:

- ☐ a. If the process is switched out, its "context" pointer (in the PCB) must point within the kstack.
- ☐ b. If the process is switched out, its "tf" pointer (in the PCB) can be NULL (or any random value) as its value is not used in the future (i.e., it will be overwritten when the process is switched in).
- ☐ c. If the process is switched out, its "context" pointer (in the PCB) can be NULL (or any random value) as its value is not used in the future (i.e., it will be overwritten when the process is switched in).
- ☐ d. If the process is switched out, its "context" pointer (in the PCB) must point within the kstack.

Question 16

Not yet answered

Marked out of 4.00

Which of the following statements are true about the order in which the xv6 locks are acquired (whenever multiple locks need to be held simultaneously in xv6)? Mark all that apply.

Select one or more:

- ☐ a. The ptable.lock must be the outermost lock that is taken (whenever one or more locks need to be acquired simultaneously, including the ptable.lock)
- ☐ b. Because xv6 locks disable interrupts, the locks can be acquired in any order without the possibility of a deadlock, i.e., there does not need to be a global order of lock acquisition.
- ☐ c. Multiple locks are never acquired simultaneously in xv6
- ☐ d. The ptable.lock must be the innermost lock that is taken (whenever one or more locks need to be acquired simultaneously, including the ptable.lock)

Question 17

Not yet answered

Marked out of 4.00

The CLOCK algorithm is used for cache replacement while managing the main memory and the swap space. Which of the following statements is/are true?

Select one or more:

☐ a. A two-hand CLOCK always produces better hit rates than one-hand CLOCK

☐ b. The LRU algorithm is guaranteed to produce better hit rates than CLOCK

☐ c. The CLOCK algorithm is guaranteed to produce better hit rates than LRU

☐ d. The CLOCK algorithm can be implemented more efficiently than FIFO

☐ e. The CLOCK algorithm can be implemented more efficiently than LRU

Question 18

Not yet answered

Marked out of 4.00

Consider the following function:

```
void transfer(account *a, account *b) {
  acquire(&a->lock);
  acquire(&b->lock);
  a->money--;
  b->money++;
  release(&a->lock);
  release(&b->lock);
}
```

Assume that "account" is a C struct with "lock" and "money" fields. The transfer function is supposed to execute atomically with respect to other transfers. The acquire and release functions are used to acquire and release the lock (supplied as argument) respectively. Which of the following is true?

Select one or more:

☐ a. This implementation cannot provide mutual exclusion if the machine has more than one processors

☐ b. This implementation does not guarantee mutual-exclusion between two simultaneous executions of transfer, i.e., two threads may access the same account simultaneously.

☐ c. At most one transfer can proceed at any given time

☐ d. This implementation allows multiple transfers between distinct pairs of accounts to execute simultaneously.

☐ e. This implementation can have a deadlock

Question 19

Not yet answered

Marked out of 1.00

A situation where several concurrent processes access the shared memory and the outcome of the execution depends on the particular order in which access takes place is called

Select one or more:

- ☐ a. Exception
- ☐ b. Data Hazard
- ☐ c. Read-after-Write-after-Read
- ☐ d. Race condition

Question 20

Not yet answered

Marked out of 5.00

Which of the following statements is/are true about condition variables?

Select one or more:

- ☐ a. Condition variables are stateless
- ☐ b. A condition variable is associated with acquire and release functions
- ☐ c. Condition variables must not be used on a uniprocessor system; they always result in a deadlock if they are used on a uniprocessor system
- ☐ d. A condition variable may be associated with either Hoare or Mesa semantics
- ☐ e. Condition variables depend on locks for mutual exclusion

Question 21

Not yet answered

Marked out of 4.00

Which of the following statements are true about the context-switching logic in xv6? Mark all that apply.

Select one or more:

- ☐ a. It is necessary to save all callee-save registers in swtch.S (and thus it is necessary to have space for them in "struct context"
- ☐ b. It is possible to add more registers to "struct context". These registers should be saved and restored (using push/pop) in swtch.S, This will not have any effect on the correctness of the xv6 kernel, although it may degrade performance a bit.
- ☐ c. The values of the registers that get saved in "struct context" (on a context switch) must be the values that were generated in the user mode
- ☐ d. On a single processor, it is possible to context-switch from one process (say process A) to another process (say process B) without entering the kernel mode

Question **22**

Not yet answered

Marked out of 3.00

Which of the following are true about the copy-on-write optimization during process fork? Mark all that apply.

