5.5/12

COSC 450 Operating System Mini-3

4/18/24

Name: Kyle Trafaglia.

1. (2 pt.) Consider the following sequence of memory references from a 780-byte program:
   712, 86, 123, 234, 34, 312, 77, 412, 211, 334, 34, 305, 287, 111, 201, 76, 145, 742, 89, 166

   a) Give the reference string, assuming a page size of 100 bytes.
   Since page size is 100 bytes, virtual space of 780 byte program can be saved in 8 pages.
   Page 0 ~ page 7

   -.2 X

   b) Find the page faults (number of page faults) for the reference string in part a) assuming 300 bytes of physical memory available to the program and LRU replacement algorithm

   X

   c) With same assumption, how many page faults would be if you use an optimal replacement algorithm?

   X

2. (1 pt.) A system use bit-map to keep track of free-blocks. Let' say a block size is 2KB. The system use 2^{12} blocks for bit-map. What is the total disk size? (MB or GB)

-.7

2^{12} blocks $\cdot$ 2×2^{10} $= 2^{23}$ byte $\times 2^3 = 2^{26}$ bit

$\times 2\times 2^{10} = 2^{37}$ bytes $\rightarrow 2^7 \times 2^{30} \rightarrow \boxed{128\ GB}$

X

3. (1 pt.) Page size is one of most important design issue in the operating system. We can mathematically analyze page size based on following assumptions:
   - S: average size of process (byte)
   - P: the size of page (byte)
   - E: Each page entry requires (byte)
   - 50% of memory in the last page of the process is wasted due to internal fragmentation

   a. Define the total overhead function based on page size P.

   $$\text{Total overhead} = \frac{SE}{P} + \frac{P}{2}$$ ✓

   b. Find the optimal page size formula based on the total overhead (by minimize the total overhead)

   $$\text{Total overhead}' = -\frac{SE}{P^2} + \frac{1}{2}$$ ✓

   $$\text{Optimal Page size} \rightarrow P^2 = 2SE \rightarrow P = \boxed{\sqrt{2SE}}$$ ✓

4. (2 pt.) In the file system, two methods are widely used to keep track of free blocks: a linked list and a bitmap. Let's say a block size is 4-KB and 32-bit disk block number in a file system.

   a. How many maximum blocks are needed for keep track 128-GB disk with linked list?

   $$128\,GB / 4\,KB = 2^7 \times 2^{30} / 2^2 \times 2^{10} = 2^{37} / 2^{12} = 2^{25} \text{ blocks}$$

   $$2^3 \cdot 2^{12} / 32 = 2^{15} - 1 / 2^5 = 2^{10} - 1$$ ✓

   $$2^{25} / 2^{10} - 1 = 2^{15} - 1 = \boxed{32767 \text{ blocks}}$$

   b. How many blocks are needed for keep track of 128-GB disk with bitmap?

   2^{25} blocks ... calculated above 2^{25} bit $/ 2^3 = 2^{22}$ byte

   $$2^{22} / 4 \times 2^{10} = 2^{22} / 2^{12} = \boxed{2^{10} \text{ blocks}}$$ ✓

   c. What is the maximum disk size supported by this file system?

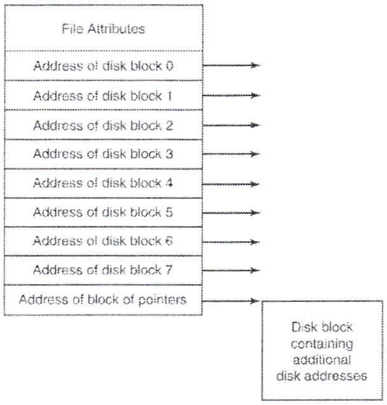
   $$2^{32} \times 4\,KB = 2^2 \times 2^{30} \times 2^2 \times 2^{10} =$$

   $$2^{32} \times 2^{12} = 2^{44} \rightarrow \boxed{16\,TB}$$ ✓

5. (2 pt.) LINUX like system use i-node to maintain the file system. Attributes and block addresses are saved in i-node. One problem with i-nodes is that if each one has room for a fixed number of disk addresses, what happens when a file grows beyond this limit? One solution is to reserve the last disk address not for a data block, but instead for the address of block containing more disk-block addresses as shown following picture. Picture shows that i-node contains 8 direct addresses and these were 64 bits each. A block size is 4 KB. If a file use i-node and one extra block to save block information, what world be the maximum file size?

-1

$$8 \times \underline{4 \cdot 2^{10}} / 2^6 = 2^6 = 64 - 1$$



$$63 + 8 = 71 * 4 \cdot 2^{10} = \boxed{290816}$$

Block size / ~~B~~ blocks per bit address + extra direct addresses * block size = max file size

6. (2 pt.) System need save backups to handle recover from disaster. Two strategies can be used for dumping a disk to tape: a physical dump or logical dump. Briefly discuss **physical dump** and **logical dump algorithm used in UNIX system**.

-.8

logical dump: All directories changed in a given time period are recursively dumped.

Physical dump: Starting at 0, changed files are stored to tape until last changed file is read, then all are dumped.

did not discuss algorithm!