COSC 450 Operating System Test #2

11/21/2024

Name:_____.

1. (10 pt.) In the file system, two methods are widely used to keep track of free blocks: a linked list and a bitmap.
   a) A system uses a bit-map to track free blocks in its file system. Each block is 4 KB in size, and the system allocates 2^{18} blocks for the bit-map. What is the total size of the disk?

      Size of bit-map = $4 \times 2^{10} \times 2^{18}$ Byte = $2^2 \times 2^{10} \times 2^{18} \times 8$ Bit = 2^{33} bit.
      There are 2^{33}blocks
      Total disk size = $2^{33} \times 4 \times 2^{10} = 2^{45}$ Byte = 32 TB

   b) A system uses a free-list to track free blocks in its file system. Each block number in the free-list is represented using 32 bits, and the block size is 4 KB. The system allocates 2^{20} blocks for the free-list. What is the total size of the disk?

      sol) # of block information per block = (block size / bit used for a block #) − 1
      = $8 \times 4 \times 2^{10}$ / 32 bit = 2^{10} − 1 = 1024 − 1 = 1023
      total # block number = 1023×2^{20}
      total disk size = $1023 \times 2^{20} \times 4 \times 2^{10}$ =1023×2^{32} Byte = about $2^{10} \times 2^{32}$ Byte = 2 TB.

2. (5 pt.) Discuss each of followings.
   a) What are four necessary conditions for a deadlock
      - Mutual exclusion
      - Hold-and Wait
      - No preemption
      - Circular wait
   b) Four strategies for dealing with a deadlock
      - Ignore
      - Detection and recovery
      - Avoidance with dynamic allocation
      - By attacking one of necessary deadlock condition

3. (15 pt.) Consider a system with 5 processes (P_0 … P_4) and 3 resources types (A, B, C) with E = (10, 5, 7). Resource-allocation state at time t_0 shows in table.

Process	Allocated			Max Need		
	A	B	C	A	B	C
P_0	0	1	0	7	5	3
P_1	2	0	0	3	2	2
P_2	3	0	2	9	0	2
P_3	2	1	1	2	2	2
P_4	0	0	2	4	3	3

a) What are the values of Request Matrix R, and available matrix A

A = (3, 3, 2) R =

R		
A	B	C
7	4	3
1	2	2
6	0	0
0	1	1
4	3	1

b) Will a request of (1, 0, 2) by P_1 be granted? (it is not yes/no problem)

Process	C			R			A		
	A	B	C	A	B	C	A	B	C
P_0	0	1	0	7	4	3	2	3	0
P_1	3	0	2	0	2	0			
P_2	3	0	2	6	0	0			
P_3	2	1	1	0	1	1			
P_4	0	0	2	4	3	1			

A = (3, 3, 0) – P_1 – (5, 3, 2) – P_3 – (7, 4, 3) –P_0 –(7, 5, 3)- P_2 –(10, 5, 5) –P_4 –(10, 5, 7)

c) Will a request of (3, 2, 0) by P_4 be granted? (it is not yes/no problem)

Process	C			R			A		
	A	B	C	A	B	C	A	B	C
P_0	0	1	0	7	4	3	0	1	2
P_1	2	0	0	1	2	2			
P_2	3	0	2	6	0	0			
P_3	2	1	1	0	1	1			
P_4	3	2	2	1	1	1			

A = (0, 1, 2) – P_3 – (2, 2, 3) – P_1 – (4, 2, 3) –P_4 –(7, 4, 5)- P_0 –(7, 4, 5) –P_2 –(10, 5, 7)

d) Will a request of (3, 3, 0) by P_4 be granted? (it is not yes/no problem)

Process	C			R			A		
	A	B	C	A	B	C	A	B	C
P_0	0	1	0	7	2	3	0	0	2
P_1	2	0	0	1	2	2			
P_2	3	0	2	6	0	0			
P_3	2	1	1	0	1	1			
P_4	3	3	2	1	0	1			

Answer) non- of process can

4. (5 pt.) To solve deadlock problem, we can attack one of necessary deadlock condition. Mr. Computer provide following solutions. What necessary deadlock condition he attack and what are problem with his solution?
   Solution 1)

   - Allows a process to request resources only when the process has none
   - To get a new resource, first, release all the resources currently holds and request all at time same time
     Sol)
     Attacking hold and wait,
     starvation

   Solution 2)
   - A process can hold only one resource.
   - If a process need second resource, the process need release the first one.
     Sol)
     Attacking circular wait,
     If a process need two resource at a same time, this solution have problem

5. (10 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

   Sol) 7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1

   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

7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1

7	7	7	2		2		4	4	4	0			1		1		1		
	0	0	0		0		0	0	3	3			3		0		0		
		1	1		3		3	2	2	2			2		2		7		

