University of Newcastle

School of Electrical Engineering and Computing

**COMP2240 - Operating Systems**

**Workshop 09 - Solution**

**Topics: File Management**

|  |  |
| --- | --- |
| **1.** | In a hashed file organisation, the division method is used to compute the hash address of a record. This method can be stated as follows:  Choose a large prime number *m* which is close to the number of keys *n*. Define the hash function ***h****(k) = k (****mod*** *m) +c*, where *c* is the lower limit of address.  If a set of records needs to be stored in 100 locations, starting from the address 7865, compute the address for the records having IDs 1234, 2345, 3333, and 4433.  **Answer**:  Since there are 100 locations, a suitable value of *m* will be 97.  The value of *c* = 7865  Hence,  ***h****(1234) = 1234(****mod*** *97) + 7865 = 7935*  ***h****(2345) = 2345(****mod*** *97) + 7865 = 7882*  ***h****(3333) = 3333(****mod*** *97) + 7865 = 7900*  ***h****(4433) = 4433(****mod*** *97) + 7865 = 7933* |
| **2.** | For a B-Tree (degree of the B-Tree is *d=3*) in Figure 1, show the result of inserting the key 97.    Figure 1: An Instance of B-Tree  **Answer:**  Since the Maximum *keys* that a *node* can have is 2d-1=2\*3-1=5, which is reached,  and the maximum *children* that a *node* can have is 2d=2\*3=6 also reached, we need to split the tree to add the *key=97*. |
|  |  |
| **3.** | 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.   1. Assuming a seek time of 5 msec, a rotational delay of 4 msec, a transfer rate of 8 MB/sec, and an average file size of 8KB, how long does it take to read a file into main memory and then write it back to the disk at a new location? 2. Using these numbers, how long would it take to compact half of a 16-GB disk?   **Answer:**   1. It takes 9msec to start the transfer (due to 5msec seek and 4msec rotation delay).   To read 213 bytes (8KB) at the transfer rate of 223 bytes/sec (8MB/sec) requires 2−10 sec (0.977msec).  Hence the total time to seek, rotate and transfer is 9.977msec. Writing back takes another 9.977msec.  Thus copying an average file takes 19.954msec.   1. To compact half of a 16GB disk would involve copying 8GB of storage, which is 220 files. At 19.954 msec per file, this takes 20,923 seconds, which is 5.8 hours.   Clearly, compacting the disk after every file removal is not a great idea. |
| **4.** | A sequential file is stored in a disk occupying 100 contiguous disk blocks. The disk has an average rotational delay of 2.5 ms. The time taken to seek the head of the drive to the required cylinder is 25 ms and the time taken to read a block is 0.25 ms. Find the minimum, maximum, and average time to search for a record using a linear search process.  **Answer:**  Number of contiguous disk blocks = 100  Average rotational delay of disk = 2.5 ms  Seek time = 25 ms  Time to read a block = 0.25 ms  Time to access the first block = Rotational delay + Seek time  = 2.5 + 25 = 27.5 ms  If the first record is the record to be searched, then the time of access is same as above. This corresponds to the minimum time for the search.  Time to read 100 blocks = Time to read one block × Number of blocks  = 0.25 × 100 = 25 ms  Time of searching  if the last record is to be searched = Time to access first block + Time to read 100 blocks  = 27.5 + 25 = 52.5 ms  This corresponds to the maximum time for the search.  Average time = (Minimum time + Maximum time)/2  = (27.5 + 52.5)/2 = 40 ms |
| **5.** | A sequential file has 10 million records. How does efficiency in access improve by using a two-level index? Assume 100 entries in a higher-level index and 10,000 entries in a lower-level index.  **Answer:**  Number of records = 10,000,000  Using sequential access, average search length  = ½ × Total number of records = ½ × 10,000,000  = 5,000,000  Using 2-level indexes:  High-level index has 100 entries.  Average search length for high-level index = ½ × 100 = 50  Low-level index has 10,000 entries.  Number of low-level entries for each high-level entry = 10000/100 = 100  Average search length for entry point to low-level index = ½ × 100 = 50  Number of records in file is 10,000,000.  Number of records for each low-level entry =10,000,000/10,000 = 1000  Average search length for entry point to file = ½ × 1,000 = 500  Thus, average search length  = Average search length for high-level index  + Average search length for entry point to low-level index  + Average search length for entry point to file  = 50 + 50 + 500 = 600  The above calculations reveal that the average search length has reduced from 5 million records in sequential access to 600 records when two levels of indexing is used; hence, a much greater efficiency in access is obtained. |
|  |  |

