BLOCK II:
MEMORY AND I/O MANAGEMENT, SYSTEM DEADLOCK AND MULTIPROGRAMMING SYSTEM

Unit 1 : Memory Management
Unit 2 : Input-Output Organization
Unit 3 : Introduction to Deadlock
Unit 4 : Deadlock Prevention, Detection and Avoidance
Unit 5 : Multiprogramming System
Unit 6 : Secondary Storage Management
Unit 7 : Security
Unit 8 : Distributed Operating System

UNIT 1: MEMORY MANAGEMENT

Unit Structure:

1.1 Introduction
1.2 Unit Objectives
1.3 Hierarchy of Memory Types
1.4 Cache Memory
1.5 Associative Memory
1.6 Address Protection
1.7 Paging
    1.7.1 Paging Hardware Support
1.8 Segmentation
    1.8.1 Segmentation Hardware
1.9 Virtual memory
    1.9.1 Demand Paging
1.10 Page Replacement Algorithms
    1.10.1 FIFO Page Replacement
    1.10.2 LRU Page Replacement
    1.10.3 Optimal Page Replacement
1.11 Summing Up
1.12 Answers to Check Your Progress
1.13 Possible Questions
1.14 References & Suggested Readings

1.1 INTRODUCTION

The unit deals with management of main memory during process execution. One of the most important functions of operating system is memory management that includes the hardware support in processor for paging, virtual memory and segmentation. Virtual memory allows a program with memory space larger than the size of the main memory available in the system. This is possible by allowing only that section of the code that is active at that point of time without the need of having all instructions and data of the process being present in main memory at the same time. The concept of paging and segmentation eliminates the need of allocating main memory to the process in contiguous manner. Also if the overall memory requirement exceeds the physical memory limit,

Space for learners:

pages from memory may need to be replaced to make room for new pages. Various page replacement algorithms like FIFO, LRU and Optimal are used in such case.

Space for learners:

1.2 UNIT OBJECTIVES

After going through this unit, you will be able to:

- explain memory hierarchy, cache memory and associative memory
- explain the working of memory address protection.
- explain the paging memory management scheme.
- analyze and solve problems on paging.
- explain the working of paging hardware.
- explain the concept of segmentation and solve problems on segmentation.
- describe the benefits of Virtual memory management system
- explain and solve problems on different page replacement algorithms.

1.3 HIERARCHY OF MEMORY TYPES

The memory in a computer system can be divided into a hierarchy as shown in Figure 1.1. The hierarchy is based on access time, speed, cost and capacity of the memory. The five memory types in the hierarchy are registers at the top followed by the cache memory, main memory, hard disk and magnetic tapes. The first three memory types register, cache and main memory are volatile memories that is they lose their stored data in absence of power supply. The last two memory types, hard disk and magnetic tape keeps the stored data permanently even in the absence of power.

In the Figure 1.1, capacity that is the volume of information the memory can store increases as we move from top to bottom in the hierarchy.

Access time that is time required to perform read/write request increases as we move from top to bottom in the hierarchy. Similarly, the speed gap between CPU and memory decreases as we move from bottom to top of the hierarchy and finally cost per bit increases going from bottom to top of the hierarchy.

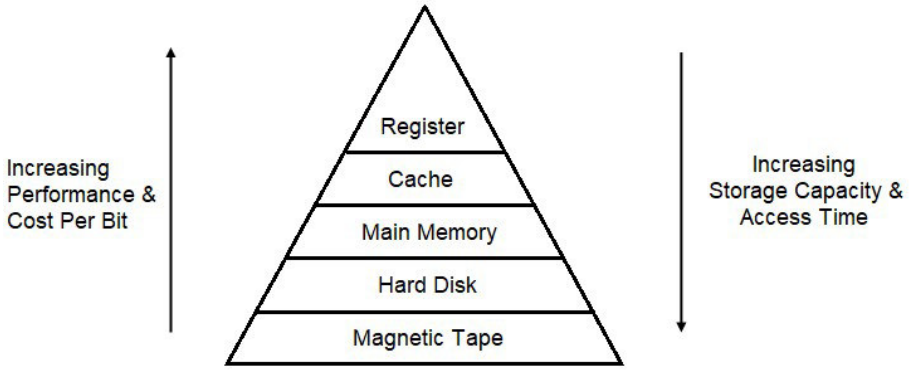


Figure 1.1: Memory Hierarchy

1.4 CACHE MEMORY

Cache Memory is a type of memory that operates at a very high speed. It's used to boost performance and synchronize with a high-speed CPU. Although cache memory is more expensive than main memory but it is less expensive than CPU registers. Cache memory acts as a buffer between the main memory and the CPU as shown in Figure 1.2. Cache memory stores frequently requested instructions and data so that they may be accessed quickly by the CPU. It smaller and faster memory that reduces the average access time of main memory by storing copies of most frequently used data.

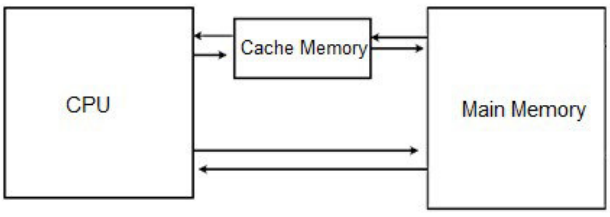


Figure 1.2: Cache Memory acting as a buffer between CPU and Main Memory.

Space for learners:

1.5 ASSOCIATIVE MEMORY

Associative memory is also known as Translation Lookaside Buffer (TLB) is a special type of memory that is optimized to perform parallel searches on data, in contrast to sequential search of data.

Operating system provides support for storing page table of a process. Generally, a page table can be stored in following ways:

- Set of dedicated registers
- In main memory
- Associative Memory or Translation lookaside buffer (TLB)

The feasibility of the first approach using a set of dedicated registers is that the page table should be reasonably smaller in size like 256 entries. With the second approach page table can be very large like millions of entries can be stored in the main memory with a pointer to the starting address of the page table for referencing. However, in this case the time required to access the page table is slower by a factor of two as it involves first accessing memory for the page table to locate the frame number which is combined with the displacement to get the physical address and then a second memory access to read the byte.

The solution to the disadvantages of the first two approaches is resolved using a fast lookup hardware support called Associative memory. Associative memory or TLB is a small, expensive but very fast associative memory. It can store entries in the range of 64 to 1024. Associative memory has two parts: a tag and a value. When a page/key needs to be searched the key is compared simultaneously with all the tags of the in the associative memory.

1.6 ADDRESS PROTECTION

In a main memory there can be several user process and operating system running at a time. To protect the address space of the operating systems as well as user processes, so that they do not run into to each other's address space, the concept of hardware address protection is introduced. Address protection is implemented with the help of two registers, the *base register* and the *limit register*. The base register holds

Space for learners:

the starting address of the process address and the limit register specifies the range. For example, in Figure 1.3, the base register holds the starting address *5500* of *Process-2* in the main memory and the limit register specifies range of *750* meaning that the range of legal address of *Process-2* is from *5500* to *6249* (inclusive).

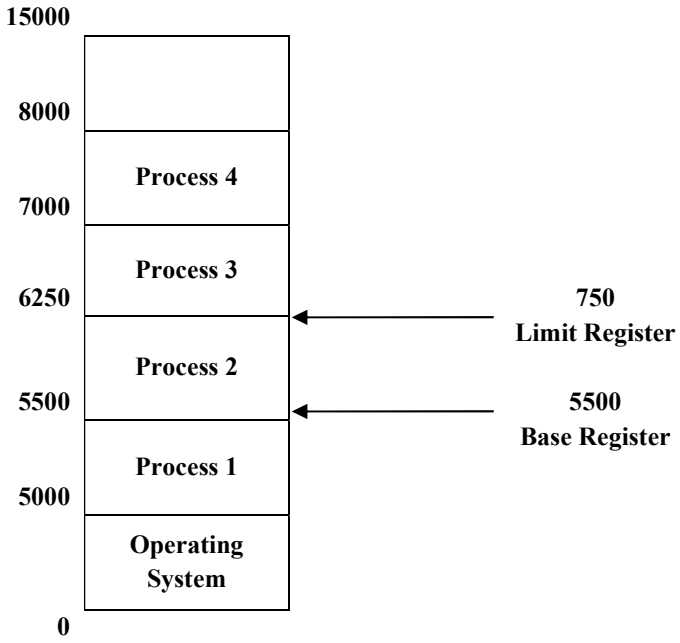


Figure 1.3: A logical address defined by base and limit register.

The Memory address protection is accomplished with the help of hardware support as shown in Figure1.4. The hardware checks that the CPU generated address is within the range specified by *base register* and *base + limit register.* A memory access attempted outside the valid range, results in trap or a fatal error.

Space for learners:

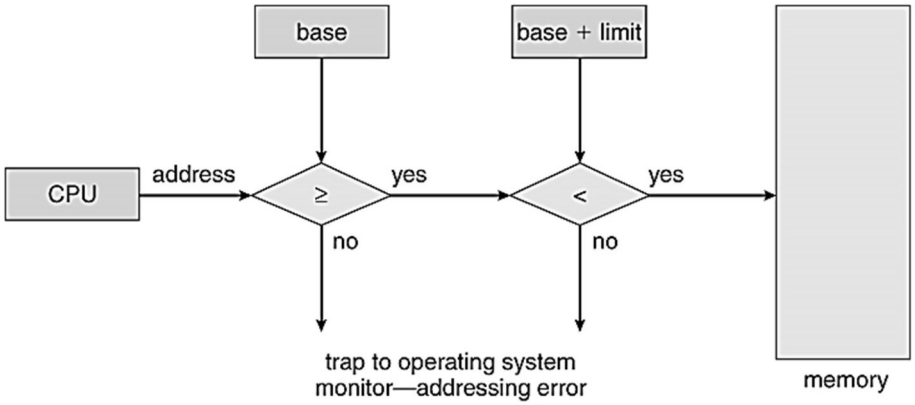


Figure 1.4: Memory address protection using base and limit register[1].

1.7 PAGING

To understand the concept of Paging we have to go through the following concepts:

- Process: It is a program in execution or a program placed in main memory for execution.
- Logical Address: It is the address that is generated by the CPU for a program while it is running. As the address does not exist physically it is also called virtual address. The hardware unit of memory known as memory management unit (MMU) maps logical address to physical address.
- Physical Address: A physical address is the actual address in the main memory.

Paging is a memory management scheme that is used to map CPU generated logical address of a process to physical address in main memory. A process consists of fixed size blocks; Figure 1.5 shows an example of a process with 4 blocks each of size 1 kilobyte. Size of a block depend upon architecture of the computer and varies between 512 bytes to 16 megabytes.

Space for learners:

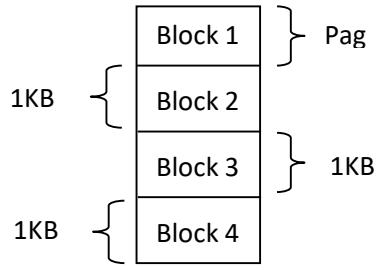


Figure 1.5: A Process with 4 blocks each of size 1 kilobyte.

The paging technique divides the logical memory to blocks of the fixed size known as Pages and divides physical memory into blocks of fixed-size known as Frames. Figure 1.6 shows an example of pages and frames in logical and physical memory respectively.

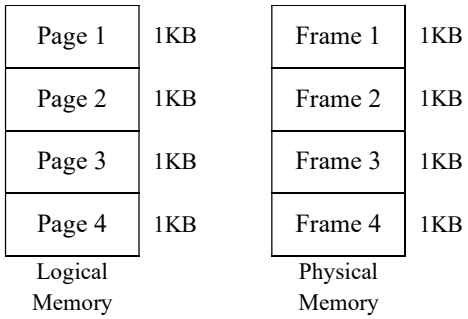


Figure 1.6: A Process with 1KB block size in logical and physical memory.

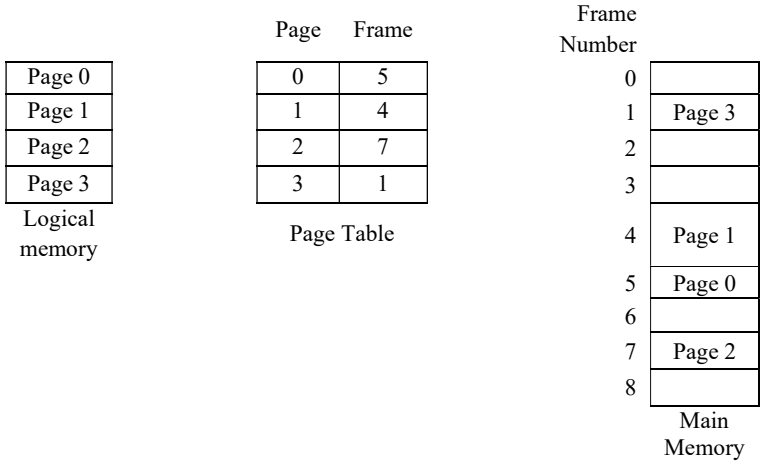


Figure 1.7: Paging model of physical and logical memory.

Space for learners:

Paging scheme allows a process to be stored in the main memory in noncontiguous manner. It also solves the problem of searching and fitting blocks of different sizes in main memory by having all block of same size. One more advantage of the paging scheme is that it prevents from external fragmentation that is if the main memory blocks are of varying sizes and the size of the free blocks are smaller than the size of the pages, then the operating will be required to merge two or more blocks into a single block large enough to fit a page. By keeping block of equal sizes for both pages and frames, such problems are resolved. The Figure 1.7 shows paging model of physical and logical memory. A page table is used for mapping between logical addresses and physical addresses. A page table resides in the main memory. The Figure 1.7 shows noncontiguous allocation of a process in main memory. The mapping of logical address to physical address is achieved using the page table.

The hardware support for paging is demonstrated using an example in Figure 1.8. The logical address generated by the CPU is divided into two parts namely *page number* and *displacement* with the page. The page number is used as an index in the page table to search for the corresponding frame number. The displacement is combined with frame number to get the physical address. In the Figure 1.8, the logical address having *page number 3* is searched for the corresponding frame number in the page table which is *frame number 15*. The *frame number 15* is combined with the *displacement 7* to form the physical address.

Space for learners:

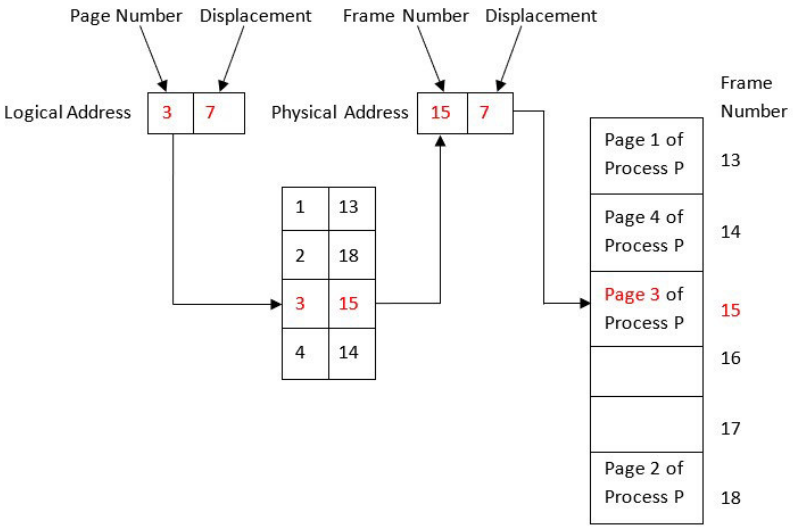


Figure 1.8: Paging hardware support.

If the size of the logical address space is 2^m and size of a page is 2^n bytes/words, then "*m-n*" bits of a logical address designate the page number the "*n*" bits designate the displacement or offset. Therefor the logical address is:

Page Number	Displacement
p	d
m - n	n

Paging Example -1:

Assume a page size of 1K and a 15-bit logical address space. How many pages are in the system?

Solution:

Page size = 1K = 2^{10}i.e. displacement, n=10 bits

No. of bits in logical address = 15, i.e. m=15 bits.

Therefore, no. of bits used for page number is, m - n = 5 bits

Total no. of pages in the system is 2^5 =32.

Paging Example -2:

Assume that a CPU has a 15-bit logical address space with 8 logical pages. How large are the pages?

Solution:

Space for learners:

There are 8 logical pages, that means 3 bits are required to address 8 logical pages ($2^3 = 8$).
Therefore, m - n=3 bits
Logical address is 15 bits, m=15 bits
Displacement = 15 -3 = 12 bits.
So, the pages are of size $2^{12} = 4096 = 4K$ bytes

Space for learners:

1.7.1 Paging Hardware Support

The operating system provides hardware support for quick search in the form of Associative memory or TLB. There are possibly two cases for a page search in TLB, Figure 1.9 illustrates the paging hardware with Translation Look aside Buffer for these two cases:

- If the search key/page is found it is called as a*TLB hit* and corresponding value/frame is returned from the TLB. Displacement is combined with frame number and the physical address is accessed.
- If the search key/page is not found it is called as a*TLB miss* and the page is searched in the page table stored in main memory. The frame number corresponding to the search page is combined with the displacement to access the address in the physical memory. Also the page number and frame number is added to the TLB so that if the same page is referred next time it is found quickly. In case the TLB is full, operating system selects a page replacement algorithm to replace an existing page with the new entry.

The percentage of times that a particular page number is found in the TLB is called the *hit ratio*. If the hit ratio is 60% that means 60 times out of 100 references the page will be found in TLB and remaining 40 times the page is found in the page table.

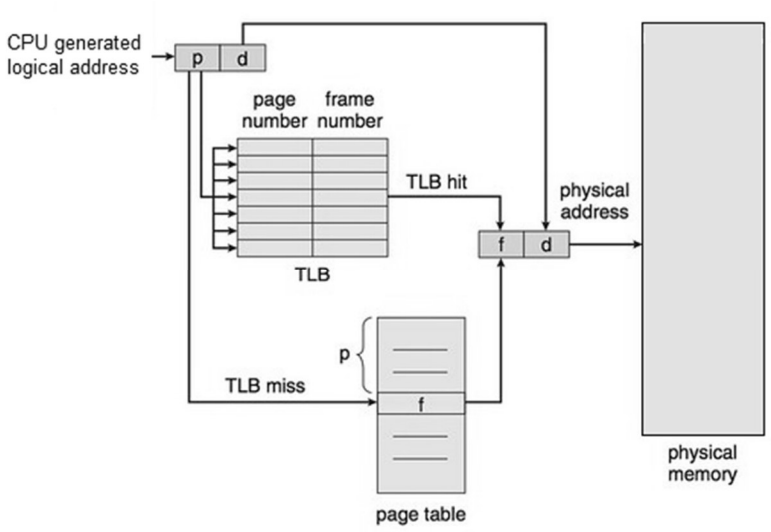


Figure 1.9: Paging hardware with Translation Look aside Buffer [1].

Paging Example -3:

If it takes 25 nanoseconds to search the TLB and 75 nanoseconds to access memory. If the hit ratio is 70%, calculate effective memory access time.

Solution:

If the page is in the TLB, time taken to access the physical address

= Time taken to search the TLB + Time taken to access memory

= 25 +75 =100 nanoseconds

If the page is in not in the TLB, time taken the physical address

= Time taken to search the TLB + Time taken to access page table + Time taken to access memory

= 25 +75 +75

= 175 nanoseconds

Hit ratio is 70%, therefore

Effective access time = 0.70 X 100 + 0.30 X 175 =122.5 nanoseconds.

Space for learners:

1.8 SEGMENTATION

Segmentation is a memory management scheme similar to paging that allows a process to be stored in the main memory in noncontiguous manner. Unlike paging where all the pages or frames are of fixed size, segmentation allows blocks or segments of variable size. Segmentation maps the user's view of a program onto the physical memory. Looking at the user's view in Figure 1.10, a program contains several variable size segments, such as the main program, subroutine, symbol table, methods etc. It also includes data structures like arrays, objects, variables, stacks etc. These segments and data structures are referred by their name without concerning about the address these segments are stored in memory. Users are not concerned about the order in which the segments are stored in the memory.

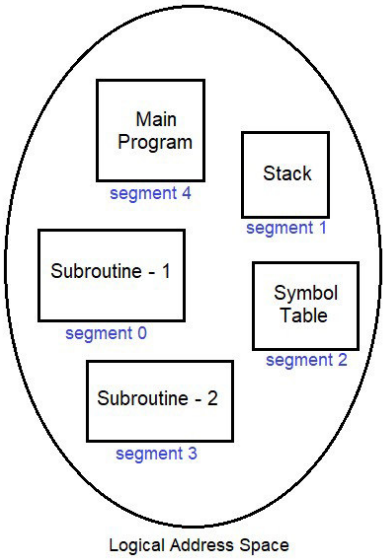


Figure 1.10: User's view of a program

The logical address space is a group of segments. Each segment has a name and a length. From the implementation point of view, segments are numbered instead of using name and the logical address is represented using the *two tuple:*

Segment-number	Displacement

Space for learners:

1.8.1 Segmentation Hardware

The mapping of the logical address <segment-number, displacement> to the physical address is achieved with the help of segment table and the segmentation hardware as shown in Figure 1.11. Each entry of the segment table has a segment limit and segment base. The base represents the starting address of the segment in the main memory and the limit specifies the length of the segment. The segment table is indexed on the segment number.

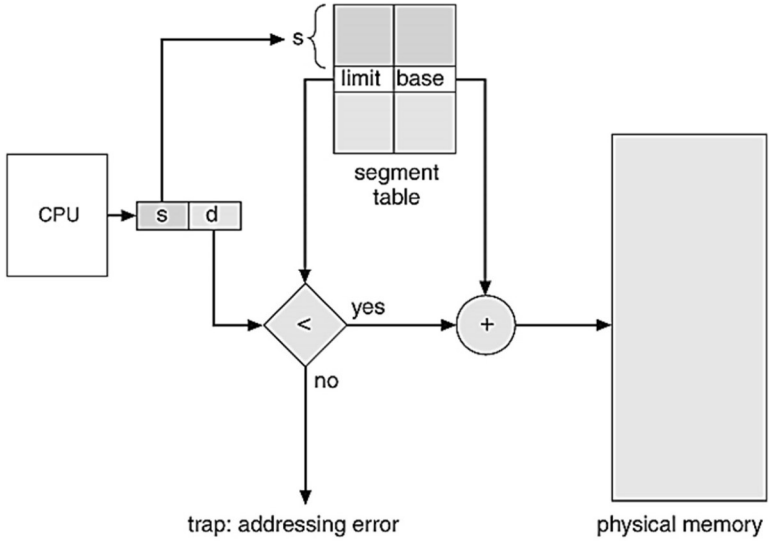


Figure 1.11: Segmentation Hardware[1].

The working of segmentation hardware starts by first identifying the segment number, *s* and the displacement, *d*of the logical address. The segment number is used to search the segment table, which is indexed on the segment number. The displacement, *d*of the logical address should be between 0 and limit. If the condition is not satisfied, it means that the logical address is going beyond the segment limit and a trap interrupt is initiated which is handled by the operating system.

A segmentation example is shown in Figure 1.12. There are 5 segments numbered from 0 through 4. The segments are stored in physical memory in noncontiguous manner. Also no specific ordering is followed for storing the segments as can be observed in the example. The segment table has an entry for each of the segment, the starting

Space for learners:

address of the segment mentioned as *base* and the length of the segment mentioned as *limit*. For example, segment 0 begins at address 5100 and length of the segment is limited to 500 bytes. Therefore, a reference to byte 17 of segment 0 is mapped to 5100 (base of segment 0) + 17 = 5117. Similarly, a reference to byte 88 of segment 4 is mapped to 7300 + 88 = 7388. A trap interrupt will be called if byte 1700 of segment 4 is referenced as the limit is 1500.

