Rashed Abdulla Alyammahi

1. rm command line use unlink to remove file names or directories.
Files existing in hard disk contains their name and memory data, in addtion, before remove files, linux OS needs to check files status such as permission, files are opening or not. So there isn't any 1 system call to remove file name and their data.

2. Linux manages files in directory by linking tables, and use path name to know where files&directories are. In linux, root usually is "/", and files/directories path name start from root. Rename system call can change their path name, it means to change the links between tables. This doesn't change file data, just change path name, so it reduces workload instead of copy&delete.

3.
/var/log/boot.log
direcroty for /
inode for /*var*
direcroty for /*var*
inode for /*var/log*
direcroty for /*var*/*log*
inode for /*var/log/boot.log*
There are 6 disk operations.

4.
a. inodes near start of disk in HDD have advantage when access inodes to extract information because read/write speed is faster near the start of HDD.
Inode contains file information, so it'll be easy to implement such as extract file status when putting inodes at the start of the first data blocks of the file.

b. In FFS, when access file data, HDD pointer will go to where data is stored in, so it takes time when go to data block then come back to start of disk to search another inode. FFS arranges related inodes and their data near each other, so it'll be faster when go to another inodes. Also, there is risk when let all inodes near each others, If there are any mistake in inodes blocks, That can cause miss information blocks.

c. There is no different with FFS approach in SSD because the speed in every sector in SSD is the same.

5. The OS cannot know the free space to allocate, so it causes corrupt system.
We can recover the bitmap by scanning all inodes to find free blocks.

6.
a. This would decrease internal fragmentation. Example if OS needs to allocate a 6KB file, then it'll allocate a 4KB block and 4 contiguous 512-byte blocks.
b. OS needs to maintain bitmap to manage sub-blocks are using inside a block, so OS needs to add an extra state to know which block contains used sub-block and still have free sub-block. The allocator would have to check this extra state to allocate sub-blocks and combine the sub-blocks to obtain the larger block when all of the sub-blocks is free.

7. Direct blocks = 12, block size = 16Kb = 2^14 Bytes, pointer = 8 Bytes
a. maximum file size = (12 + 2^14/8 + (2^14/8)^2 + (2^14/8)^3)*2^14
= 3*2^16 + 2^25 + 2^36 + 2^47 = 192KB + 32MB + 64GB + 128TB
= 128.064032192 TB
b.
The index block can hold 16KB/8 = 2048 pointer to data block
The single indirect pointer indirectly points to 2048*16KB = 33554432 Bytes
The double-indirect pointer indirectly points to 2048^2 * 16KB = 67 108 864 000 bytes
The triple-indirect pointer indirectly points to 2048^3 * 16KB = 128TB

- ➔ 4096 need 2 direct data blocks (as the size of block is 2048)

32768 need 12 direct data blocks and (32768-12*2048)/2048 = 4 data blocks

- ➔ 32768 need 12 direct data blocks and should be stores in 4 blocks using 1 single indirect pointer.

1048576 need 12 direct data blocks and (1048576-12*2048)/2048 = 500 data blocks

- ➔ 1048576 32768 need 12 direct data blocks and should be stores in 500 blocks using 1 single indirect pointer.

268435456 need 12 direct data blocks + 1 single indirect pointer (2048 data blocks) +
(268435456 -12*2048 – 2048*2048)/2048 = 129012 data blocks in double indirect pointer.

- ➔ 268435456 need 12 direct data blocks and 1 single indirect pointer and should be stores in 129012 block using 1 double indirect pointer.

8.
reading one page takes 25µs, programming one page takes 200µs and erasing one block takes 1500µs
Block 0 has three dead blocks (page 0 1 2) and 1 live block (page 3, which contain block 100).
Read live data (page 3) from block 0. It takes 25µs.
Write 1 live data to end of the log. It takes 200µs.
Erased block 0. It takes 1500µs.
It takes 1500µs+200µs+ 25µs = 1725µs to garbage collection to reclaim physical block 0.