Select one or more:

- ☐ a. It reduces disk space requirements
- ☐ b. It allows faster implementations of the open() and close() system calls in UNIX
- ☐ c. It allows faster forks
- ☐ d. It can be counter-productive (i.e., lead to poor performance compared to an OS where there is no copy-on-write optimization) if both the child and parent processes write to all the pages mapped in their address space (after fork).

Question **23**

Not yet answered

Marked out of 4.00

Which of the following statements are true about implementing copy-on-write optimization? Indicate which of the following statements are true about the actions that need to be taken at the time of creating the forked process.

Select one or more:

- ☐ a. To implement copy-on-write, we need to mark the PTEs of the parent process so that they only allow read-only access. The PTEs of the child process must be left unmodified (i.e., they can allow both read and write accesses).
- ☐ b. To implement copy-on-write, we need to mark the PTEs of the one of the two processes (child and parent) so that they only allow read-only access. The PTEs of the other process (it can be any one) must be left unmodified (i.e., they can allow both read and write accesses).
- ☐ c. To implement copy-on-write, we need to mark the PTEs of the child process so that they only allow read-only access. The PTEs of the parent process must be left unmodified (i.e., they can allow both read and write accesses).
- ☐ d. To implement copy-on-write, we need to mark the PTEs of the both processes (child and parent) so that they only allow read-only access.

Question **24**

Not yet answered

Marked out of 2.00

A counting semaphore was initialized to 10. Then 6 P (wait) operations and 4 V (signal) operations were completed on this semaphore. The resulting value of the semaphore is

Select one or more:

- ☐ a. 10
- ☐ b. 12
- ☐ c. 0
- ☐ d. 8

Question **25**

Not yet answered

Marked out of 2.00

A critical section is a program segment ...

Select one or more:

- ☐ a. where shared resources are accessed
- ☐ b. which should run in a certain specified amount of time
- ☐ c. which _must_ be enclosed by a pair of semaphore operations, P and V
- ☐ d. which avoids deadlocks

Question **26**

Not yet answered

Marked out of 2.00

Which of the following is NOT a necessary condition for a deadlock?

Select one or more:

- ☐ a. Uniprocessor system
- ☐ b. Circular wait
- ☐ c. Mutual exclusion
- ☐ d. No preemption of resources

Question **27**

Not yet answered

Marked out of 3.00

When an executable program is loaded into memory using the exec() system call, it is possible that only a few pages are loaded initially and the rest are demand-paged. Which of the following are advantages of demand paging? Mark all that apply.

Select one or more:

- ☐ a. Process creation takes less time
- ☐ b. It saves disk space
- ☐ c. Demand paging reduces the probability of thrashing
- ☐ d. It saves memory space because typically only those pages that are used are allocated in memory

Question **28**

Not yet answered

Marked out of 4.00

In demand paging, the swap space on the disk can be used to hold virtual memory pages that do not fit in memory. In other words, the main memory acts as a cache for the virtual memory pages, and a cache-replacement policy decides which pages reside in main memory and which pages reside on the disk. Which of the following statements is/are true?

Select one or more:

- ☐ a. If a page has not been modified, it does not need to be written back to the swap space on eviction
- ☐ b. The main memory is typically managed as a direct-mapped cache
- ☐ c. A hit rate of 90% in the cache produces acceptable system performance
- ☐ d. The cache is managed at page-granularity

Question **29**

Not yet answered

Marked out of 2.00

Which of the following are the methods to communicate the completion of the scheduled tasks on the devices to the CPU?

Select one or more:

- ☐ a. Polling
- ☐ b. Exception
- ☐ c. External Interrupt
- ☐ d. Exokernel
- ☐ e. System call
- ☐ f. Multi-level feedback queue

Question **30**

Not yet answered

Marked out of 4.00

Many disks use a sector size of 512B. Many OSes use a "block size" of 4096B (or 4KB). Which of the following statements are true about an ideal block size.

Select one or more:

- ☐ a. A smaller block size (say 512B) will typically result in better disk bandwidth utilization, even if the workloads exhibit high spatial locality.
- ☐ b. A larger block size (say 64KB) has more potential to exploit spatial locality, but suffers from higher internal fragmentation.
- ☐ c. The ideal block-size depends on the typical size of the files that are created/used by the end-users
- ☐ d. A larger block size (say 64KB) produces better hit rates in the buffer cache

Question **31**

Not yet answered

Marked out of 3.00

Many operating systems provide a ‘disk defragmenter’ utility. This utility can be invoked by the user to reorganize his/her disk contents, for better future disk performance (after reboot). Which of the following, do you think, are potential operations that such a utility might perform?