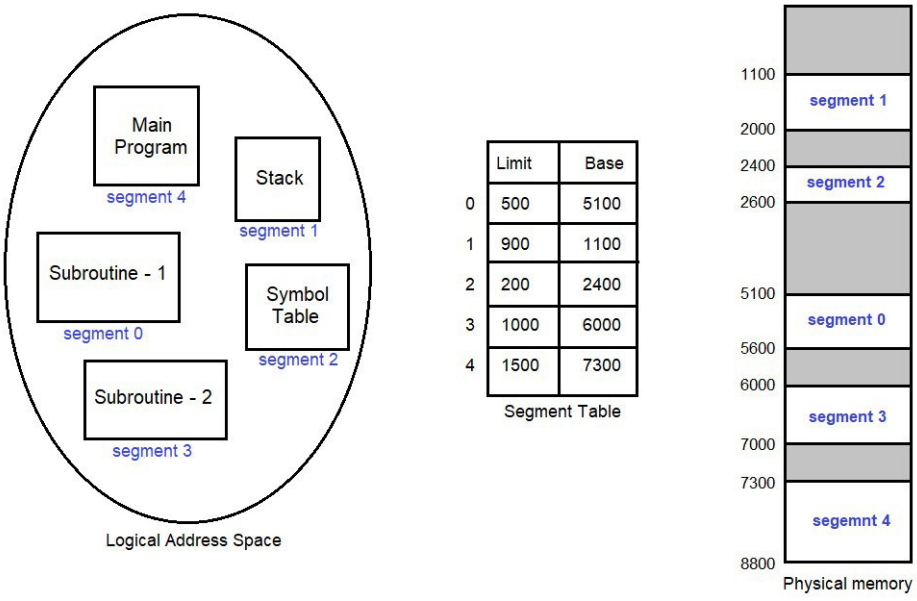


Figure 1.12: Example of Segmentation.

1.9 VIRTUAL MEMORY

The memory management scheme discussed in previous section requires the entire process to be in the main memory for execution. Most of the times there can be a requirement of many processes to be in the memory simultaneously for execution. This situation can prevent simultaneous execution of multiple processes due to the size of the main memory, which may not be large enough to hold all the processes. So, a concept of virtual memory was introduced.

A virtual memory management scheme allows execution of a process even if it is not completely in memory. That is, it requires only that

Space for learners:

section of the process's code to be in the memory that will be executed. Generally, a process contains several functions or procedures and not all the functions are required to be in the memory at the same time. So the function or the procedure that will be executed needs to be in the main memory, while the other functions or procedures can be placed in the secondary memory and wait for their turn of execution. So whenever a function is not available in the main memory, it is brought from the secondary memory to main memory for execution. The main advantage of this scheme is that a program larger than main memory can still run on a smaller physical memory. This is how a games like *Need for speed* or *Call of Duty* which require respectively 30 GB and 90 GB of memory can still run on a system having 6 GB RAM with sufficient hard disk space. Also, as only a section of the code of a process needs to be in memory so many process can be there in memory simultaneously. Thereby increasing CPU utilization and throughput.

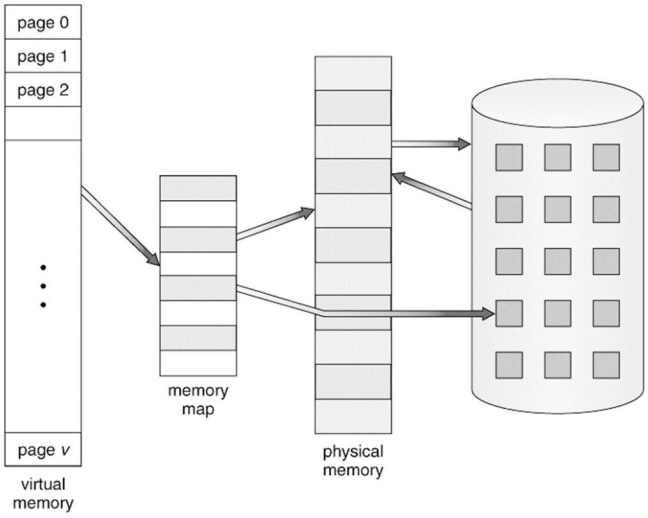


Figure 1.13: Example showing virtual memory larger than physical memory[1].

Figure 1.13 shows an example of a larger virtual memory than physical memory. The programmer thus need not have to worry about the size of the main memory available, thus can concentrate on the problem to be programmed. As can be seen in the Figure 1.13, pages from the large virtual memory address space is stored in the secondary memory and the pages are brought back to main memory whenever a call to those

Space for learners:

pages are required. If the main memory does not have any free slot for the pages, then some page replacement algorithms are used to replace the pages in main memory with the pages from secondary memory.

Figure 1.14 shows dynamic memory allocation, where the stack grows upward and the heap grows downward. The gap shown in the figure between the heap and the stack is the part of virtual address space and will require physical memory space only if either heap or stack grows or both of them grows.

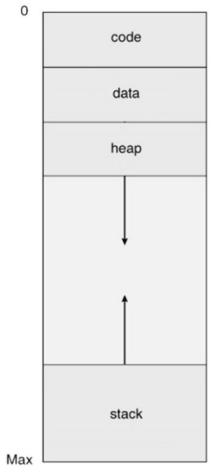


Figure 1.14: Virtual memory address space.

1.9.1 Demand Paging

Suppose a user wants to run a program, so the entire program is loaded to main memory from the secondary memory. However, if the program runs one option/case out of the several cases based on the user input, it is impractical to load the code for all the cases, other cases my never be called for execution. So a virtual memory technique known as demand paging is used to load only those pages of the process when they are required or whenever there is a demand for the page occurs during the program execution.

Space for learners:

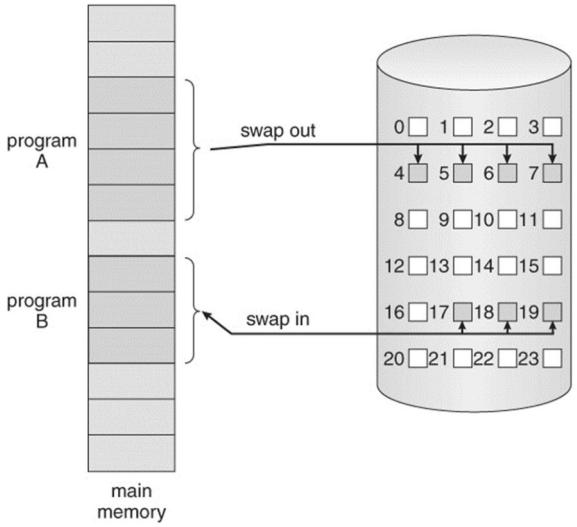


Figure 1.15: Example showing Demand Paging [1].

In Figure 1.15 shows and example of demand paging where pages 4, 5, 6 and 7 of Program A is swapped out of memory and pages 17, 18 and 19 of Program B is moved in to the memory because of the demand for the pages 17, 18 and 19. The method is implemented by a *pager* program responsible for demand paging.

1.10 PAGE REPLACEMENT ALGORITHMS

Since operating system allows virtual memory to be larger than the main memory, as a result a page fault may occur. A page fault occurs when a running process tries to accesses a memory page that is not loaded in main memory. In the event of a page fault, the operating system may have to replace an existing page with the new page. A page replacement algorithm is required in an operating system that utilizes paging for memory management. It determines which page has to be replaced when a new page arrives. Different page replacement algorithms offer various methods for determining which pages to replace. All methods have the same goal: to decrease page faults.

Space for learners:

1.10.1 FIFO Page Replacement

This is the most basic algorithm for replacing pages. The operating system uses this technique to maintain track of all memory pages in a queue, with the oldest page at the top. When a page has to be replaced, the first page in the queue is removed.

For example, consider the reference string 4, 0, 2, 5, 3, 5, 4, 0, 4, 5, 2 as shown in Figure 1.16 that is the order in which the memory references for the pages will be made. Assume that the memory which can accommodate three frames/pages at a time. The replacement algorithm uses the FIFO approach that is the first page moved to memory will be the first one to be replaced, this is followed by replacing second page, third page and so on with a new page. Initially, all the frames are empty so first three references (4, 0, 2) will result in page fault and are brought into the empty frames. The next reference 5 will replace the page 4 as it was the first page moved to the memory. Similarly, reference 3 will replace page 0 as it was the second page moved to memory. The next reference 5 is already in memory so no page fault and hence no page replacement. The process continues until all the page request in the reference string are processed. The total number of page faults using FIFO page replacement algorithm is 9.

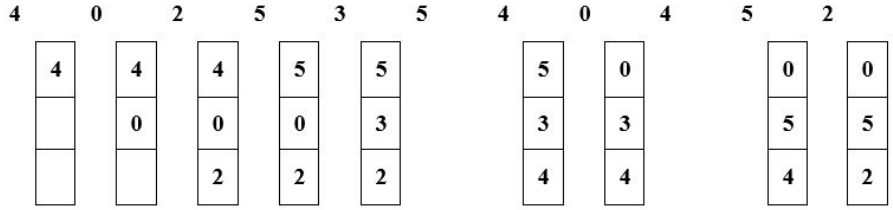


Figure 1.16: FIFO page replacement algorithm.

1.10.2 LRU Page Replacement

Least Recently Used (LRU) page replacement algorithm is a Greedy algorithm where the page to be replaced is the page which has not been used for the longest duration of time in the past. LRU keeps track of page usage over a period of time. It is based on the assumption that the

Space for learners:

pages that have been extensively utilized in the past will also be heavily used in the future.

For example, consider the reference string 4, 0, 2, 5, 3, 5, 4, 0, 4, 5, 2 as shown in Figure 1.17 that is the order in which the memory references for the pages will be made. Assume that the memory which can accommodate three frames/pages at a time. Initially, all the frames are empty so first three references (4, 0, 2) will result in page faults and are brought into the empty frames. The next reference 5 will replace the page 4 as on scanning left starting at reference 5, we find that among the pages (4, 0, 2), page 4 is the least recently used page. Similarly, the next reference 3 will replace page 0 as on scanning left starting at reference 3, we find that among the pages (5, 0, 2), page 0 is the least recently used page. The next reference 5 is already in memory so no page fault and hence no page replacement. The process continues until all the page request in the reference string are processed. The total number of page faults using LRU page replacement algorithm is 8.

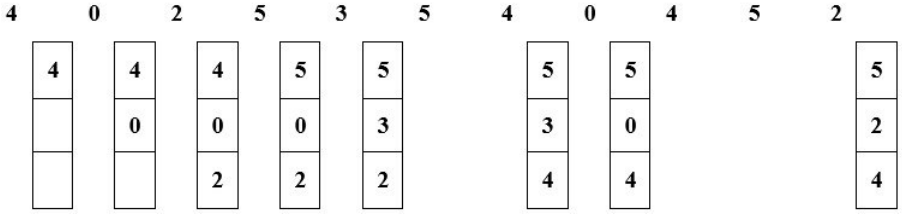


Figure 1.17: LRU page replacement algorithm.

1.10.3 Optimal Page Replacement

The best page replacement algorithm is the Optimal Page Replacement algorithm, which produces the fewest page faults. This method replaces pages that will not be utilized for the longest period of time in the future. The algorithm is difficult to implement because it requires future knowledge of the pages referenced pages.

For example, consider the reference string 4, 0, 2, 5, 3, 5, 4, 0, 4, 5, 2 as shown in Figure 1.18 that is the order in which the memory references for the pages will be made. Assume that the memory which can accommodate three frames/pages at a time. Initially, all the frames are empty so first three references (4, 0, 2) will result in page faults and are brought into the empty frames. The next reference 5 will replace the

Space for learners:

page 2 as on scanning right starting at reference 5, we find that among the pages (4, 0, 2), page 2 is not used for the longest duration of time. Similarly, the next reference 3 will replace page 0 as on scanning right starting at reference 3, we find that among the pages (4, 0, 5), page 0 is not used for the longest duration of time. The next reference 5 is already in memory so no page fault and hence no page replacement. The process continues until all the page request in the reference string are processed. The total number of page faults using LRU page replacement algorithm is 7.

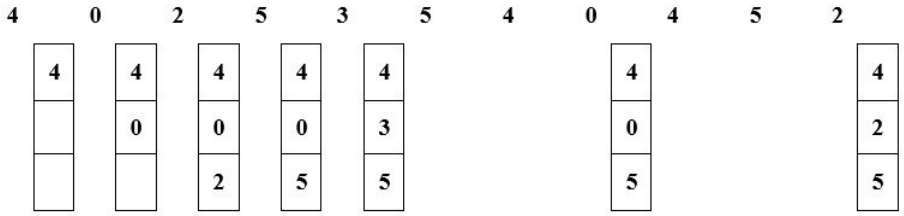


Figure 1.18: Optimal page replacement algorithm.

Space for learners:

CHECK YOUR PROGRESS

1. A memory buffer used to minimize the speed difference between CPU and Main memory is called _____.
   a) Main memory
   b) Cache memory
   c) register
   d) disk buffer

2. Increasing the RAM improves performance because of _____
   a) Increase in Virtual memory
   b) Bigger RAMs are faster
   c) Less page faults occur
   d) All of the above

3. Page fault occurs when
   a) Exception is thrown
   b) Requested page is not in memory
   c) Page is corrupted
   d) Requested page is in memory

4. Each logical address must be _____ than the value in limit register.
   a) less than
   b) equal to
   c) Not equal to
   d) greater than

5. Which one is the fastest memory
   a) Cache Memory
   b) Associative Memory
   c) Main Memory
   d) Secondary memory

6. Fixed-sized blocks in physical memory is called _____
   a) Block
   b) Frame
   c) Pages
   d) Segment

7. In paging CPU generated logical address has two parts _____and _____.
   a) Page offset & frame bit

Space for learners:

b) Page number & Page offset
c) Frame offset & displacement
d) Frame number & page offset

8. Paging does not suffer from _____.
   a) Internal Fragmentation
   b) External Fragmentation
   c) Both a) and b)
   d) None of the above

9. If it takes 10 milliseconds to search the TLB and 80 milliseconds to access the physical memory. If the TLB hit ratio is 0.6, the effective memory access time (in milliseconds) is _____.
   a) 120
   b) 122
   c) 134
   d) 124

10. The displacement 'd' in a logical address must be _____
   a) Greater than segment limit
   b) Greater than the segment number
   c) Between 0 and the segment number
   d) Between 0 and segment limit

11. In segmentation, each address is specified by _____
   a) A key and value
   b) A displacement and value
   c) A segment number & displacement
   d) A value and segment number

12. The virtual memory manager loads only those component of a program during execution as a when required is known as
   a) Segmentation
   b) Swapping

Space for learners:

c) Virtual memory
d) Demand Paging

Space for learners:

1.11 SUMMING UP

- The five memory types in the hierarchy are register, cache, main memory, hard disk and magnetic tapes based on access time, speed, cost and capacity of the memory.
- Cache memory acts as a buffer between the main memory and the CPU.
- Associative memory is also known as Translation look aside Buffer (TLB) is a special type of memory that is optimized to perform parallel searches on data.
- Address protection is implemented with the help of two registers, the base register and the limit register.
- Paging is a memory management scheme that is used to map CPU generated logical address of a process to physical address in main memory.
- Logical Address is the address that is generated by the CPU for a running program.
- A physical address is the actual address in the main memory.
- Paging is a memory management scheme that is used to map CPU generated logical address of a process to physical address in main memory.
- The logical address generated by the CPU is divided into two parts namely page number and displacement with the page.
- Translation look aside Buffer is a small, expensive but very fast associative memory.
- In a translation look aside buffer, if the search page is found it is called as an TLB hit if the page is not found it called as TLB miss.
- The percentage of times that a particular page number is found in the TLB is called the hit ratio.

- Segmentation is a memory management scheme similar to paging that allows a process to be stored in the main memory in noncontiguous manner.
- The mapping of the logical address <segment-number, displacement> to the physical address is achieved with the help of segment table and the segmentation hardware.
- A virtual memory management scheme allows execution of a process even if it is not completely in memory.
- A virtual memory technique known as demand paging is used to load only those pages of the process when they are required or whenever there is a demand for the page occurs during the program execution.
- A page fault occurs when a running process tries to accesses a memory page that is not loaded in main memory.
- A page replacement algorithm is required in an operating system that utilizes paging for memory management. It determines which page has to be replaced when a new page arrives.

Space for learners:

1.12 ANSWERS TO CHECK YOUR PROGRESS

1. b	2. c	3. b	4. a	5. b
6. b	7. b	8. b	9. b	10. d
11.c	12. d			

1.13 POSSIBLE QUESTIONS

1. What is an associative memory? Why it is used?
2. How does the operating system ensure that two or more processes do not use the same address space?
3. Explain Paging memory management scheme.
4. What is hit ratio? Why page should be replaced in the memory?
5. Consider a logical address space of 16 pages of 512 words each, mapped on to a physical memory of 64 frames. How many bits are

there in the logical address? How many bits are there in the physical address?

6. If it takes 125 nanoseconds to search the TLB and 500 nanoseconds to access memory. If the hit ratio is 90%, calculate effective memory access time.
7. Assume a page size of 4K and an 18-bit logical address space. How many pages are in the system?
8. Assume that a CPU has a 16-bit logical address space with 4 logical pages. How large are the pages?
9. What is segmentation? Explain.
10. Define a virtual memory. With a neat diagram, explain the working of a virtual memory. What are the benefits of a virtual memory?
11. What is demand paging? Explain.
12. Consider logical address 1025 and the following
13. page table for some process P0. Assume a 15-bit address space with a page size of 1K. What is the physical address to which logical address 1025 will be mapped?

6
2
3

14. Consider the following segment table:

Segment	Base	Length
34	100	100
21	2500	200
0	1200	50
90	1700	300
7	500	500
2	600	50
99	650	200

Space for learners:

What are the physical address for the following logical address?

i. 0,25
ii. 2,89
iii. 90,345
iv. 34,50
v. 99,201

15. Consider the reference string 0, 3, 0, 4, 5, 3, 2, 0, 5, 4, 6, 7, 3, 4

    Find the number of Page faults in each of the following cases assuming that memory can accommodate 4 pages/frames at a time.

    i. FIFO Page Replacement
    ii. LRU Page Replacement
    iii. Optimal Page Replacement

Space for learners:

1.14 REFERENCES & SUGGESTED READINGS

- Operating System Principles 8th edition by Abraham Silberschatz, Greg Gagne, and Peter Baer Galvin, Willey
- Operating Systems: Internals and Design Principles 9th edition by William Stallings, Pearson Education
- Madnik and Donovan, Operating systems, McGraw Hill.
- Andrew, S. Tannenbaum, Modern operating system, PHI.

UNIT 2: INPUT-OUTPUT ORGANIZATION

Unit Structure:

2.1 Introduction
2.2 Unit Objectives
2.3 Input/Output peripherals
2.4 Accessing I/O devices
2.5 Polling
2.6 Interrupts
  2.6.1 Handling multiple devices
  2.6.2 Polling scheme
  2.6.3 Vectored interrupt
  2.6.4 Priority interrupt
  2.6.5 Daisy chain
2.7 Direct memory access
  2.7.1 Bus arbitration
2.8 Buses
2.9 Application I/O interface
2.10 Kernel I/O subsystem
  2.10.1 I/O scheduling
  2.10.2 Buffering
  2.10.3 Caching
  2.10.4 Spooling
  2.10.5 Error handling
2.11 Summing Up
2.12 Answers to Check Your Progress
2.13 Possible Questions
2.14 References &Suggested Readings

Space for learners:

2.1 INTRODUCTION

Input and output peripherals are the key components of a computer system. The main task of a computer system can be categorized as Input/output and processing. In a computer system, the operating system (OS) is used to manage and control the input/output devices and perform operations on the data receives from I/O devices and output it. In this chapter, we will discuss the basic input/output hardware, Input/output services and interface provided by OS, how

OS bridges the gap between Input/output hardware interface and Input/output application interface, interrupts handling, etc.

Space for learners:

2.2 UNIT OBJECTIVES

After going through this unit, you will be able to:

- know about the I/O devices.
- understand different implementation issues related to I/O devices.
- explain how the I/O interface manages the gap between I/O devices and other units of a computer system.
- understand How to I/O devices are connected to a computer.
- learn about the I/O device controller.
- Learn how to access the I/O devices.
- explain how an OS handled interrupts and its different cases.
- understand how DMA is used to improve the throughput of a system.
- learn about the bus organization of a computer system.
- know about the functionalities of the kernel of an OS.

2.3 INPUT/OUTPUT PERIPHERALS

A computer system consists of four basic building blocks such as ALU, control unit, memory unit, and input/output unit. An input device can be defined as a hardware unit used to provide inputs into a system. The inputs may be a piece of data, information, control instruction, control signal, etc. The data or information can be of a different format – text, graphics, signals, etc., which is converted into a machine-understandable format by the input devices. The output hardware used in a computer system is keyboard, mouse, joystick, scanner, electronic pen, microphone, sensor devices, CCTV, light pen, trackball, graphic tablet, etc. The hardware peripherals used to get the output from the processor, project them or reproduce them in a human-understandable format can be defined as an output device/hardware. Output hardware's are monitor, printer, headphones, speaker, sound card, video card, plotter, screen

projector, speech synthesizer, GPS, etc. input/output hardware are may be wired or wireless.

To communicate with a machine, the I/O devices are connected with connection points of a machine known as a port. The I/O devices may use a common set of wires to transfer data/signals/addresses are known as a bus. A bus system in a computer can have three different types such as – to transfer data, data bus, control bus for transferring control signals, and address bus for transferring addresses in between processor and I/O devices or memory units.

A controller is used to control the I/O devices, system buses, and ports. A processor can send data and commands to the controller for performing I/O transfer. The controller has one or more registers to hold the data and commands. To read and write the device control registers the processor uses a set of standard data-transfer instructions and thus executes the I/O requests. The I/O device ports also have four registers namely – status, control, data in, and data out registers. The bits contained in the **status register** are used to depicting the states of completeness, availability of a byte of information to be read from the **data-in** register and occurring of a device error. To start a control command or to change the mode of a device, the bits in the control register are used. Reading the contents from the data-in register a host can access the inputs and by writing into the data-out register, a host can send output. The I/O device port registers are typically 1 – 4 bytes in size.

STOP TO CONSIDER

1. Certain bits in a control register of a serial port are used to choose a communication between full-duplex and half-duplex.
2. Other bits are used to check the parity.
3. Third bits are used to set the word-length between 7-8 bits.
4. Fourth bit is used choose the supported speed of the serial port.

2.4 ACCESSING I/O DEVICES

The computer system uses a common line to connect all the I/O devices with it called the bus. The bus system allows the I/O devices to exchange information as shown in **figure 2.1.** The bus system

Space for learners:

typically consists of three different sets of lines: to transfer addresses – address

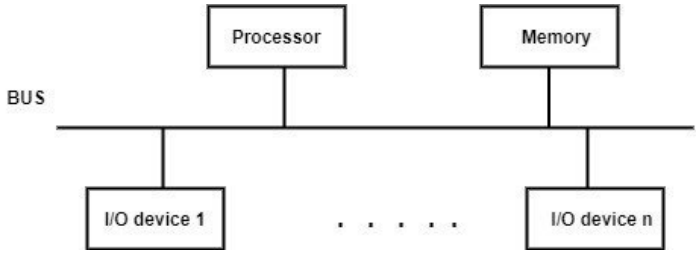


Figure 2.1: Bus structure of a computer system

bus, data – data bus, and control signals – control bus. Each I/O device has a unique set of specified addresses. When the processor placed a request by placing an address into the address line, any one of the connected devices will recognize it and respond to the command issued on the control line. During execution the processor request either a read or write command through the command line and transferred over the data bus. If the I/O device and the memory shared the same address space then the mechanism will be known as memory-mapped I/O.

The accessing speed of the I/O devices is varied from device to device and with the CPU also. The speed of the CPU is very high in comparison to the I/O devices. The CPU can execute millions of instructions when a user supplies input through an input device such as the keyboard. The CPU can execute the input character received from the keyboard, only after available in the input buffer of the keyboard interface.

The I/O interface for each I/O device provides a platform to connect between the buses and the devices such that it can communicate data to and fro between the CPU and I/O devices. It has been depicted in figure 2.2. The interfaces are consists of four different set of registers – data in, data out, status and control registers.