12 page fault

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

7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1

7	7	7	2		2		2			2			2				7		
	0	0	0		0		4			0			0				0		
		1	1		3		3			3			1				1		

9 page faults

6. (10 pt.) A system use the paging for managing virtual memory. The system has four page frames. The time of loading, time of last access, and the reference bit **R**, and modified bit **M** for each page are as shown below (the times are in clock ticks):

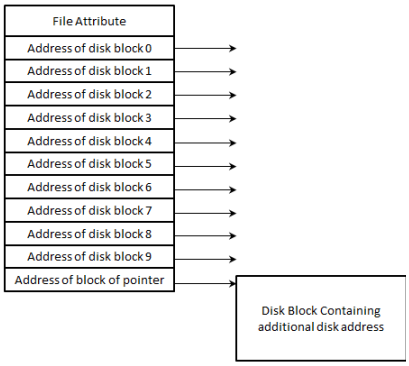
Page	Loaded	Last referenced	R	M
0	220	265	0	0
1	245	255	0	1
2	115	270	1	0
3	126	280	1	1

a) Which page will FIFO (First In First Out) replace? Page 2
b) Which page will NRU (Not Recently Used) replace? Page 0
c) Which page will LRU (Least Recently Used) replace? Page 1
d) Which page will Second chance replace? Page 0

NRU

- Class 0: Not referenced, not modified
- Class 1: not referenced, modified
- Class 2: referenced, not modified
- Class 3: referenced, modified

7. (10 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 10 direct addresses and these were 32 bits each. A block size is 2 KB. If a file use i-node and one extra block to save block information, what world the largest possible file size could be?



Sol) since 1 block is 2KB, and 32bit = 4 Byte per block address, it can save $2 \times 2^{10} / 4 = 2^9 = 512$ block information
Total = 512 + 10 = 522 block information.
Since a block size is 2KB, largest file will be 2KB × 522 =1044 KB

8. (10 pt.) A computer system generates 32-bit virtual addresses for processes. The system has 16 GB of RAM, and the page size is 2 KB..

a) If each entry in the page table requires 64 bits, calculate the maximum size of the page table in bytes.

- Maximum virtual address space = $2^{32} = 2^{22} \times 2^{10} = 2^{22}$ **KB**
- ∴Maximum # of pages per a process = virtual space / a page size = $2^{22}/2 = 2^{21}$ pages .
- Maximum size of page table per a process = number of page × one entry size
  = $2^{21} \times 64$ bits = $(2^{21} \times 64)/8$ Byte= $2^{21} \times 8$ byte= **16 MB**

b) How many bits are required to store the page frame number information in each entry of the page table?

- need calculate the number of page frame
  # of page frame = size of RAM / size of page
  = 16GB / 2KB = $16 \times 2^{30} / 2 \times 2^{10} = 2^{23}$ page frames
  ∴ **23 bits for page frame number.**

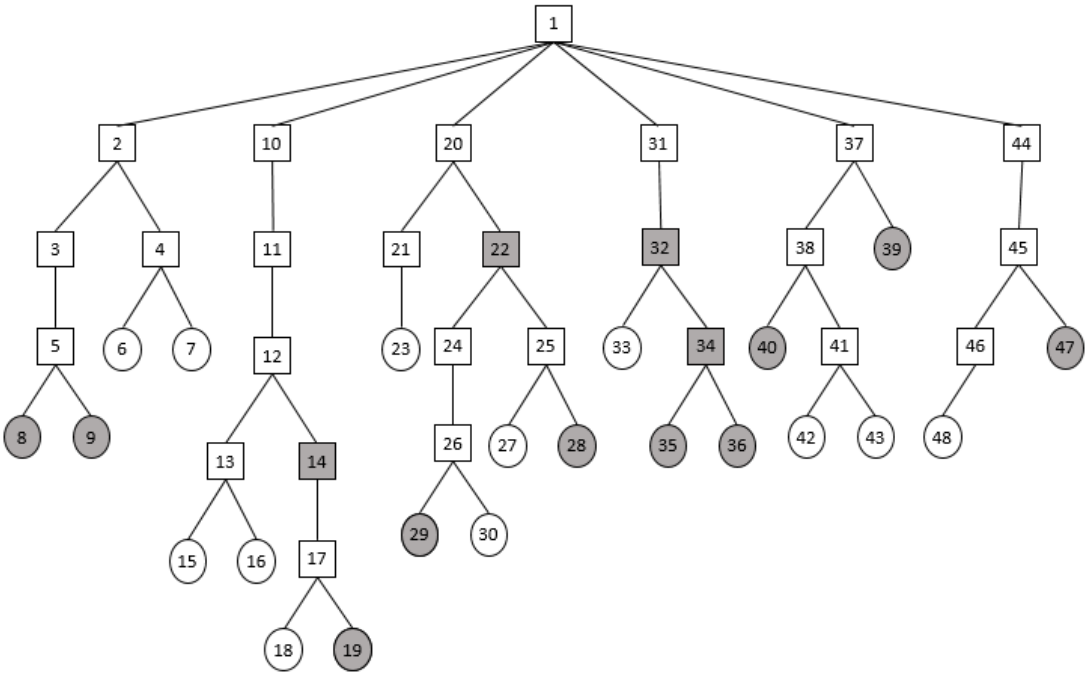
9. (10 pt.) Let's assume that a LINUX system use bitmap for maintain free disk block information. Let assume the bitmap was completely lost due to the crash. Is it possible to recover bitmap? If possible, discuss your algorithm to recover the bitmap. If not, discuss why.

