Operating Systems Design
Exam 2 Review: Spring 2011

Paul Krzyzanowski
pxk@cs.rutgers.edu

1

Question 1

CPU utilization tends to be *lower* when:

a. There are more processes in memory.
b. There are fewer processes in memory.
c. There is a higher degree of multiprogramming
d. Processes perform very little I/O.

CPU Utilization = $(1 - p^n)$

Estimate of probability that the CPU is not idle
p = probability that the process is waiting on I/O
n = degree of multiprogramming

More processes in memory → higher CPU utilization
Fewer processes in memory → lower CPU utilization

2

Question 2

A major problem with the base & limit approach to memory translation is:

a. It requires the process to occupy contiguous memory locations.
b. The translation process is time-consuming.
c. The translation must be done for each memory reference.
d. A process can easily access memory that belongs to another process.

(b) Not if done in hardware.
(c) Not if done in hardware.
(d) No.

3

Question 3

Page-based virtual memory is subject to:

a. Internal fragmentation.
b. External fragmentation.
c. Both.
d. Neither.

External fragmentation = unused memory between allocation units.
Internal fragmentation = unused memory within an allocation unit.

Page-based VM: fixed-size allocation units

4

Question 4

Memory compaction is a technique to:

a. Free up unused memory in a process to create more free memory.
b. Remove redundant data in a process so it takes up less space in memory.
c. Apply real-time data compression to a process' memory to reduce its footprint.
d. Move a process to a different part of memory to create a larger region of contiguous memory

5

Question 5

The difference between base & limit addressing and segmentation is:

a. A process may have a discontiguous address space with segmentation.
b. A process cannot access another process' memory under segmentation.
c. Segmentation requires dynamic address translation.
d. Segmentation does not have problems with external fragmentation

Each segment (code, data, stack, etc.) can be in its own memory region.
Each of these regions has to be contiguous

b, c, d: apply to base & limit addressing as well.

6

Question 6

In contrast to segmentation, paging:

a. Requires real-time address translation.
b. Requires less memory to store memory address translation tables.
c. Divides memory into fixed-size chunks.
d. Requires that a process be allocated a contiguous chunk of memory.

(a) So does segmentation
(b) Segmentation requires a lot less memory for address translation
(d) Base & limit (simplest form of segmentation) requires a contiguous chunk of memory; segmentation requires several contiguous chunks

(c) Fixed-size chunk of memory: page frame

7

Question 7

With a direct mapping paging system on a 32-bit processor with 32 KB pages, what is the size of each process' page table?

a. 32K entries
b. 64K entries
c. 128K entries
d. 4G entries

Page size = 32 KB → 15-bit page offset (2^{15} = 32K)

If offset = 15 bits then the page # in an address is 32-15 = 17 bits

2^{17} = 128K

8

Question 8

In contrast to a single-level page table, multilevel page tables:

a. Reduce the maximum amount of memory that a process can need for a page table.
b. Result in faster page table lookups.
c. Allow processes to share regions of memory.
d. Typically require much less memory per process.

(a) Worst case: you need all the PTEs (= the entire page table) + the index table
(b) Multilevel page tables result in *slower* lookups!
(c) So do single-level page tables
(d) Processes do not exhibit worst-case use; a typical process has large regions of memory that are not allocated. Hence, multiple partial page tables do not have to be allocated.

9

Question 9

A logical address of 0x514413ab on a 32-bit page-based virtual memory system with 4 KB pages has the following offset:

a. 0x3ab
b. 0x413ab
c. 0x51441
d. 0x514

4 KB pages → 12-bit offset = bottom three nibbles

10

Question 10

Effective memory access time on a TLB miss with a 4-level page table compared with a 2-level page table is:

a. 167% the speed of a two-level page table (5/3x slower).
b. 200% the speed of a two-level page table (2x slower).
c. 60% the speed of a two-level page table (5/3x faster).
d. 50% the speed of a two-level page table (2x faster).

More levels mean more main memory lookups → slower access

With a 2-level table:
2 memory accesses for page table walk + 1 memory access for data = 3 memory access cycles
With a 4-level table:
4 memory accesses for page table walk + 1 memory access for data = 5 memory access cycles
4-level vs. 2-level: 5 cycles vs. 3 cycles = 5/3 speed

11

Question 11

The ARM v7 MMU architecture is best described as:

a. Combined direct mapped paging and segmentation.
b. Combined associative and direct mapped paging.
c. Pure direct mapped paging.
d. Pure segmentation.

Practically every CPU except for Intel & small microcontrollers uses (b): page-based virtual memory with an associative TLB.

12

Question 12

The *ideal* page replacement algorithm would evict:

a. Any page that has not been used since the last periodic interrupt.
b. The oldest page.
c. The least frequently used page.
d. The least recently used page.

In the *ideal* world, we'd like to use LRU. Unfortunately, the hardware on MMUs makes this difficult.

(a) No distinction between recently used or not.
(b) An old page may still be used recently and frequently.
(c) A new page that is recently accessed may not have yet had a high frequency of accesses.