Space for learners:

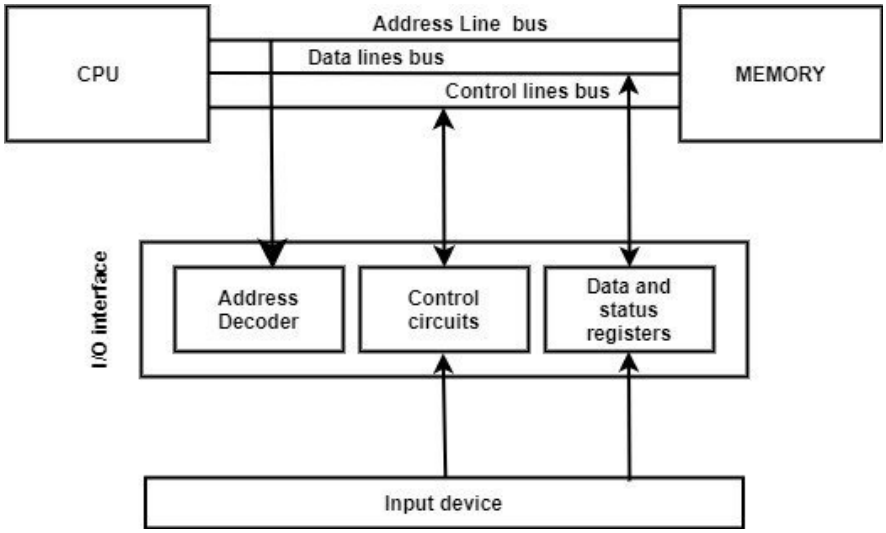


Figure 2.2: Communication of data between CPU and I/O devices

An input device interface for the keyboard sets the SIN bit of the status register as 1 if it has to perform an input operation. If the data is read by the processor for execution, it will be reset to 0. The SIN bit is regularly checked and read the data register. When the processor repeatedly checks for the status flag to get the required synchronization between the I/O device and the processor, is known as *program-controlled I/O.* Two other techniques to control the I/O device are interrupt-driven I/O and Direct memory access (DMA). Interrupt-driven I/O devices use to send an interrupt request over the bus to indicate that the device is ready for transferring data. In DMA, the I/O devices can directly communicate with the memory without intervening with the CPU frequently.

2.5 POLLING

The controller can announce its status using the **busy-bit** in the **status register**. The busy bit is set to 1 to state the status of the controller as busy and set to 0 when it is ready to accept the next command. The host machines announce the availability of commands to be executed by setting the **command-ready** bit. To place a command, a host repeatedly checks the busy bit until it becomes clear. The host sets the write bit into the command register and writes a byte into the data-out register. Then the host set the command bit ready. When the controller gets the command-ready as 1, immediately it sets the busy bit. The controller will read the command register and notice the write command. The controller

Space for learners:

reads the data-out register to get the bytes of information and do the input/output to the device. After completion, the controller clears the command-ready bit, clears the error bit in the status register indicating the successful completion of the I/O operation, and clears the busy bit to indicate that the task is finished. When the host checks for the busyness of the controller by checking the status register repeatedly until the busy bit becomes clear or 0 is known as busy-waiting or polling. If the duration of a wait is long, the host may switch to another task. In many computer architectures, three CPU instruction cycles such as read a device register, logical-and to extract a status bit, and branch if not zero are sufficient to poll a device. Polling may be inefficient if the host repeatedly attempted for busy-bit but does not find any device ready for service due to the involvement of the CPU with incomplete processing. To overcome this problem, the hardware controller should inform the CPU, when the device becomes ready for service, rather than require the CPU to poll repeatedly for an I/O completion. The hardware mechanism that enables a device to notify the CPU is called an interrupt.

Space for learners:

2.6 INTERRUPTS

When an I/O device is busy performing a task for a long time, and the processor repeatedly asks for the status of the device, the processor may not execute any necessary computation within this time and seat idle. But the CPU can perform some other necessary computations while waiting for the I/O device to become ready. It is possible by sending a signal called an ***interrupt*** to the processor. Using the concept of interrupt the waiting cycle can be eliminated and increase the throughput of the system.

Interrupts is a hardware mechanism where the CPU senses the interrupt-request line after the execution of each instruction. Interrupts can take place at any time. The I/O device controller sets the interrupts-request line to get the CPU cycle at any time. If the CPU detects an interrupt in the interrupts request line, it immediately saves the current state on a processor stack and jumps to the interrupts handler routine. Suppose the CPU executing the i^{th} instruction during execution while the interrupt request arrives as shown in **figure 2.3**. The processor will complete the execution of the instruction ***"i"*** and load the program counter by the first instruction of the interrupt service routine. The address of the next

instruction ***i+1*** will be stored onto the processor stack and after completion of the interrupt service routine, the PC will load the instruction ***i+1***.

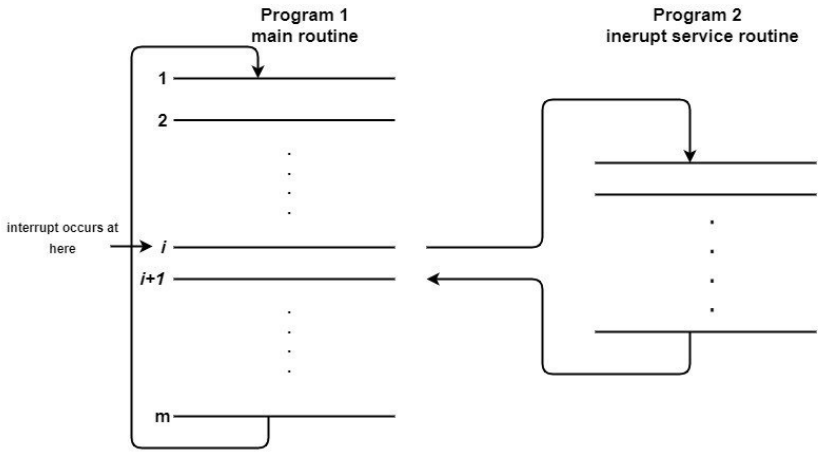


Figure 2.3: Execution of a program instruction by CPU

The interrupt handler routine determines the cause of the interrupt, performs the required operations, and executes the return-from operation to resume the CPU states before the interrupt. The following **figure 2.4** depicts a complete interrupts-driven input/output cycle. Input/output operations can be classified as synchronous and asynchronous.

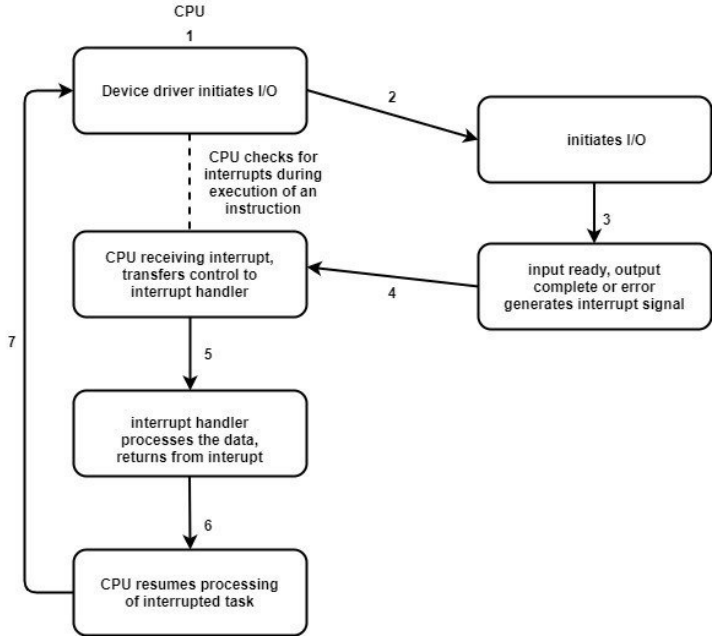


Figure 2.4: Complete interrupt driven input/output cycle

Space for learners:

In the case of a synchronous scheme, the CPU execution has to wait when the I/O devices are proceeds. But in the case of the asynchronous scheme, I/O operations can proceeds simultaneously with the CPU execution. The basic interrupt mechanism of a system allows the CPU to respond to an asynchronous event. To have an efficient input/output, the interrupt-controller hardware provides some more sophisticated features such as – ability to handle interrupt during complex processing tasks, the need to know about the interrupt initiating device without polling all the devices, and should support multilevel interrupt.

Two request lines – maskable and non-maskable are used by the CPU to identify the interrupt request type. In the case of a non-maskable interrupt, the CPU has to respond immediately by switching itself from its current execution. In case of a maskable interrupt, the CPU can turn off it such that the current execution is not interrupted.

Space for learners:

2.6.1 Handling Multiple Devices

A computer system may be connected with several I/O devices. There is no definite order in which I/O devices can request an interrupt, as the I/O devices are operationally independent. More than one device can activate the interrupt request line at the same time. This can raise some difficulties or issues in the system:

i. How a processor can recognize the interrupt generating device.

ii. If more than one device generates interrupt and maintain different interrupt service routine, then how the processor can recognize the starting address of the appropriate routine.

iii. Is it necessary to allow another device to set the interrupt request line, while one device is already being served its interrupt service routine?

iv. How to handle more than one interrupt generated exactly at the same time.

It is possible to handle more than one interrupt generated at the same time by breaking the tie and select any one of the two for service. After completion of the service routine of the selected device, the second one can be served. Some of the ways to handle multiple interrupts are:

2.6.1.1 Polling Scheme

A device can indicate its interrupt request by placing 1 in one of the bits of the status register. The particular bit in the status register is known as IRQ. To identify the interrupt requested device, the interrupt service routine has to poll all the I/O devices connected with the bus system. For servicing, a dedicated subroutine will call a device, which encountered its IRQ bit set at first and is being served. It is easy to implement but wastes notable time during interrogating the IRQ bits of all the devices which are not requesting any service. To overcome this problem vectored interrupt mechanism has been used.

2.6.1.2 Vectored Interrupt

To reduce the time used in interrogating IRQ bits of each device in the polling scheme, vectored interrupt mechanism is used. Here the I/O device itself has to inform the CPU directly about its interrupt request. The interrupt requesting device sends the first address of its interrupt service routine to the processor over the bus as an indication of generating an interrupt. Then the processor can start the execution of the corresponding interrupt service routine from the specified starting address send by the device. This system enables the processor to recognize the interrupt request if any I/O device even activates a single interrupt request line. The location shared by the device with the processor has to be considered as the starting address of the interrupt service routine. The CPU loads the starting address onto the program counter which is known as the ***interrupt vector***. Interrupt vector typically sends by the I/O devices over the data bus and the address length ranges in between 4 – 8 bits.

The processor may not respond to the interrupt vector immediately after requesting. The interrupts requesting devices have to wait to get the acknowledgment from the processor until the completion of the current execution. The interrupt requesting device can load its interrupt vector onto the bus if the CPU is ready to read it. While the processor is ready to read the interrupt vector, it enables the interrupt acknowledgment (INTA) line. The I/O device responds to the CPU by placing the interrupt vector onto the bus and turned off the INTR signal.

Space for learners:

2.6.1.3 Priority Interrupt

Space for learners:

When more than one device is involved in requesting an interrupt, the processor may have an arrangement to not allow other devices when one interrupt service routine is already is in process. Once an interrupt service routine is started to serve by the processor, it continues until completion of it and before the processor accepts an interrupt request from a second device. i.e. the second device has to wait until the completion of the current interrupt service routine execution. Sometimes the delay of execution may lead to an erroneous operation. Some of the waiting interrupt requests may have more priority than the executing one. To overcome this situation, the I/O devices have to arrange in a priority-based structure. Here the processor will accept the higher priority device request while servicing a device with lower priority.

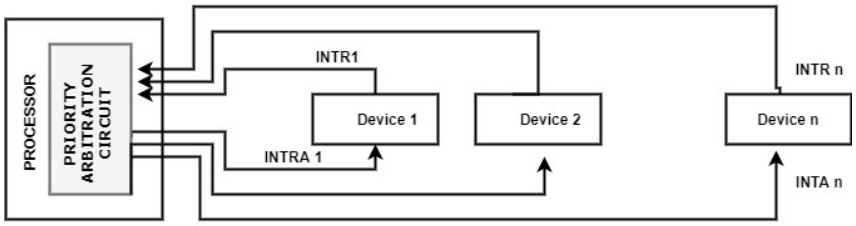


Figure 2.5: priority interrupt

If the processor accepts interrupt requests from other devices during the execution of an interrupt service routine; the accepting device will be selected based on device priority. This type of arrangement of I/O devices is known as a multiple-level priority organization. During the execution of a device request, the processor can accept interrupt requests from other devices which have a higher priority level than the current one. Once the processor has started to serve an interrupt service routine of a particular priority level, it disables interrupts from devices that have the same or lower priority level. The interrupt from higher priority devices may continue and be accepted. A multiple priority scheme can be implemented easily by using separate INTR and INTA lines for each device as shown in **figure 2.5**. In the diagram, each INTR line is assigned a different level of priority.

2.6.1.4 Daisy Chain

If more than one device generates interrupts simultaneously, in a multiple-level priority organization it is clear that the processor will serve the device with the highest priority. But in the case of vectored interrupt one device is select to send the interrupt request. Another efficient mechanism to solve this problem is the ***daisy chain***. In this widely used scheme, all the I/O devices are shared a common interrupt request line for sending interrupt requests. But the processor uses only one interrupt acknowledgment (INTA) line to acknowledge the devices. The INTA line is connected in a daisy chain fashion, such as it passes through all the I/O devices as shown in **figure 2.6**. When the several I/O devices are activatedthe INTR line, the processor responds to it by enabling the INTA line. At first, the CPU serves device 1. If device 1 has a pending request, it will hold the INTA signal line until the operations have been completed. In the daisy chain arrangement, the device which is electronically closest to the processor has the highest priority. The second device along the chain has the second-highest priority and so on. In daisy chain arrangement the requirement of wires is less in comparison to the priority base structure as shown in **figure 2.6**.

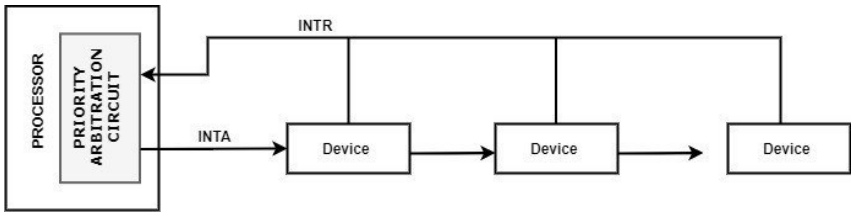


Figure 2.6: Daisy chain

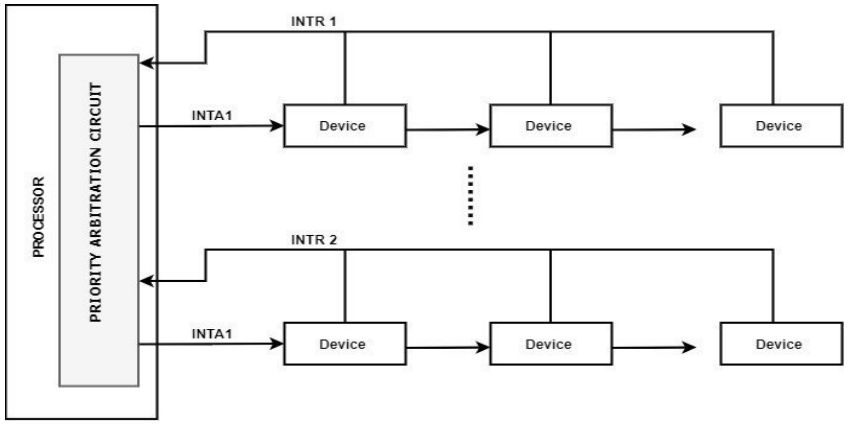


Figure 2.7: Daisy chain with priority based structure

Space for learners:

Combining both the priority interrupt and daisy chain mechanism, a more general and useful interrupt handling mechanism can be achieved. An example of such a hybrid structure is depicted in the following diagram 2.7. Here the devices are combined to form a group of a particular priority level. Within the group, devices are connected in a daisy chain fashion.

Space for learners:

2.7 DIRECT MEMORY ACCESS

It has been observed from the previous sections that the I/O operations are mainly concentrated on the transfer of data between the processor and I/O devices. The processor can perform this by polling a device or the device itself can send an interrupt request to the processor. During this process involvement of the processor is very high. When the processor served an interrupt service routine, several program instructions have to be executed for each data word transfer. Additionally, the processor may be busy with polling the status register of each device, instruction to increment the memory address and keep a record of the word count. When an interrupt occurred, the additional overhead associated with the saving of currently executed instruction address into a stack, load the program counter by starting address of the interrupt service routine and again resume the previous execution. For transferring a large block of data directly in between the I/O device and the main memory, a different technique may use known as Direct Memory Access (DMA). In DMA the continuous intervention of the CPU is reduced by allowing the I/O devices to access the memory unit directly under the control of a special control unit. The control unit is a part of the device interface and performed the DMA transfer. This control circuit is known as the DMA controller. In general, the I/O devices are accessing the main memory through the processor. But in the case of DMA, the role of the processor is replaced by the DMA controller. The DMA controller is responsible for providing the required memory addresses and control signals needed for data transfer.

The controller unit performs the data transfer operation without interrupting the CPU, but the complete operation is under the control of the main program executed by the processor. To start an operation, the CPU sends the starting address, data words in the block, and the data flow direction to the controller. Once the DMA controller receives this information, it started to perform the

requested operation. After completion of the data transferring, the CPU is informed by the DMA through an interrupt signal and the processor removes the control from the DMA controller.

Space for learners:

STOP TO CONSIDER

1. DMA transfers may have several attributes such as: Source address, destination address, transfer length, transfer type, block size, line stride, line length, etc.
2. DMA transfers can be categorized into two forms based on the hardware design and the involved peripheral devices, such as – single cycle DMA and burst transfer.

During the data transfer using DMA, if the current execution cannot continue by the CPU, then CPU can switch the operation to some other which is ready in the ready queue. After receiving the interrupt signal from the DMA controller, the processor can return to the process requested for data transfer.

The entire operations of input/output are always performed by the operating system. OS is responsible for suspending a program executed by the processor and starting another one. Initiation of a DMA is also a task of the OS.

For example, to transfer a data block from the main memory to disk, a dedicated program writes the starting address and the word count of the data block into the corresponding registers of the disk controller. The DMA controller performs this operation independently without intervening in the CPU. After completion of the transfer of the data block, the done bit of the status and the control register are set. Simultaneously the controller sends an interrupt request to the CPU and sets the IRQ bit. The status register is used to store the information about proper transferring of the data block or if there occurred any error during data transfer.

The priority of demanding the bus system by DMA devices is always more than the processor. Among the DMA devices, the highest speed devices are getting higher priority than the others. Memory accesses by the processor and the DMA devices are interwoven. In a computer system, most of the memory access requests are generated by the processor itself. Thus it can be said that the DMA devices are stealing the memory cycles from the CPU.

This mechanism is known as ***cycle stealing***.It can be stated that the DMA devices are allowed to access the memory of a computer system exclusively to transfer a block of data without interruption and it can be defined as ***block*** or ***burst*** mode. But if the processor and the DMA controller or two DMA controllers request the main memory at the same time, then a conflict may arise. As a remedy or to resolve the conflict, a mechanism is used by the bus system to coordinate among the memory accessing devices is known as ***bus arbitration***.

Space for learners:

2.7.1 Bus Arbitration

A device known as a ***bus-master*** is used to control the initiation of the data transfer through the bus system at any time. When the bus master relinquishes control of the bus, another device may acquire it immediately. But using the bus arbitration mechanism, the next device which is going to be the bus master will be selected and the bus master ship will be transferred. There are two arbitration processes namely centralized and distributed arbitration. A single bus arbiter is used to perform the required arbitration in centralized arbitration. In the case of distributed bus arbitration, all the devices participating in the selection process of the next bus master.

2.8 BUSES

The prime units of a computer system are interconnected through a common bus system. The common bus is used to transfer data, addresses, and control signals among the prime computer units such as memory, processor, I/O devices, and the control unit. The line required in bus arbitration and interrupt are also included in this common bus system. During transferring information a set of rules have to be followed by the buses known as protocols. A bus protocol can be defined as a set of rules to govern the behaviour of interconnected devices. There are three kinds of buses available in a system. To transfer data – data bus, to transfer addresses – address bus, and to transfer control signals – control bus.

In the control lines, a single R/W signal is used to indicate, either read or write operation to be performed on memory. If the signal bit is set to 1 means a read operation, and 0 indicates a write operation. These lines are also used to carry time information, at what time a

device will perform the read/write operation i.e. at what time a device will place data onto the bus or at what time receives data from the bus. Based on the timing of data transfer over a bus, two different categories can be obtained – synchronous and asynchronous bus systems.

In an asynchronous bus system, all the devices derived the time from a common clock. Equal time duration is assigned for each device in the synchronous bus. Each of the time intervals of equal size is known as the bus cycle. One word of data can transfer in a bus cycle.

In an asynchronous bus system, the common clock is replaced by two-time control lines such as ***Master–ready*** and ***Slave–ready***. This method is based on the use of a handshake between the master and slave. Here at first, the master indicates about the data whether it is ready for transmission or not, and second, the slave will respond to it. According to the handshaking protocol – the master will place the command information and addresses on the bus. It indicates the activation of the master–ready line and it is received by all the interconnected devices. At this point, all the devices have to decode their addresses. The slave line performs the required operation and informs the CPU by activating the slave–ready line. A full handshaking method can provide the highest degree of flexibility and reliability.

Space for learners:

2.9 APPLICATION I/O INTERFACE

I/O interfaces are used to enable the I/O devices and treat them in a standard and uniform manner. I/O interfaces can be customized with a layer known as the device drivers. Device drivers are used to hiding the differences between the device controllers from the I/O subsystem of the kernel. The use of the driver application encapsulates the behaviour of the devices in a few generic classes that hide the hardware differences from applications. It makes the operating system (OS) independent of the hardware and simplifies the job of the OS developers. This process restricts the device manufacturer either to manufacture a product that is compatible with the existing host controller interface of the OS or write a device driver to interface the new hardware to facilitate the OS. Thus a new device can be added to a computer through an OS. For different OS types, the device drivers may vary. For example, a graphics driver

for MS-DOS may not be supported by the OS, MS- Windows 2000, or in MAC OS.

The devices can be categorized based on the data transfer style. The ***character-stream*** device transfers the data byte by byte whereas the ***block device*** transfers a block of bytes as a unit at a time. The keyboard is an example of a ***character stream*** interface. In a ***sequential device*** data transfer occurred in a fixed order determined by the device, whereas a ***random access device*** can instruct the device to search data on any available memory location randomly. Some of the devices perform data transfer within a predictable response time known as ***synchronous devices***, whereas some of them show irregularity or in-predictable response time known as an ***asynchronous device***. A ***sharable device*** can be accessed by several processes or threads but a ***dedicated device*** cannot. Some of the devices can perform both ***read/write*** operations, but some of them can perform either read or write operations i.e. transfer of data in only one direction.

STOP TO CONSIDER

1. The device drivers are always operated within the kernel of an OS.
2. Kernel is the core part of an OS, which has direct access to the computer hardwares.
3. Device drivers are splitted into two layers such as – logical and physical layer.
4. Broad classification catrgories of device drivers are: kernel device drivers and user mode device drivers.

The block device interface will collect all the related information for accessing disk drivers and other block-oriented devices. These devices are expecting commands like *read()* or ***write()***. Random access devices can expect to have a ***seek ()*** command to locate the address of the next block to be transferred. To interact with the network devices, most of the OS including UNIX, Windows NT have used a network socket interface.

The computer system has ***clock*** and ***timer*** hardware to provide some basic functions such as current time, elapsed time, and a timer to perform trigger operations. The hardware used to maintain the trigger and the elapsed time is known as a ***programmable interval***

Space for learners:

timer. The OS provides an interface to the user to control the timer. During the power cut or shut-down mode of a system, a CMOS cell is used to supply power to the clock and timer.

Space for learners:

2.10 KERNEL I/O SUB SYSTEM

The kernel of an OS provides lots of functionalities related to input/output such as – scheduling, caching, buffering, spooling, device reservation, error handling, etc. The kernel also protects the system from malicious software and errant processes.

2.10.1 I/O Scheduling

The kernel subsystem scheduled the I/O request such that the devices can perform their operations in an ordered manner. I/O scheduling can improve the system performance by fairly distributing the devices among the processes and thus improve the average waiting time. To implement I/O scheduling a wait queue containing I/O requests for the devices to be maintained. If an application is issued a blocking I/O system call, then the I/O request will be kept in the wait queue for that particular device. The I/O scheduler may rearrange the contents of the wait queue to improve the system performance and average access time experienced by the applications. In the case of an asynchronous I/O, the I/O scheduling has to keep track of many I/O requests simultaneously. The efficiency of a computer system can be improved by using other techniques that use storage in main memory or a disk via buffering, caching, and spooling.

