**Assignment 3**

**CSC 4420 Computer Operating System**

**Chapter 3 and Chapter 4**

**Due: Mar. 24, 2025**

**50 Points**

1. **In this problem, you are to compare the storage needed to keep track of free memory using a bitmap versus using a linked list. The 8‐GB memory is allocated in units of n bytes. For the linked list, assume that memory consists of an alternating sequence of segments and holes, each 1MB. Also assume that each node in the linked list needs a 32‐bit memory address, a 16‐bit length, and a 16‐bit next-node field. How many bytes of storage is required for each method? Which one is better? (2.5 points)  
     
   - Memory Allocation Size (n bytes)  
   - Total Memory Size – 8GB (8,000,000,000 bytes )  
   - Linked List:  
    i.nodes contain:  
    32-bit memory address (start of segment)  
    16-bit length (size of segment)  
    16-bit next-node field (points to the next node in the list)  
    ii. Segments (used memory) – size = 1MB  
    iii. Holes (free memory) – size = 1MB  
     
   i.Calculating the number of bits for the bitmap:  
    - # of memory units = memory size / memory allocation size = (8,000,000,000)/n  
   - # of bits in bitmap will be = 8,000,000,000/n  
   - converting bits into bytes since (1 byte = 8 bit) = (8,000,000,000/n)/8 = 8,000,000,000/8n = 1,000,000,000/n -> (number of bytes for the bitmap depending on n)  
   ii. Calculating how many nodes and bytes are required for the linked list to track the memory using 1MB segments and holes.  
   - 32-bit memory address (start of segment) = (4bytes)  
   - 16-bit length (size of segment) = (2bytes)  
   - 16-bit next-node field (points to the next node in the list) (2bytes)  
   - Node size = 4 (memory address) + 2 (length) + 2 (next pointer) = 8 bytes  
   - 1MB = 1,048,576 bytes  
   - Number of segments/holes = 8,000,000,000 bytes / 1,048,576 bytes = 7,629.394 ≈ 7,630 segments/holes  
   - Number of segment nodes = 7,630/2 = 3,815  
   - Number of hole nodes = 7,630/2 = 3,815**

**- Total storage for linked list = 7,630 \* 8 bytes per node = 61,040 bytes  
  
iii. Comparing bitmap and linked list**

**- The storage required for the bitmap is 1,000,000,000 / n bytes, where n is the size of each memory unit. The bitmap method may require much more storage, more so when the memory unit is small (if n = 1 byte, storage will be 1GB)  
- The storage required for the linked list is 61,040 bytes. Linked list method requires a fixed storage size of 61,040 bytes, regardless of the memory unit size  
- In conclusion bitmap method can be better in terms of storage if n is large. But if n is small, the linked list method is better since it uses a fixed storage.**

1. **What is the difference between a physical address and a virtual address? Explain the difference between internal fragmentation and external fragmentation. Which one occurs in paging systems? Which one occurs in systems using pure segmentation? (5 points)  
     
   i. Virtual Address:  
    - Used by programs to access memory locations  
    - It is generated by the CPU and refers to a location in virtual memory.  
    - Virtual addresses are translated into physical addresses via address translation (mechanism), that is managed by the memory management unit (MMU)  
    - Allows for isolation between processes, so each process thinks it has access to the entire memory space.  
   ii. Physical Address:  
    - It is an actual location in the system’s physical memory (RAM)  
    - Is the address at which data is actually stored on the memory chips  
    - Handled by the hardware, and the OS’s memory manager translates from virtual to physical addresses.  
   iii. Internal Fragmentation  
    - Occurs when memory is allocated in fixed sized blocks/partitions and a process/program does not use the entire allocated memory.  
    - If memory is allocated in 4KB but program needs 3KB, the 1KB left in block is wasted.  
    - The wasted space is internal to allocated memory block 🡪 Internal Fragmentation  
   iv. External Fragmentation  
    - Occurs when free memory is split into small scattered blocks over time. Makes it difficult to allocate large contiguous blocks of memory to a program.  
    - The enough total free memory, might not be in a single contiguous block, causing problems in allocating large objects.  
    - When memory is allocated and freed dynamically in varying sizes, (common in systems using variable-sized memory allocation) 🡪 External Fragmentation.  
   v. Which Fragmentation Occurs in Paging Systems?  
    - Internal Fragmentation occurs in paging systems because memory is allocated in fixed-size pages, and a process may not always use the entire page, leaving unused space within each page.  
   vi. Which Fragmentation Occurs in Systems Using Pure Segmentation?  
    - External Fragmentation occurs in pure segmentation because segments are allocated based on the size of the data they store, and over time, memory may become fragmented into small, non-contiguous free spaces that cannot be used to allocate large segments.**
2. **Suppose that a machine has 48‐bit virtual addresses and 32‐bit physical addresses**

