Bryan Arnold

CSE 4300

Homework 4

5/2/18

1.1) The second chance algorithm approximates the LRU paging algorithm to speed it up. The key parts of this second chance algorithm are the inclusion of a reference bit associated to each page of the LRU stack of most recent page uses. Each page is associated with a default 0 bit, and when referenced that bit becomes 1. The second chance algorithm then takes a first in first out design along with the reference bit. It is run by a clock and checks each page referenced in the LRU stack. If the reference bit for a page checked is 0, replace it. If the bit is 1, set the reference bit to 0, leave it in memory, and go on to the next page. The enhanced second chance algorithm takes a similar approach to this, but adds another bit associated with each page in an ordered pair as (reference bit, modify bit). Now, as the clock makes the algorithm check each page, now the ordered pair of bits is checked. If (0, 0), the page hasn't been modified or used recently, so it is the best page to replace. If (1, 0), the page has been used recently but has been cleaned out, so it will probably be used again soon making it the third best page to replace. If (0, 1), the page hasn't been used recently but has been modified, so it is the second-best page to replace as write out must be done first. Lastly, (1, 1) is a page that has recently been used and modified, which means it will probably be used again soon as well as needing write out before replacement, making it the worst page to replace. When replacement is called for, use the clock to do so, but the four cases must be considered to always prioritize (0, 0) to be replaced. This may take several searches of the stack/queue to fulfill.

1.2) Thrashing occurs when a process doesn't have the minimum required pages, so the number of page-faults it undergoes is very high. There are page faults to acquire pages, replace frames, as well as replacing the new frames back. This frantic alteration of page-faults and frame replacements is what thrashing is, a process rapidly and busily swapping pages in and out. When this occurs, the CPU utilization becomes low and causes the OS to think it needs to increase its multiprogramming, leading to additional process creation. This can drastically decrease performance. This is exactly how the OS detects and eliminates thrashing. First, when the evaluation of the CPU utilization is compared to the level of multiprogramming required, the system can detect thrashing occurring since the page swapping is causing high multiprogramming needs. A way for the system to help eliminate thrashing is to reduce this multiprogramming level, by adding more processes to aid in the process that is thrashing. This allows for more CPU to be used and lower multiprogramming needs. The page swapping will be more efficient as it now has the number of desired pages to operate.

2) A single-level directory organization is a single directory for all the users. All the files have been placed into a single directory. The advantages are it is very easy to implement

and understand how it functions and is structured. All files are mapped in one directory, simple as that. The disadvantages are the fact that there are large amounts of files to be put into the directory, presenting a limitation on memory of the directory. The other issue is when multiple users utilize the directory, there is no way to group what files belong to what users or which files should be grouped together in categories, as well as the vast number of different files they could have allow no duplicate file names.

In a two-level directory organization, there are two different directories. There is a master file directory, which is separated into each user. All the users in the master directory then have their own separate file directories for their individual files. The advantages include there is now a path name to get to specific files in a user's directory, allowing duplicate files names for different users, as well as efficient file searching in each individual directory. The disadvantages are that there is still no way in which to group files together into categories or groups as well as file access sharing.

In a tree structured directory organization, the user can create a tree structure of arbitrary height for their files using subdirectories they can create with their files. The advantages of this organization allow for efficient searching for files as well as the ability to group together similar files as the user wants. A user can create files in a directory, delete them, create directories, delete them, all by utilizing working directory commands with absolute and relative path names. There are no real clear disadvantages to this than just being more difficult to implement than other organizations.

An acyclic graph directory organization has the same structure as a tree structure, but it allows different directories to share the same file, as well as directories to share directories with other directories. Essentially, a tree structure with free sharing amongst directories of files and directories. This sharing makes it so any modifications to a file or directory are immediately seen by anything sharing them. Files are shared using a link, or an absolute or relative path name to files or directories. When a reference to file or directory is made, the directory is search for if the directory is marked as a link. The name of the real file is then included in the link information to anything sharing it. The advantages to this are just the same as a tree structure, but it also allows for sharing of files or directories between multiple different directories and allows for multiple avenues for accessing them. The disadvantages to this is that only one copy of a file or directory can be stored, so any changes to that alter what is seen by anything that shares them. This also doesn't allow for files of the same name or directories of the same name to be made in different paths of the tree structure. Deleting a directory causes a dangling pointer, but this I solved through backpointers to delete all pointers.