STOP TO CONSIDER

1. Different scheduling algorithms are used by an OS to scheduled the operations of I/O devices.
2. First Come First Serve (FCFS), Shortest Job First (SJF), Priority scheduling, Round robin etc. are the prime scheduling algorithms.

2.10.2 Buffering

Before transferring data from a device to another device or device to application, maybe store it in a memory area temporarily known as a buffer. Buffering is done due to three reasons. First, cope up with the speed mismatch between the speed of the producer and the consumer. The second, to provide adaptation for devices that have different data transfer sizes. A third use of buffering is to copy semantics for application I/O. Copying of data between kernel buffer and application data space is common in the operating system despite the overhead that this operation introduces, because of the clean semantics.

2.10.3 Caching

Cache memory is a faster memory placed in between the processor and the main memory. Caching is used to reduce the speed compatibility of the processor and the memory access time. It stores a block of data word into it which is being used by the processor shortly. The difference between buffering and caching is that in buffering an existing copy of data is hold whereas in cache a copy of data items can be store that can remain elsewhere.

2.10.4 Spooling

The output stream of data has to store in a buffer before going to an output device. It is known as spooling. A printer can be used to print the output of a process at a time, but many applications can request the printer at a time to print their output concurrently without mixing the outputs. The OS allows this by intercepting all the outputs to the printer. The output of each application is spooled into a separate disk file. Once the current printing process is being over, spooling system copied the next output to be print from the queue. It can copy one output from the queue to the printer at a time.

2.10.5 Error Handling

Using protected memory, an OS can protect a system from loss of data or information due to any errors that occurred in hardware or at the application level. Thus a system can protect from small mechanical faults. Devices and I/O data transfer may fail due to

Space for learners:

several reasons. It may be either transient such as when a network becomes overloaded or for permanent reasons such as when a disk controller becomes defective. An OS can handle transient kinds of failures effectively.

CHECK YOUR PROGRESS

Choose the correct options for the following questions:

1. Which of the following is a major part of the time taken when accessing data on the disk?

A. Settle time

B. Rotational latency

C. Seek time

D. Waiting time

2. How does the hardware trigger an interrupt?

A. Sending signals to CPU through the system bus

B. Executing a special program called interrupt program

C. Executing a special program called system program

D. Executing a special operation called system call

3. Which operation is performed by an interrupt handler?

A. Saving the current state of the system

B. Loading the interrupt handling code and executing it

C. Once done handling, bringing back the system to the original state it was before the interrupt occurred

D. All of these

4. Which of the following is an example of the spooled device?

A. A graphic display device

B. A line printer used to print the output of several jobs

C. A terminal used to enter input data to a running program

D. A secondary storage device in a virtual memory system

5. An application loads 100 libraries at start-up. Loading each library requires exactly one disk access. The seek time of the disk to a random location is given as 10 ms. The rotational speed of the disk is 6000 rpm. If all 100 libraries are loaded from random locations on the disk, how long does it take to load all libraries? (The time to

Space for learners:

transfer data from the disk block once the head has been positioned at the start of the block may be neglected)

A. 0.50 s

B. 1.50 s

C. 1.25 s

D. 1.00 s

6. Consider the following table of arrival time and burst time for three processes P0, P1, and P2.

Process	Arrival time	Burst Time
P0	0 ms	9 ms
P1	1 ms	4 ms
P2	2 ms	9 ms

7. The pre-emptive shortest job first scheduling algorithm is used. Scheduling is carried out only at the arrival or completion of processes. What is the average waiting time for the three processes?

A. 5.0 ms

B. 4.33 ms

C. 6.33 ms

D. 7.33 ms

8. Let the time taken to switch between user and kernel modes of execution be t1 while the time taken to switch between two processes be t2. Which of the following is TRUE? (GATE CS 2011)

A. t1 > t2
B. t1 = t2
C. t1 < t2

D. Nothing can be said about the relation between t1 and t2

9. A set of wires and a rigidly defined protocol that specifies a set of messages that can be sent on the wires.

A. Port

B. Node

C. Bus

D. None of these

Space for learners:

10. The _____ presents a uniform device-access interface to the I/O subsystem, much as system calls provide a standard interface between the application and the operating system.

A. Devices

B. Buses

C. Device drivers

D. I/O systems

11. The interrupt vector contains

A. The interrupts

B. the memory addresses of specialized interrupt handlers

C. the identifiers of interrupts

D. the device addresses

Space for learners:

2.11 SUMMING UP

- An input device can be defined as a hardware unit used to provide inputs into a system.
- The hardware peripherals used to get the output from the processor, project them or reproduce them in a human-understandable format can be defined as an output device/hardware.
- To communicate with a machine, the I/O devices are connected with connection points of a machine known as a port.
- A controller is used to control the I/O devices, system buses, and ports.
- The controller has one or more registers to hold the data and commands.
- The I/O device ports have four registers namely – status, control, data in, and data out registers.
- The I/O device port registers are typically 1 – 4 bytes in size.
- The computer system uses a common line to connect all the I/O devices with it called the bus.
- If the I/O device and the memory shared the same address space then the mechanism will be known as memory-mapped I/O.

- The CPU can execute the input character received from the keyboard, only after available in the input buffer of the keyboard interface.
- When the processor repeatedly checks for the status flag to get the required synchronization between the I/O device and the processor, is known as *program-controlled I/O.*
- The controller can announce its status using the **busy-bit** in the **status register**.
- Using the concept of interrupt the waiting cycle can be eliminated and increase the throughput of the system.
- In the case of a synchronous scheme, the CPU execution has to wait when the I/O devices are proceeds.
- In the case of the asynchronous scheme, I/O operations can proceeds simultaneously with the CPU execution.
- Two request lines – maskable and non-maskable are used by the CPU to identify the interrupt request type.
- In the case of a non-maskable interrupt, the CPU has to respond immediately by switching itself from its current execution.
- A device can indicate its interrupt request by placing 1 in one of the bits of the status register.
- The CPU loads the starting address onto the program counter which is known as the interrupt vector.
- In DMA the continuous intervention of the CPU is reduced by allowing the I/O devices to access the memory unit directly under the control of a special control unit.
- The controller unit performs the data transfer operation without interrupting the CPU, but the complete operation is under the control of the main program executed by the processor.
- The priority of demanding the bus system by DMA devices is always more than the processor.
- A device known as a bus-master is used to control the initiation of the data transfer through the bus system at any time.
- Before transferring data from a device to another device or device to application, maybe store it in a memory area temporarily known as a buffer.

Space for learners:

2.12 ANSWERS TO CHECK YOUR PROGRESS

Space for learners:

1.C

2.A

3.D

4.B

5. B

Explanation: Since transfer time can be neglected, the average access time is the sums of average seek time and average rotational latency. The average seeks time for a random location time is given as 10 ms. The average rotational latency is half of the time needed for a complete rotation. It is given that 6000 rotations need 1 minute. So one rotation will take 60/6000 seconds which is 10 ms. Therefore average rotational latency is half of 10 ms, which is 5ms.

Average disk access time = seek time + rotational latency

= 10 ms + 5 ms

= 15 ms

For 100 libraries, the average disk access time will be 15*100 ms

6. A

Explanation: Process P0 is allocated processor at 0 ms as there is no other process in the ready queue. P0 is preempted after 1 ms as P1 arrives at 1 ms and burst time for P1 is less than the remaining time of P0. P1 runs for 4ms. P2 arrived at 2 ms but P1 continued as the burst time of P2 is longer than P1. After P1 completes, P0 is scheduled again as the remaining time for P0 is less than the burst time of P2.

P0 waits for 4 ms, P1 waits for 0 ms, and P2 waits for 11 ms. So average waiting time is (0+4+11)/3 = 5.

7. C

Explanation: Process switching involves a mode switch. Context switching can occur only in kernel mode.

8. C
9. C
10. B

2.13 POSSIBLE QUESTIONS

A. Answer the following questions:

1. What is an interrupt?
2. What is the use of the INTA line in a processor?
3. What is a device driver?
4. What is a socket?
5. What is the function of an interrupt handler?
6. What is the role of a scheduler in an OS?
7. What types of errors can detect and correct by an OS?
8. What is a vectored interrupt?
9. What do you mean by polling?
10. What is spooling?

B. Answer the following questions:

1. Explain different types of interrupts.
2. Explain the process how the CPU identifies an interrupt requesting device.
3. Prepare a list of devices with its priority value for a particular operating system.
4. Explain the working principle of daisy chain.
5. Explain atleast two computer operations in details where DMA transfer is required.
6. How DMA transfer can be used in storing data explain.
7. How DMA transfers take place when transfer a large block of data explain.
8. Explain different categories of device drivers with example.

Space for learners:

9. How the device drivers work?
10. Explain the architectures of device drivers.
11. Explain the concept of virtual device driver.
12. Explain round robin and SJF scheduling algorithm with example.
13. How buffering is used in printing process explain briefly.
14. What do you mean by bus arbitration? Explain distributed bus arbitration with a block diagram.
15. Explain the bus structure of a computer system.
16. How daisy chain is used to handle interruptions? Explain the procedure to make a priority base daisy chain with a block diagram.
17. Explain how DMA is used in a system.
18. Explain the term memory-mapped I/O and program-controlled I/O.
19. Explain the process of accessing I/O devices.
20. What is an interrupt? Explain how multiple I/O devices can be handle by an OS?
21. Explain the concepts of bus arbitration.
22. Differentiate between polling and vectored interrupt.

Space for learners:

2.14 REFERENCES &SUGGESTED READINGS

- Operating System Principles 8th edition by Abraham Silberschatz, Greg Gagne, and Peter Baer Galvin, Willey
- Operating Systems: Internals and Design Principles 9th edition by William Stallings, Pearson Education
- Madnik and Donovan, Operating systems, McGraw Hill.
- Andrew, S. Tannenbaum, Modern operating system, PHI.

UNIT 3: INTRODUCTION TO DEADLOCK

Unit Structure:

3.1 Introduction

3.2 Unit Objectives

3.3 Definition of deadlock

3.4 Types of resources

3.5 Different types of deadlocks

3.5.1 Resource deadlock

3.5.2 Communication deadlock

3.6 Conditions for resource deadlock

3.7 Deadlock modelling

3.8 Strategies to deal with deadlock

3.9 Starvation

3.10 Summing Up

3.11 Answers to Check Your Progress

3.12 Possible Questions

3.13 References & Suggested Readings

Space for learners:

3.1 INTRODUCTION

In this unit you will learn about basic of deadlock. Deadlock is a situation where a set of processes are blocked because each process is holding a resource and waiting for another resource acquired by some other process. There are two types of resources available preemptable and non-preemptable. Deadlock can be resource deadlock or communication deadlock. There are four conditions those must hold to occur deadlock. The resource deadlock can be modelled using resource graphs. If the resource graph contains a cycle, then it means that deadlock present. There are different strategies to deal with deadlock which will be discussed in next chapter. Starvation is another process closely related to deadlock.

3.2 UNIT OBJECTIVES

After going through this unit, you will be able to:

- Understand the concept of deadlock
- Know about different types of resources
- Learn about different types of deadlock
- Learn about the four condition those must be hold to occur deadlock
- Learn about resource graph
- Learn how to modelled deadlock for single resource type
- Know about different strategies to deal with deadlock
- Learn about starvation

Space for learners:

3.3 DEFINITION OF DEADLOCK

Computer systems are full of resources that can be used only by one process at a time. Common examples include printers, tape drives etc. If a set of processes, try to simultaneously access the same resources then sometimes situation like deadlock may arise. Deadlock is a situation where a set of processes are blocked because each process is holding a resource and waiting for another resource acquired by some other process. Because all the processes are waiting, none of them will ever cause any event that could wake up any of the other members of the set, and all the processes continue to wait forever.

Deadlock can be defined formally as follows:

A set of processes is deadlocked if each process in the set is waiting for an event that only another process in the set can cause.

Deadlocks can also occur across machines. For example, many offices have a local area network with many computers connected to it. Often devices such as scanners, Blu-ray/DVD recorders, printers, and tape drives are connected to the network as shared resources, available to any user on any machine. If these devices can be reserved remotely, deadlocks can occur. Again for example, in a database system a program may have to lock several

records it is using, to avoid race conditions. If process *A* locks record *R1* and process *B* locks record *R2* and then each process tries to lock the other one's record, then the system will be deadlock. Thus, deadlocks can occur on hardware resources or on software resources.

Space for learners:

3.4 TYPES OF RESOURCES

A major class of deadlocks involves resources to which some process has been granted exclusive access. A computer will normally have many different resources that a process can acquire. A resource can be a hardware device (e.g. a Blu-ray drive) or a piece of information (e.g., a record in a database). A resource is anything that must be request, used, and released over the course of time.

If the resource is not available when it is requested, the requesting process has to wait. The process may wait a little while and try again or it may automatically have blocked and awakened when it becomes available. Usually a request or open system calls are provided to allow processes to explicitly ask for resources.

The resources mainly classified into two types-

a) Preemptable resources

b) Non-preemptable resources

A *preemptable resource* is the resource that can be taken away from its current owner (and given back later) without causing any effect. One preemptable resource is memory. For example, a system has 1 GB of memory, one printer and two 1-GB processes A and B. Each process A and B want to print something. At first process *A* requests and gets the printer, then starts to print. But, before it has finished the computation, it exceeds its time quantum. Process *B* now runs and tries, unsuccessfully as it turns out, to acquire the printer. Now this is a deadlock situation, because *A* has the printer and *B* has the memory, and neither one can proceed without the resource held by the other. But we can get rid of this deadlock situation because it is possible to preempt (take away) the memory from *B* by swapping it out and swapping *A* in. Now *A* can run, do its printing, and then release the printer. No deadlock occurs.

A *non-preemptable resource*, in contrast, is one that cannot be taken away from its current owner without causing any effect. If a process has begun to burn a Blu-ray, suddenly taking the Blu-ray recorder away from it and giving it to another process will result in a garbled Blu-ray. Blu-ray recorders are not preemptable at an arbitrary moment.

3.5 DIFFERENT TYPES OF DEADLOCK

3.5.1 Resource Deadlock

There are different types of deadlocks. The resource deadlock is one of the common deadlock. As mentioned earlier if each member of the set of deadlocked processes is waiting for a resource (non-preemtable) that is owned by a deadlocked process then none of the processes can run, none of them can release any resources and none of them can be awakened. This kind of deadlock is called a **resource deadlock**. Here the number of processes, the number of resources possessed and requested, hardware or software resources these things are unimportant.

3.5.2 Communication Deadlocks

Another kind of deadlock can occur in communication systems (e.g. networks), in which two or more processes communicate by sending messages. For example, consider a situation where process *A* sends a request message to process *B* and then blocks until *B* sends back a reply message. Suppose the request message gets lost. Then *A* is blocked waiting for the reply and *B* is blocked waiting for a request asking it to do something. This is a deadlock situation. But as we see that there are no resources involve in the above situation, so this is not classical resource deadlock. This situation is called a communication deadlock. Since there are no resources communication deadlocks cannot be prevented by ordering the resources. Again since there are no moments when a request could be postponed communication deadlocks cannot be avoided by careful scheduling. *Timeouts* is a technique to break communication deadlock. In most of the network communication

Space for learners:

systems, whenever a sender sends a message, it also starts a timer for a specific time duration. If an acknowledgment is not received from the receiving end before the timer timeouts, then the sender has to retransmit the message again. In this way, the deadlock is broken.

Resource deadlocks can also occur in communication network. As we know that in a network when a packet comes into a router from one of its hosts, it is put into a buffer for forwarding to another router and then to another until it gets to the destination. These buffers are resources and there are a finite number of them. Now consider four routers A, B, C and D. Each one has equal numbers of buffers. Suppose that all the packets at router *A* need to go to *B* and all the packets at *B* need to go to *C* and all the packets at *C* need to go to *D* and all the packets at *D* need to go to *A*. No packet can move because there is no extra buffer at the other end and we have a classical resource deadlock.

Space for learners:

3.6 CONDITIONS FOR RESOURCE DEADLOCKS

Coffman et al. (1971) showed that four conditions must hold to occur resource deadlock. If one of them is absent, no resource deadlock is possible.

1. The first condition is mutual exclusion condition according to which each resource is either currently assigned to exactly one process or is available.

2. The second condition is hold-and-wait condition. According to this condition processes currently holding resources that were granted earlier can request new resources.

3. The third condition is no-preemption condition. This condition holds when resources are non-preemptable i.e. resources previously granted cannot be forcibly taken away from a process. They must be explicitly released by the process holding them.

4. The fourth condition is circular wait condition. According to this condition there must be a circular list of two or more processes, each of which is waiting for a resource held by the next member of the chain.

Space for learners:

3.7 DEADLOCK MODELLING

Holt (1972) showed how the four conditions of resource deadlock can be modelled using a directed graph as follows-

- A *circle* represents a process.
- A *square* represents a resource.
- A *directed arc from a resource (square) to a process (circle)* represents that the resource is currently held by that process. In Figure 3.1(a) resource *R* is currently assigned to process *A*.
- A *directed arc from a process to a resource* means that the process is currently blocked waiting for that resource. In Figure 3.1(b) process *B* is waiting for resource *S*.
- A *cycle* in the graph means that there is a deadlock involving the processes and resources in the cycle. In Figure 3.1(c), process *C* is waiting for resource *T*, which is currently held by process *D*. Process *D* is not about to release resource *T* because it is waiting for resource *U*, held by *C*. Both processes will wait forever.

This directed graph can be mentioned as resource allocation graph. Resource allocation graphs are a tool using which we will be able to see if a given request/release sequence leads to deadlock. The requests and releases works are performed step by step and after every step the resource allocation graph is checked to see if it contains any cycles. If so, deadlock occur; if not, there is no deadlock. Here we consider the resource allocation graphs for the case of a single resource of each type. These resource allocation graphs can also be generalized to handle multiple resources of the same type (Holt, 1972).

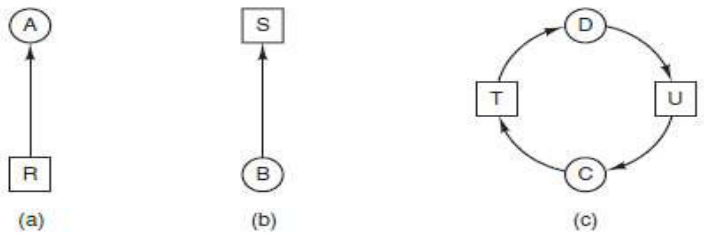


Figure3.1:Resource allocation graphs. (a) Holding a resource (b) Requesting a resource (c) Deadlock

(Reference: "Modern Operating Systems" by Andrew S Tanenbaum)

Now, how these resource allocation graphs can be used? For example, suppose there are three processes, *A*, *B*, *C* and three resources *R*, *S*, *T*. All these processes do both I/O and computing. The operating system is free to run any unblocked process at any instant. Thus, it could decide to run *A, B, C* sequentially without any pre-emption. Since all three processes are running sequentially, so there is no competition for the resources. Hence deadlock will not occur. When the processes are run sequentially, there is no possibility that while one process is waiting for I/O, another can use the CPU. Thus, running the processes strictly sequentially may not be optimal in this situation.

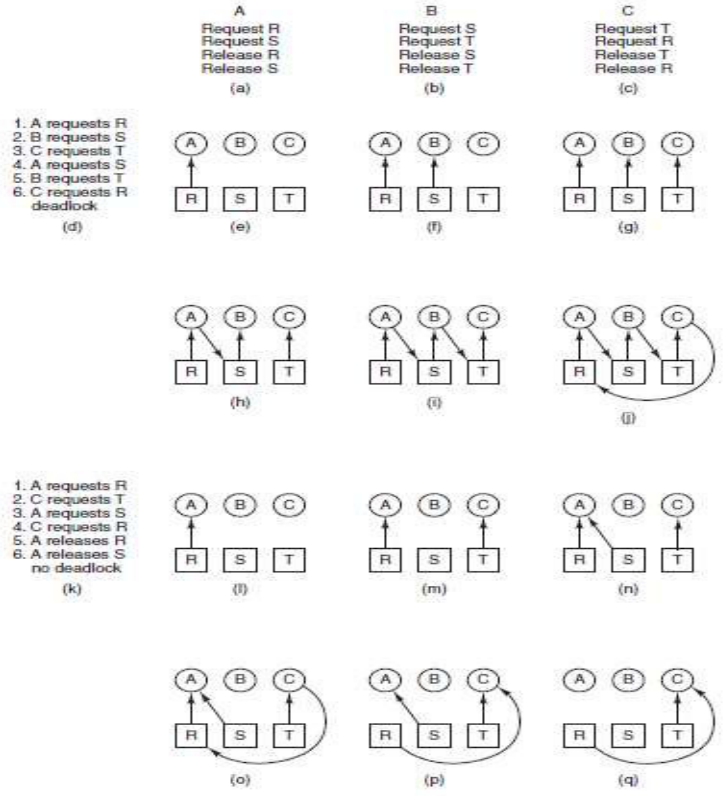


Figure3.2: An example of how deadlock occurs and how it can be avoided.
(Reference: "Modern Operating Systems" by Andrew S Tanenbaum)

Suppose the resource requests occur in the orders as shown in Figure 3.2(d). For this order the six resulting resource allocation graphs are as shown in Figure 3.2(e)–(j). From the Figure 3.2(j) it can be conclude that this order leads to deadlock as there is a cycle A→S→B→T→C→R→A present in the graph.

Space for learners:

However, if operating system knew that granting a particular request might lead to deadlock then it can simply suspend the process without granting the request until it is safe. In this example, it could suspend *B* instead of granting it *S* as the orders shown in Figure2.2(k). This order sequence leads to the resource allocation graphs of Figure 3.2(l)–(q), which do not lead to deadlock. After step (q), process *B* can be granted *S* because *A* is finished and *C* has everything it needs. Even if *B* blocks when requesting *T*, no deadlock can occur. *B* will just wait until *C* is finished.

3.8 STRATEGIES TO DEAL WITH DEADLOCKS

In general, there are four strategies to deal with deadlocks –

- Just ignore the problem. Maybe if you ignore it, it will ignore you.
- Detection and recovery. Let them occur, detect them, and take action.
- Dynamic avoidance by careful resource allocation.
- Prevention, by structurally negating one of the four conditions.

3.9 STARVATION

A problem closely related to deadlock is starvation. Starvation occurs if a process is indefinitely delayed. This may happen if the process wants a resource for execution which is never provided to the process or if the process is never provided the processor for some reason.

Some of the common causes of starvation are as follows –

- If a process is never allotted the resources it wants for execution.
- If high priority processes keep executing and low priority processes get blocked for indefinite time
- If there are not enough resources to provide to every process as required.

Space for learners:

- If processes are selected randomly for execution, then a process may wait for a long time because of non-selection.

Some ways to handle starvation are as follows –

- An independent manager can be used for allocation of resources. This resource manager distributes resources fairly and tries to avoid starvation.
- Random selection of processes for resource allocation or processor allocation should be avoided as they encourage starvation.
- The priority scheme of resource allocation should include concepts such as aging, where the priority of a process is increased the longer it waits. This avoids starvation.

CHECK YOUR PROGRESS

State TRUE or FALSE:

1. Deadlock occur when resources are pre-emptable.
2. In resource allocation graph a circle represents a process.
3. Resource allocation graphs are undirected.
4. Resource deadlock does not occur in network.
5. Starvation and deadlock are closely related.
6. A resource can be a hardware device or a piece of information.

3.10 SUMMING UP

- A set of processes is deadlocked if each process in the set is waiting for an event that only another process in the set can cause.
- A resource can be a hardware device (e.g. a Blu-ray drive) or a piece of information (e.g., a record in a database).
- A resource is anything that must be request, used, and released over the course of time.
- If the resource is not available when it is requested, the requesting process has to wait.

