Homework6

Name Yu Yang Student ID 892449550

11.10

Open-file table entry of a file could only be removed from the table if all users close that file. One table would suffice because OS could keep track of the count number of how many processes are still using it. When the number is zero, the entry could be removed for that one table.

If two or more users are accessing one file, processes may perform different operation to different location of a file (or same location), the system should be able to separately keep track of those ops, which require separate entries for different processes

11.11

A mandatory lock will prevent any other process from accessing the locked file. But there are many cases which processes could share a file concurrently (like a reader lock), where mandatory locks would significantly decrease efficiency and flexibility.

But in other hand, advisory locking could cause unexpected race conditions and errors.

11.12

The answer to which one is better depends on the demands of users. Leaving it to users would make the OS size small and maximize the flexibility.

Keeping track of the type or implement specific types would provide speed up for specific application cases if most users in the OS share common tasks.

11.14

If arbitrarily long names could be used, the simulation could be simply done by using specific “separator” to indicate the boundaries of directories. For a example, if we use “\” as separator, “/usr/doc/a.file” could be represented as “\usr\doc\a.file”.
If names’ length is limited (to seven chars in this case), we need to use up to seven characters to represent arbitrarily long directory path. By limited space we could not finish this a possible solution is using hashing to hash multilevel directory files, using additional separating chaining lists to solve conflicts.

12.9

Scheme a has the simplest block allocation because all extends are the same size. But a has big internal fragmentation.
Scheme b has the most complex allocation. B has no internal fragmentation (variable size), there might be external fragmentation in B.
Scheme C has medium level of allocation complexity and fragmentation between a and b.

12.11

When the target block is at the middle of a file, we could traverse in pointers stored in FAT rather than through blocks. And because FAT could be cached memory, we could get a improved random-access time by reading FAT information in memory.

12.12

a. Could be reconstructed by doing garbage collection, linking all the unallocated pages in to free-space list.
b. 4 ops. 1). Reading in block which contains the root dir 2) reading in block which contains a3) reading in block which contains b 4) reading in block which contains file c
c. We could store the pointer on the disk, with RAID supporting multiple backups.

12.19

In order to recover a file system, we should make the recovered one consistent.
In log-structured file system, all metadata changes are written sequentially to a log (which

guarantee we maintain a same order of operations). Once the changes are written to the log, they are considered to be committed. Meanwhile, these log entries are replayed across the actual file system. Only when a committed transaction is completed, it is removed from the log file. Therefore, when the system crashes, the log file will contain transactions not completed in actual file system. So that we should complete those transactions in the same odder, file system is recovered.

The only problem is aborted transaction which is committed before the system crash. For such cases, any changes should be undone.

By performing the actions above, any problems with consistency checking is eliminated.