Select one or more:

- ☐ a. Move pages from the filesystem to the swap space
- ☐ b. Move pages from the swap space to the filesystem
- ☐ c. Populate the buffer cache in main memory with the frequently accessed blocks in the disk
- ☐ d. Lay out the sequential contents of a file on sequential blocks in the disk. This would optimize the common case of reading the file sequentially on a magnetic disk.

Question 32

Not yet answered

Marked out of 2.00

Which of the following is in correct order of size, smallest first?

Select one or more:

- ☐ a. Cylinder, track, sector
- ☐ b. Track, cylinder, sector
- ☐ c. Sector, track, cylinder
- ☐ d. Cylinder, sector, track
- ☐ e. Sector, cylinder, track

Question 33

Not yet answered

Marked out of 2.00

A file control block contains the information about

Select one or more:

- ☐ a. Location of file contents
- ☐ b. File ownership
- ☐ c. File permissions

Question 34

Not yet answered

Marked out of 4.00

In UNIX, the open/read/write/close system calls can be used to access files stored in a filesystem. A program writes data to a file using a sequence of open-write-close system calls. Which of the following are true?

Select one or more:

- ☐ a. For the ext3 filesystem, if a power failure occurs after the write system call returns, then the data would be definitely available on the disk
- ☐ b. For the ext3 filesystem, if a power failure occurs after both the write and the close system calls return, then the data would be definitely available on the disk
- ☐ c. If a power failure occurs, the data would not be available on the disk irrespective of when it occurred (e.g.,, even after one hour) as long as the program does not call the sync() system call
- ☐ d. Depending on the filesystem, the data that is written may or may not be available on the disk, after the write() system call returns.

Question 35

Not yet answered

Marked out of 4.00

Which of the following is true about filesystem crash recovery in a non-logging filesystem like ext2?

Select one or more:

- ☐ a. With careful ordering of filesystem block writes, all types of filesystem errors due to power failures can be avoided
- ☐ b. Ensuring crash-recovery through careful ordering of writes may require more disk accesses (than if we didn't have to worry about crash recovery)
- ☐ c. The order in which filesystem blocks are written to disk is important to ensure that the filesystem is not corrupted on power failure
- ☐ d. It is possible to eliminate the possibility of data loss by careful ordering and disk writes; however it can cause disk blocks to get leaked (wasted)

Question 36

Not yet answered

Marked out of 4.00

Consider the DOS FAT filesystem, the Indexed filesystem, and the Multi-level indexed filesystem. Which of the following statements is/are true?

Select one or more:

- ☐ a. A DOS FAT32 filesystem cannot be used to support a very large filesystem sizes (e.g., NTFS can support larger filesystems than DOS FAT32)
- ☐ b. To access file data, a single-level indexed filesystem (with only direct blocks) requires fewer disk accesses on average than a multi-level indexed file system
- ☐ c. For small files, indexed filesystem has better performance than multi-level indexed filesystem
- ☐ d. A multi-level indexed filesystem. such as NTFS, can support files of unbounded size (assuming that the disk is large enough)

Question 37

Not yet answered

Marked out of 4.00

Which of the following statements are true about first-fit and best-fit algorithms?

Select one or more:

- ☐ a. The first-fit algorithm is typically used while allocating pages to processes
- ☐ b. Best-fit algorithm suffers from the sawdust problem
- ☐ c. Typically, first-fit starts behaving somewhat like best-fit, when run over a long period of time.
- ☐ d. Best-fit always produces less fragmentation than first-fit algorithm

Question 38

Not yet answered

Marked out of 3.00

In Unix if a process forks a child process and forgets to call wait() syscall, which of the following statements are true?

Select one or more:

- ☐ a. None of these choices are true
- ☐ b. If the child never exits but the parent exits, then some memory is leaked (wasted) forever
- ☐ c. If the child exits but the parent never exits, then some memory is leaked (wasted) forever
- ☐ d. If the child never exits and the parent exits, then some memory is leaked (wasted) forever
- ☐ e. If the child exits and then the parent exits, then some memory is leaked (wasted) forever

Question 39

Not yet answered

Marked out of 4.00

Which of the following statements is/are true for a logging-based (journaling) filesystem like ext3?

