EEL-4736/EEL-5737 Principles of Computer System Design

Homework #2 solutions

Part A

A-1) Textbook exercise 2.3

2.3a The trouble with synonyms is that if name1 and name2 refer to the same object, {name1, object} is in the cache, and someone refers to name2, we are likely to not realize that we already have object in the cache, and we will look it up again and add it to the cache as the pair {name2, object}. We have wasted time, we have used up cache space, and we have set ourselves up for a disaster if the system permits modifying objects by name: a modification to the copy of object associated with name1 should be propagated to the copy of object associated with name2, but we don't have enough information to realize that.

2.3b If each object has a unique ID one could change the cache to hold {name, object, UID} triples. Then, whenever someone modifies an object by name, search the cache for any other entries that have the same value for UID in their third field and either modify or invalidate those copies.

A-2) Textbook exercise 2.4

2.4 Louis has mistakenly assumed that repeated searches for the same name always resolve to the same object. There are several situations in which this assumption may be false; here are a few examples:

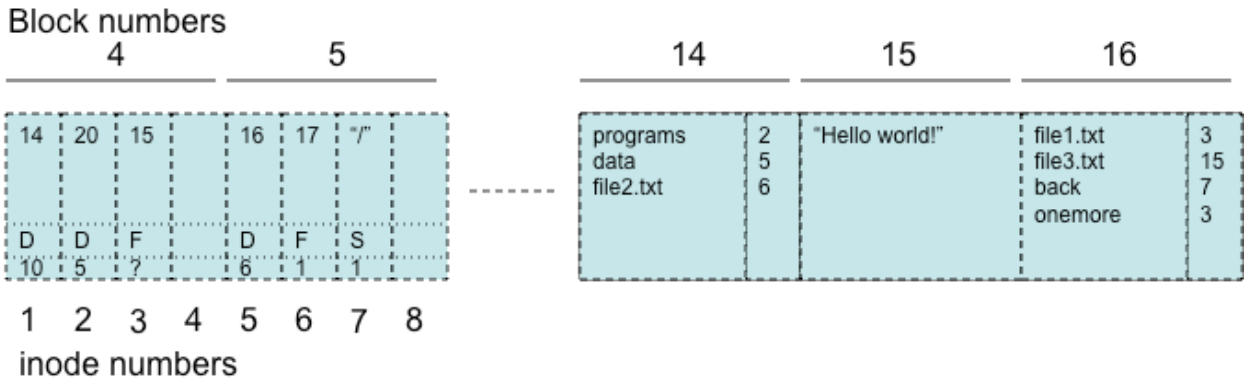
- When one of the items in the search path is the working directory, and the object is found in the working directory. If the user changes the working directory, and tries to use an object that happens to have the same name in the new working directory, the ROT will have priority in resolving the name and the user will get the object from the old working directory, which is probably not what he or she expects.

- When the user changes the search path with the intent of changing the way names get resolved, the ROT will interfere with that intent; all names resolved previously will continue to be resolved the way they were before the change.

- A user refers to an object through a link, causing the path name of the object to be placed in the ROT with the name of the link, and then changes the link to point to something else. The next time the user refers to the link the ROT will lead to the thing the link used to refer to rather than the thing it currently refers to.

In each of those cases, the user will notice a difference. Louis might try to patch around these problems by, for example, resetting the ROT whenever the current thread changes the search path, the user changes the working directory, or anyone on the system modifies a link. But these fixes might eliminate much of the speedup that Louis hoped to obtain, and there are probably other cases he hasn't discovered yet.

A-3) Consider the figure below representing data stored on a computer's hard disk and mounted as the *root file system*. At the bottom of each inode, its **type** (D (directory), F (file), and S (symlink)) and **reference count** are shown (e.g. D, 10 for inode 1), based on the UNIX file system described in class.



i) Suppose blocks are 8KBytes (8192 Bytes) in size, and there are 8 inodes per block. Assume an inode uses 8 Bytes to store type and reference count; the rest of the inode space is used to store block numbers. Suppose the disk has 2^32 blocks, and 2048 blocks are used to store inodes.

a. What is the largest number of files that can be stored?

we have 2048 blocks storing inodes, and 8 inodes per block; each file needs an inode, so we can have up to 8*2048 = 16384 files (technically up to 16383, since the root inode is needed)

b. What is the largest file size that can be stored (in KBytes)?

an inode uses 1024 Bytes of a block; 8 Bytes are used for type and ref count, so there are 1016 Bytes left to store block numbers. Since there are 2^32 blocks, you need 4 Bytes (32 bits) for a block number. So, there can be up to 1016/4 = 254 blocks per inode, and the largest file size is 254 * 8KB = 2032KB

ii) Consider the LOOKUP resolution procedure as discussed in class.

a. What is returned from LOOKUP("data",14)?

not found (inode #14 is not shown)

b. What is returned from LOOKUP("file1.txt",5)?

inode 3

c. What is returned from LOOKUP("programs",5)?

not found

iii) Suppose you create a new hard link “xyz” in directory “/data” that links to “/data/file1.txt”. Describe (in words, or use a drawing based on the figure above) the new state *of all blocks that may have changed* due to this operation

in block 16, you need to add a binding “xyz - 3”. You also need to increment the reference count for inode 3

iv) Assume the sequence of events for two processes (A and B) *running in the same computer where this file system is mounted*:

Time T1: A issues fdA=open(“/data/file1.txt”), and obtains fdA=3
Time T2: B issues fdB=open(”/data/file1.txt”) and obtains fdB=3
Time T3: thread A issues read(fdA,bufA,n=5)
Time T4: thread A issues close(3)
Time T5: thread B issues read(fdB,bufB,n=5)

What values (if any) are returned in the buffer bufA of thread A at time T3, and bufB in thread B at time T5?

@T3: “Hello”
@T5: “Hello” (each thread has their own cursor)

v) Suppose the system has a second hard disk, with a second file system as depicted in the figure below. Suppose you mount this second file system under directory “/mnt” of the root file system. Is it possible to create a hard link “file4.txt” under “/data” that links to “file4.txt” in the second file system? Is it possible to create a symbolic link “file4.txt” under “/data” that links to “file4.txt” in the second file system? Describe (in words, or use a drawing based on the figure above) the new state *of all blocks that may have changed* due to these operations, if they are possible.

It is not possible to create a hard link to an inode in a different file system.
It is possible to create a symbolic link.
In block 16 of the first file system, you need to add a binding “file4.txt - xx”, where xx is any free inode (e.g. inode 8). In inode 8, its type should be “S”, reference count 1, and it stores the string “/mnt/file4.txt” instead of block numbers.