Internals - 03

OS(18CS43)

Shahul Hameed. S
1KN18CS097
CSE 'A' sec
4th Sem

(b.) If the number of frames allocated to low priority process fall below the minimum number required by the computer architecture then we suspend the process execution.

A process is thrashing if it is spending more time in paging that executing. If the process do not have enough number of frames, it will quickly page fault. During this it must replace some page that is not currently in use. Consequently it quickly faults again and again. This high paging activity is called thrashing.

Cause of thrashing:-

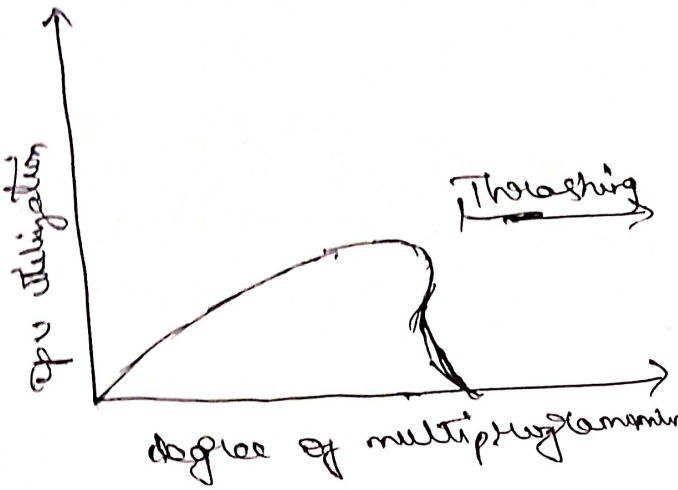
* Thrashing results in severe performance problem.

The operating system monitors the CPU utilization is low. We increase the degree of multiprogramming by introducing new process to the system.

* A global page replacement algorithm replaces pages with no regards to the process to which they belong.

→ As the degree of multiple programming increases, more slowly until a maximum is reached. If the degree of multi-programming is increased further thrashing sets in and the cpu utilization drops sharply.

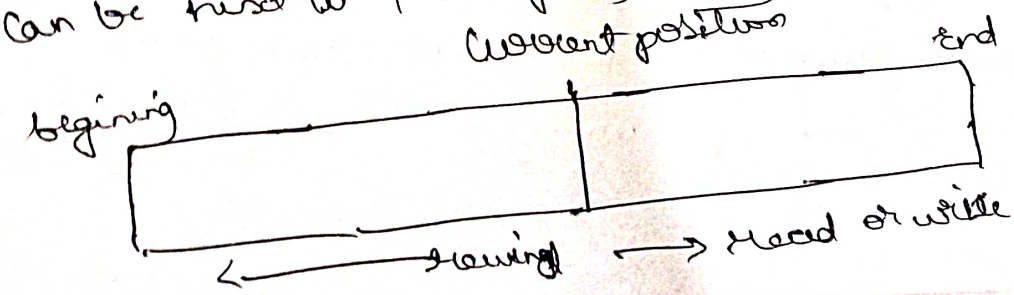
The figure shows the thrashing.



At this point, to increase CPU utilization and stop thrashing we must increase degree of multiprogramming. We can limit the effect of thrashing by using a local replacement algorithm.

To prevent thrashing we must provide a process as many frames as it needs.

(11b) Sequential access:- It is the simplest access method. Information in the file is processed in order, one record after another. Editors and compilers access the files in this fashion. Normally read and write operation are done on the files. A read operation reads the next portion of the file and automatically advances a file pointer, which tracks next I/O location. write operation appends to the end of the file and such a file can be reset to the begining.



Direct Access:

Direct access allows random access to any file block. It is based on disk model of a file. A file is made up of fixed length logical records. It allows the program to read and write records rapidly in any order. A direct access file allows arbitrary blocks to be read and/or written. Not all OS support sequential and direct access. Few OS use sequential access some OS uses direct access. It is easy to simulate access on a direct access but the reverse is extremely efficient.