Space for learners:

- Usually a request or open system calls are provided to allow processes to explicitly ask for resources.
- The resources mainly classified into two types- Preemptable and non preemptable.
- A *preemptable resource* is the resource that can be taken away from its current owner (and given back later) without causing any effect.
- A *non preemptable resource*, in contrast, is one that cannot be taken away from its current owner without causing any effect.
- If each member of the set of deadlocked processes is waiting for a resource (non-preemtable) that is owned by a deadlocked process then none of the processes can run, none of them can release any resources and none of them can be awakened. This kind of deadlock is called a **resource deadlock**.
- Communication deadlock can occur in communication systems (e.g. networks), in which two or more processes communicate by sending messages.
- *Timeouts* is a technique to break communication deadlock.
- Four conditions must hold to occur resource deadlock- mutual exclusion condition, hold-and-wait condition, no-preemption condition, circular wait condition.
- Resource deadlock can be modelled using a directed graph called resource graph.
- A *cycle* in the resource allocation graph means that there is a deadlock involving the processes and resources in the cycle.
- Starvation occurs if a process is indefinitely delayed.

Space for learners:

3.11 ANSWERS TO CHECK YOUR PROGRESS

1. False.
2. True.
3. False.
4. False.
5. True.
6. True

3.12 POSSIBLE QUESTIONS

Short answer type questions:

1. Briefly explain about preemptable and non-preemptable resources.
2. Briefly explain about different types of deadlock.
3. Mention about the four conditions which must be satisfied to occur resource deadlock.
4. From a resource allocation graph how can we conclude whether there occurs deadlock or not?
5. Mention four strategies to deal with deadlock.

Long answer type questions:

1. Briefly explain about the resource allocation graph used to modelling deadlock for single resource type each.
2. Briefly explain with the help of an example how can we use the resource graph?

3.13 REFERNCES & SUGGESTED READINGS

- "Operating System Concepts" by Avi Silberschatz and Peter Galvin
- "Operating Systems: Internals and Design Principles" by William Stallings
- "Operating Systems: A Concept-Based Approach" by D M Dhamdhere
- "Modern Operating Systems" by Andrew S Tanenbaum

Space for learners:

UNIT 4: DEADLOCK PREVENTION, DETECTION AND AVOIDANCE

Unit Structure:

4.1 Introduction

4.2 Unit Objectives

4.3 Strategies to deal with deadlock

4.3.1 Ignore the Problem All Together

4.3.2 Deadlock Detection and Recovery

4.3.3 Deadlock Avoidance

4.3.4 Deadlock Prevention

4.4 Summing Up

4.5 Answers to Check Your Progress

4.6 Possible Questions

4.7 References &Suggested Readings

Space for learners:

4.1 INTRODUCTION

In this unit, you will learn about different strategies to deal with deadlock in detail. The strategies are mainly divided into four categories- simply ignore the deadlock, early detection and recovery of deadlock, avoid deadlock and prevent deadlock. Both the detection and avoidance methods consider the facts of single resource type and multiple resource types.

4.2 UNIT OBJECTIVES

After going through this unit, you will be able to:

- Understand about different strategies to deal with deadlock
- Know about different methods to detect deadlock for both single resource type and multiple resource types
- Learn about different ways to recover from deadlock
- Learn about different ways to avoid deadlock for both single resource type and multiple resource types
- Learn about different methods to prevent deadlock

4.3 STRATEGIES TO DEAL WITH DEADLOCK

4.3.1 Ignore the Problem all Together

The simplest approach is the ostrich algorithm: stick your head in the sand and pretend there is no problem. If deadlocks only occur once a year or so, it may be better to simply let them happen and reboot as necessary than to suffer the constant overhead and system performance penalties associated with deadlock prevention or detection. This is the approach that both Windows and UNIX take.

4.3.2 Deadlock Detection and Recovery

The second technique is detection and recovery. In the technique the system tries to detect the deadlock only when it is happening. After detection it will take some action to recover.

4.3.2.1 Deadlock Detection with One Resource of Each Type

Suppose there is only one resource of each type. For example, one scanner, one Blu-ray recorder, one plotter, and one tape drive, but no more than one of each class of resource. As discussed in Unit 2, a resource allocation graph can be construct to detect deadlock in such a system.

Consider a system with seven processes *A, B, C, D, E, F,G* and six resources *R, S, U, T, V, W*. The state of the system is as follows-

- Process *A* holds *R* and wants *S*.
- Process *B* holds nothing but wants *T*.
- Process *C* holds nothing but wants *S*.
- Process *D* holds *U* and wants *S* and *T*.
- Process *E* holds *T* and wants *V*.
- Process *F* holds *W* and wants *S*.
- Process *G* holds *V* and wants *U*.

The question is: ‘‘Is this system deadlocked, and if so, which processes are involved?’’

Space for learners:

The resource allocation graph for this system will be as follows-

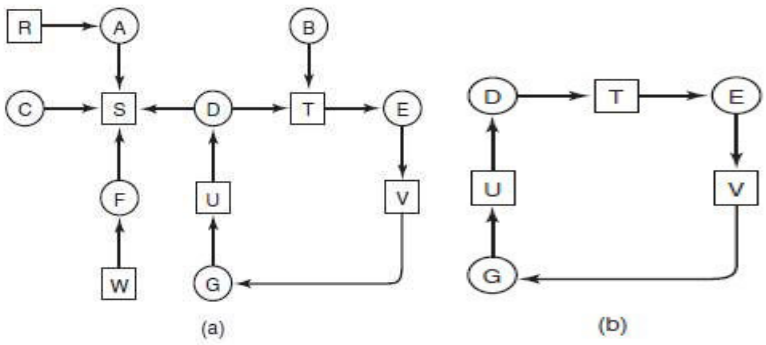


Figure 4.1 (a) Resource allocation graph (b) Cycle from (a)
(Reference: "Modern Operating Systems" by Andrew S Tanenbaum)

We can see that the graph in Figure 4.1 (a) contains one cycle and processes *D*, *E* and *G* are all deadlocked. But Processes *A*, *C* and *F* are not deadlocked because *S* can be allocated to any one of them, which then finishes and returns it. Then the other two can take it in turn and also complete.

Now to detect deadlock using this resource allocation graph, it is needed to detect cycle in a directed graph. The following algorithm can detect cycles in a directed graph. The algorithm uses one dynamic data structure *L,* a list of nodes, as well as a list of arcs. The arcs are either marked or unmarked. Marked arc means it is already visited and unmarked arc means it is not visited yet. The steps are as follows-

Algorithm 4.1

Step 1: For each node *N*, in the graph, perform the following steps with *N* as the starting node.

Step 2: Initialize *L* to the empty list and all arcs remain unmarked.

Step 3: Add the current node to the end of *L* and check to see if the node now appears in *L* two times. If it does, the graph contains a cycle and the algorithm terminates.

Step 4: From the given node, see if there are any unmarked outgoing arcs. If so, go to Step 5; if not, go to Step 6.

Step 5: Pick an unmarked outgoing arc at random and mark it. Then follow it to the new current node and go to Step 3.

Space for learners:

Step 6: If this node is the initial node, the graph does not contain any cycles and the algorithm terminates. Otherwise, we have now reached a dead end. Remove it and go back to the previous node, that is, the one that was current just before this one, make that one the current node and go to Step 3.

Working of Algorithm 4.1 on the resource allocation graph of Figure 4.1(a)

Iteration 1:

Let us first start the algorithm from a randomly selected node *R* (Step 1). After that successively consider A, B, C, S, D, T, E, F as the starting node.

Initialize *L* as empty list (Step 2).

Add *R* to the *L* and move through the only unmarked outgoing arc to node *A*. Add *A* to *L*. Thus *L* becomes *L* = [*R*, *A*] (Step 3, Step 4, Step 5).

From *A* go to node *S*. *L* becomes *L* = [*R*, *A*, *S*] (Repeat Step 3, Step 4, Step 5).

S has no outgoing arcs, so it is a dead end. Thus backtrack to node *A* (Repeat Step 3, Step 4, Step 6).

A has no unmarked outgoing arcs, so backtrack to *R*. Since *R* is the starting node so inspection of *R* has been completed (Repeat Step 3, Step 4, Step 6).

Iteration 2:

In second iteration start the algorithm from the randomly selected node *A* (Step 1).
Initialize *L* as empty list (Step 2).

Add *A* to *L*. From *A* move through the only unmarked outgoing arc to node *S*. *L* becomes *L* = [*A*, *S*] (Step 3, Step 4, Step 5).

Space for learners:

S has no outgoing arcs, so it is a dead end. Thus backtrack to node *A*. *A* is the starting node so inspection of *A* has been completed(Repeat Step 3, Step 4, Step 6).

Iteration 3:

In third iteration start the algorithm from the randomly selected node *B* (Step 1).

Initialize *L* as empty list (Step 2).

Add *B* to *L*. From *B* move through the only unmarked outgoing arc to node *T*. *L* becomes *L* = [*B*, *T*] (Step 3, Step 4, Step 5).

From *T* go to node *E*. *L* becomes *L* = [*B*, *T, E*] (Repeat Step 3, Step 4, Step 5).

From *E* go to node *V*. *L* becomes *L* = [*B*, *T, E, V*] (Repeat Step 3, Step 4, Step 5).

From *V* go to node *G*. *L* becomes *L* = [*B*, *T, E, V, G*] (Repeat Step 3, Step 4, Step 5).

From *G* go to node *U*. *L* becomes *L* = [*B*, *T, E, V, G, U*] (Repeat Step 3, Step 4, Step 5).

From *U* go to node *D*. *L* becomes *L* = [*B*, *T, E, V, G, U, D*] (Repeat Step 3, Step 4, Step 5).

From *D* go to node *T*. *L* becomes *L* = [*B*, *T, E, V, G, U, D, T*] (Repeat Step 3, Step 4, Step 5).

Now *T* is the current node and it appears two times in *L*. Thus, there is a cycle in the graph. So the algorithm terminates here (Step 3).

4.3.2.2 Deadlock Detection with Multiple Resources of each Type

Suppose there are multiple copies of some of the resources. For such case the deadlock detection algorithm has been discussed below-

Space for learners:

Let, n is the numbers of processes and m is the number of resource Classes. The i^{th} process is denoted as P_i, the i^{th} class resource is denoted as E_i.

The processes are either marked or unmarked. Initially all processes are unmarked. At the end of the algorithm if all processes are marked, then it indicates that they are able to complete and are thus not deadlocked. Other it indicates deadlock occurs.

At any instant of the algorithm some of the resources are assigned and are not available. A_i denotes the number of instances of i^{th} class resource that are currently available (i.e. unassigned).

The ***existing resource vector*** $E= (E_1, E_2, ..., E_m)$ gives the total number of instances of each resource in existence. For example, if class 1 is tape drives, then $E_1=2$ means the system has two tape drives.

The ***available resource vector*** $A=(A_1, A_2, ..., A_m)$gives the number of instances of each resource class that are currently available. If both of the two tape drives are assigned, A_1 will be 0.

Let, C be the ***current allocation matrix*** and R be the ***request matrix***.

C_{ij} is the number of instances of resource E_j that are held by process P_i. Similarly, R_{ij} is the number of instances of resource E_j that P_i wants.

Again, every resource is either allocated or is available.

$\sum_{i=1}^{n} Cij + A_j = E_j$

Suppose A and B are two vectors. Then the relation $A \leq B$ means that each element of A is less than or equal to the corresponding element of B i.e. $A \leq B$ holds if and only if $A_i \leq B_i$ for $1 \leq i \leq m$.

The steps of deadlock detection algorithm are as given below. Note that initially all processes are unmarked.

Space for learners:

Space for learners:

Algorithm 3.2

Step 1. Find an unmarked process $P_i$such that the i^{th} row of R is less than or equal to A.

Step 1.1. Add the i^{th} row of C to A.

Step 1.2 Mark the process and go back to step 1.

Step 2. If no such process exists, the algorithm terminates.

Working of Algorithm 3.2

Suppose we have 3 processes (P_1, P_2, P_3) and 4 resource classes (tape drives, plotters, scanners, and Blu-ray drives). Process P_1 has one scanner, Process P_2 has two tape drives and a Blu-ray drive, Process P_3 has a plotter and two scanners. Each process needs additional resources, as shown in the matrix R of Figure 4.2.

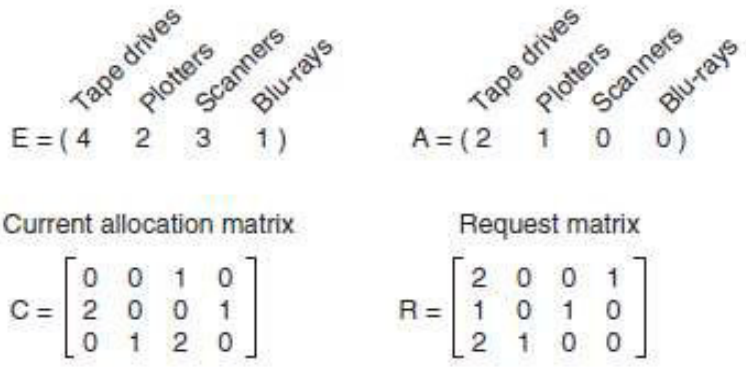


Figure 4.2: An example of deadlock detection algorithm

(Reference: "Modern Operating Systems" by Andrew S Tanenbaum)

Iteration 1:

Suppose start the algorithm by picking the unmarked process P_1. Now, is ($R_1 \leq A$) true? No. (Step 1)

Pick another unmarked process P_2 and check whether ($R_2 \leq A$) is true. No. (Step 1)

Pick another unmarked process P_3 and check whether ($R_3 \leq A$) is true. Yes. (Step 1)

Add C_3 to A and mark P_3 and go to Step 1(Step 1.1).

Thus A becomes A= (2 2 2 0)

Iteration 2:

Pick the unmarked process P_2 and check whether ($R_2 \leq A$) is true. Yes. (Step 1)

Add $C_2$to A and mark P_2 and go to Step 1(Step 1.1).

Thus A becomes A= (4 2 2 1)

Iteration 3:

Pick the unmarked process P_1 and check whether ($R_1 \leq A$) is true. Yes. (Step 1)

Add C_1 to A and mark $P_1$and go to Step 1(Step 1.1).

Thus A becomes A= (4 2 3 1)

Iteration 4:

No unmarked process found (Step 1).

So go to Step 2 and terminate.

At the end of the algorithm no unmarked processes remain. So there is no deadlock in the system.

4.3.2.3 Recovery from Deadlock

After detect a deadlock the next work will be recover from deadlock. Some ways to recover from deadlock are discussed below-

- ***Recovery through pre-emption***

One way to recover from deadlock is pre-emption i.e. temporarily take a resource away from its current owner and give it to another process. Recovering this way is frequently difficult or impossible.

- ***Recovery through rollback***

If the system designers and machine operators know that deadlocks are likely, they can arrange to have processes checkpointed periodically. Checkpointing a process means that its state is written to a file so that it can be restarted later. The checkpoint contains not only the memory image, but also the resource state, in other words, which resources are currently assigned to the process. To be most effective, new checkpoints should not overwrite old ones but should

Space for learners:

be written to new files, so as the process executes, a whole sequence accumulates. When a deadlock is detected, it is easy to see which resources are needed. To do the recovery, a process that owns a needed resource is rolled back to a point in time before it acquired that resource by starting at one of its earlier checkpoints. In effect, the process is reset to an earlier moment when it did not have the resource, which is now assigned to one of the deadlocked processes. If the restarted process tries to acquire the resource again, it will have to wait until it becomes available.

- ***Recovery through killing processes***

The simplest way to break a deadlock is to kill one or more processes. In this approach, the process to be killed is carefully chosen because it is holding resources that some process in the cycle needs.

4.3.3 Deadlock Avoidance

The deadlock detection methods assume that a process asks for all the necessary resources at once. But in most of the system resources are requested one at a time. In this situation the system should decide whether granting a resource is safe or not. The deadlock avoidance algorithms are designed based on the concept of safe states.

A state is said to be ***safe*** if there is some scheduling order in which every process can run to completion even if all of them suddenly request their maximum number of resources immediately. Otherwise the state is ***unsafe***. A safe state can guarantee that all processes will finish. In unsafe state, there is no sequence that guarantees completion. An unsafe state is not a deadlocked state.

For example- Suppose we have 10 resources of same type. These 10 resources are used by three processes A, B, C as shown in Figure 4.3 (a). Process *A* has 3 resources and it may need as many as 9 resources to complete it. Similarly, B has 2 resources and may need as many as 4 resources, C has 2 resources and may need as many as 7 resources to complete it. Now, the question is- "Is the state shown in Figure 4.3 (a) safe?"

Space for learners:

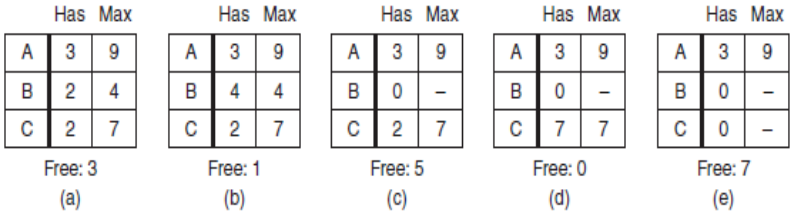


Figure 4.3: Demonstration that the state in (a) is safe
(Reference: "Modern Operating Systems" by Andrew S Tanenbaum)

From Figure 4.3 (a) it is seen that all three processes already have 7 resources. So, number of available resources is 3. The maximum number of resources needed by B is 4. Thus, scheduler can run B as B needs 2 more resources to complete it (Figure 4.3 (b)). After completion B will release the resources (Figure 4.3 (c)). At this point number of available resources is 5 and A need 6 more resources to complete, C need 5 more resources to complete. In this scenario scheduler can run C (Figure 4.3 (d)). After Completion of C number of available resources will be 7 (Figure 4.3 (e)). At this point scheduler can run A. Hence the state shown in Figure 4.3 (a) is a safe state as there is an execution order B, C, A for the processes.

4.3.3.1 The Banker's Algorithm for a Single Resource Type

The ***banker's algorithm*** is a well-known deadlock avoidance algorithm. Suppose a banker will grant loans to a group of customers. The algorithm will check if granting the request leads to an unsafe state. If so, the request is denied. Otherwise the request is carried out.

For example- Suppose there are four customers *A*, *B*, *C, D* which needs total 22 credit units (Figure 4.4 (a)). The bankers assume that not all customers will need their maximum credit immediately, so he reserves only 10 credit units out of 22 credit units. Now from these 10 credit units he granted each customer a certain number of credit units as shown in Figure 4.4 (b). Thus a total of 8 units are granted and 2 units remain free. This state is safe because there is an ordered sequence like *C, D, B, A* to grant the credit units if all the customers suddenly asked for their maximum credit units.

Space for learners:

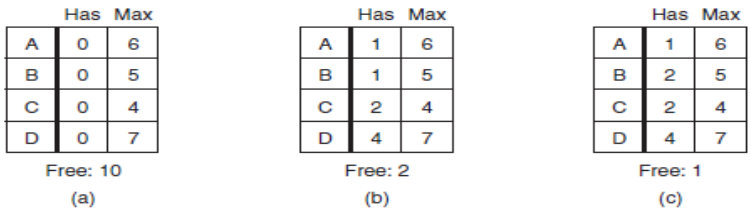


Figure 4.4Three resource allocation states: (a) Safe. (b) Safe. (c) Unsafe

(Reference: "Modern Operating Systems" by Andrew S Tanenbaum)

Suppose at this point customer B request for one more unit and it is granted (Figure 4.4 (c)). Thus a total of 9 units are granted and 1 unit remain free. This state is an unsafe state as there is no sequence available to grant the credit units if all the customers suddenly asked for their maximum credit units.

The banker's algorithm considers each request as it occurs, seeing whether granting it leads to a safe state. If it does, the request is granted; otherwise, it is postponed until later.

4.3.3.2 The Banker's Algorithm for Multiple Resource Types

The banker's algorithm can be generalized to handle multiple resources.

For example- Suppose there are 5 processes and 4 numbers of resources (Tape drives, Plotters, Printers, Blu-rays) The first matrix of Figure 4.5 is the current allocation matrix (*C*). It shows how many of each resource are currently assigned to each of the five processes. The second matrix is the request matrix (*R*) and it shows how many resources each process still needs in order to complete. The three vectors of the Figure 4.5 show the existing resources *E*, the assigned resources *P*, and the available resources *A* respectively.

Space for learners:

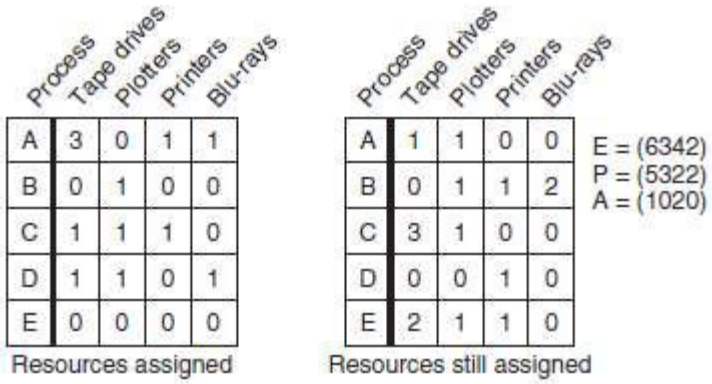


Figure 4.5 The banker's algorithm with multiple resources
(Reference: "Modern Operating Systems" by Andrew S Tanenbaum)

Steps of Banker's algorithm for multiple resources-

Step 1. Assume that a process keeps its resources until it exits. Now search for a row in *R* which is less than or equal to *A*.
Step 1.1If no such row exists, the system will eventually deadlock since no process can run to completion.

Step 1.2. If such a row exists, then assume that the process of the chosen row requests all the resources it needs and finishes. Mark that process as terminated and add all of its resources to the *A* vector.

Step 2. Repeat Steps 1 until either all processes are marked terminated (in which case the initial state was safe) or no process is left whose resource needs can be met (in which case the system was not safe).

Now, in Figure 4.5 the current state is safe. Suppose that process *B* makes a request for the printer. This request can be granted because the resulting state is still safe. After that process *D* can finish and then processes *A* or *E*, followed by the rest. Now imagine that after giving *B* one of the two remaining printers, *E* wants the last printer. Granting that request would reduce the vector of available resources to (1 0 0 0), which leads to deadlock, so *E*'s request must be deferred for a while.

4.3.4 Deadlock Prevention

Deadlock is essentially impossible, because it requires information about future requests, which is not known. Thus to avoid deadlock in

Space for learners:

real systems Coffman et al. (1971) provides four conditions. If we can ensure that at least one of these conditions is never satisfied, then deadlocks will be structurally impossible (Havender, 1968).

- ***Attacking the Mutual-Exclusion Condition***

First let us attack the mutual exclusion condition. If no resource were ever assigned exclusively to a single process, we would never have deadlocks. Avoid assigning a resource unless absolutely necessary, and try to make sure that as few processes as possible may actually claim the resource.

- ***Attacking the Hold-and-Wait Condition***

The second of the condition stated is that- if we can prevent processes that hold resources from waiting for more resources, we can eliminate deadlocks. One way to achieve this goal is to require all processes to request all their resources before starting execution. If every thingis available, the process will be allocated whatever it needs and can run to completion. If one or more resources are busy, nothing will be allocated and the process will just wait. An immediate problem with this approach is that many processes do not know how many resources they will need until they have started running. In fact, if they knew, the banker's algorithm could be used.

- ***Attacking the No-Preemption Condition***

If a process is holding some resources and requests another resource that cannot be immediately allocated to it, then all resources currently being held are pre-empted. The pre-empted resources are added to the list of resources for which the process is waiting. The process will be restarted only when the old resources are assigned to it and as well as the new resources that it is requesting.

- ***Attacking the Circular Wait Condition***

The circular wait is a scenario like there exists a set $\{P_0, P_1, \ldots, P_n\}$ of waiting processes such that: P_0 is waiting for a resource that is held by P_1, $P_1$is waiting for a resource that is held by P_2, …, P_{n-1} is waiting for a resource that is held by P_n and P_n is waiting for a resource that is held by P_0.

The circular wait can be avoided in several ways. One way is to a process is permitted only to a single resource at any moment. If it needs a second one, it must release the first one.

Space for learners:

Another way to avoid the circular wait is to provide a global numbering of all the resources, as shown in Figure 4.6. Now the rule is- processes can request resources whenever they want to, but all requests must be made in numerical order. A process may request first a printer and then a tape drive, but it may not request first a plotter and then a printer.

