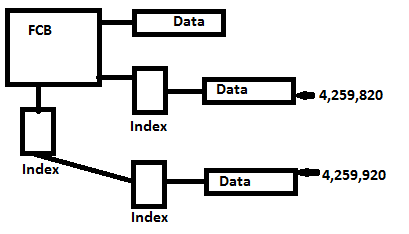
Q1.

4KiB = 2^12, 2^12/2^2=2^10, 16 = 2^4

1. 16\*2^12= 2^16 = **65,536** bytes
2. 2^10\*2^12 + 2^16 bytes = **4,259,840** bytes
3. 2^10\*2^10\*2^12 + 2^10\*2^12 + 2^16 bytes = **4,299,227,136** bytes
4. 2^10\*2^10\*2^10\*2^12 + 2^10\*2^12 + 2^16 = **4,402,345,738,240** bytes
5. bytes 1,024^6 = 2^60,

2^10\*2^10\*2^10\*2^10\*2^10\*2^10 = 2^60 bytes => 2^60 => (**5 levels**).

1. 4,050/4,096 = 0.98, 4,150/4,096 > 1 and < 2, so **2** blocks were accessed (direct block).
2. 4,259,820/4,096 = 1,039.9 – 16 = 1,023.9(last index in single indirect block), 4,259,920/4,096 = 1,040.019 – 16 = 1,024.019 (first index in double indirect block), hence 2+3=**5** blocks were accessed.
3. 4,263,900/4,096 = 1,040.99 – 16 = 1,024.99 (first index in double indirect block), 4,264,000/4,096 – 16 = 1,025.015(second index in double indirect block), hence **4** blocks were accessed.
4. 4,259,820/4,096 = 1,039.995 – 16 = 1,023.995, Read: 2+3 = 5 blocks, Write: 4,259,920/4,096 = 1,040.019 = 16+1024+0.019, 2 + 1 inode = 3, hence 5+3= **8** blocks were accessed.
5. 4,259,820/4096 = 1,039.995 – 16 = 1,023.995, Read: 1 + 1 = 2 blocks, Write: 4,259,920/4,096 = 1,040.019 = 3 + 1 = 4 blocks + 1 inode, hence 2+5= **7** blocks were accessed.



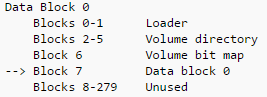
Q2

512KiB = 2^9, 2^9/2^1=2^8

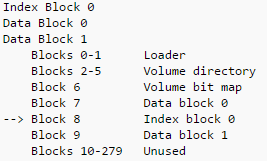
1. Table

|  |  |  |  |
| --- | --- | --- | --- |
| Block Size | 512 | Index Block | 256 |
|  | Seedling | Sapling | Tree files |
| Minimum | 0 | 512< data | 131072<data |
| Maximum | 512 | 131072 | 33554432 |

1. This file system called ProDos file system. The reason there are only 128 index block pointers in the first level index block for a tree file, due to the limitation of memory is not larger enough to hold the data.
2. If a new file of nondirectory type is open, one data block is allocated to that file. An entry is placed in the volume directory, and it points to block 7, the new data block, as the key block for the file. The key block is indicated below by an arrow.

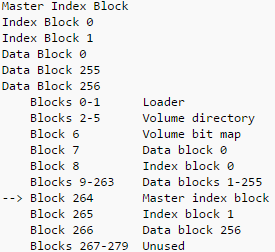


If the file grows into sapling file (more than 512 bytes). An index block is allocated, and it becomes the file's key block. A second data block is also allocated.



Hence bit map updated, Block 8, and 9 used, totally **3 blocks updated**.

1. As we can read from the following figure, a master index block is allocated, index block 1 is allocated, a new data block is allocated and bit map is updated, so totally **4 blocks are updated**.



Q3

1. FIFO, page fault: 12

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 4 | 3 | 2 | 1 | 6 | 7 | 1 | 2 | 3 | 7 | 5 | 1 |
| 0 | 1 | = | = | = | 5 | = | = | = | = | = | = | = | 2 | = | = | = | = |
| 0 | = | 2 | = | = | = | = | = | = | 1 | = | = | = | = | 3 | = | = | = |
| 0 | = | = | 3 | = | = | = | = | = | = | 6 | = | = | = | = | = | 5 | = |
| 0 | = | = | = | 4 | = | = | = | = | = | = | 7 | = | = | = | = | = | 1 |

1. LRU, page fault: 11

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 4 | 3 | 2 | 1 | 6 | 7 | 1 | 2 | 3 | 7 | 5 | 1 |
| 0 | 1 | = | = | = | 5 | = | = | = | 1 | = | = | = | = | = | = | 5 | = |
| 0 | = | 2 | = | = | = | = | = | = | = | = | = | = | = | = | = | = | 1 |
| 0 | = | = | 3 | = | = | = | = | = | = | = | 7 | = | = | = | = | = | = |
| 0 | = | = | = | 4 | = | = | = | = | = | 6 | = | = | = | 3 | = | = | = |

1. OPT, page fault: 9

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 4 | 3 | 2 | 1 | 6 | 7 | 1 | 2 | 3 | 7 | 5 | 1 |
| 0 | 1 | = | = | = | 5 | = | = | = | = | 6 | 7 | = | = | = | = | 5 | = |
| 0 | = | 2 | = | = | = | = | = | = | = | = | = | = | = | = | = | = | = |
| 0 | = | = | 3 | = | = | = | = | = | = | = | = | = | = | = | = | = | = |
| 0 | = | = | = | 4 | = | = | = | = | 1 | = | = | = | = | = | = | = | = |

Q4

Win7 -> Win8, Win10:

**Memory combing** is a technique in which Windows efficiently use RAM by free same content in the memory, which reduce the memory footprint even for services and OS components. For example, if there are multi applications are running, and the same content may in the memory, the new Windows can reduce the duplicate content by freeing the same content in the RAM, which reduce the usage of the RAM.

**Doing the same job with less memory**. As application executes, its own system housekeeping, program files and data are loaded into main memory. These pages in the memory were analysed for recognising which pages are frequently used by applications and which pages are not frequently used. Win8 improved the low-level OS to allocate the memory for frequency and non-frequency pages.

**Service changes and reductions,** some services are not running as windows starts, they are trigged as demand of any application. This also reduces the usage of memory.

**More granular prioritization of memory**: Win8 has a better scheme for the prioritisation of memory allocations made by applications and system components. This means the operation system can make better decisions about what memory (page) to keep and what memory (page) will need to be removed soon.

**Windows 10 Memory Compression.** Thisis new feature in Win10, the goal of the memory compression is let operating system allocating the memory efficiently. It reduces the amount of reads and writes to the pagefile.