Assignment 3 - Assignmnet 3

Operating Systems (Athabasca University)

Disclaimer: No part of this assignment was copied in whole or part from the textbook. Although the ideas expressed are those acquired from an understanding of the course textbook.

What are the advantages of using dynamic loading? (6 marks)

Having the entire program and data on memory all at the same time is not a good use of system resources. This is because not all of it is needed at the same time. It is much better if only data that is needed is available on memory and the rest of it is stored in a relocated load format on disk.

Dynamic loading is useful to utilize memory space more efficiently. This is achieved by not loading all routines on memory but rather having the routines loaded from a secondary source such as a disk. When the routine is needed by the program, the routine is called and then relocated by the source into memory. On relocating the routine is given control by the memory.

It is important to note that dynamic loading does not require and form of special operating system support. When programs are built, the programmer has to design their program in such as way that it can take advantage of this method by utilizing the library routines provided by the operating system.

Explain the basic method for implementing paging. (8 marks)

Paging is a memory management scheme which permits the physical address of a process to be noncontiguous such as in segmentation. Paging also has the advantage of avoiding external fragmentation and the need for compaction.

The basic method of implementing pages will involve breaking up the memory into fixed-sized blocks called frames as well as breaking up the logical memory in the blocks of the same size called pages. With this arrangement in place, when the process is executed the pages of the process are loaded into any available frames in the physical memory. This approach is useful because the size of the logical address space is not bound by the size of the physical memory.

Though this setup comes with the advantages, it is crucial that each logical address maps correctly to the right physical address. So how do we achieve this? In paging a logical address space is made up of two parts. First the page number and second the page offset. With the right page number of the logical address, you can use the page table to map that page number to a correct frame number. And combining that frame number with the offset provided allows us to locate the right physical address.

Briefly, describe the segmentation memory management scheme. How does it differ from the paging memory management scheme in terms of the user's view of memory? (8 marks)

Segmentation is a memory management scheme that allows memory to be allocated in such a way that it mimics this user's view of memory. With segmentation, the logical address space is made up of segments with each of these segments having its number("base") and a length. The length of the logical address can be referred to as the "limit". When mapping from the logical address to the physical address, we check to make sure the offset provided falls within the segment's base address plus the segments limit. If this is not the case, we know the values provided are wrong and the process is trying to illegally access a part of memory which it is not permitted to access.

The main difference between the segmentation and paging management scheme in terms of user's view of memory is that with segmentation, when the user is trying to access a memory address, they provide the segment number and the offset and the pair is used to locate the physical address.

One the other hand, with paging when the user is trying to access a memory address, they need to provide only one address which is then translated to a page number and an offset by the operating system. The whole process of translating the single address given is not known to the programmer.

Explain the distinction between a demand-paging system and a paging system with swapping. (8 marks)

The demand-paging system allows the pages of a process to be loaded from disk only when it is needed. This is particularly important because it is not all part of a program that is loaded from the disk that will be used by the user.

The concept of paging system with swapping is connected with swapping out an entire process from memory into a backing-store (secondary storage such as a disk) and swapping in an entire process as well from the backing store into the physical memory.

The two concepts above seem familiar, but the difference lies in the fact that with demand paging, not the ENTIRE process is "swapped" in and out. Only pages of a process that will be needed by the user are swapped into the memory. On the other hand with "paging system with swapping" the entire process is swapped into memory regardless of if all parts will be used.

As an important side note, its is technically incorrect to use the word "swapping" about demand paging. The word "pager" is preferable because while swapping deals with moving the entire process, a pager is most interested in moving just specific pages of a process that will be needed.

How does the second-chance algorithm for page replacement differ from the FIFO page replacement algorithm? (8 marks)

Page replacement algorithms are used to find a "victim" frame that needs to be taken away from the main memory to give room for a new page that is needed by another process.

FIFO page replacement algorithm is the simplest algorithm. As the name may suggest the first frame that was brought into memory is the best choice for the “victim” page when a page needs to be taken away to make room for an incoming page. However, it is somewhat obvious that this may not be the best replacement algorithm because if the first page is still actively in use by its process, then we will have to bring back the page almost immediately after replacing it. This will significantly increase the fault rate.

The second-chance algorithm could be said to be born out of the need to find an algorithm which is much closer to being optimal than the FIFO. At the heart of the second chance algorithm is the FIFO algorithm. However, the second chance algorithm makes use of a reference bit to be able to know if a page has been recently used. If the page has been used(i-e reference bit is set to 1), the second chance algorithm resets it reference bit to zero and ”saves” it. It goes ahead and checks the next page using the same criteria. However, this time if the page has not been used it is the most like choice to be a “victim” page, and it is then selected.

The second-chance algorithm tries as much as possible to remove the least recently used frame though it is not 100% efficient at this. Achieving the Least recently used algorithm will require hardware which not all computers offer.

