NETID:_____

[15pts] 7. All's Well That Ends Well

A slight variant of this question was asked on Prelim 2. Suppose you have a Terabyte partition on a disk. To be precise, the partition has 2^{40} bytes on it, subdivided into blocks of 8 Kbytes (8192 = 2^{13} bytes).

a) [1] How many blocks are on the partition?
(Write your answer in the format 2^{xxx}.)

2^{27} (≈ 128 million)

Briefly explain (or provide the work for) your answer:

$2^{40} / 2^{13}$

You want to put a Unix-like file system on the partition, with one superblock in position 0, followed by a sequence of blocks filled with i-nodes. Each i-node is 128 = 2^7 bytes. You want to have enough i-nodes to store 2^{20} (about a million) files.

b) [1] How many i-node blocks do you need to store all these i-nodes?
(Answer in 2^{xxx} format.)

2^{14}

Briefly explain (or provide the work for) your answer:

$2^{20} \times 2^7 / 2^{13}$

A block pointer identifies a block on the partition, and is 4 bytes long (enough to identify 2^{32} blocks). An "indirect block" (a block filled with block pointers) can have 8192 / 4 = 2048 (2^{11}) block pointers.

Suppose now that an i-node contains 13 block pointers. The first 10 point to the first 10 data blocks. The next three point to an indirect block, a double indirect block, and a triple indirect block. The maximum file size can be approximated by just the number of data blocks reachable from the triple indirect block pointer (the rest is negligible).

An i-node also contains a "last modified time" that is updated whenever a file is written, but in our case there are no other timestamps in an i-node.

c) [2] In theory, how much data (in bytes) could be accessed from the triple indirect block pointer in the i-node? For this question, assume the size of the disk is unbounded. (Answer in 2^{xxx} format.)

2^{46}

Briefly explain (or provide the work for) your answer:

$= (2^{11})^3 \times 2^{13}$

Question 7. (cont'd)

d) [2] Assume now that the file system cache is empty except for the superblock. Assume the file with i-node #2015 has the string "Hello World" in it (that is, the file is just 11 bytes long). How many disk accesses would be necessary to read the contents of this file, given that you (and the kernel) know the i-node number?

2

Briefly explain your answer:

- read inode block containing inode #2015 to find the data block number
- read the data block itself

e) [3] In reality this same file's i-node number has to be retrieved first. Suppose the name of the file is /etc/test.txt. Assume that the contents of each directory fits in a single block. The root directory / is described in i-node #2 by convention. Assume /etc is in i-node #5 (you know this, but the kernel doesn't). Again, assuming only the superblock is in the cache and a cache large enough so the same block never has to be read more than once, how many *disk* accesses are required to read the file?

5

Briefly explain your answer:

- read block with inode of root directory (#2) to find location of root directory
- read root directory to find inode # of etc. (#5)
- ~~read block with etc inode (#5) to read etc. to find location of etc directory~~ – *already in cache*
- read etc directory to find inode # of test.txt
- read inode block with inode #2015
- read the content of the file itself

NETID:_____

Question 7. (cont'd)

f) [3] File /etc/shakespeare.txt (which you know to be in i-node #7, but the kernel doesn't) contains the complete works of Shakespeare (2^{22} bytes or about 4 Megabytes). Assuming only the superblock is in the cache, how many disk accesses are required to retrieve the whole thing?

516

Briefly explain your answer:

Shakespeare's complete works fit in $2^{22} / 2^{13} = 2^9$ (512) blocks.

- (1) read block with inode of root directory (#2) to find location of root directory
- (1) read root directory to find inode # of etc (#5)
- ~~read inode block with etc inode (#5) to find location of etc directory~~ – *already in cache*
- (1) read etc directory to find inode # of shakespaere.txt
- ~~read inode block with shakespeare inode (#7)~~ – *already in cache*
- (10) read the first 10 direct blocks
- (1) read the indirect block
- (1014) read the remaining 512 – 10 = 502 blocks

g) [3] Suppose somebody wants to add the text "All's Well That Ends Well." to the end of the complete works of Shakespeare (the new text will be contained in a new data block at the very end). Suppose that the file system has only the superblock in its cache and can allocate free blocks without going to the disk. How many disk reads and how many disk writes are necessary (assuming a "write-through" cache? (Assume that among the i-nodes only i-node #7 has to be updated for this operation.)

4 reads / 3 writes

Briefly explain your answer:

- (1) read block with inode of root directory (#2) to find location of root directory
- (1) read root directory to find inode # of etc (#5)
- ~~read inode block with etc inode (#5) to find location of etc directory~~ – *already in cache*
- (1) read etc directory to find inode # of shakespaere.txt
- ~~read block with shakespeare inode (#7)~~ – *already in cache*
- (1) read the indirect block
- (1) allocate and write the new data block
- (1) write the updated indirect block
- (1) update size and last modified time in inode #7 and write the inode block