1.

a. RAID-5:2 RAID-6:3

b. RAID-5:7 RAID-6:7

2.

a. FCFS:13011

b. SSTF:7559

c. SCAN:7492

d. LOOK:7424

e. C-SCAN:9917

f. C-LOOK:9137

3.

open-file table:打开文件表，内核对所有打开的文件维护一个系统级别的文件打开表，存储了与一个打开文件相关的全部信息，包括文件偏移量，状态标志信息，i-node 对象指针。

why we need it：通过打开文件表，将已打开的文件信息缓存在内存中，进而通过打开文件表完成文件索引，避免每次访问都需要遍历目录，减少 IO 开销，提高性能。另一方面，全局下的打开文件表可以维护一个文件的打开状态，确认当前是否正在被使用，正确进行删除操作等。

4.

755：755 的文件权限，权限按照 R、W、X 对应排列，第一个 7 代表文件拥有者具有读写和执行的权利，后面两个 5 分别代表文件所属组的其他用户、组外用户具有读和执行的权利。

5.

问题：连续分配会产生外部碎片，导致创建大文件时没有足够的连续空间。当文件需要追加写入时，不能正常拓展文件大小。

解决方法：将文件分块存储。

6.

使用 FAT 的好处：更好的随机访问性能，访问文件中间部分的块时，查找存储在 FAT 中的指针来确定其位置，而非按顺序访问文件的所有块来找到指针指向目标块的指针。大多数 FAT 都可以缓存在内存中，因此可以通过内存访问来确定指针，而不必访问磁盘块。

FAT 的主要问题：部分缓存较难实现，使用空间换时间，缓存 FAT 以获取更好的访问性能，会产生内存空间的浪费。

7.

a. 无缓存

root directory -> inode for /a -> disk block for /a -> inode for /a/b -> disk block for /a/b -> inode for /a/b/c -> disk block for /a/b/c 7 次

b. 缓存 inode

root directory -> inode for /a -> disk block for /a -> inode for /a/b -> disk block for /a/b -> inode for /a/b/c -> disk block for /a/b/c 4 次

8.

(12 * 8 KB) + (2048 * 8 KB) + (2048 * 2048 * 8 KB) + (2048 * 2048 * 2048 * 8 KB)=68,753,047,648k

9.

hard link:1.创建一个目录条目指向一个存在的文件。

2.硬链接条目和原条目共同指向一个 inode，并不创建新的文件内容。

3.对应 link count+1,实际上是一个文件有两个路径

symbolic link:1. 创造了一个新的目录条目和 inode，它的目录条目指向的是这个新的 inode
2.这个 inode 里存储的是指向的文件路径。
3.对应 link count 没有改变，实际上可以看作是一个文件的快捷方式

10.

Polling, interrupt, DMA

11.

1.I/O 调度(I/O scheduling)
2.缓冲(Buffering)
3.缓存(Caching)
4.假脱机(Spooling)
5.错误处理和 I/O 保护(Error handling and I/O protection)
6.电源管理(Power management),etc