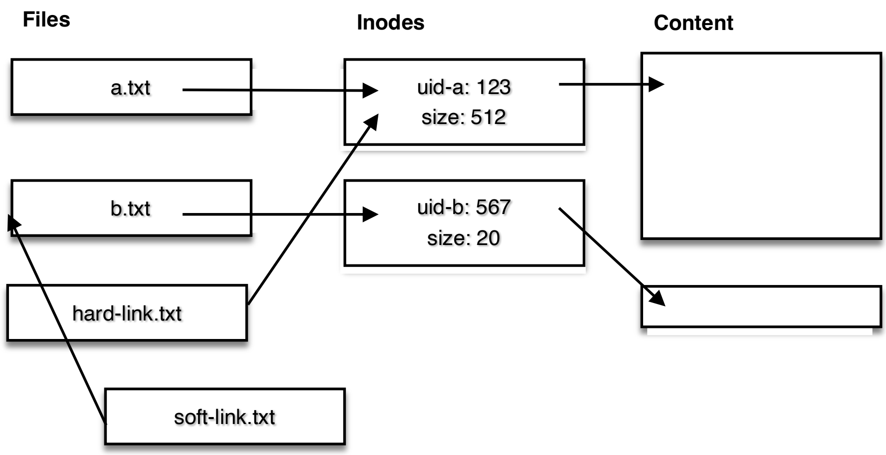
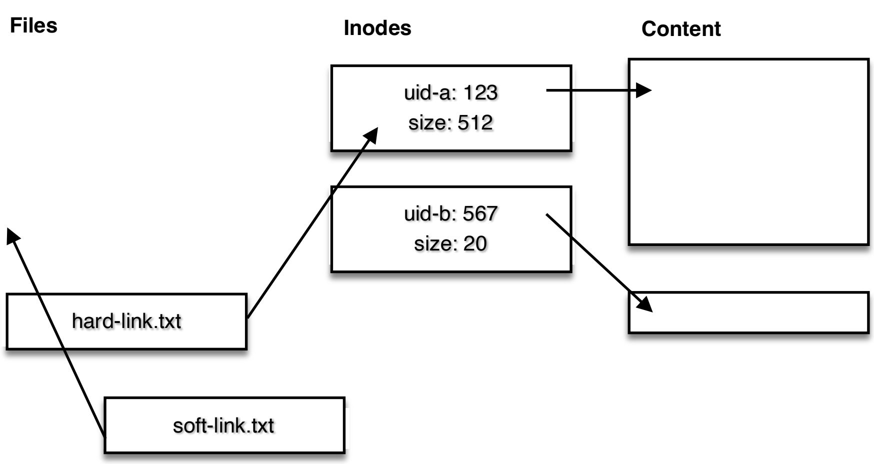
File System Management
Practice Exercises – Solutions
COMP-310/ECSE-427 Winter 23

Question 1

A.



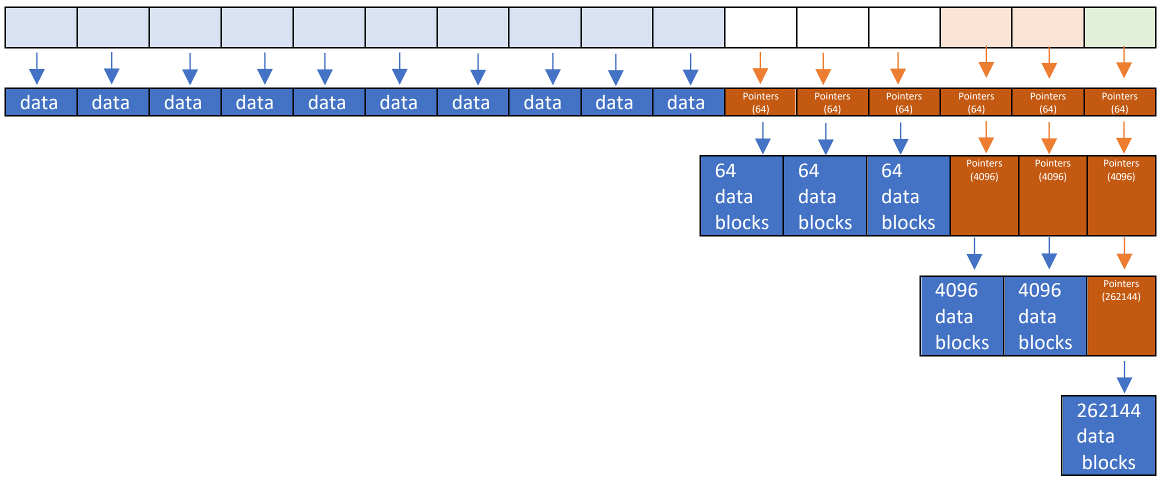
B. soft-link.txt will be a **dangling reference**, while hard-link.txt will **still point to the inode** of a.txt.



Question 2

A. 10 * 512B + 3 * 64 * 512B + 2 * 64^2 * 512B + 64^3 * 512B = 138,515,456 B ~= 13 MB

B.