Select one or more:

- ☐ a. Large transactions in a logging-based filesystem increase the probability of data loss on power failure
- ☐ b. Multiple filesystem write operations can be grouped into a single transaction
- ☐ c. The write of the a transaction to a disk log would involve a contiguous write to the disk; this is usually much cheaper than making all the individual changes to different parts of the disk
- ☐ d. A logging-based filesystem effectively provides half the amount of raw disk space for filesystem data; the other half is used by the log

Question 40

Not yet answered

Marked out of 4.00

In xv6, when a process is killed through the kill system call, the "killed" flag is set. Then, at an *appropriate* time, the "killed" flag is checked, and if it is set, the process is made to exit. Which of the following is NOT an appropriate time for checking the "killed" flag and exiting if it is set?

Select one or more:

- ☐ a. At system call entry, i.e., when the process calls the system call
- ☐ b. At system call exit, i.e., when the system call is about to return back to the process
- ☐ c. On external interrupt handler entry, if an external interrupt is received while the process was executing in kernel mode
- ☐ d. On external interrupt handler entry, when an external interrupt is received while the process was executing in user mode
- ☐ e. On exception entry, i.e., when an exception occurs while the process was executing in user mode

Question 41
Not yet answered
Marked out of 3.00

Which of the following is true about lock-free synchronization primitives, such as compare-and-swap?

Select one or more:

- ☐ a. There can be no deadlocks with lock-free primitives. This is because even if there was a concurrent access, and so one thread may have to retry, there will always be progress.
- ☐ b. Lock-free primitives can only be used in kernel mode, and cannot be used in user mode
- ☐ c. Lock-free primitives can be used even if the OS implements the demand-paging and copy-on-write optimizations
- ☐ d. Like locks, lock-free primitives like the compare-and-swap instruction can be used to protect critical sections of arbitrary size.

Question 42
Not yet answered
Marked out of 3.00

Which of the following are true statements about diffferent types of locks?

Select one or more:

- ☐ a. It is not possible to implement "try reader-writer locks", i.e., a combination of try locks and reader-writer locks (e.g.., try_read_acquire(), try_write_acquire()).
- ☐ b. Reader-writer locks enable more concurrency (than plain locks) because they allow multiple readers to execute in the critical section simultaneously.
- ☐ c. Recursive locks is usually not a good programming practice, because the programmer can no longer assume that the invariants hold at the entry of the critical section that is protected by the recursive lock.
- ☐ d. Reader-writer locks can exhibit deadlocks.

Question 43
Not yet answered
Marked out of 4.00

Which of the following statements are true about a logging based filesystem (like ext3). Mark all that apply.

Select one or more:

- ☐ a. A logging based filesystem experiences 2x slowdown during disk writes (over non-logging based filesystems like ext2)
- ☐ b. A logging based file system can quickly run out of log space on disk.
- ☐ c. A logging based filesystem generates more disk write traffic than a non-logging based filesystem like ext2
- ☐ d. On a crash, the entire contents of the filesystem are available in the log.
- ☐ e. A logging-based filesystem would have unacceptable performance with a write-through buffer cache

Question 44
Not yet answered
Marked out of 3.00

Which of the following statements is/are true about a monitor?

Select one or more:

- ☐ a. A monitor's queue can have at most one process in it at a time.
- ☐ b. Only one process can be active at a time within the monitor.
- ☐ c. In the absence of any other synchronization mechanism, it is not possible to have a deadlock involving only monitors.

Question 45
Not yet answered
Marked out of 2.00

Consider a memory page containing a heavily used variable that was initialized very early and is in constant use. This page may be removed from the page cache (and written to the swap space) for which of the following page replacement algorithms?

Select one or more:

- ☐ a. LRU (least recently used)
- ☐ b. LFU (least frequently used)
- ☐ c. Optimal page replacement algorithm (oracle-based)
- ☐ d. FIFO (first in first out)
- ☐ e. Random

Question 46
Not yet answered
Marked out of 2.00

The page table cache is also known as

Select one or more:

- ☐ a. Swap space
- ☐ b. storage buffer
- ☐ c. page table storage
- ☐ d. Critical section
- ☐ e. L1 cache
- ☐ f. translation look aside buffer

Question 47
Not yet answered
Marked out of 3.00

Which of the following statements are true about polling and interrupts? Mark all that apply.