13

Question 13

If evicted, the following pages do not need to be written to a swap file:

a. Program text (pages containing executable code).
b. Heap (allocated memory).
c. Stack.
d. Any of the above must be written to a swap file if evicted.

(a): The text segment (program) is not modified and can be reloaded from the program file. It can be evicted without saving the contents.

(b), (c): contain data that was written by the user program and must be saved.

14

Question 14

Page fault frequency is a way of:

a. Measuring the effectiveness (hit ratio) of a TLB.
b. Identifying the specific pages that are best targets for eviction.
c. Measuring the efficiency of page fault processing within an operating system.
d. Ensuring that all processes have their working sets in memory.

(a) Page faults occur on an MMU translation failure, not a TLB miss.
(b) Page fault frequency is a number that identifies the rate of page faults. It does not identify which pages have not been recently accessed and are good candidates for eviction.
(c) Whether you page or not has nothing to do with how efficiently you process a page fault.
(d) PFF is one technique for deciding if one process does not have enough of its working set in memory (high PFF) or has too much (low PFF).

15

Question 15

Thrashing in a virtual memory system is caused by:

a. A process making too many requests for disk I/O.
b. Multiple processes requesting disk I/O simultaneously.
c. Processes not having their working set resident in memory.
d. Slow operating system response to processing a page fault.

(a), (b) Improper disk I/O scheduling may cause delays and lead to thrashing BUT the question asks about VM.
(c) Not having a WS in memory causes a process to have excessive page faults (high PFF), leading to thrashing.
(d) This increases the perceived effects of thrashing.

16

Question 16

A buffer cache is used with:

a. Character devices.
b. Block devices.
c. Network devices.
d. All categories of devices

(a) No. Character device data is not addressable
(c) No for the same reason

17

Question 17

On POSIX systems, devices are presented as files. When you send an I/O request to such a file, it goes:

a. to the device driver that was identified in the metadata of the file when it was opened.
b. to the device driver, which is contained within data of the device file.
c. to the device driver based on the major and minor numbers contained in the file's data.
d. the file system driver (module), which then interprets it as a device request.

(b) The device driver is not stored in the device file. The device file has no data.
(c) The file has no data, just metadata.
(d) Once the file is identified as a device file, I/O requests do not go through the file system driver. Character drivers implement file_operations for VFS.

18

Question 18

When a user process makes a system call, the kernel code is run in the:

a. User context.
b. Kernel context.
c. Interrupt context.
d. Device context.

(a) Process-related requests are run in *kernel mode* in the *user context*.
(b) Kernel threads run in the kernel context.
(c) Asynchronous device interrupts are handled in interrupt context.
(d) No such thing.

19

Question 19

A buffer cache:

a. Allows user programs to read and write byte streams when the back-end device supports only block I/O.
b. Stores frequently used blocks of data.
c. Is an intermediate place for data to reside between the user program and device driver.
d. All of the above.

20

Question 20

What data identifies the device driver that should be used for a specific device?

a. Major number.
b. Major number, minor number.
c. Device type (block or character), major number.
d. Device type (block or character), major number, minor number.

(a) No. Need to identify the device table: block or character.
(b) No. Need to identify the device table: block or character.
(c) Yes.
(d) TMI. The minor number is interpreted only within the context of the device driver.

21

Question 21

Interrupt handling is separated into two parts to:

a. minimize the amount of work done in the interrupt service routine.
b. separate generic interrupt processing operations from device-specific ones.
c. allow a user process to handle all non-critical parts of interrupt processing.
d. make the code more modular.

(a) Yes. Anything that requires a lot of work or sleeping is handled by a kernel worker thread (bottom half). Immediate service is handled by the top half.
(b) Yes, but this isn't the 2-part separation of handling an interrupt.
(c) No.
(d) Sometimes.

22

Question 22

The purpose of device frameworks is to:

a. Allow devices to present themselves as dynamically loaded modules.
b. Provide an alternate interface to devices that don't fit the character or block model.
c. Ensure that devices can properly implement a file interface.
d. Provide a well-defined set of interfaces for specific device types

(a) Device drivers do this without frameworks.
(b) Devices have to fit the block, character, or network interface. Frameworks only extend it.
(c) No.
(d) Yes.

23

Question 23

Disk scheduling algorithms try to minimize the effects of:

a. Seek time.
b. Rotational latency.
c. Transfer rate.
d. All of the above.

(a) Yes. Seek time is by far the dominant performance factor of a disk. Disk scheduling algorithms try to sort requests to minimize unnecessary seeks.
(b) They used to try to do this, but it was a distant #2 to (a).
(c) Can't address this with a disk scheduling algorithm.
(d) No.

24

Question 24

A system has the following blocks queued for writing: 8000, 3000, 5000, 2000. The most recently written block was 4500. The block read before that was 2500. What sequence of writes will a Circular-LOOK (or C-SCAN) algorithm generate?

a. 5000, 8000, 3000, 2000 *LOOK*
b. 5000, 8000, 2000, 3000 *C-LOOK*
c. 5000, 3000, 2000, 8000 *SSTF*
d. 8000, 3000, 5000, 2000 *FCFS*

Previous block access was 2500, then 4500
Disk direction = toward higher blocks
C-LOOK schedules requests to higher blocks, then goes to the lowest block and starts scheduling all over.
LOOK also schedules I/O as we move from high to low.

