Operating Systems Design
Exam 2 Review: Fall 2010

Paul Krzyzanowski
pxk@cs.rutgers.edu

1

---

1. Why could adding more memory to a computer make it "run faster"?

If processes don't have their working sets in memory, that leads to thrashing, which has a big impact on performance. We want system memory to be at least as large as the ∑ of the working sets of all processes.

Not: *more programs can be loaded into memory (good answer for swapping systems)*

Not: *we can have a bigger buffer cache*

2

---

2. Some processors do not support an address space identifier in their TLB. Under what circumstances does its absence affect system performance?

Under context switching.

Without an address space identifier (ASID), context switching will require flushing the TLB, which will lead to a very low initial hit ratio.

Not: *when we have multiprogramming systems.*

Not: *a process may accidentally access another process' memory*

3

---

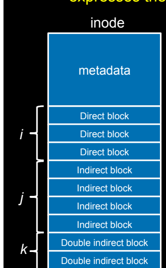
3. A hard link is having two or more file names refer to the same file. Why are hard links easy to implement on an inode-based file system (such as FFS) but difficult on a FAT-based file system (such as Microsoft's FAT32)?

- Under FFS, the directory entry points to the inode (metadata). Multiple directory entries may point to the same inode as long as the inode keeps reference (link) counts.
- Under FAT, the directory entry contains the metadata (there's no place for it in the File Allocation Table). Multiple directory entries could, in theory, point to the same starting block # but the metadata won't stay in sync (you'd also need to figure out how to do reference counting).

4

---

4. An inode based file system has blocks of size *b* bytes. Each block pointer is *w* bytes long. An inode entry contains *i* direct blocks, *j* indirect blocks, and *k* double indirect blocks. What is the maximum possible file size that can be supported? Be sure that your answer expresses the size in bytes and not blocks.

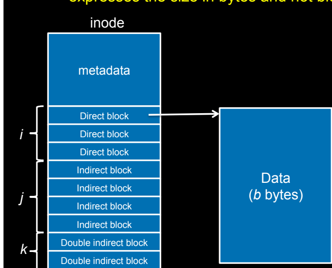


Lots of students got this wrong!!

5

---

4. An inode based file system has blocks of size *b* bytes. Each block pointer is *w* bytes long. An inode entry contains *i* direct blocks, *j* indirect blocks, and *k* double indirect blocks. What is the maximum possible file size that can be supported? Be sure that your answer expresses the size in bytes and not blocks.

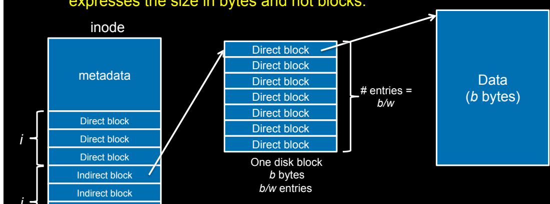


<u>Direct blocks</u>

Capacity = *ib*

6

4. An inode based file system has blocks of size *b* bytes. Each block pointer is *w* bytes long. An inode entry contains *i* direct blocks, *j* indirect blocks, and *k* double indirect blocks. What is the maximum possible file size that can be supported? Be sure that your answer expresses the size in bytes and not blocks.



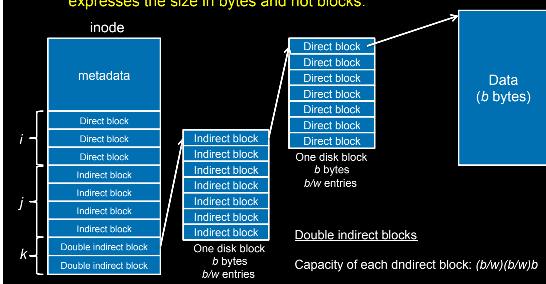
Indirect blocks

Capacity of each indirect block: $(b/w)b$

Capacity: $j(b/w)b$

7

4. An inode based file system has blocks of size *b* bytes. Each block pointer is *w* bytes long. An inode entry contains *i* direct blocks, *j* indirect blocks, and *k* double indirect blocks. What is the maximum possible file size that can be supported? Be sure that your answer expresses the size in bytes and not blocks.



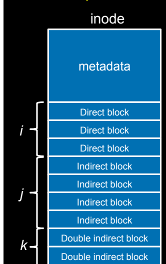
Double indirect blocks

Capacity of each dndirect block: $(b/w)(b/w)b$

Capacity: $k(b/w)^2b$

8

4. An inode based file system has blocks of size *b* bytes. Each block pointer is *w* bytes long. An inode entry contains *i* direct blocks, *j* indirect blocks, and *k* double indirect blocks. What is the maximum possible file size that can be supported? Be sure that your answer expresses the size in bytes and not blocks.



Maximum file size:

$ib + j(b/w)b + k(b/w)^2b$ =

$= b(i + jb/w + kb^2/w^2)$

9

5. How does an extent differ from a cluster?

- Cluster = fixed-size allocation unit on the disk; a logical block.
- Extent = variable-size allocation unit: multiple clusters (blocks)

10

Questions 6-8

6. Base and limit addressing is most useful in:
   a) an inverted page table.
   b) a single partition monoprogramming system.
   c) a direct mapping paging system.
   d) a segmentation system.
7. A hit ratio in a paging system is:
   a) The fraction of memory translations that are made by a memory-based page table lookup.
   b) The fraction of memory translations that are made by the TLB.
   c) The ratio of successful lookups to page faults.
   d) The ratio of memory writes to memory reads.
8. A Translation Lookaside Buffer (TLB) caches:
   a) the contents of frequently accessed memory locations.
   b) frequently accessed pages.
   c) partial page tables.
   d) frequently accessed page table entries.