**a). If pages are 4KB, how many entries are in the page table if it has only a single level? Explain.**

**b). Suppose this same system has a TLB (Translation Lookaside Buffer) with 32 entries. Furthermore, suppose that a program contains instructions that fit into one page and it sequentially reads long integer elements from an array that spans thousands of pages. How effective will the TLB be for this case? (5 points)  
  
- Virtual Address: 48 bits**

**- Physical Address: 32 bits  
- Page size : 4KB 🡪 4 kilobytes = 4096 bytes = 2^12 (12 bits will be used for the page offset)  
  
i. How many entries are in the page table if it has only a single leve?  
 - # virtual pages = 2^(48 – page offset) = 2^(48 – 12) = 2^36   
 - The page table must have 2^36 entries in the single level page table for the system, because each page table entry corresponds to one virtual page.  
ii. How effective will the TLB be?  
 - TLB with 32 entries, reading sequentially long integer elements from an array that spans thousands of pages.  
 - TLB will be ineffective because the program is accessing a large array that spans thousands of pages. It will become insufficient for caching the many pages being accessed  
- Most of the accesses will result int TLB misses, leading to slower performance due to the need to perform frequent page table lookups.**

1. **A computer with an 8-KB page, a main memory, and a virtual address space uses an inverted page table to implement its virtual memory. How big should the hash table be to ensure a mean hash chain length of less than 1? Assume that the hash table size is a power of two. (2.5 points)  
   - Page Size: 8 KB = 2^13 bytes  
   - # virtual pages : (assuming 32-bit virtual address) = 2^32 bytes/ 2^13 bytes per page = 2^19 pages  
   - # entries in inverted page table = # physical memory frames (which depends on size of physical memory) N  
   - Hash Table Size (H) > # virtual pages & chain length < 1 🡪 H = 2^(log\_2(#virtual pages) = 2^20  
   - Hash Table Size = 2^20 entries (1,048,576 entries)  
   - Mean hash chain length < 1**
2. **Write a program that simulates a paging system using the aging algorithm in Fig. 3-17. The number of page frames is 4 and the number of pages is 8. The sequence of page references is (4, 1, 5, 6, 2, 1, 2, 7, 6, 3, 2, 1). Output a page table for each referred page with page miss or hit, replaced page number for miss and clock tick as well as total number of hits and misses. Partial outputs are shown below (Please submit your .c file 10 points)**

**A screenshot of a computer screen

AI-generated content may be incorrect.**

1. **Explain contiguous allocation and block-based strategies in file systems. Some digital consumer devices need to store data, for example as files. Name a modern device that requires file storage and for which contiguous allocation would be a fine idea (5 points)  
     