Select one or more:

- ☐ a. Interrupts produce better response latency
- ☐ b. Polling performs better than interrupts when the expected rate of external events (that are being monitored through polling or interrupt mechanism) is very low
- ☐ c. Interrupts can be lost if the CPU clears its interrupt flag. In this case, the interrupt should be generated again by the external device (that is generating the external event).
- ☐ d. Polling does not suffer from the receive livelock problem

Question 48

Not yet answered

Marked out of 3.00

In xv6, consider the following situation:

a. A thread (say thread1) is executing on CPU0, and acquires spinlock X

b. A timer interrupt occurs and the thread1 gets context-switched out (while holding X)

c. Another thread (say thread2) starts executing on CPU0 and tries to acquire spinlock X, and spins (because thread1 is holding X)

Assume that threads in xv6 can share address space both within the kernel and in user-space (i.e., multiple processes can map the same pages in the user address space and access them concurrently).

Based on the this sequence of events and your knowledge of the xv6 kernel, which of the following statements are true?

Select one or more:

- ☐ a. Thread2 will yield CPU0 to Thread1 after it starts spinning (without any other external interrupt and without Thread1 releasing the lock on another CPU).
- ☐ b. It is possible that thread1 gets scheduled on CPU1 (another CPU distinct from CPU0) and releases the lock, so that thread2 can make progress before the next timer interrupt on CPU0.
- ☐ c. Thread2 will keep spinning forever
- ☐ d. It is possible that thread2 keeps spinning till the next timer interrupt, and then gets context-switched out.

Question 49

Not yet answered

Marked out of 3.00

Priority scheduling allows some processes to have strictly higher priority over others. Which of the following are true?

Select one or more:

- ☐ a. If a low-priority process holds a lock L, and a high-priority process wants to acquire L, the high-priority process must wait for the low-priority process to release the lock L.
- ☐ b. If a low-priority process holds a lock L, and a high-priority process wants to acquire L, the priority of the high-priority process is donated to the low-priority process (to make it a high-priority process as well) until L is released by the low-priority process
- ☐ c. If a low-priority process holds a lock L, and a high-priority process wants to acquire L, then the low-priority process is killed and the lock L is given to the high-priority process.

Question 50

Not yet answered

Marked out of 3.00

Which of the following statements are true about UNIX processes and programs? Mark all that apply.

Select one or more:

- ☐ a. A single program may run in multiple processes simultaneously.
- ☐ b. At any instant of time, all running processes must be running distinct programs (i.e., no two processes must be running the same program).
- ☐ c. A single process may run different programs at different instants of time.
- ☐ d. Two or more running processes may have the same process-ID simultaneously.

Please turn over for the next page

Question 51

Not yet answered

Marked out of 4.00

Both Read-Copy-Update (RCU) mechanism and reader-writer locking allow multiple readers to execute simultaneously within the critical section, but allow only a single writer to execute at any time (when a writer executes, no other reader/writer should be executing simultaneously). Which of the following statements are true about RCU vis-a-vis reader-writer locks?

Select one or more:

☐ a. RCU mechanisms must be accompanied with spinlocks, not blocking locks

☐ b. RCU can typically be implemented only in special situations (e.g., in the OS kernel) where it may be possible to assume certain things about the environment (e.g., thread scheduling behaviour). In contrast, reader-writer locks are general and can be used in any context.

☐ c. Wherever it is possible to implement RCU, if the workload is read-mostly (i.e., large number of reads and occasional writes), it produces faster implementations than reader-writer locks because it avoids the synchronization overheads associated with reader locks.

☐ d. RCU can only be used for synchronizing threads that are executing on the same CPU

Question 52

Not yet answered

Marked out of 4.00

Which of the following statements are true about scheduling at the OS level? Mark all that apply.

Select one or more:

☐ a. When a scheduling policy minimizes average response latency, it also maximize fairness among different processes

☐ b. In a typical computer system with a disk, memory, CPU, etc., there are several schedulers at each level, e.g., OS scheduler, disk scheduler, application-level scheduler, etc.

☐ c. If the system is performing at optimum throughput (the best achievable throughput is being realized), then it must be true that the obtained average response latency is the minimum achievable.

☐ d. If we use a bad CPU scheduling policy, it is possible that the wall-clock time (as displayed to the user) deviates from its actual value over time.