In this case, the user cannot open a 1GB file, because the maximum file size is 13MB. No disk accesses occur because the requested file is too large.

C. Since we have 10 direct pointers, it is likely that the typical file size is < 5KB because files <5KB are the fastest to access. Good file systems design needs to optimize for the common scenarios, while supporting the uncommon cases (e.g., files larger than 5KB).

Question 3

Max file size = PD * S + PI * P * S + PDI * P^2 * S = S(PD + PI*P + PDI * P^2) Bytes

s

Question 4

Out of SCAN (moving up), C-LOOK (moving up, serving requests on the way up), and SSTF, which disk scheduling policy minimizes the total hard disk arm motion?

→ SSTF

Question 5

Linked Allocation

a) Add block in the beginning: 1 I/O

b) Add block in the middle: 102 I/Os

c) Add block in the end: 202 I/Os

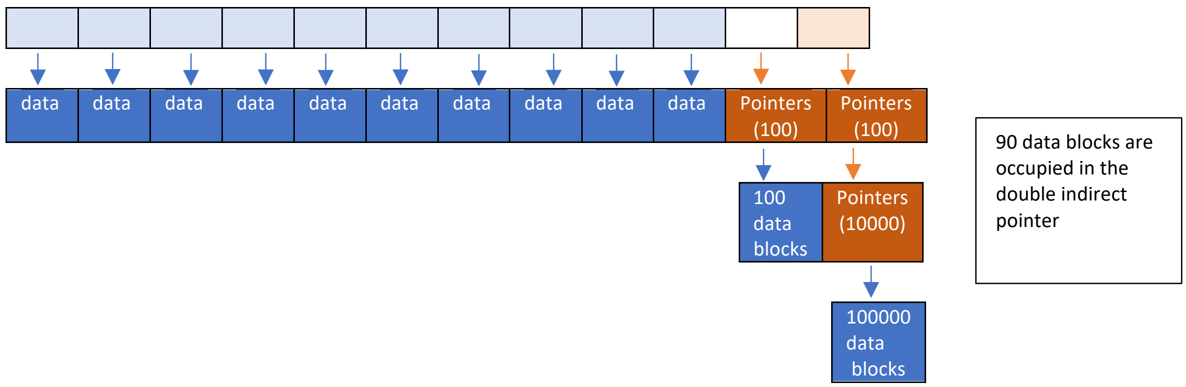
FAT

a) Add block in the beginning: 1 I/O

b) Add block in the middle: 1 I/O

c) Add block in the end: 1 I/O

Indexed allocation with 10 direct pointers, 1 indirect pointer, and 1 double-indirect pointer, where each pointer block can hold 100 pointers
The Inode is in memory, so the first layer of pointers is cached.



a) Add block in the beginning: 1 I/O to write block to disk + 1 + 2*2 I/Os to modify the indirect pointer and the second level of the double-indirect pointer = 6 I/Os

b) Add block in the middle: 1 I/O to write block to disk + 1 + 2*2 I/Os to modify the indirect pointer and the second level of the double-indirect pointer = 6 I/Os

c) Add block in the end: : 1 I/O to write block to disk + 1 + 2 I/O to modify the second level of the double-indirect pointer = 4 I/Os

Question 6

```
fd = Open(bigfile)
for (i = 0; i < 100; i++) {
  R = randomfilelocation()
  Seek( fd, R)
  if (R<2^30-10)
    Read( fd, buffer, 10)
  else
    Read(fd, buffer, 2^30-R)    //if we're close to eof we cannot read 10 Bytes
printf(buffer)
}
Close(fd)
```

Question 7

A. The main advantage of RAID-5 over RAID-4 is more efficient handling of random writes, as RAID-5 does not suffer from the small writes problem. The reason is that the parity blocks are distributed across the entire disk array in RAID-5, as opposed to RAID-4 where the parity blocks reside in a single disk. The parity disk becomes the bottleneck in RAID-4.

B. The minimum number of disks to configure a RAID-5 array is 3. RAID 5 can sustain the loss of a single drive. In the event of a drive failure, data from the failed drive is reconstructed from parity striped across the remaining drives. Therefore, RAID-5 cannot be configured with just 2 drives, as the equivalent capacity of one of the drives needs to be used for the parity blocks.

Question 8

A. In memory: LFS maintains pointers to the Imap pieces, which, when accessed can retrieve the full Imap from disk.
B. On disk: LFS maintains the actual Imap, spread across different segments, as well as the data that is referenced by the Imap. Optional: LFS can maintain a checkpointing region for the Imap on disk, but this is an optimization. It is possible to decide what blocks are old without using the checkpointing region.
C. To figure out whether a block B is logically overwritten by a later write to the same block, LFS does the following:
   - Scan the entire Imap and check whether B is referenced by any of the Inodes.
   - If B is referenced by an Inode, it means that B has not been logically replaced by a newer version.
   - If B is not referenced, it means B has been logically replaced and should be garbage collected.