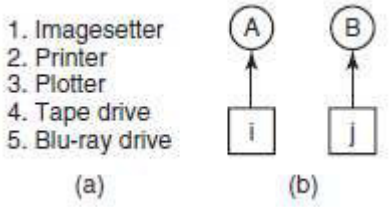


Figure 4.6(a) Numerically ordered resources. (b) A resource graph. (Reference: "Modern Operating Systems" by Andrew S Tanenbaum)

With this rule, the resource allocation graph can never have cycles. In figure 4.6 it is seen that a deadlock will occur if and only if *A* requests resource *j* and *B* requests resource *i*. Assuming *i* and *j* are distinct resources, they will have different numbers. If $i > j$, then *A* is not allowed to request *j* because that is lower than what it already has. If $i < j$, then *B* is not allowed to request *i* because that is lower than what it already has. Either way, deadlock is impossible.

CHECK YOUR PROGRESS

State TRUE or FALSE:

1. In safe state there is an order sequence to complete the processes.
2. Unsafe state is a deadlock state.
3. Banker's algorithm is a well-known deadlock avoidance algorithm.
4. To avoid circular wait one technique uses global numbering.

Space for learners:

4.4 SUMMING UP

- Using deadlock detection algorithm, the system tries to detect deadlock detect the deadlock only when it is happening.
- A resource allocation graph can be construct to detect deadlock in a system with single resource type each.
- To detect deadlock (single resource type each) using resource allocation graph, it is needed to detect cycle in a directed graph
- Suppose there are multiple copies of some of the resources. For such case the resource allocation graph cannot detect deadlock. So, a different algorithm is needed to deadlock detection in such situation.
- The deadlock detection algorithm (multiple copies of some of the resources) considered all processes are either marked or unmarked. Initially all processes are unmarked. At the end of the algorithm if all processes are marked, then it indicates that they are able to complete and are thus not deadlocked. Other it indicates deadlock occurs.
- The ***existing resource vector*** gives the total number of instances of each resource in existence.
- The ***available resource vector*** gives the number of instances of each resource class that are currently available.
- Suppose A and B are two vectors. Then the relation $A \leq B$ means that each element of A is less than or equal to the corresponding element of B i.e. $A \leq B$ holds if and only if $A_i \leq B_i$ for $1 \leq i \leq m$.
- One way to recover from deadlock is pre-emption.
- The second way to recover from deadlock is rollback.
- The simplest way to break a deadlock is to kill one or more processes.
- The deadlock detection methods assume that a process asks for all the necessary resources at once. But in most of the system resources are requested one at a time. In this situation the system should decide whether granting a resource is safe or not.
- A state is said to be ***safe*** if there is some scheduling order in which every process can run to completion even if all of them

Space for learners:

suddenly request their maximum number of resources immediately. Otherwise the state is ***unsafe***.

- A safe state can guarantee that all processes will finish.
- In unsafe state, there is no sequence that guarantees completion.
- An unsafe state is not a deadlocked state.
- The ***banker's algorithm*** is a well-known deadlock avoidance algorithm. Suppose a banker will grant loans to a group of customers. The algorithm will check if granting the request leads to an unsafe state. If so, the request is denied. Otherwise the request is carried out.
- To avoid deadlock in real systems Coffman et al. (1971) provides four conditions.
- The first condition stated prevent deadlock is attack the mutual exclusion condition. If no resource were ever assigned exclusively to a single process, we would never have deadlocks.
- The second of the condition stated is that- if we can prevent processes that hold resources from waiting for more resources, we can eliminate deadlocks.
- The third condition is if a process is holding some resources and requests another resource that cannot be immediately allocated to it, then all resources currently being held are pre-empted.
- The fourth condition to prevent deadlock is the circular wait. It can be avoided in several ways. One way is to a process is permitted only to a single resource at any moment. If it needs a second one, it must release the first one.
- Another way to avoid the circular wait is to provide a global numbering of all the resources.

Space for learners:

4.5 ANSWERS TO CHECK YOUR PROGRESS

1. True
2. False.
3. True.
4. True.

4.6 POSSIBLE QUESTIONS

Short answer type questions:

1. What is checkpointing a process mean?
2. What is safe and unsafe state?
3. What is circular wait of processes?
4. Briefly explain two techniques to avoid circular wait of processes?

Long answer type questions:

1. Briefly explain the ways to detect deadlock with one resource of each type.
2. Briefly explain the ways to detect deadlock with multiple resource of each type.
3. Briefly explain the techniques to recovery from deadlock.
4. Briefly explain the Banker's algorithm to avoid deadlock with one resource of each type.
5. Briefly explain the Banker's algorithm to avoid deadlock with multiple resource of each type.
6. Briefly explain different ways to prevent deadlock.
7. For example- Suppose we have 12resources of same type. These 12 resources are used by three processes A, B, C, D as shown in Figure 3.3 (a). Process *A* has 1 resources and it may need as many as 8 resources to complete it. Similarly, B has 2 resources and may need as many as 4 resources, C has 3 resources and may need as many as 6 resources to complete it, D has 4 resources and may need as many as 5 resources to complete it Now, the question is- "Is the state safe?"

4.7 REFERENCES & SUGGESTED READINGS

- "Operating System Concepts" by Avi Silberschatz and Peter Galvin
- "Operating Systems: Internals and Design Principles" by William Stallings

Space for learners:

- "Operating Systems: A Concept-Based Approach" by D M Dhamdhere.
- "Modern Operating Systems" by Andrew S Tanenbaum

Space for learners:

UNIT5: MULTIPROGRAMMING SYSTEM

Space for learners:

Unit Structure:

5.1 Introduction
5.2 Unit Objectives
5.3 Basic Concepts of Multiprogramming
5.4 I/O System
5.5 Memory Management
5.6 File System
5.7 Summing Up
5.8 Answer to Check Your Progress
5.9 Possible Questions
5.10 References & Suggested Reading

5.1 INTRODUCTION

As the name suggest, multiple process are in the main memory. In this case, CPU utilization is increased. Two or more processes reside in main memory are executed concurrently. This is done by switching the CPU from one program to another program almost instantaneously. To manage the entire resources of the system is the main motive of multiprogramming. Command processor, file system, I/O control system, and transient area are the primary components of multiprogramming system.

5.2 UNIT OBJECTIVES

After going through this unit you will be able to

- Understand the concept of multiprogramming
- Learn different scheduling algorithm used in multiprogramming
- Know the structure and operations of I/O
- Learn about the data transfer using direct memory access
- Understand memory management in multiprogramming
- Understand the algorithm for memory allocation
- Explain about paging and segmentation.

- Understand the file system Structure
- Understand the operations on File system
- Learn the file access methods

Space for learners:

5.3 BASIC CONCEPTS OF MULTIPROGRAMMING

The concept of multiprogramming depends on the capability of a computer to store instructions (programs) for long-term use. Multiple programs are loaded in main memory. Operating system assigns CPU to the first program. If that particular program needs some I/O operations then CPU instead of waiting for that program, OS allocates the next program to CPU. Once the I/O operation of the first program is completed, CPU continues with that program. In this fashion, execution takes place in multiprogramming system. The objective is to reduce CPU idle time by allowing new jobs to take over the CPU whenever the currently running job needed to wait (e.g. for user I/O).

Before multiprogramming was introduced, operating system working was very simple- it executes only one program at a time via CPU.

With the introduction of multiprogramming, operating system now executes multiple programs using different mechanism and several options existed for allocating CPU time

For decision-making two types of scheduling were introduced - **Job scheduling** and **CPU scheduling**. The selection of jobs to load into memory is termed as Job Scheduling and the selection of a job existing in memory to execute via the CPU is known as CPU scheduling. Both these decisions of Job scheduling and selection of a job are made by the operating system.

5.3.1 Process Scheduling

A process is a program which is in execution. The activity which involve removal of running process from the CPU and selection of another process based on some strategy is known as Process Scheduling. The objective of multiprogramming is to have some process running at all times, to maximize CPU utilization and the

objective of time sharing is to switch the CPU among processes so frequently that users can interact with each program while it is running. To meet these objectives, the process scheduler selects an available process (possibly from set of several available processes) for program execution on the CPU.

In Multiprogramming operating systems, process scheduling is an essential function. Multiprogramming system allows a number of process to load in main memory and scheduler decides which program to remove and to execute next.

Space for learners:

5.3.1.1Process Scheduling Queues

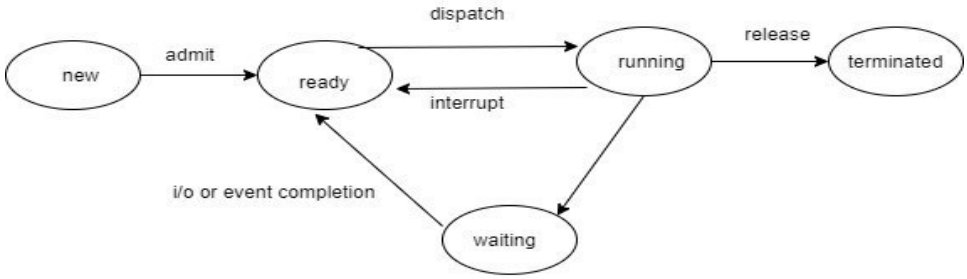
All PCBs are maintained by operating system in Process Scheduling Queues.PCB stands for Process Control Block. PCB is a data structure created by operating system. When a process is created, operating system maintains certain information in Process Control Block. A separate queue for each of the process states and PCBs are maintained by operating system. When the state of a process is changed, its PCB is unlinked from its current queue and moved to its new state queue.

The Operating System maintains the following important process scheduling queues −

- **Job queue** − This queue keeps all the processes in the system.
- **Ready queue** − This queue keeps a set of all processes residing in main memory, ready and waiting to execute. A new process is always put in this queue.
- **Device queues** − The processes which are blocked due to unavailability of an I/O device constitute this queue.

Space for learners:

5.3.1.2 Process State Transition Diagram



PROCESS STATE TRANSITION DIAGRAM

1. **Running:** The process in execution by the CPU.
2. **Waiting/Blocked:** When a process needs some I/O operation or waiting for some other event to happen, it goes into waiting/blocked state.
3. **Ready:** A process that is waiting to be executed is placed in the ready queue.
4. **New:** The process just being created is in new state. The process will be in new state until long-term scheduler brings it to the ready state.
5. **Terminated/Exit:** A process that is finished its execution or aborted due to some reason.

STOP TO CONSIDER

Process Scheduling is mainly use to Schedule the process. Many processes waiting in the ready queue are assigned to CPU one by one according to scheduling algorithm. A PCB is also called Process Descriptor. Whenever a new process is created, operating system creates a PCB which contains information like Pointer, Process state, Process state, list of open files, list of open devices, general purpose registers, priority.

5.3.2 Types of Scheduler

Following are the various scheduler used for scheduling process by the operating system.

1. Long term scheduler

Long term scheduler is responsible for creating processes and to bringing them into the system. It place processes from new state to ready queue. Since it creates processes that are why it is known as Job Scheduler also. It controls the degree of Multiprogramming. It chooses a perfect mix of IO bound and CPU bound processes among the jobs present in the pool. If maximum processes are I/O bound then CPU will remain ideal as state most of the time. This will decrease the degree of multiprogramming. So the job of long term scheduler is very critical and it may affect the system for long time.

2. Short term scheduler

This CPU scheduler is responsible for scheduling one of the processes from ready state to running state. In order to select which job is going to assigned CPU, scheduling algorithm is used.

If the selected job by Short term scheduler is CPU burst time, then for a long time processes have to wait in ready queue and a situation occur which is known as starvation.

3. Medium term scheduler

This medium term scheduler is also known as swapper as it swaps processes from main memory to secondary memory and vice versa. Processes which are IO bound are removed from the running state and placed in the waiting queue so that perfect mixes of processes are in the ready queue. Suspending and resuming of the processes is responsible of this scheduler. Various algorithms are used by operating system for this purpose.

STOP TO CONSIDER

Long term scheduler brings process from Job POOL to the main memory ready for execution. It has access to only job pool and ready queue. Whereas Short term scheduler selects job from ready queue and assigns CPU to that process. It has access to ready queue and CPU.

Space for learners:

Medium term scheduler is used to keep the flow smooth. Sometimes some process reserves memory in ready queue but may do nothing except reserving memory space. Medium term scheduler takes this process out of the memory.

Space for learners:

5.3.3 Scheduling Algorithm

The Purpose of a Scheduling algorithm is:

1. Maximum CPU utilization
2. Fare allocation of CPU
3. Maximum throughput
4. Minimum turnaround time
5. Minimum waiting time
6. Minimum response time

The following algorithms can be used to schedule the jobs:

1. First Come First Serve

It is the simplest algorithm to implement. The process with the minimal arrival time will get the CPU first. The lesser the arrival time, the sooner will the process gets the CPU. It is the non-preemptive type of scheduling.

2. Round Robin

In the Round Robin scheduling algorithm, the OS defines a time quantum (slice). All the processes will get executed in the cyclic way. Each of the process will get the CPU for a small amount of time (called time quantum) and then get back to the ready queue to wait for its next turn. It is a preemptive type of scheduling.

3. Shortest Job First

The job with the shortest burst time will get the CPU first. The lesser the burst time, the sooner will the process get the CPU. It is the non-preemptive type of scheduling.

4. Shortest remaining time first

It is the preemptive form of SJF. In this algorithm, the OS schedules the Job according to the remaining time of the execution.

STOP TO CONSIDER

A Scheduling Algorithm tells us how much each processes will get CPU time. When a high priority process enters in the system, it preempts a low priority process in between and executes the high priority process first. These algorithms are either **non-preemptive or preemptive**. Non-preemptive cannot be preempted until it completes its allotted time, whereas preemptive process can be forcefully removed when a high priority process comes for execution.

FCFS Scheduling

As the name implies, the processes which enters the ready queue will get the CPU first. If the arrival time of the process is lesser then sooner the job will get CPU. It may cause the problem of starvation if the burst time of the currently running process is more and thus the process waiting in the ready queue has to wait for longer time.

Advantages:

1. The logic of this algorithm is very simple; execution will be based on first cum first serve. Very easy to understand and easy to implement algorithm
2. Each and every process gets a chance to execute.

Disadvantages:

1. This algorithm is non-preemptive means the process once gets the CPU will continue till its completion.
2. The problem of starvation may occur, due to non-preemptive nature of this algorithm.
3. The average waiting time is higher as compare to other scheduling algorithms, thus poor performance.

Space for learners:

Example

Let's consider 5 processes with process ID **P0, P1, P2, P3 and P4**. Arrival time of P0 is 0, P1 arrives at 1, P2 arrives at time 2, P3 arrives at time 3 and Process P4 arrival time is 4 in the ready queue. Arrival and Burst time of the processes are given in the following table respectively.

The Turnaround time and the waiting time are as follows-

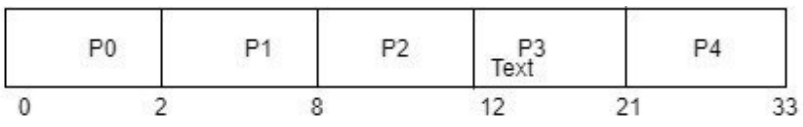
Turn Around Time = Completion Time - Arrival Time

Waiting Time = Turnaround time - Burst Time

The average waiting Time is calculated by adding the respective waiting time of all the processes and divided the sum by the total number of processes.

Process ID	Arrival Time	Burst Time	Completion Time	Turn Around Time	Waiting Time
0	0	2	2	2	0
1	1	6	8	7	1
2	2	4	12	10	6
3	3	9	21	18	9
4	6	12	33	29	17

Average waiting time=31/5



GANTT CHART

Space for learners:

STOP TO CONSIDER

FCFS simply allocates the process according to the arrival time. Process which enters the ready queue first will be allocating the CPU. The problem of starvation may occur if the first process has largest CPU burst among all the process.

CHECK YOUR PROGRESS-I

1. Consider five processes with arrival and burst time given. Calculate average waiting time and turnaround time

PROCESS ID	ARRIVAL TIME	BURST TIME
P1	4	5
P2	6	4
P3	0	3
P4	6	2
P5	5	4

Round Robin Algorithm

In Round Robin algorithm, the processes are dispatched in first in first out (FIFO) manner. Each processes are given limited amount of CPU time for execution in round robin fashion. This limited amount of CPU time is known as quantum time or time-slice or fixed time. If the process does not complete before CPU time expires, the CPU is preempted and given chance to next process waiting in queue.

ADVANTAGES:

1. All processes are given fair treatment.

2. Starvation does not takes place since each and every process given CPU time.

3.Simple and widely used algorithm.

DISADVANTAGES:

1. Determination of time quantum is too critical. If it is too short, it causes frequent context switching and lower CPU efficiency.

Space for learners:

If it is too big, it causes poor response time for short interactive process.

2. Process with long burst time may be starved.

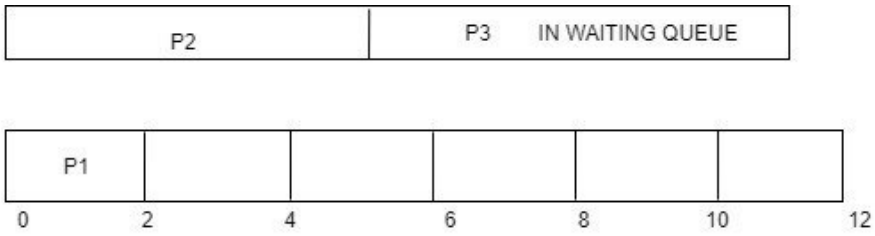
Example of Round-robin Scheduling

Consider this following three processes P1,P2,P3

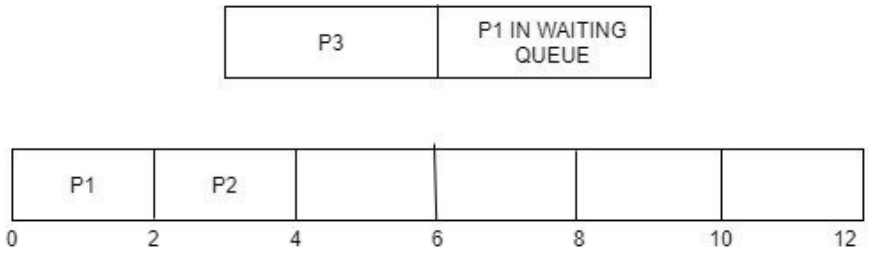
PROCESS QUEUE	BURST TIME
P1	4
P2	3
P3	5

Time slice =2

Step 1) The execution begins with process P1and it has burst time 4. As time slice given is 2 so every process executes for 2 seconds. P2 and P3 are still in the waiting queue.



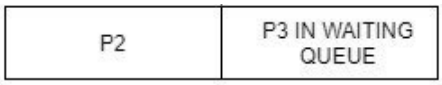
Step 2) At time =2, P2 starts executing and P1 is added to the end of the Queue

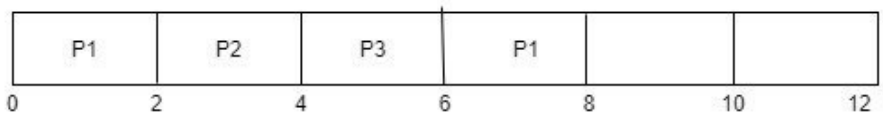


Space for learners:

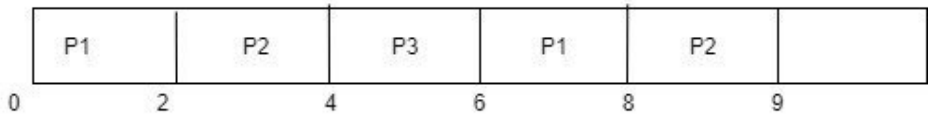
Step 3) At time=4 , P2 is preempted as its burst time is 3 and given time slice is 2 and added at the end of the queue. P3 starts executing.

Step 4) At time=6 , P3 is preempted after time slice of 2 and add at the end of the queue. P1 starts executing.

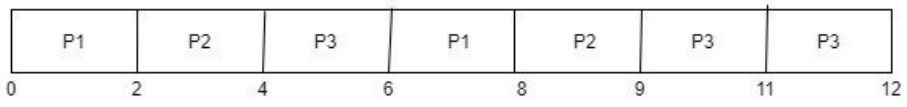




Step 5) At time=8 , P1 has a burst time of 4. It has completed execution. P2 starts execution



Step 6) P2 has a burst time of 3. P2 has already executed for 2 interval. P2 completes execution at time=9. Then, P3 starts execution till it burst time completed.



Step 7)T**he** average waiting time for above example is calculated as

Space for learners:

Wait time

P1= 0+ 4= 4

P2= 2+4= 6

P3= 4+3= 7

CHECK YOUR PROGRESS-II

Q2. Consider five processes P1,P2,P3,P4,P5 with arrival and burst time. Given time quantum=2 units. Calculate Average turnaround time and average waiting time using Round Robin algorithm.

PROCESS ID	ARRIVAL TIME	BURST TIME
P1	0	5
P2	1	3
P3	2	1
P4	3	2
P5	4	3

Shortest Job First Algorithm

Out of all available (waiting) processes, it selects the process with the smallest burst time to execute next. If process with small waiting time occurs frequently then problem of starvation occurs. This can be solved with the concept of ageing. Two version of SJF exist- preemptive or non-preemptive.

Advantages:

- Minimum average waiting time and minimum turnaround time.
- Maximum throughput provided.
- It provides a standard for other algorithms incase of average waiting time.

Disadvantages:

- Processes having larger burst time may face starvation.
- It is difficult to know the length of the upcoming CPU request.
- Requires prior knowledge of how long a process or job will run.

Non-Preemptive SJF:

Space for learners:

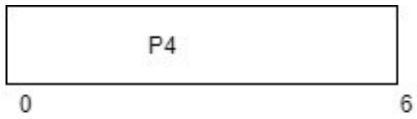
Once the CPU is executing a process then that process can be released only after completing its execution. That is in the middle of execution the process cannot be released.

Space for learners:

PROCESS QUEUE	ARRIVAL TIME	BURST TIME
P1	2	1
P2	1	5
P3	4	1
P4	0	6
P5	2	3

GANTT CHART

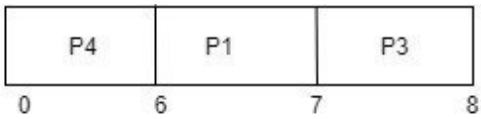
1)At T=0,P4 arrives in ready state, so CPU will be allocated to it. And it will continue till T=6 since forcefully cannot removed it.



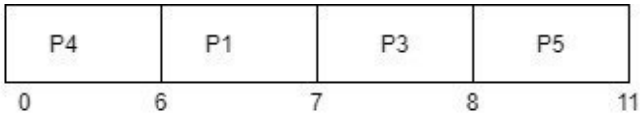
Note that in ready queue all the processes(P1,P2,P3,P5) has arrived within time period 6(which is burst time of P4).

2)At T=6, P1 arrives and it executes till T=7 since it has burst time of 1.

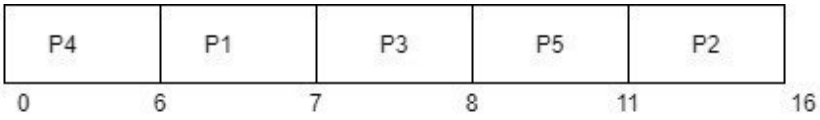
3)At T=7, P3 arrives and it executes till T=8,since it has burst time of T=8



4)At T=8 ,P5 arrives and executes till T=11 since it has burst time T=3.



5) At last T=11,P2 arrives and it executes till T=16 since it has burst time T=5.



At T=16, no more processes are left for execution.

Now lets see Completion time(CT),turn around time(TAT), waiting time(WT) and response time(RT).

PROCESSES	ARRIVAL TIME(AT)	BURST TIME	COMPLETION TIME	TAT=CT-AT	WT=TAT-BT	RESPONSE TIME
P1	2	1	7	5	4	4
P2	1	5	16	15	10	10
P3	4	1	8	4	3	3
P4	0	6	6	6	0	0
P5	2	3	11	9	6	6

Therefore from the above we can easily calculate average turnaround time which is 39/5=7.8 and average waiting time which is 23/5=4.6.

Note: Response time is the time at which CPU responded for first time minus arrival time. Eg for P2=11-1=10

STOP TO CONSIDER

Shortest Job First is a non preemptive algorithm. This algorithm associates with each process the length of the processes next CPU burst. The process having least next CPU burst will be one to get the CPU first. If two processes having same length arrives and waiting for CPU then FCFS scheduling takes place to break the tie.

Space for learners:

CHECK YOUR PROGRESS-III

Q3. Consider five processes with arrival and burst time. Calculate average waiting time. Apply Shortest job scheduling with non-preemption.