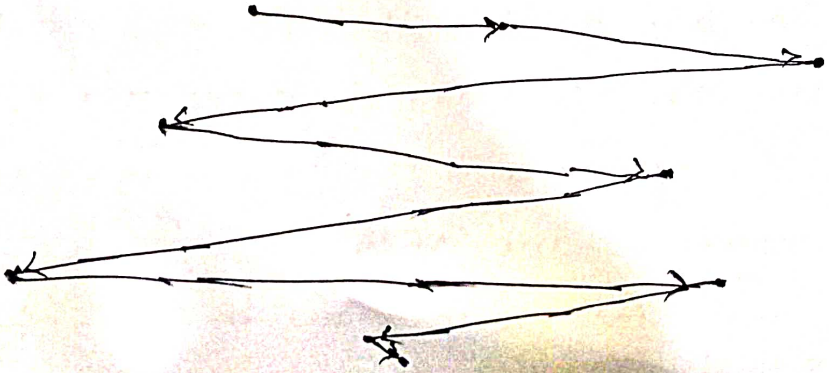
(3) ## FCFS Scheduling:

Simplest form of disk scheduling, generally does not provide fastest service.

Eg 1: A disk sequence with requests I/O blocks on cylinders 98, 183, 37, 122, 14, 124, 65, 67. and disk head is initially at cylinder 53.

It will first move to 98, then to 183, 137, 122, 14, 124, 65 & 67, as shown in figure below. total of 640 cylinders.

0 14 37 53 65 67 98 122 124 183 194

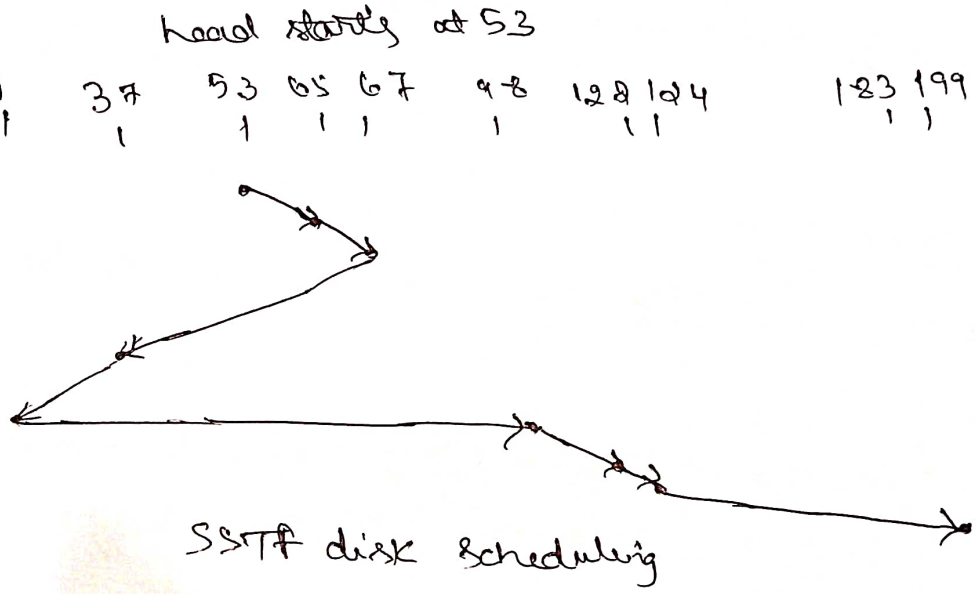


SSTF Scheduling

SSTF assumes that it is better to service all the requests close to the current head position before moving the head far away to service other requests.

Eg:- For the queue 98, 183, 37, 122, 14, 124, 65, 67 with head position = 53. Closest request to the initial head position is 65. Once we are at cylinder 65 next request served is 67, next is 37 and so on as shown in below.

Total head movements is 236.



SSTF disk scheduling

SCAN Scheduling:-

The disk arm starts at end of the disk and moves towards the other end, servicing requests as it reaches each cylinder, until it gets to other end of the disk.

