Solutions to exercises on file systems

1. How do caches improve performance of the file system?

2. The open-file table is used to maintain information about files that are currently open. How the OS knows when to close an entry in this table, given that many processes may have opened the same file?

3. Assume we have an i-node with 10 direct address pointers and 1 single indirect address pointer. The address pointers are 4 bytes long, a block is 1024 bytes long. What would the largest possible file in this system?

   Sol: So, we have 10 x 1024 or 10240 bytes of capacity through the direct address pointers, plus what is reachable from the indirect pointer. Assume that points to a block of 256 4-byte pointers, allowing 256 x 1024 or 262,144 bytes. Add these to get 272,384 bytes as the maximum file size.

4. Consider a 4-TB disk that uses 4-KB blocks and the free-list method. How many block addresses can be stored in one block?

   Sol: 4-TB is 2^{42}, a block is 4-KB = 2^{12}, thus the number of blocks is $2^{42}/2^{12} = 2^{30}$. We need 30 bits to address all the blocks, however assuming block addresses are expressed in bit strings that are a power of 2, then we need 32 bits = 4 bytes to address a block. One block is 4096 bytes, $2^{12}/2^2 = 2^{10} = 1024$, thus one of the free-list can potentially hold 1024 block addresses. However, 4 bytes in a block of the free-list must be used to link with the next block in the free-list, thus one block can store 1023 block addresses.

5. The beginning of a free-space bitmap looks like this after the disk partition is first formatted: 1000 0000 0000 0000 (the first block is used by the root directory). The system always searches for free blocks starting at the lowest-numbered block, so after writing file A, which uses six blocks, the bitmap looks like this: 1111 1110 0000 0000. Show the bitmap after the following actions: File B is written, using five blocks, File A is deleted, File C is written, using eight blocks, File B is deleted.

6. What would happen if the bitmap or free list containing the information about free disk blocks was completely lost due to a crash? Is there any way to recover from this disaster, or is it bye-bye disk? Discuss your answers for UNIX and the FAT -16 file system separately.

7. Consider a disk that has 10 data blocks starting from block 14 through 23. Let there be 2 files on the disk: f1 and f2. The directory structure lists that the first data blocks of f1 and f2 are respectively 22 and 16. Given the FAT table entries as below, what are the data blocks allotted to f1 and f2?

   (14,18); (15,17); (16,23); (17,21); (18,20); (19,15); (20, -1); (21, -1); (22,19); (23,14).

   In the above notation, (x, y) indicates that the value stored in table entry x points to data block y.

8. A Unix file system has 1-KB blocks and 4-bytes disk addresses. What is the maximum file size if i-nodes contain 10 direct entries and one single, double and triple indirect entry each?

   Sol:

   - One direct block address points to a 1 KB data block, so there is space for 10 KB of data using the direct pointers.
   - Address pointers take 4 bytes. # of address pointers in an indirect block is 1KB / 4 = 256 addresses. A single indirect block points too 256 data blocks, thus 256 $\times$ 1 KB = 256 KB.
   - In double indirect block, the first indirection point to 256 address blocks, each address block point to 256 data blocks. Thus the number of data blocks that are pointed to is 256 $\times$ 256 = 65536 $\times$ 1 KB = 64 MB.
   - Triple direct block, this can be understood as adding a block of address pointers, each pointing to a double indirect block. A block of address pointers has 256 pointers, each double indirect block hold 64MB, thus 256 $\times$ 64 MB = 16 GB.

   The maximum file is 10 KB + 256 KB + 64 MB + 16 GB

9. How many disk operations are needed to fetch the i-node for afile with the path name /usr/ast/courses/os/handout.t? Assume that the i-node for the root directory is in memory, but nothing else along the path is in memory. Also assume that all directories fit in one disk block.

   Sol: There are ten disk operations are needed to fetch the inode for the file /user/ast/courses/os/handout.t. Since i-node for root is already in memory, the ten operations are:

   - 1 fetch the / directory
   - 2 fetch the i-node for usr
   - 3 fetch the /usr directory
   - 4 fetch the i-node for ast
   - 5 fetch the /usr/ast directory
   - 6 fetch the i-node for courses
   - 7 fetch the /usr/ast/courses directory
   - 8 fetch the i-node for os
   - 9 fetch the /usr/ast/courses/os directory
   - 10 fetch the i-node for handout.t