PROCESS ID	ARRIVAL TIME	BURST TIME
P1	6	2
P2	2	5
P3	8	1
P4	3	0
P5	4	4

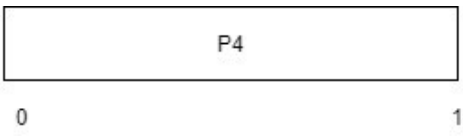
Space for learners:

PREEMPTIVE SHORTEST JOB FIRST (SJF WITH PREEMPTION)

This algorithm is also known as Shortest remaining time first(SRTF).Whenever new process arrives, there may be preemption of the running process. The process can be removed while it is executing before termination of that process. It happens if the newly arrived process has shorter burst time then the currently running process. This algorithm gives optimal solution compared to all other algorithm.

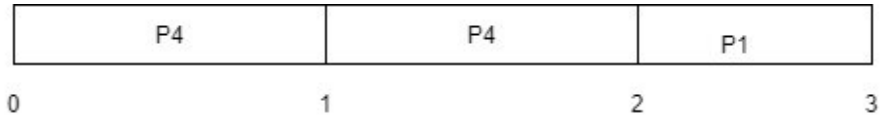
PROCESS QUEUE	ARRIVAL TIME	BURST TIME
P1	2	1
P2	1	5
P3	4	1
P4	0	6
P5	2	3

1)At time T=0,P4 arrived at ready state, so CPU is allocated to P4



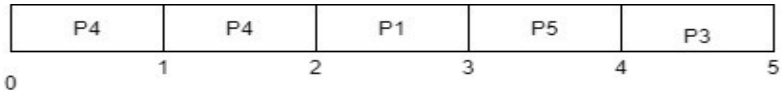
2) At time T=1,P2 arrived but since remaining burst time of P4=5 and burst time of P2=5 is same so no context switching takes place and P4 will continue till next process comes.

3)At T=2, P1 and P5 has arrived but since burst time of P1=1 so CPU is allocated to P1 .

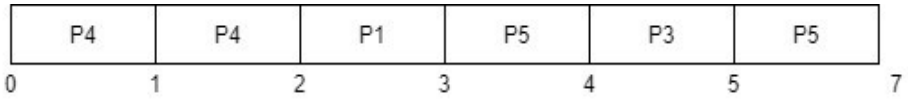


4)At T=3, P5 is allocated since it is in ready queue.

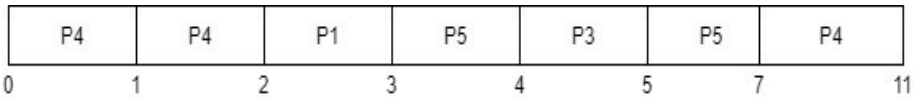
5)At T=4, P3 has arrived and is allocated the CPU.



6) Till now all the processes are arrived in the ready queue, so from this time SJRN will work as same as SJF with non preemption. So at T=5, P5=2 (remaining burst time) is short of all remaining available process , so P5 allocated to CPU.

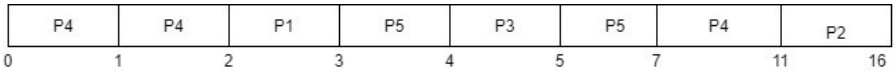


7)At T=7, burst time of P4 =4 and P2=5, since P4 burst time is short so P4 is allocated to CPU.



8)At T=11 , only left process in the ready queue is P2 with burst time P2=5 and is allocated CPU.

Space for learners:



Now lets see Completion time(CT),turn around time(TAT), waiting time(WT) and response time(RT).

PROCESSES	ARRIVAL TIME(AT)	BURST TIME	COMPLETION TIME	TAT=CT-AT	WT=TAT-BT	RESPONSE TIME
P1	2	1	3	1	0	0
P2	1	5	16	15	10	10
P3	4	1	5	1	0	0
P4	0	6	11	11	5	0
P5	2	3	7	5	2	1

Therefore from the above we can easily calculate average turnaround time which is 33/5=6.6 and average waiting time which is 17/5=3.4

STOP TO CONSIDER

In Pre-emptive Shortest Job First Scheduling., when a new process arrives with shorter burst time then currently running process is pre-empted or removed from CPU, and executed process with shorter burst time. Once completes the previous suspended process is executed.

Space for learners:

CHECK YOUR PROGRESS-IV

Q4. Consider five processes with arrival and burst time. Apply SJF with preemption in order to calculate average waiting time.

PROCESS ID	ARRIVAL TIME	BURST TIME
P1	6	2
P2	2	5
P3	8	1
P4	3	0
P5	4	4

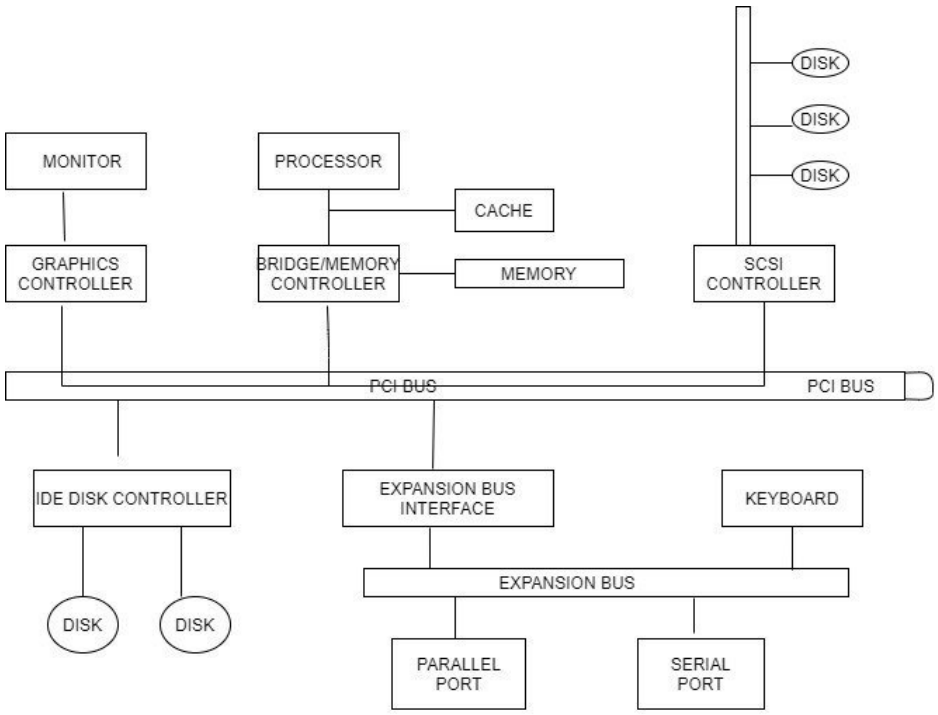
Space for learners:

5.4 I/O SUPERVISORS

5.4.1 I/O Structure

We know without I/O a computer system is useless. I/O is gateway for the outside world. There are thousands of devices, each slightly different from one another. How we will standardize the interfaces to those devices? Some devices provide single byte at a time (i.e. keyboard),other devices provide whole blocks(i.e. disks, networks etc.). Some devices must be accessed sequentially (i.e. tape),other can be accessed randomly (i.e. disk, CD etc.).Some devices require continual monitoring. Others generate the interrupts when they need service. A large portion of operating system code is devoted to managing I/O, both due to its importance to the reliability and performance of a system and since of the varying nature of the devices .A general purpose computer system consists of CPUs and multiple device controllers that are connected through a common bus. Typically, operating systems have a device driver for each device controller. The device driver understands the device controller and presents a uniform interface to the device to the rest of the operating system.

Space for learners:



A TYPICAL PC BUS STRUCTURE

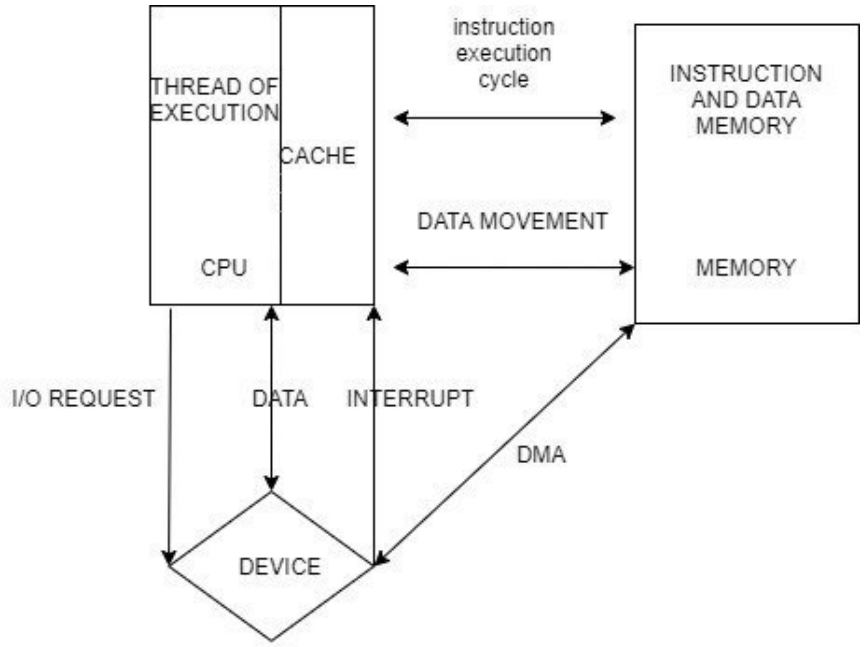
This figure depicts how different I/O devices are connected. Processor is connected with cache memory which is required to store frequently required data and instruction. Processor is connected with bridge/memory controller. Data comes from bridge to memory may be for some I/O devices. It is connected with PCI bus. PCI stands for peripheral component interconnect. This PCI bus is also known as I/O bus because it is dedicated to established communication between the system and I/O devices. Monitor is an output device and is connected via graphics controller. SCSI is a small computer system interface. With SCSI high speed hard disks are connected. SCSI provides SCSI bus and with SCSI, disks are connected. Integrated device electronics (IDE disk) where multiple disk are connected with local dedicated disk buses. Also expansion bus interfaces are available, which provides more port. Other peripheral devices are connected with the help of port. Ports are categorized into two-Serial port in which against each and every clock pulse, one

bit at a time is sent for communication. Parallel port where multiple number of bits can be transferred against each clock.

STOP TO CONSIDER

In modern PC, three out of four bus types commonly found are 1.PCI bus 2.Expansion Bus. 3.SCSI Bus. High speed bandwidth devices connected to the memory subsystem(and the CPU) via **PCI Bus**, Slower low-bandwidth devices which transfers one character at a time are connected by **expansion** bus. SCSI devices are connected to a common SCSI controller via **SCSI bus**.

5.4.2 Working of an I/O Operation



WORKING OF AN I/O OPEARATION

Space for learners:

- When any I/O operation has to be performing by any device, the respective device driver loads appropriate registers within the device controller.
- The device controller in turn examines the contents of these registers because in that registers the information or data about what is the exact I/O operation that has to perform is available.
- Whatever has to perform that data transferred to the local buffer of the device controller.
- Once the transfer of data is complete, the device controller informs the device driver that it's finished its operation via an interrupt.
- The device then returns control to the operating system.
- This form of interrupt-driven I/O is good for transferring small amount of data but its produce overhead when need to transfer large amount of data. In that case Direct Memory Access (DMA) is used. CPU is free to do other work while DMA is used.
- After setting up pointers, buffers and counters for the I/O device, the device controller transfer an entire block of data directly to or from its own buffer storage to memory with no intervention by the CPU.
- In DMA only one interrupt is generated per block to tell the device driver that the operation has completed. CPU perform other task while device controller is performing these operation.

Space for learners:

5.4.3 Direct Memory Access

We know if we involve our CPU to perform read/write operation from the peripheral devices, then actually we are mis-utilising the potentiality of the CPU.CPU performs best if we involves CPU for the I/O device operation. But all the I/O devices are very slow; it cannot get synchronized with CPU. So CPU will have long waiting time. And CPU is not actually meant for this. So concept of DMA

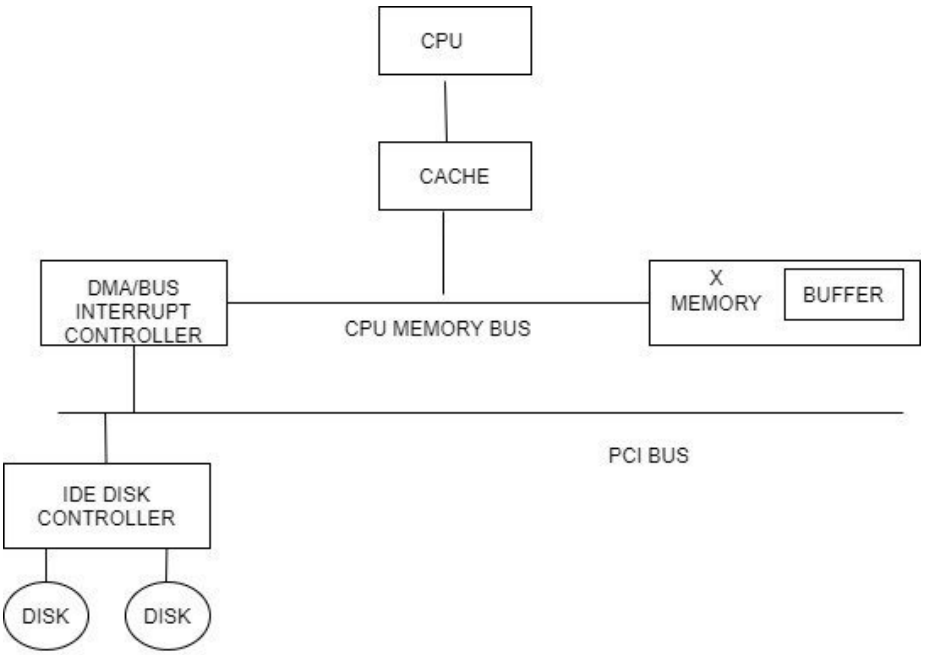
controller has come. DMA controller does the needful data transfer between I/O devices and the memory. DMA controller gets four parameters-

1. Source address from where data is to be read.
2. Target address here this data is to be transferred.
3. Bytes count that is how many data is to be transferred.
4. Whether it is read or writes operation so that it can decide that from which direction data is to be transferred.

So, after getting all this parameter the DMA controller will do the needful data transfer in between I/O device and the memory and this way CPU keeps busy in some other tasks. But it is suggested use of CPU instead of DMA if device speed is fast or if CPU has nothing to do as DMA cost is very high. There are six steps to perform data transfer-

1. Device driver is told to transfer disk data to buffer at address x.
2. Device driver tells disk controller to transfer C bytes from disk to buffer at address x
3. Disk controller initiates the DMA transfer.
4. Disk controller initiates the DMA transfer.
5. DMA controller transfers bytes to buffer x, increasing memory address and decreasing c until c=0.

Space for learners:



DATA TRANSFER USING DMA

6. When c=0,DMA interrupts CPU to signal transfer completion.

5.5 MEMORY MANAGEMENT

We know computer memory is a location of computer system used to store information. It is a very important function of operating system. In order to run any program, that program has to be load in computer memory. So computer memory is important resources to execute a program inside the computer. Therefore we should use this resource to our fullest usage, we should not keep any program in memory which is not executed or required in near future. So only those programs are to be loaded in memory which is demanded in near future. Therefore the concept of memory management comes here.

5.5.1 Binding

A) Address Binding:

1)**Compile time binding**: In this compilation, the absolute address will get embedded in executable code. That means the program knows from

Space for learners:

which address it is supposed to get loaded and where supposed to get executed.

Advantage: It requires minimum set up time.

Disadvantage: If the memory space is occupied by another program, collision takes place. The current program will overwrite the previous existing program in memory.

2) **Load time address binding**: When the program gets compiled then all the addresses will be in re-locatable address. It is **the translation of the logical addresses to physical addresses at the time of loading**. The relocating loader contains the base address in the main memory from where the allocation would begin

3)**Execution time address binding**: The program has got loaded into memory might be executing but during period of time if operating system can move the program to one block of memory to another memory block then it is said to be execution time address binding. The new address will be allocated to the respective program.

B) Dynamic loading:

A program written is divided into number of modules. When the program is in execution, the program must be in main memory. All the modules at the same time are not required to bring in the main memory. Dynamic loading says load the main module at first and load the other module when they are required. That means all the module will not be jumbled up in the main memory, which may or may not be required in future.

C)Overlays:

Whenever assembly language program gets executed, assembler comes into play. Assembles does this in two phase-pass 1 and pass 2. Both the phase may not be required to be loaded in the main memory. In this overlay comes into play. It decides which pass to reside in the memory for execution.

5.5.2 Logical Address Versus Physical Address

The address generated by CPU is called logical address and the address generated by the memory unit is called physical address. Memory

Space for learners:

management unit converts logical address into physical address. Logical address space is a set of all logical address generated by CPU and the physical address space is the set of all physical address generated by the program. Logical address is Virtual and physical address is real.

Space for learners:

5.5.3 Contiguous Memory Allocation

In contiguous memory allocation, each process is contained in single contiguous section of memory. All available memory resides at one place which means unused memory are not scattered here and there across the whole memory space.

Fixed Partitioning

Operating system uses various techniques to manage the memory. The degree of multiprogramming says that keep more and more program in main memory so that whenever CPU needs a process for execution, it is easily available. Main memory must accommodate both operating system and various user processes. The memory is divided into two partitions. One for resident operating system and one for user processes. We want several user processes to reside in memory at the same time.

In contiguous memory allocation, one of the simplest methods is fixed size partitioning. In fixed size partitioning, memory is divided into several fixed size partition. Each partition may contain exactly one process. Degree of multiprogramming depends on the number of partition.

Variable Size Partitioning

In variable size partitioning, initially all memory is available for user processes and is considered one large block of available memory, a hole. Eventually memory contains a set of holes of various sizes. The operating system keeps a table initially which parts of memory are available and which are occupied. The memory blocks available comprise a set of holes of various sizes scattered throughout the main memory.

So whenever new process arrives, the system searches for a hole large enough for that process. If the hole is larger then what is required by the process; in that case the hole is divided into two parts. One part is

allocated to arriving process and other is written to the set of holes. Now when a process terminates, it releases its block of memory and which is then placed back to set of holes. If the new hole is adjacent to other hole, then these adjacent holes are merged to form a larger hole.

Space for learners:

5.5.4 Partitioning Algorithm

Infixed partition, memory partition is fixed, once we allocate a process, the leftover space is unused. For example P0 is assigned to the first hole which is large enough for the process but the space leftover is left unused. Since every partition is allocate to one process only so the process left cannot be allocate to any other process. Thus causes internal fragmentation. So it does not make sense to use various algorithms in Fixed size partition.

The various algorithms for memory allocation for variable size partitioning are:

Here also it creates many holes but the spaces left over can be used for other process. That's why it makes sense to use various algorithm in variable size partition.

1. First fit
2. Next fit
3. Best fit
4. Worst fit

First fit: Allocate the first hole that is big enough. Scanning is done from the beginning in order to find the hole which is big enough to size of the process need to allocate.

Next fit: Here it allocates the first hole that is big enough. It is similar to first fit but scanning is done from the location at which previous search ended. Example say there are two processes. P0 is allocated first hole which is big enough, after that scanning is performed for P1 where last search ended.

Best fit: Allocate the smallest hole that is big enough thus it produces smallest leftover hole. While scanning many holes may be bigger than

the size of process which we are going to load. So minimum of these holes which fits better will be taken.

Worst fit: Allocate the largest hole which is big enough. It produces largest leftover hole which may be more useful than a leftover hole from a best fit approach. After scanning one having the maximum size hole will be taken for loading the correct process.

First Fit:

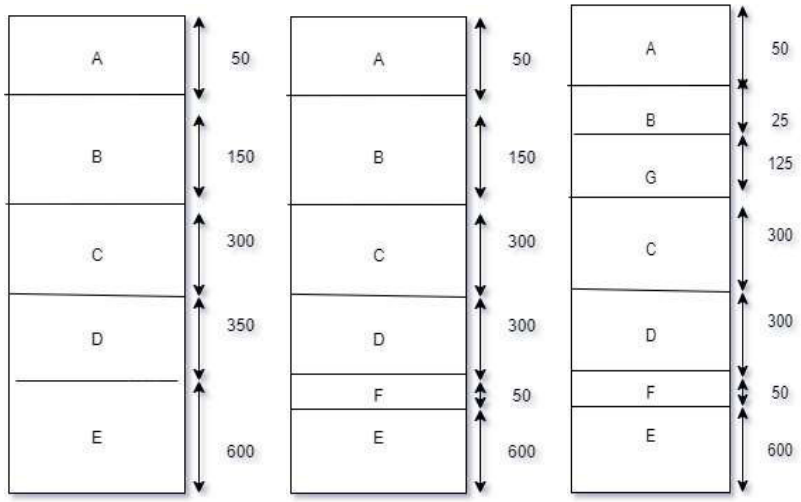


Fig 1 Fig 2 Fig 3

The figure shows memory layout. Consider A, C and E memory space is occupied by some process. B,D are empty and is consider as available memory hole. Now we have four process arriving. First P1 having memory requirement 300, secondP2 having requirement 25, third P3 having 125 and fourth P4 having 50. Now we have to check whether the first fit able to satisfy the entire four requests or not.

So for first process with memory requirement 300, first fit checks the first hole empty from the beginning whether hole is greater than the arriving process size. Starting available space is 150which is smaller then 300, so it searches for next available space i.e. 350(D) in fig 1. Since 350 is larger than 300,process P1 will be allocated to that memory and thus 350-300=50 (F)is new leftover hole(fig 2). Next process size is 25 and starting available hole is 150. Since it is satisfied so P2 is allocate to memory hole i.e. B and thus 150-25=125(G) hole is left in fig 3. Next third arriving process size is 125 and available space is also

Space for learners:

125(G) in fig 3. Thus this memory is occupied. Last process size is 50 and also available hole is of size 50(F) in fig 3. Thus memory is occupied by P4. In this way, first fit works.

Next Fit:

We have to check whether the next fit able to satisfy all the four request or not. First P1 having memory requirement 300,second P2 having requirement 25,third P3 having 125 and fourth P4 having 50.

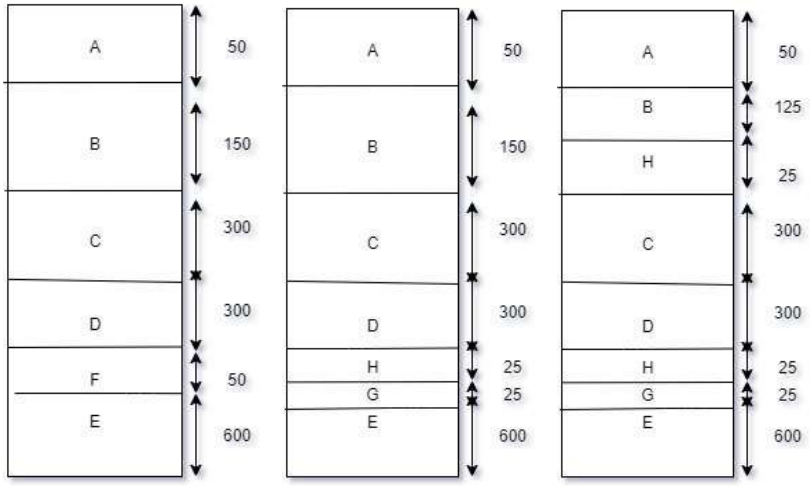


Fig 4 Fig 5 Fig 6

Consider figure 1 , Since hole D is satisfying the size of Process P1 i.e 300,so D will be occupied by Process P1,leaving 350-300=50(F) as leftover hole shown in fig 4. Now next Process is P2 with size 25. Next fit searches from where last searches took place i.e from D. Since F satisfies the process P2 memory size i.e 25.So F will be allocate to process P2 and thus 50-25=25(G) leftover hole available shown in fig 5. Next arriving process is P3 with 125 memory requirement. G is smaller so next fit searches next larger hole. B is satisfying thus P3 will be allocate to hole B and thus 150-125=25(H) leftover hole available as shown in fig 6. For the next process P4 with size 50, available hole is H (25) which is not sufficient and also G(25) which is not sufficient. Thus the last process P4 is not allocated memory by next fit algorithm.