   i. Contiguous allocation  
    - Each file is stored as a single contiguous block of memory. All the blocks used by a file are adjacent to each other in memory or on the storage device.  
    - There is minimal seeking or jumping around on the disk leading to faster read and write operations.  
    - Easy to implement and track where the file starts and ends.  
    - Over time as files are added, deleted, and resized, fragmentations can occur between allocated blocks.  
   ii. Block based allocation  
    - Files are divided into fixed-size blocks or chunks. The blocks can be scattered across the storage medium.  
    - The file system maintains a block map or a linked list of the blocks that comprise a file.  
    - Files do not need contiguous free space to be allocated. Reducing fragmentation.  
    - Efficient use of disk space  
    - Accessing a file would require more seeks and disk head movements. (slow)  
    - File system must keep track of all the scattered blocks (overhead and retrieval.)  
   iii. Modern device requiring file storage w/ contiguous allocation  
    - SD card (digital camera)  
    - Works well because of sequential file writing and infrequent file modification or deletion.**
2. **The beginning of a free-space bitmap looks like this after the disk partition is first formatted: 1000 0000 0000 0000 (the first block is used by the root directory). The system always searches for free blocks starting at the lowest-numbered block, so after writing file *A*, which uses six blocks, the bitmap looks like this: 1111 1110 0000 0000. Show the bitmap after each of the following additional actions: (5 points)** 
   1. File *B* is written, using five blocks.  
      1111 1111 1111 0000
   2. File *A* is deleted.  
      1000 0001 1111 0000
   3. File *C* is written, using eight blocks.  
      1111 1111 1111 1100
   4. File *B* is deleted.  
      1111 1110 0000 1100
3. **Give the definition of incremental logical backup in file systems. Suppose that file 21 in Fig. 4-27 was not modified since the last dump. In what way would the four bitmaps of Fig. 4-28 be different? (5 points)  
   i. Incremental logical backup:  
    - Backup in which only the files and directories that have changed or been created since the last backup are saved. This approach reduces the amount of data stored and speed up the backup process compared to a full backup, which copies all files regardless of whether they have changed.  
   ii. Fig 4.28  
    a) The first bitmap is not marking all the files that have not been changed.  
    b) Then it is unmarking the unchanged directories with those unchanged files (if the there are files that have been changed in that directory then that directory is not marked.)  
    c) Then it is unmarking all the changed files  
    d) Lastly, it is unmarking all the unchanged directories with changed directories, (unmarking unchanged directories alone as well). Then marking back up all the changed files.**

|  |  |
| --- | --- |
| **A diagram of a tree  AI-generated content may be incorrect.** |  |
| **Fig. 4.27** | **Fig. 4.28** |

1. **Explain what writethrough cache and block cache are. For an external USB hard drive attached to a computer, which is more suitable: a writethrough cache or a block cache? (5 points)   
   i. writethrough cache  
    - Caching mechanism where every write to the cache is immediately written to the underlying storage or memory. Ensuring the data in the cache and storage are always consistent.  
    - When W occurs, data is written to both cache and main memory/storage at the same time  
    - No risk of data loss in the event of power failure or crash  
    - W operations tend to be slower.  
   ii. Block cache  
    - Temporarily stores data in the cache before it is written to the underlying storage.   
    - W are initially done to the cache, and the data is written to the disk or storage in the background or at a later time (when cache is flushed  
    - When W occurs, data is written to the cache and not immediately to the underlying storage, then eventually written to the disk (cache flushing)   
    - Performs W much faster than writethrough  
    - If system crashes or loses power before the data in the cache is written to the storage, there could be a risk of data loss.  
   ii. Which is more suitable for an external USB hard drive?  
    - Block cache because it improves write performance by reducing the number of operation that need to be written to disk immediately.**
2. **Discuss the design issues involved in selecting the appropriate block size for a file system. Explain why a file reading requires more block access in i-node based file system compared to a FAT file system. (5 points)  
   i. Choosing the right block size in a file system requires balancing storage efficiency, performance, and metadata overhead.  
   ii. I-node file systems require more block access compared to FAT file systems because they store metadata separately and use indirect block pointers for large files. However, they provide better random access performance.  
   iii. FAT file systems are more efficient in terms of sequential access but become inefficient for large files due to the need to traverse a linked list of blocks.**