1. When remove files or directories, linux OS also needs to remove their name in inodes and data. *unlink* system call removes name.

Before linux OS removes files or directories, it needs to check permission, file is be using in some applications. So there’s no way only 1 system call to do all this stuff.

2. Linux use inodes to manage files and directories in system, to get a file path name, linux OS needs to search in inodes. Rename system call changes the name in inodes, which allow not to change or copy data but rename the path name of file/directory, this can save effort of CPU and make performance better.

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.

The speed is faster when HDD cursor is near the center of the disk, so you’ll quickly get information from the inodes and also it’s easy to implement when putting inodes at the start of the disk.

putting inodes at the start of the first data blocks of the file help OS manages inodes and their data more effective.

b. When access files, OS needs to browse the directory inode and look up their data to know the next inode is. So if the inodes and their data is far away, it’ll spend a lot of time to go to where the file is. So a file are placed near the start of the same block group as the file’s data blocks help save time to access file.

c. The speed while access the sectors of SSD is the same because of SSD architecture. So maybe FFS approach doesn’t have more advantages.

5. The OS cannot know the free space to allocate, so it causes corrupt system.

This can cause corrupt all system because allocator cannot find any free space for another programs execute. You can recover the bitmap by scanning all inodes to find free blocks.

6.

a. This would reduce internal fragmentation because file data can be store more effective by allocate sub-block 512 bytes.

b. bitmap management needs to be modified by ticking which block contains used&unused sub-blocks. So the allocator will know the free sub-blocks to fill data into.

7.

Direct blocks: 12

block size: 16Kb  
pointer: 8B

a. maximum file size = (12 + 16KB/8 + (16KB/8)^2 + (16KB/8)^3)\*16KB

= 192KB + 32MB + 64GB + 128TB

= 128.064032192 (TB)

b.

The index block can hold 16KB/8 = 2048 pointers to data block

The single indirect pointer: 16KB/8\*16KB = 32 (MB)

The double-indirect pointer: (16KB/8)^2 \* 16KB = 64 (GB)

4096: 2 direct data blocks (as the size of block is 2048)

32768: 12 direct data blocks and (32768-12\*2048)/2048 = 4 data blocks using 1 single indirect pointer.

1048576: 12 direct data blocks and (1048576-12\*2048)/2048 = 500 data blocks using 1 single indirect pointer.

268435456: 12 direct data blocks + 1 single indirect pointer (2048 data blocks) +

(268435456 - 12\*2048 - 2048\*2048)/2048 = 129012 data blocks in double indirect pointer using 1 double indirect pointer.

8.

Block 0 contains 3 dead blocks (page 00 01 02) and 1 live block (page 03).

Read page 03 in block 0: 25us

Write page 03 to end of the log: 200us

Erased block 0: 1500us

So, it takes 25+200+1500=1725us for garbage collection to reclaim physical block zero.