Question 53
Not yet answered
Marked out of 1.00

Which of the following is true about secondary disk storage?

Select one or more:

- ☐ a. Flash memory is faster than magnetic disk
- ☐ b. For a magnetic disk, accessing 10MB of data at random addresses is more expensive than 10MB of sequentially placed data
- ☐ c. On power failure, a magnetic disk loses all its contents
- ☐ d. Magnetic disk storage is typically at least 1000x slower than main memory

Question 54
Not yet answered
Marked out of 3.00

A semaphore is associated with two functions: P (or wait) and V (or signal). Which of the following statements is/are true about semaphores?

Select one or more:

- ☐ a. Inside the semaphore's P() function (or wait() function), one must use a blocking mechanism to wait if required; it is not acceptable to use spinning for waiting in this case.
- ☐ b. Semaphores can be used to implement mutual exclusion
- ☐ c. In the absence of any other synchronization mechanism, it is not possible to have a deadlock involving only semaphores
- ☐ d. Semaphores are stateless

Question 55
Not yet answered
Marked out of 1.00

If process A has higher temporal and spatial locality than process B (with all other characteristics being the same), then:

Select one or more:

- ☐ a. Execution will likely be faster in process A
- ☐ b. page-fault rate will be unaffected by locality
- ☐ c. page-fault rate will likely be less in process B
- ☐ d. Execution will likely be faster in process B
- ☐ e. page-fault rate will likely be less in process A

Question 56
Not yet answered
Marked out of 4.00

Which of the following are true about spinlocks and blocking locks?

Select one or more:

- ☐ a. On a multiprocessor system (that has more than one CPUs), blocking locks cannot provide mutual exclusion
- ☐ b. Blocking locks cannot result in deadlocks, but spinlocks may result in deadlocks
- ☐ c. Spinlocks always result in poorer resource utilization when compared to blocking locks
- ☐ d. For kernel-level threads, blocking locks require support of the OS scheduler

Question **57**

Not yet answered

Marked out of 3.00

Which of the following is true about thrashing in Virtual Memory (VM)?

Select one or more:

☐ a. If thrashing occurs, the OS is helpless and the system must be rebooted

☐ b. If the system reaches a deadlock state, then the system will start thrashing

☐ c. Thrashing causes memory fragmentation

☐ d. Thrashing occurs if the VM hit rates (for finding a page in main memory) are lower than the desired value

☐ e. A global page replacement policy does not suffer from the problem of thrashing, i.e., thrashing cannot occur if a global page replaement policy is used.

☐ f. Thrashing occurs if the amount of disk space is lesser than required

Question **58**

Not yet answered

Marked out of 3.00

Which of the following statements are true about the timer interrupt?

Select one or more:

☐ a. If there is only a single process running in the system, it would still get interrupted after every periodic time interval (by the timer interrupt)

☐ b. A timer interrupt is often used as one of the ways to keep track of wall-clock time (i.e., the time of the day as displayed to the user).

☐ c. The typical timer interrupt frequency is 100,000 Hz

☐ d. It is not possible for the timer interrupt to get ignored (e.g., due to clearing of interrupts on the CPU).

☐ e. In xv6, if a timer interrupt is received while the processor was executing in kernel mode, then the interrupt would be ignored.

Question **59**

Not yet answered

Marked out of 4.00

Which of the following is true?

Select one or more:

☐ a. TLBs are usually located on the same chip as processor to have low access latency

☐ b. To be effective, TLBs should have a hit ratio of 99+ percent

☐ c. TLBs are usually fully-associative to have high hit rates

☐ d. The typical TLB size is 4-8MB

Question **60**

Not yet answered

Marked out of 4.00

Which of the following statements are true about a two-handed CLOCK algorithm vs. a one-handed CLOCK algorithm?

Select one or more:

- ☐ a. A two-handed CLOCK algorithm offers more flexibility to the OS than a one-handed CLOCK algorithm. For example, the OS administrator can choose to reduce the angle between the two hands if the size of the physical memory is very large.
- ☐ b. A two-handed CLOCK algorithm, along with the timer interrupt, is used as a method for maintaining the wall-clock time (as displayed to the user)
- ☐ c. It is possible to extend the idea to use *n* hands (for *n*>= 3) by generalizing the idea from *n*=2 (for a two-handed CLOCK) to higher value of *n*.
- ☐ d. A two-handed CLOCK algorithm is guaranteed to produce better hit rates than a one-handed CLOCK algorithm
- ☐ e. A two-handed CLOCK algorithm is more efficient to implement than a one-handed CLOCK algorithm

Question **61**

Not yet answered

Marked out of 2.00

There is no need for synchronization abstractions for Uniprocessor systems.