11

Questions 9-11

9. A certain MMU converts 16-bit virtual address with 512-byte pages into 20-bit physical addresses. How big is the page table?
   a) 128 entries (2^7)
   b) 2048 entries (2^{11})
   c) 65536 entries (2^{16})
   d) 1048576 entries (2^{20})

10. Which memory allocation strategy picks the largest hole?
    a) First fit
    b) Best fit
    c) Worst fit
    d) Next fit
11. The following page table has the same number of entries as there are page frames:
    a) multilevel page table
    b) inverted page table
    c) direct mapped page table
    d) associatively mapped page table

12

Questions 12-14

12. Which page replacement algorithm best approximates LRU (least recently used)?
    a. First In, First Out
    b. Not Frequently Used [at each timer interrupt, increment counters on used pages – no history]
    c. Clock (second chance) [pick a page that has not been referenced]
    d. N^{th} chance [increment a per-page counter if not referenced; pick page with that hasn't been referenced over N faults]

13. Suppose that you have a 32-bit address with a two-level paging hierarchy and a 4 KB page size. The top-level index table has 1024 entries. How many entries does each partial page table have?
    a. 1024 (2^{10})
    b. 262144 (2^{18})
    c. 1048576 (2^{20})
    d. 4194304 (2^{22})

32 bits

1st level	2nd level	offset
10 bits: 1024 bytes (2^{10})	32 - 10 - 12 = 10 bits	12 bits: 4096 bytes (2^{12})

14. Page fault frequency monitoring is a technique for:
    a. managing the resident set for all processes in the system.
    b. evaluating the effectiveness of the TLB.
    c. evaluating the effectiveness of the MMU.
    d. optimizing the interrupt response time for page faults.

13

Questions 15-17

15. An example of a block device is a:
    a) Bluetooth headset.
    b) Scanner.
    c) Video frame buffer.
    d) CD-ROM.

16. A thread should not block if it is executing in:
    a) User context.
    b) Interrupt context.
    c) Kernel context.
    d) (b) or (c).

17. On POSIX systems, a device is identified within the file system by:
    a) A file name that has a unique meaning to the kernel.
    b) A file name that is a hard link to the device driver.
    c) A file name that points to the device table instead of to an inode.
    d) An inode entry that contains a number that is the index into the device table as well as an additional parameter.

14

Questions 18-19

18. The goal of the top half of a device driver is to:
    a) Delegate as much work to the bottom half as possible.
    b) Forward user requests from a system call to the device controller.
    c) Initialize the device driver and register it with the kernel.
    d) Keep track of the number of references to this specific driver.

19. Assume that we're using logical block addresses instead of tracks and sectors to refer to disk locations and assume that low block numbers are on the outside of the disk and high block numbers are on the inside. The last block written is 10,000. The block written before that was 8,000. Our queue for read/write requests contains:

    6000, 11000, 30000, 9000, 20000

    What is the sequence in which the requests will be scheduled with a LOOK algorithm?
    a) 9000, 11000, 6000, 20000, 30000 [Shortest Seek Time First]
    b) 11000, 20000, 30000, 9000, 6000 [LOOK (=SCAN but just seeks until no more requests in that direction]
    c) 11000, 20000, 30000, 6000, 9000 [Circular-LOOK (or C-SCAN): seek to lowest track (or start)]
    d) 6000, 11000, 30000, 9000, 20000 [First come, first served]

15

Questions 20-22

20. The NOOP disk scheduler in the Linux 2.6 kernel is best suited for:
    a) flash memory.
    b) database systems.
    c) real-time systems.
    d) I/O-intensive systems.

21. In the POSIX VFS architecture, which object do you need to access to remove a file?
    a) superblock
    b) inode
    c) dentry
    d) File

    In-class hint: *or change permissions; or change the owner*

22. The File Allocation Table of Microsoft's FAT32 file system is an example of:
    a) linked allocation.
    b) contiguous allocation.
    c) indexed allocation.
    d) combined indexing.

    In-class hint: *derivative, not a direct example*

16

Questions 23-25

23. Larger cluster sizes DO NOT:
    a) improve file data access performance. [Usually yes: we read more contiguous data]
    b) reduce external fragmentation. [Not an issue when allocation units are the same size]
    c) reduce file fragmentation. [A file has more contiguous chunks so it has less opportunity to get fragmented]
    d) increase internal fragmentation. [larger allocation units INCREASE internal fragmentation]

24. Linux's ext2, ext3, and ext4 file systems DO NOT use block groups to:
    a) keep a file's blocks close to each other.
    b) keep an i-node in close proximity to the file.
    c) provide a certain degree of fault resiliency: the file system is not destroyed if one group is destroyed.
    d) make it easy to locate free blocks.

25. Which form of journaling has the least risk of data corruption?
    a) ordered journaling [write data; then just journals metadata]
    b) full data journaling [journal data and metadata]
    c) writeback journaling [metadata journaling but no ordering of data blocks first]
    d) asynchronous journaling [huh?]

17

Questions 26-27

26. Recovery after a crash in a journaled file system involves:
    a) replaying all properly terminated transactions in the log (those marked with a transaction end).
    b) replaying all transactions in the log, including partially-logged ones.
    c) marking all transactions in the log as invalid and clearing the log.
    d) checking the file system integrity.

27. The JFFS2 file system performs
    a) static wear leveling [we move static data to used blocks to distribute the wear]
    b) dynamic wear leveling
    c) both static and dynamic wear leveling
    d) no wear leveling but it is optimized for the longer write times of flash media

18