Explain how copy-on-write operates. (8 marks)
When a new child process is created from a parent process, it is unnecessary that all the pages of the parent process need to be duplicated. Copy on write helps us to minimize the number of pages that are created. This technique allows the child process to be able to share pages of its parent's process and if a child process will like to make changes to any page, it has to create a copy of that page for it’s use rather than making changes to the page that belongs to the parent process. With this in place, the child process is not given a duplicate of every page that may or may not be used by the child process. This technique is used by some operating systems such as Windows XP, Solaris, etc.. With this, process creation is much faster and efficient.

It is important to note that the pages that are shareable by a child process always have two characteristics. The page needs to be able to be modified because pages that cannot be modified cannot be shared between the parent and child process. And secondly, they need to be explicitly marked as “copy on write pages.”

If you were creating an operating system to handle files, what are the six basic file operations that you should implement? (8 marks)
An operating system that hopes to handle files should be able to do the following

Create a file
Write a file
Read a file
Deleted the file
Truncate the file

Reposition the file

Though this can be seen as the basic operation, it is possible to combine these operations to achieve more operations. E.g copying a file can be achieved by combining both the create a file, read and write operation.

Creating a file requires both finding space in the file system for the file and also making an entry into the file directory using the name and attribute of the file. Writing a file requires both the name of the file and the data that is meant to be written to the file. With the name of the file, the operating system can locate the file in the file system via the help of the file directory. However, after writing has been done, the write pointer is used to keep track of where the next write is to start from.

Reading a file requires knowing the name of the file and the part of the memory that the next block of the file should be used. Just as in the case of writing, a read pointer is used as well to know the next read start point.

It is important to note that since processes usually just read and write from the same file, for the sake of efficiency we can use one pointer to make the next start position, rather than having one read pointer and a different write pointer. This one general pointer can be called the “current-file position pointer”.

If we ever need to change the position of the current-file position pointer we can reposition it just by calling on the name of the file and the value it is to be repositioned to.

Deleting a file will require just the name of the file.When the file is found within the file structure, the space of memory it occupies can be released for use by other files. Also, note that we have to remove the file name from the file directory.

Truncating a file is useful in cases where we will just like to delete the content of the file without having to delete the file itself.

To create a new file, an application program calls on the logical file system. Describe the steps the logical file system takes to create a file. (8 marks)

When creating a new file, the logical file system allocates a new File control block(FCB). The right file directory is then updated with the new file name and the FCB with has been allocated to the file. After this update is done, the file directory is written back to the disk.

It is important to note that depending on the file system, FCB’s can be all created initially at the file-system creation time and the free ones are then allocated when they are needed.

In general, to create a file, we will need the name of the file. Space needs to be found in the current file structure for the new file when it is created. On creating the fie, the file name and attributes also need to be added to the file directory.

How is a hash table superior to a simple linear list structure? What issue must be handled by hashtable implementation? (8 marks)

Both the linear list structure and the hash table structure are data structures used for file directories.
The simple linear list structure is easy to implement and has a natural appeal. However, it is slow during execution. This leads to significant lags which are not suitable for great user experience. The linear list structure makes use of a simple list of the file name with pointers to the data block. Adding a new file, deleting a file, updating a file all requires a search to be implemented. This search(a linear search)is slow because of the way the data is stored. When a file is to be created we need to make sure that a file with the same file name doesn't previously exist therefore we need to search the directory. In deleting or updating a file, we need to search for the file to be deleted or updated. To help out with the search, some operating systems make use of a cache to store recently used files so they can be easily retrieved if needed again rather than re-reading them from the disk.

A hash table is a more sophisticated data structure for a file directory used together with a linear list. The hash table aims to reduce the search time by converting the file names to a "hash"(special identifier integer). Each has a pointer to a file name in the linear list.

The hash table implementation, however introduces a new problem which needs to be dealt with. This is the problem of collision. A collision is a situation that arises when two files names hash to the same location.

What are the factors influencing the selection of a disk scheduling algorithm? (8 marks)

Disk scheduling algorithms all have their pros and cons. Selecting a disk scheduling algorithm largely depends on two things

1. The number of requests
2. The type of requests.

All algorithms are essentially the same when you have just one request in the work queue because there is only one position they could go to at that point. However, with more request, the movement needs to be coordinated in such a way that it is efficient as possible.

To ensure efficiency, we need to consider the type of the request. A request requires the disk to read data is largely dependent on how the data is stored. If the data is stored contiguously, then it reduces the amount of disk head movements. When the data to be read is not stored contiguously, the disk head will have to perform multiple disk head movements.

For efficiency, an operating system can implement multiple disk scheduling algorithms and choose an appropriate one based on the request.