25

Question 25

The VFS interface to set a file's attributes is present in this component:

a. inode
b. dentry
c. file
d. superblock

(a) Inodes hold metadata about the file.
(b) Dentries hold directory contents.
(c) File contains info about the status of an file (contents).
(d) Superblock contains info about the file system.

26

Question 26

Which of the following is not a valid function in the file_operations interface for the VFS file object?

a. open
b. lock
c. rename
d. write

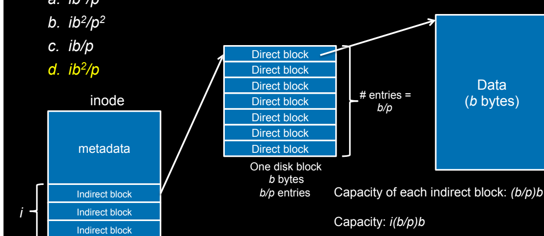
file_operations interact with the VFS file object: info about the file contents.
Rename operates on metadata.

27

Question 27

An inode-based file system that contains *i* indirect blocks, has a block size of *b* bytes, and the length of a block pointer is *p* bytes. The maximum file size is:

a. ib^3/p
b. ib^2/p^2
c. ib/p
d. ib^2/p



28

Question 28

Microsoft's FAT32 file system is an example of:

a. Contiguous allocation.
b. Linked allocation.
c. Indexed allocation.
d. Combined indexing.

The FAT contains a sequence of links to define the allocation of each file.
The directory entry contains the starting block.

29

Question 29

Block groups in the ext2 file system primarily serve to:

a. Improve the reliability of file storage.
b. Provide better wear leveling.
c. Reduce seek latency when accessing files.
d. Reduce rotational latency when accessing files

(a) No.
(b) No.
(c) Berkeley FSS idea: try to allocate disk blocks in the same group as an inode to reduce seek time when accessing a file or related files.
(d) No.

30

Question 30

Differing from *full data journaling, ordered journaling*:

a. Does not ensure that the file system is kept in a consistent state.
b. Ensures that journal entries appear in the exact same order as file operations are issued.
c. Does not perform journaling for all files.
d. Commits a journal entry after writing file data.

(a) File data may be undefined in some cases, but the FS is consistent (no unaccounted for blocks or doubly-referenced blocks).
(b) No different than full data journaling.
(c) No & no different than full data journaling.
(d) Yes. Data blocks are committed to the disk before the journal entry is committed.

31

Question 31

Differing from its predecessor, ext3, the ext4 file system:

a. Uses block groups.
b. Uses extents instead of cluster pointers.
c. Allows journaling to be enabled.
d. No longer uses inodes

(a) Used by BSD FFS, ext2, ext3, ext4.
(b) Yes, arranged in an Htree to speed up seeks.
(c) Yes, but so does ext3.
(d) No. It still uses inodes to store file metadata.

32

Question 32

The logs in a log structured file system such as YAFFS2 differ from journaling in that:

a. Logs are the primary storage structure of the file system.
b. Logs are cleared after the data writes associated with them are complete.
c. Logs improve the integrity of the file system.
d. A sequence of logs represents an indivisible transaction.

(a) Yes. All file system operations are just log writes.
(b) No. The logs make up the file system.
(c) No.
(d) No. There is no concept of transactions with LFS.

33

Question 33

Many mobile devices running Android OS use the YAFFS2 file system because it:

a. Provides wear leveling.
b. Is transaction-oriented for reliability.
c. Allows systems to boot quickly.
d. Avoids the overhead of more complex file systems such as ext4 or FAT32.

(a) Yes. This is the primary motivator. Most phones & tablets use unmanaged NAND flash.
(b) No.
(c) YAFFS boots quicker than other log-structured file systems since it doesn't need to be scanned to reconstruct the FS if it was unmounted previously. It's not better than other non-log-structured file systems.
(d) Not really. The LFS overhead is greater.

34

Question 34

A TLB miss will cause a page fault to occur.

True **[False]**

False.
It will result in a *page table walk*.
If the page table entry (PTE) is not valid, then a page fault will occur.

35

Question 35

A device driver is responsible for implementing the mechanism & policy for device access .

True **[False]**

False.
Device drivers implement mechanism only.

36

Question 36

A cluster is a series of extents.

True **[False]**

False.
A cluster is the smallest allocatable unit of storage on a disk.
1 cluster = N blocks, defined by the file sytem.
An extent is a variable number of consecutive clusters.

37

Question 37

C-SCAN disk read/write scheduling has no benefit on NAND flash memory.

[True] False

True.
Disk scheduling algorithms try to minimize seek time. There is no concept of seek time on NAND flash memory.

38

Question 38

Block groups in the ext2/3/4 file systems have no benefit on NAND flash memory.

[True] False

True.
Block groups were created to minimize seek time: to group allocated data blocks close to the inodes (files) and directory files that reference them.

Block groups actually hurt NAND flash memory since they encourage frequent reuse of the same blocks.

39