A general graph directory organization is the same as an acyclic graph structure but allows for cycles with its pointer utilization. A directory being referred to can now point to the directory that is pointing at it, which is a cycle. The advantages to this is that you can refer to a directory higher and its contents a lower directory may need. The disadvantages are

that if a cycle exists, access of contents of this cycle can go on infinitely, causing an infinite loop which is never good. There is no real way to avoid traversing each component in the cycle while searching it, which can also take time. Also, whenever a new link is added, a cycle must be detected to determine if the cycle is allowed. Lastly, there is no way to release garbage collection of deleted or allocated data.

3a-f)

Block placement (Down) / Allocation strategy (Right)	Contiguous Allocation	Linked Allocation	Indexed Allocation
Block added at the beginning (a)	201	1	1
Block added in the middle (b)	101	52	1
Block added at the end (c)	1 (or roughly 101)	3	1
Block removed at the beginning (d)	198	1	0
Block removed from the middle (e)	98	52	0
Block removed from the end (f)	0	100	0

4) By having different levels of allocation permissiveness, you could decrease the level of internal fragmentation of memory allocation. If a file is 6KB in size, you first allocation a 4 KB block as the system allows. Since 512-byte sized blocks are also allowed for allocation of data as well, you could have 4 of these blocks to finish the allocation of the 6KB file. So, instead of having to allocate another 4 KB block and only utilize half of it, you simply have 512-byte blocks to compensate and improve allocation maximization. This also in turn increases performance, as you do not need to allocate more data, leading to less paging, context switching, etc. Also, the bitmap of free blocks that must be managed in allocation, must also maintain the extra states which sub blocks are currently used inside a block. The allocation would examine the extra state to allocate these sub blocks and bring together these sub blocks to obtain a larger block when all the sub blocks are freed.

5a) In word at a time mode, the CPU needs to acquire the system bus to send a command to the disk controller. To do this step, the acquiring of the bus takes t1 nsec, as indicated, and sending a word command over the bus takes t3 nsec, as indicated. This is (t1 + t2) nsec of time. Now, the CPU requests ability to transfer the word, which is another (t1 + t3) nsec. Once the transfer is completed into memory, the CPU would acknowledge so, which is another (t1 + t3) nsec. Adding up this total elapsed time for each component of the task would be 3(t1 + t3) nsec. Since there are 1000 words to be transferred, simply multiply the elapsed time for one word 1000

times: 1000 * 3(t1 + t3) nsecs = 3000(t1 + t3) nsecs. Therefore, it would take (3000t1 + 3000t3) nsecs to transfer 1000 words using the word at a time mode.

b) For burst mode, each step is also required that occurred in word at a time mode, but slightly altered. First, the bus will be acquired and the command to be sent to the disk take (t1 + t3) nsecs, just like in word at a time. The disk would acquire the bus in t1 nsecs, followed by a transfer of 200 words. Each word takes t3 nsecs, so 200t3 nsecs for each burst to take place. The bus waits for acknowledgment that the transfer occurred, again taking (t1 + t3) nsecs like in word at a time mode. Since there are 1000 words, there will need to be 5 total instances of this. So, there will be 5 times the CPU acquires the bus and the command is sent to the disk (5t1 + 5t3) nsecs, as well as acknowledgment 5 times, (5t1 + 5t3). Now, 1000 words needs to be transferred, so 5(200t3) nsecs = 1000t3 nsecs. Adding these all together, we have (5t1 + 5t3) + (5t1 + 5t3) + 1000t3 nsecs = (10t1 + 1010t3) nsecs to transfer all the words. This is significantly faster than word at a time mode.