**Supplementary problems:**

|  |  |
| --- | --- |
| **S1.** | An alternative algorithm for insertion into a B-tree is the following: As the insertion algorithm travels down the tree, each full node that is encountered is immediately split, even though it may turn out that the split was unnecessary.   1. What is the advantage of this technique? 2. What are the disadvantages?   **Answer:**   1. This technique allows us to insert a key into a B-tree in a single pass down the tree from the root to a leaf. This saves time in the current operation. 2. The principal disadvantage is that the tree is subdivided and, potentially, new layers are added to the tree unnecessarily. This may slow down future searches. |
| **S2.** | Consider the organization of a UNIX file as represented by the inode shown in Figure 3.  Assume that there are 12 direct block pointers, and a singly, doubly, and triply indirect pointer in each inode. Further, assume that the system block size and the disk sector size are both 8K. If the disk block pointer is 32 bits, with 8 bits to identify the physical disk and 24 bits to identify the physical block, then   1. What is the maximum file size supported by this system? 2. What is the maximum file system partition supported by this system? 3. Assuming no information other than that the file inode is already in main memory, how many disk accesses are required to access the byte in position 13,423,956?     Figure 2: Structure of FreeBSD Inode and File  **Answer:**   1. Find the number of disk block pointers that fit in one block by dividing the block size by the pointer size:   8K/4 = 2K pointers per block  The maximum file size supported by the inode is thus:  12 + 2K + (2K × 2K) + (2K × 2K × 2K)  Direct Indirect – 1 Indirect – 2 Indirect – 3  12 + 2K + 4M + 8G blocks  Which, when multiplied by the block size (8K), is  96KB + 16MB + 32GB + 64TB  Which is HUGE.   1. There are 24 bits for identifying blocks within a partition, so that leads to:   224 × 8K = 16M × 8K = 128 GB   1. Using the information from (a), we see that the direct blocks only cover the first 96KB, while the first indirect block covers the next 16MB. The requested file position is 13M and change, which clearly falls within the range of the first indirect block.   There will thus be two disk accesses. One for the first indirect block, and one for the block containing the required data. |
| **S3.** | A i-node in Linux system is 128 bytes long and describes exactly one file. An i-node contains accounting information (including all the information returned by stat, which simply takes it from the i-node), as well as enough information to locate all the disk blocks that hold the file’s data  A Linux i-node has 12 disk addresses for data blocks, as well as the addresses of single, double, and triple indirect blocks. If each of these holds 256 disk addresses, what is the size of the largest file that can be handled, assuming that a disk block is 1 KB?  **Answer:**  The i-node holds 12 addresses. The single indirect block holds 256. The double indirect block leads to 65,536, and the triple indirect leads to 16,777,216, for a total of 16,843,018 blocks. This limits the maximum file size to 12 + 256+ 65,536 + 16,777,218 blocks, which is about 16 gigabytes. |
| **S4.** | After a system crash and reboot, a recovery program is usually run. Suppose this program discovers that the link count in a disk i-node is 2, but only one directory entry references the inode. Can it fix the problem, and if so, how?  **Answer:**  All it has to do is set the link count to 1, since only one directory entry references the inode. |
| **S5.** | Both the search and the insertion time for a B-tree are a function of the height of the tree. We would like to develop a measure of the worst-case search or insertion time. Consider a B-tree of degree d that contains a total of n keys. Develop an inequality that shows an upper bound on the height *h* of the tree as a function of *d* and *n*.  **Answer:**  Proof: The root of a B-tree T contains at least two pointers, and all other nodes contain at least *d* pointers. Thus T, whose height is *h*, has at least 2 nodes at depth 1, at least *2d* nodes at depth 2, at least nodes at depth 3, and so on, until at depth h it has at least nodes. Thus, the number of keys *n* satisfies: |