Explain the disadvantage(s) of the SSTF scheduling algorithm. (8 marks)
Disk scheduling algorithms are algorithms utilized by the disk to determine which requests from the work queue should be handled next. The SSTF scheduling algorithm stands for the shortest seek time first algorithm. As the name implies, this algorithm tries to handle the request that requires the shortest movement no matter the direction. In principle, if the disk has it's read head at 70, and it receives a request to read one from a sample string, e.g., 83, 76, 72 and 100. 72 will be read first as opposed to 83. After 72 is read, 76 is read, then 83 and finally 100.

Though this algorithm seems logical on the surface, it has a huge problem of possible starvation to a request. In principle, starvation in SSTF will occur like this. If the disk has, it's read head at 70, and it receives a request to read one from a sample string, e.g., 74 and 199. Assuming that while 74 is read, a new request for 78 comes in, 78 will be read next as opposed to 199. Assume yet again that while 78 is read another request for 76 comes in, 76 is read again at the expense of 199. In a scenario where the incoming request is always close to the current position of the disk head, we will have a starvation of request 199.

When the disk head has to switch directions, this slows things down as well.

Explain the concepts of a bus and a daisy chain. Indicate how these concepts are related. (8 marks)
A computer is made up of multiple devices. These devices range from CD-ROMS to scanners to keyboards, etc... Each of these devices does not work in isolation and needs a way to communicate with other devices.

If these devices share a common set of wires, it can be referred to as a bus. Buses are common in computer systems architecture. A bus has a protocol that defines how messages are sent on this set of wires. Buses are of different kinds. Some of them include an expansion bus and PCI bus. A PCI bus typically is used for connection to fast devices, and the expansion bus is typically used for connection to slower devices

It is also possible that these devices are connected to one another through a cable, and the last device in the "series" is connected to the computer system by the mean of a port. This arrangement is referred to as a daisy chain.

The bus and daisy chain have one thing in common. They both serve as a communication channel between devices. One big advantage of a daisy-chain is scalability. New devices can be added to the "chain" of devices by adding new nodes to the chain.

What are the three reasons that buffering is performed? (6 marks)
When data is being transferred between two devices or even between a device and an application, a buffer serves as a "middleman" storage to facilitate the transfer. There are three main reasons we may need a middleman for data transfer.

1. Difference in speed of the sending device and the receiving device
2. Difference in data transfer size of the sending device and the receiving device
3. Having a buffer could help with data integrity.

If the sending device has a much lower speed than the receiving device, it is important to avoid multiple write() operations for very little data. With a buffer in place, the data sent by a much slower device can be accumulated in the buffer and when the buffer is full, the write operation is done as a batch.

The second reason for buffering is most useful in computer networking. With transfer size rarely being the same for two computer n a network, a buffer serves as a "space" where packets of data that has been sent over the network can be reassembled correctly to produce the original sequence in which the data was sent.

Data integrity is very useful when data is copied from a source to a destination. Assume that a system call is made for data to be transferred and after that, the data in the application buffer is changed. To make sure that the correct data was copied, the operating system makes use of a system kernel buffer to save a copy of the data so in cases where the source data have been edited, we are sure that the right piece of data is kept.

References

Silberschatz, A., & Silberschatz, A. (2013). Operating system concepts (pp. 357). John Wiley & Sons, Inc.

Silberschatz, A., & Silberschatz, A. (2013). Operating system concepts (pp. 366). John Wiley & Sons, Inc.

Silberschatz, A., & Silberschatz, A. (2013). Operating system concepts (pp. 364-365). John Wiley & Sons, Inc.

Silberschatz, A., & Silberschatz, A. (2013). Operating system concepts (pp. 401). John Wiley & Sons, Inc.

Silberschatz, A., & Silberschatz, A. (2013). Operating system concepts (pp. 418-419). John Wiley & Sons, Inc.

Silberschatz, A., & Silberschatz, A. (2013). Operating system concepts (pp. 408). John Wiley & Sons, Inc.

Silberschatz, A., & Silberschatz, A. (2013). Operating system concepts (pp. 504-506). John Wiley & Sons, Inc.

Silberschatz, A., & Silberschatz, A. (2013). Operating system concepts (pp. 544-547). John Wiley & Sons, Inc.

Silberschatz, A., & Silberschatz, A. (2013). Operating system concepts (pp. 552-553). John Wiley & Sons, Inc.

Silberschatz, A., & Silberschatz, A. (2013). Operating system concepts (pp. 472-478). John Wiley & Sons, Inc.

Silberschatz, A., & Silberschatz, A. (2013). Operating system concepts (pp. 474). John Wiley & Sons, Inc.

Silberschatz, A., & Silberschatz, A. (2013). Operating system concepts (pp. 588-589). John Wiley & Sons, Inc.

Silberschatz, A., & Silberschatz, A. (2013). Operating system concepts (pp. 605-606). John Wiley & Sons, Inc.