Select one or more:

- ☐ a. True
- ☐ b. Depends on the Instruction Set Architecture
- ☐ c. False

Question **62**

Not yet answered

Marked out of 4.00

The function walkpgdir() in xv6 is used to walk the page table. Which of the following is true?

Select one or more:

- ☐ a. This function is invoked by the OS while conducting special operations like mapping the pages into the address space of a process
- ☐ b. This function is invoked on every memory access by the hardware
- ☐ c. This function sets all page table entries (PTEs) that it visits to 0
- ☐ d. The results of calling this function are cached in the TLB
- ☐ e. This function returns a pointer to the PTE (page table entry) for a given virtual address

Question 63

Not yet answered

Marked out of 4.00

In xv6, every process has a separate kernel stack, also called the kstack. The kstack has bounded size. A trapframe is an object of "struct trapframe" that is used to store the contents of the CPU registers (and other relevant state) at the time of trap (e.g., at the time of receiving the interrupt). The maximum number of trapframes that may be present on the kstack of a process at any given instance of time is:

Answer:

Question 64

Not yet answered

Marked out of 3.00

In xv6, if a process "proc" is currently switched out (e.g., it is in RUNNABLE or BLOCKED state), then it is possible that proc.context points outside proc's kernel stack.

Clarifications: the kernel stack is represented by the region proc.kstack to (proc.kstack + KSTACKSIZE). Here "proc" represents the PCB of the process (as in xv6).

Select one:

a. ○ True

b. ○ False

Question 65

Not yet answered

Marked out of 3.00

In xv6, if the processor is in kernel mode, and is not executing inside one of the functions in the trapasm.S file, the following relation must hold:

proc.kstack <= %esp < proc.tf < (proc.kstack + KSTACKSIZE)

Here, proc represents the current process (in whose context, the processor is running in kernel mode). proc.kstack and proc.tf represent the pointers to the current process's kernel-stack and trapframe respectively. %esp represents the CPU register ESP. KSTACKSIZE represents the maximum size of the kernel stack (as allocated by the kernel).

Select one:

a. ○ True

b. ○ False

Question 66

Not yet answered

Marked out of 20.00

In the lectures, we have seen how the acquire() and release() functions for a lock can be implemented using the atomic "xchg" instruction on the x86 architecture.

Consider a different architecture which does not provide the "xchg" instruction but only provides the "compare-and-swap" (CAS) instruction (also discussed in the lectures when discussing transactions).

Is it possible to implement the acquire() and release() functions of a lock using the CAS instruction? If yes, provide an implementation. If not, explain why not.

Question 67

Not yet answered

Marked out of 20.00

Briefly answer the following questions on xv6:

1. What happens when an xv6 lock (e.g., ptable lock) is held and an interrupt occurs? [5]

2. When does the xv6 file system write dirty buffers (buffers to which modifications have been made) back to disk? [5]

3. What happens when the xv6 kernel tries to allocate a buffer in the buffer cache, but finds that all the available buffers are BUSY? [5]

4. What happens when in the middle of an xv6 filesystem operation (e.g., file creation), a crash occurs? [5]

Question 68

Not yet answered

Marked out of 20.00

A local filesystem is a filesystem on a magnetic disk attached to the computer's I/O bus. A remote filesystem is a filesystem on a remote server (e.g., NFS). Answer the following questions:

1. Are random/sequential reads/writes typically faster or slower on the remote filesystem (compared to a local filesystem)? Answer for each of the four settings: random reads, random writes, sequential reads, sequential writes. For each setting, if your answer is "faster", indicate by how much. Similarly, if your answer is "slower", indicate by how much. [10]

2. In what ways do you think that the typical design of a remote filesystem is likely to be different from the design of a local filesystem? If you think there will be no differences, explain why. If you point out a design difference, briefly explain why that difference would be typically required. Write each difference separately (in a separate bullet) with a brief explanation. [10]

Question **69**

Not yet answered

Marked out of 2.00

After going through the full course, please indicate your favourite OS topic giving reasons (in short). (You will earn marks for this question only if you provide an answer).