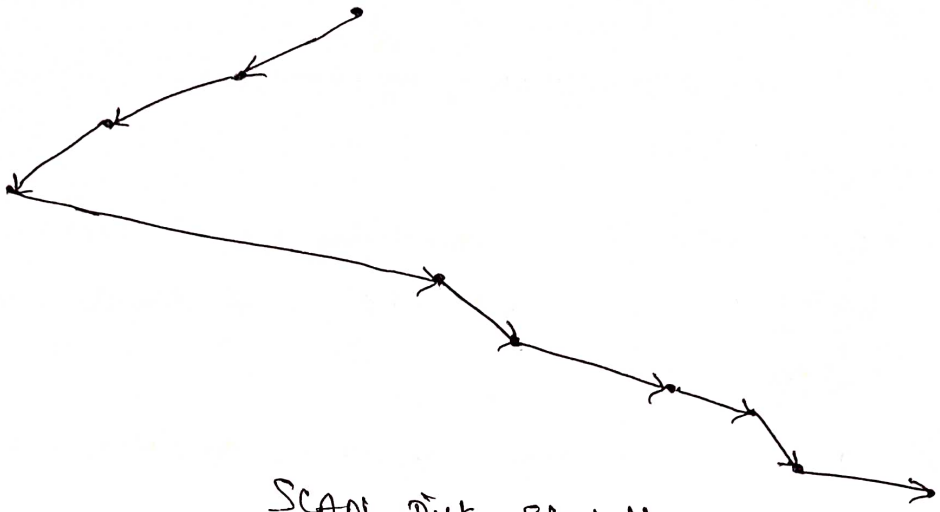
Consider, 98, 183, 37, 122, 14, 124, 65 & 67, head = 53

For this algorithm we need to know the direction of head movement. If the disk arm is moving towards 0, the head will service 37 and then 14.

At 0, the arm will reverse and move towards the other end servicing the request at 65, 67, 98, 122, 124 and 183.

Total head movement is 208 cylinders.

0 14 37 53 65 67 98 122 124 183 199



SCAN Disk Scheduling.

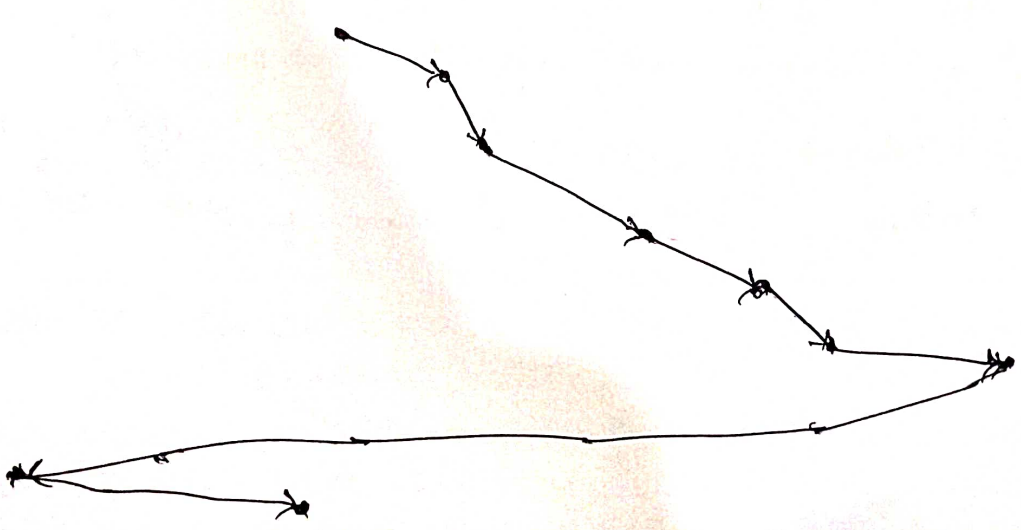
LOOK Scheduling

In this disk arm doesnot move across the full width of the disk. The arm goes only as far as the final request in each direction. Then it reverses direction immediately, without going all the way to the end of the disk.

queue = 98, 183, 37, 122, 14, 124, 65, 67

head at 53.

0 14 37 53 65 67 98 122 124 183 199



(3b) Design principle of Linux:-

* Linux resembles any other traditional, non micro kernal UNIX implementation. It is multi user, multitasking system with a full set of UNIX - compatible tools.
* Linux's file system adhere to traditional UNIX semantics.
* Linux can run smoothly on a microprocessor machine with hundreds of MB's of main memory and many GB's of disk space.
  But it is still capable of operating usefully in under 4MB RAM.
* Speed and efficiency are still important design goals, but much recent and current work on linux has concentrated on a third major design goal - standardization.
* The linux programming interface adheres to SVR4 UNIX semantics.
* A separate set of libraries is available to implement BSD semantics in places where the two behaviours differ significantly.
* Linux currently supports the POSIX threading extensions - Pthreads and a subset of the POSIX extensions for real time process control.