   Sol) In LINUX system, size of a block and number of blocks information are saved in super block in the active partition. Since used block information for a file is saved in it's i-node, we can get used blocks information by scanning all i-nodes.

   ```
   Create new bitmap (size of bit = number of block from super block)
       Reset (set as 0)
   For each i-node do
       For each entry of i-node do
           Set bitmap to 1 (based on used block information)
   ```

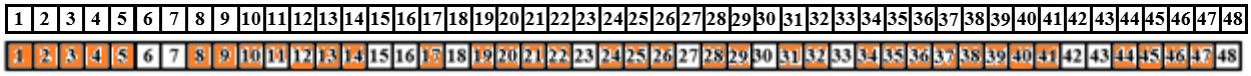
10. (10 pt.) The following figure shows a file tree with directories (squares) and files (circles). The shaded items have been modified since the base date and thus need to be dumped.

The unshaded ones have not been modified since the base date. The dump algorithm maintains a bitmap indexed by i-node number. Bits per i-node. Bits will be set and cleared in this map as the algorithm proceeds. We discussed logical dump algorithm which has four phases. Show bitmap after each phase by shading.



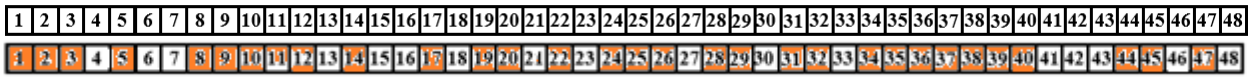
Phase 1)

1,2,3,4,5,8,9,10,11,12,13,14,17,19,20,21,22,24,25,26,28,29,31,32,34,35,36,37,38,39,40,41,44,45,46,47



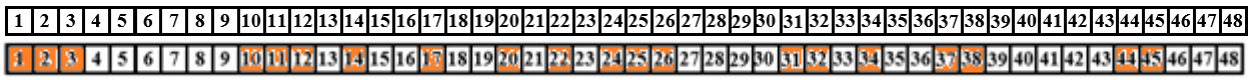
Phase 2)

1,2,3, 5,8,9,10,11,12,14,17,19,20,21,22,24,25,26,28,29,31,32,34,35,36,37,38,39,40, 44,45,47



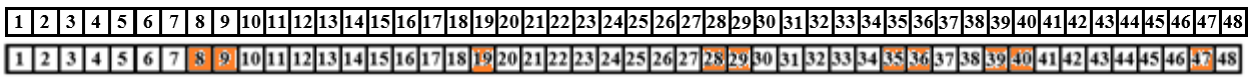
Phase 3)

1,2,3,10,11,12,14,17,20,22,24,25,26 ,31,32,34,37,38, 44,45



Phase 4)

8,9,19,28,29,35,36,39,40,47



11. (5 pt.) One way to use contiguous allocation of the disk and not suffer from holes is to compact the disk every time a file is removed. Since all files are contiguous, copying a file requires a seek and rotational delay to read the file, followed by the transfer at full speed. Writing the file back requires the same work. Assuming a seek time of 6 msec, a rotation delay of 5 msec, a transfer rate of 16 MB/sec, and an average file size of 8 KB, how long does it take to read a file into main memory and then write it back to disk at a new location? Using these numbers, how long would it take to compact half of a 32 GB disk. (1 KB = 2^{10} B, 1 GB = 2^{30} B, 1 sec = 1000 msec, 1 hour = 3600 sec)
    Sol)
    Seek time + rotation delay = 6 + 5 = 11 msec
    Average file size = 8×2^{10} Byte = 2^{13} Byte,
    Transfer rate = 16MB/sec = 16×2^{20} Byte/sec = 2^{24} Byte/sec
    A file with average size can transfer 11 + (2^{13} Byte /2^{24} Byte/sec) × 10^3 = 11.49 msec
    Read + write takes 11.49 + 11.49 = 22.98 msec

    8KB takes 22.98 msec
    16G / 8 K = 2×2^{20}, so 16GB space take 22.98×2^{21} msec = 48,192,552.96 msec = 48,192.55296 sec = 13.387 hour