BEST FIT:

Space for learners:

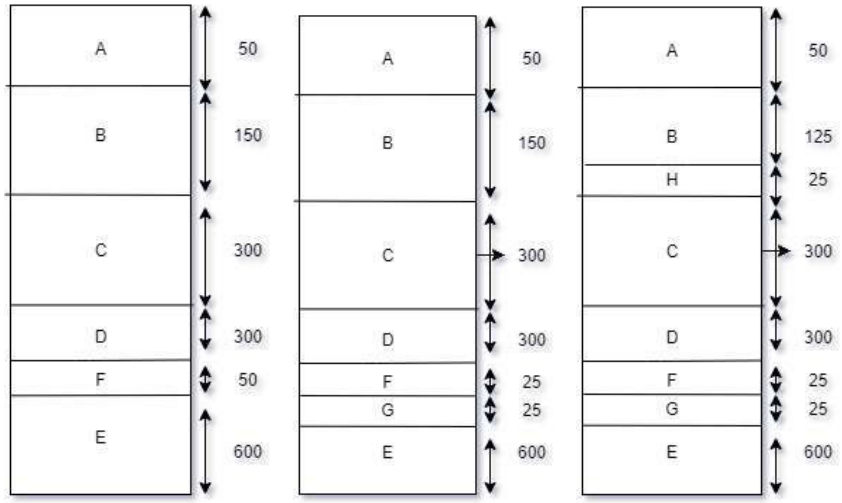


Fig 7 Fig 8 Fig 9

Initially we have two free memory hole B with 150 and D with 350 in fig 1. Consider four same process P1 with 300,P2 with 25,P3 with 125 and P4 with 150. Among this memory hole P1 occupies memory hole D leaving 350-300=50 say F hole as shown in Figure 7. Now the available memory holes are B with 150 and F with 50. Next process P2 has size 25 which both memory holes B and F satisfy. But since best fit searches smallest best hole that satisfies that process so P2 occupies space F leaving 50-25=25 say G as shown in figure 8. Now memory hole B of size 150 and G of size 25 available. Process P3 require 125 memory which is provided by B hole thus 150-125=25 left over hole say H in fig 9. Now, available memory hole are H and G of size 25 each. Process P4 with size 50arrives but none of the holes satisfies P4 request. Two holes if we sum up it gives 50 but since they are not contiguous it is not satisfying P4 request. This is known as **External fragmentation**. So Process P4 is not allocated with memory space.

Worst Fit:

Space for learners:

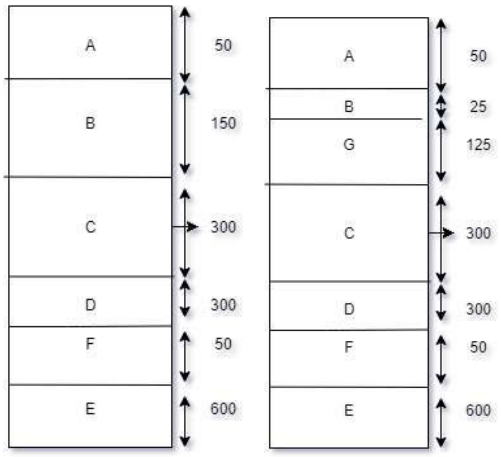


Fig 10 Fig 11

Initially holes of size 150 that is Band D of size 350 available as in fig 1. Process P1 size 300 is satisfied by hole D leaving 350-300=50 say F as shown in fig 10. Next process P2 of size 25 arrives, both hole B and F satisfies the requirement. But worst fit choose the largest hole that satisfies the process. Thus B is occupied by process P2, leaving 150-25=125 say G as shown in fig 11. Process P3 arrives with size 125 and is satisfied by G. Process P4 of size50 is satisfied by F. Thus the entire requests are satisfied by this algorithm.

STOP TO CONSIDER

First fit scans from beginning and chooses first available block which is large enough. Best fit chooses among the free memory partition, the smallest sufficient partition among for a process. Worst fit is just the opposite of best fit, allocates the process in the largest sufficient partition among freely available partition. Next fit search for the first sufficient partition from the last allocation point.

Space for learners:

CHECK YOUR PROGRESS-V

Q5. 100 KB, 500 KB, 200 KB, 450 KB and 600 KB are five memory partitions in the same order. If requests for blocks of size 212 KB, 417 KB, 112 KB and 426 KB in same order comes sequentially; then which of the following algorithm makes the efficient use of memory?

A. Best fit algorithm

B. First fit algorithm

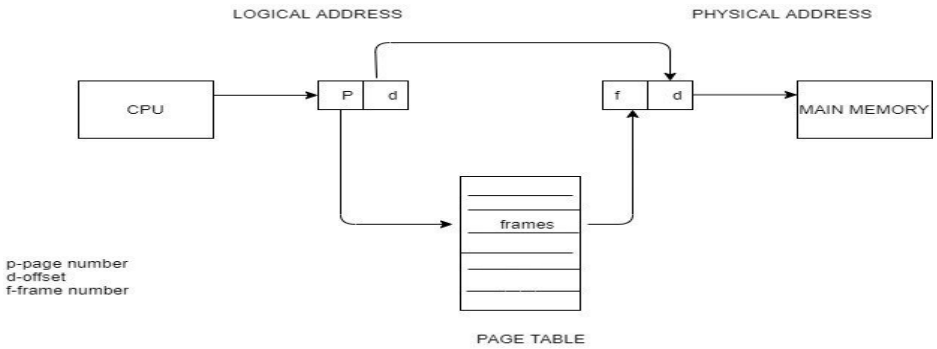
C. Next fit algorithm

D. Both next fit and best fit results in same

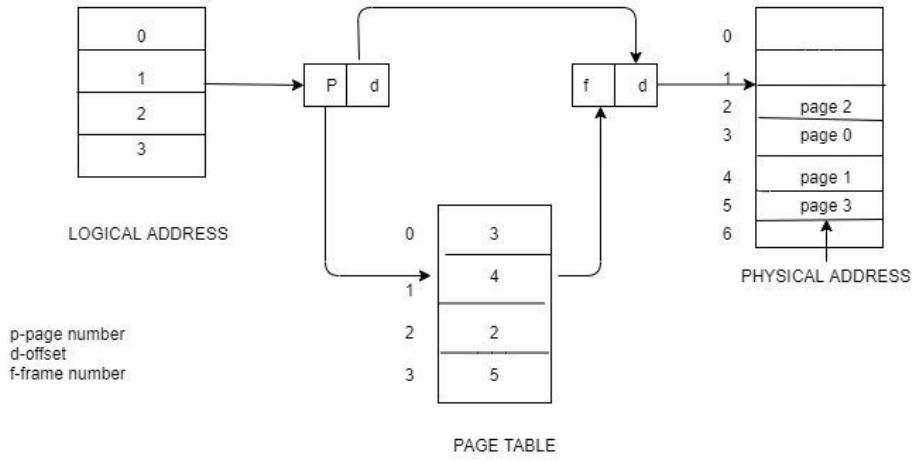
Space for learners:

5.5.5 Paging

Paging is a physical mapping. It avoids external fragmentation. The need of compaction is also reduced here. This is physical mapping in which physical memory is divided into fixed size called frames. The size of each frame is same. The logical memory is divided into fixed size block called pages. The size of each page is same in logical memory. Whenever a process to be executed, the pages are to be loaded into the frames. That means size of page is equal to size of frames. Size of pages and frames are same but number may be different. The logical address space is generated by CPU is divided into parts page number and page offset. By using page number we identify the corresponding frame in physical memory using Page table. The page table contains base address of page in the physical memory. That means in which location Page is loaded in physical memory. The page table contain frame number at which address location the base address of the page resides. Frame number is added to offset by adding simply location of physical memory.



For example, suppose we have logical memory with four pages say page 0,1,2,3. Page table contain page number 0,1,2,3. Again physical memory is supposed divided into 7 frames. In the page table page 0 contains 3,page 1 contain 4,page 2 contain 2 and page 3 contain 5 simultaneously. Page 0 contains 3 means it is the frame number where page 0 will be loaded in physical memory. Page 1 is loaded in frame 4 and so on. This is how paging works. See the diagram below for understanding of above explanation.



Advantages:

1.No external fragmentation takes place

2.Efficient use of memory

3.User's view of memory and actual physical memory are separate. The user view memory as a simple contiguous memory that contains only one process. But the user process is non-contigous in physical memory.

Space for learners:

Disadvantages:

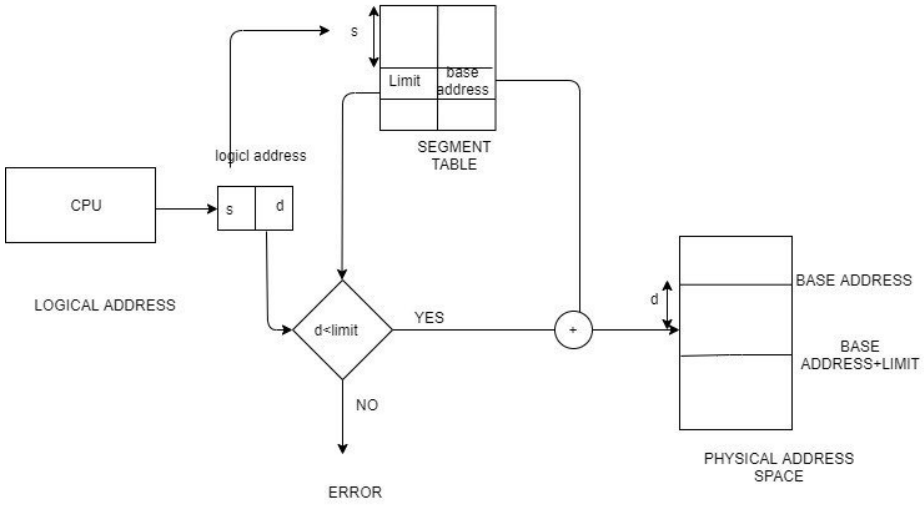
0 It suffers from internal fragmentation.

1 Page table requires extra memory. So it may not be good for a system having small RAM.

STOP TO CONSIDER

It is a fixed-size partitioning scheme**.** The secondary memory and main memory are divided into equal fixed-size partitions in Paging. It helps to avoid external fragmentation**.** The paging technique divides the main memory also called physical memory into fixed-size blocks that are known as Frames and divides the secondary memory also called logical memory into blocks of the same size that are known as Pages.

5.5.6 Segmentation



Translation of 2-dimentional logical address to one dimention Physical Address

Here CPU generates logical address. Logical address is divided into two parts-segment number(s) and offset (d).

Segment number (s): It is the number of bits required to represent the segment.

Space for learners:

Segment offset (d): It is the number of bits required to represent the size of the segment.

Each segment number and offset is stored in the form of segment table. Limit is the length of each segment and base is the starting physical address where segments reside. Each limits and base and each segment is checking whether it is less than equal to limit or not. If it is yes, the physical address will be base address added with the offset. If it is greater than segment table length register then addressing error will occur. This is how hardware implementation for the segmentation takes place

Advantages of Segmentation –

- No Internal fragmentation takes place.
- Segment Table consumes less space in comparison to Page table in paging.

Disadvantage of Segmentation –

- External fragmentation takes place when the processes are loaded and removed from the memory; the free memory space is broken into little pieces.

STOP TO CONSIDER

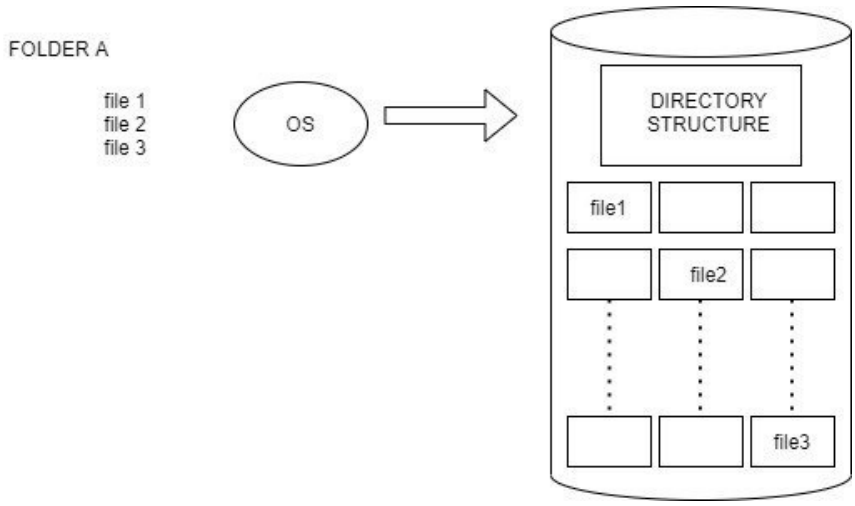
External fragmentation occurs when sufficient quantity of memory is available for a process but that available memory is not contiguous. Internal fragmentation occurs when memory block assigned to process is bigger than process. Some portion of memory is left unused, as it cannot be used by another process. Thus results in internal fragmentation.

5.6 FILE SYSTEM

A file is a named collection of related information recorded on a secondary storage device. File can be anything. It can be document file, audio, video etc. Secondary storage can be anything magnetic tape,

Space for learners:

optical device, hard disk etc. In the computer, file is the basic logical storage unit. According to user perspective everything is kept on a file and the file is stored at a single place on secondary storage device. But in secondary storage device, these files are stored as blocks. These files may occupy a block or may occupy more than a single block and these blocks may not be contagious. Consider an example, user kept file1, file2, file 3 in a folder named A. From user point of view, all the three files are in the same folder but actually these files may occupy different blocks as shown in figure. All the information is maintained in a directory structure which is stored at the beginning of secondary storage device. This directory structure keeps all the information about the files that is where the file exactly located, its address and so on. These collections of files on secondary storage device together with directory structure to manage, organize and to provide all the information together form the file system or file management system task of operating system.

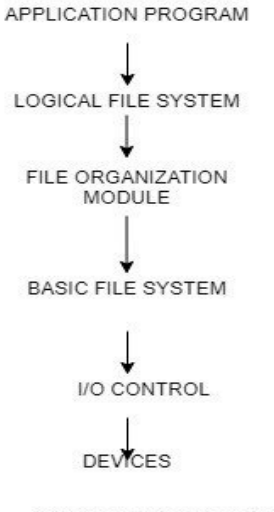


Organization of files in secondary memory

5.6.1 File System Structure

File system is stored generally in secondary storage device such as hard disk. To perform different operations file system is organized into different layers. So the layers are

Space for learners:



LAYERED FILE SYSTEM STRUCTURE

The top layer is application program and the bottom layer is devices. In between logical file system, file organization module, basic file system, I/O control.

Application Program-The program which is developed by user is called application program. That application program is given as input to the logical file system.

Logical file system-It accepts the file name as input and checks whether that file is available in directory structure. If the file is available in directory structure then it finds location of the file as well as logical block member of that file.

File organization module-That logical block member will be given as input to the file organization module and it performs a mapping in order to find physical block member in which location file is stored in hard disk. The task of this module is to find physical member block of the corresponding logical member.

Basic file system-Physical block member will be input to the basic file system. Basic file system issues a command to I/O control with the help of block member.

I/O control-I/O control accepts the command given by basic file system. Every I/O control contains device drivers. It is duty of device drivers to work with device. Device driver takes responsibility of interacting with

Space for learners:

devices so that corresponding operation takes place. This is how the layered structure of file system works.

Space for learners:

5.6.2 Operations on File

The various operations which can be implemented on a file such as read, write, open and close etc. are called file operations. Some common operations are as follows:

1. Creating a file-
   - Space for newly created file must be found in the file system.
   - After that directory must have entry for the newly created file.
2. Open operation-
   - Once file is created, it has to open in order to perform any operations.
3. Writing a file-
   - Make a system call specifying both the name and the info to be written to the file.
   - A write pointer must be maintained by the system to the location where next write should take place.
4. Reading a file-
   - Use a system call that specifies the name of the file and where in memory the next block of the file should put.
   - The read pointer is updated, once the read operation taken place.
5. Repositioning or Seek operation –
   - The seek system call reposition the file pointer to particular position in the file. Movement may be forward or backward depending on requirement of the user.
6. Delete operation-

- Delete operation not only remove contents of file but also remove the file from disk in order to freed memory space occupied by it.

7. Truncate operation-
   - This operation no doubt deleted the contents of a file but the file is not deleted completely.
8. Close operation-
   - After performing all operation ,it is suggested to close the file so that the changes made are saved permanently and also resources used by the file releases.
9. Append operations-
   - This operation add data to the end of the file.
10. Rename operation-
    - This operation is used to rename the existing file.

Space for learners:

5.6.3 Access Methods

There are three methods to access the files

1. Sequential access
2. Direct access or relative access or random access
3. Indexed sequential access

Sequential access:

It is the simplest method of all the three method. As the name implies information of the file are accessed one by one sequentially. For example let's say a file contains 100 records and the file pointer is in 50^{th} record and we need 75^{th} record to access. Here the file pointer moves one by one from 50^{th}record sequentially in order to read 75^{th} record. Thus it is not possible to access the 75^{th} record directly. Magnetic tape is an example where files are accessed sequentially.

Operations are:

Read next// It reads the next instruction pointed by pointer

Write next// It writes the instruction in the next position

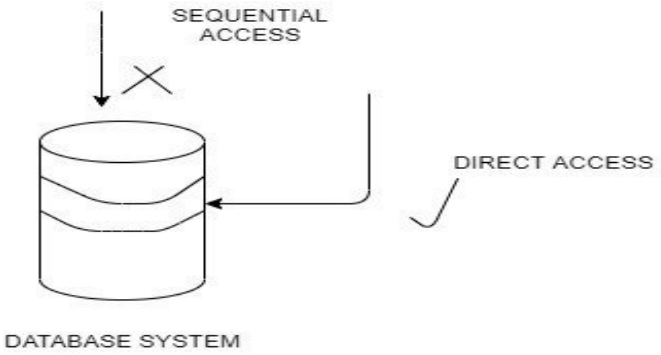
Reset (or) rewind//moves the pointer to the beginning

Advantage: Simplest of all the three method.

Disadvantage: Time consuming since need to access each record sequentially.

Direct access or relative access or random access:

By using this technique we can access the record directly. For the previous example, using this technique it is possible to access the 75th record directly without accessing one by one record sequentially. Example Hard disk, magnetic are random access device.



Let's see the operation of random access

Read n// read the instruction at position n pointed by pointer

Write n//write content in the location n pointed by pointer.

Position to n//file pointer can be move to any location say n here

Advantages: Pointer can be move directly to particular position in order to access.

Disadvantage: Implementation of direct-access systems is often difficult because of the complexity and the high level of programming (software) support that such systems need. In addition, the cost of developing and maintaining is greater than the expense of a sequential processing system.

3. Indexed sequential access:

It is used mainly in order to remove drawback of sequential access. Indexing is a data structure technique which is used to quickly locate

Space for learners:

and access the data in database. From the name itself it is saying that index are used in sequential order to access the data from the database. It is a static index structure in order for creating, maintain and manipulating files of data. Here records can be retrieved sequentially or randomly by one or more keys. It is an advanced sequential file organization.

Advantage: Searching a data in large database is very easy and quick using this technique.

Disadvantage: Extra space in the memory is required in order to store the index value.

STOP TO CONSIDER

Sequential access allows file to read/write sequentially up to the location where it is attempting to read or write. In direct access the records does not need to be in sequence because they are updated directly and rewritten back in the same location directly. In Index sequential index are used in sequential order to access the data from database.

5.7 SUMMING UP

- Scheduling algorithm is use in order to schedule the process.
- Four types of scheduling algorithm are mainly discussed here. They are round robin algorithm , First cum first serve algorithm , Shortest job first ,shortest remaining time first.
- How memory is allocated to different process are explained using the concept of paging and segmentation.
- Different memory partition algorithm like first fit, next fit, best fit and worst fit are explained using example.
- Concept of file along with file structure, file access methods are explained very clearly.

Space for learners:

5.8 ANSWERS TO CHECK YOUR PROGRESS

1. Average turnaround time=8 units

   Average waiting time=4.4 units

2. Average turnaround time=8.6 units

   Average waiting time=5.8 units

3. Average waiting time(non-preemption)=5.2 units

4.Average waiting time(preemption)=4.6 units

5. D

5.9 POSSIBLE QUESTIONS

1. Among all memory management techniques …….. is simple to implement little operating system overhead.

2. Write difference between fixed and variable partition.

3. What is contiguous and non-contiguous memory allocation. Explain with example.

4. What is the function of DMA? Explain with diagram.

5. Why DMA data transfer is necessary?

6. What are job queues, ready queues and device queues?

8. What are the benefits of multiprogramming?

9. What is PCB? What are the information associate with it?

10. Explain FCFS, SJF with example.

11. Explain the concept of pages, frames. What is physical and logical memory?

12. Define swapping.

13. What do you mean by compaction?

14. Process Burst time P1 10, P2 29, P3 3, P4 7,and P5 12 given for five different processes in milliseconds. Consider the First come First serve (FCFS), Non Preemptive Shortest Job First (SJF), Round Robin (RR) (quantum=10ms) scheduling algorithms. Calculate average

Space for learners:

waiting time and turn around time. Illustrate the scheduling using Gantt chart.

15. Write about various scheduling algorithm.

5.10 REFERENCES & SUGGESTED READINGS

- Operating System Concept by Abraham Silberschatz,Peter Baer Galvin,Greg Gagne
- Operating System Principles (Internal and design principles) by William Stallings
- Modern Operating Systems by Andrew S Tanenbaum/ Herbert BOS
- Operating System Concept by Willey

Space for learners:

UNIT 6: SECONDARY STORAGE MANAGEMENT

Unit Structure:

6.1 Introduction

6.2 Unit Objectives

6.3 Mass Storage Structure

6.4 Disk Structure

6.5 Disk Scheduling Algorithm

6.6 Swap Space Management

6.7 RAID Structure

6.8 Stable Storage

6.9 Tertiary Storage Structure.

6.10 Summing Up

6.11 Answers to Check Your Progress

6.12 Possible Questions

6.13 References &Suggested Readings

Space for learners:

6.1 INTRODUCTION

This unit gives an overview of secondary storage management. The secondary storage means non-volatile memory management. The unit explains the disk structure along with the different disk scheduling algorithms. The swap space management is also discussed in the unit. The Redundant Arrays of Independent Disks (RAID) architectures are also discussed in the unit. The stable-storage implementation is also highlighted in the unit along with the outcomes of the disk. The tertiary storage consists of high-capacity data achieves and it is discussed nicely in the unit.

6.2 UNIT OBJECTIVES

After going through this unit, you will be able to

- Understand about the mass storage structure.
- Learn about the disk Structure

- Learn about the different disk scheduling algorithm
- Understand about the swap space management
- Understand about the RAID structure
- Explain the stable storage and tertiary storage

Space for learners:

6.3 MASS STORAGE STRUCTURE

The secondary storage is those where the memory is non-volatile. It means that the data will be intact with the device even if the device or system turns off. The secondary storage structure is auxiliary storage and is less expensive but has less speed than primary storage. Non-frequent data are saved in the secondary storage. Examples of secondary steerages are magnetic disk, pen drive, HDD, etc.

Mass storage refers to the storage of large amounts of data. The data are stored in persisting and machine-readable form. The mass storage device includes tape, RAID system, HDD, magnetic tape, optical disk, memory cards, and SSD, etc. The mass storage doesn't include the RAM. There are two types of the mass storage structure. The first one is local data storage which is a Smartphone or local computer. The other one is global data storage which includes servers, data centres, cloud, etc.

The mass storage device is characterized by:

i) Sustainable speed of the device
ii) Seek time of the device
iii) Cost of the device
iv) The capacity of the device.

6.4 DISK STRUCTURE

The modern disk structure contains tracks and each sector contains multiple sectors. The disks are arranged in a 1-D array of blocks. These blocks are the storage unit of the disk structure which is known as the sector. For each surface, a read-write desk is available in the disk structure. The tracks on the surfaces are known as a cylinder. The disk has the basic following structure.

i) The disks are in the form of platters that are covered with magnetic media. The hard disk platters are metal whereas the floppy disk platter is plastics.

ii) Every platter has two working surfaces and each working surface has some rings called tracks. The tracks which are in the same distance from the edge of the platter is known as a cylinder.

iii) Each track is further divided into sectors. The sector contains 512 bytes of data. Some sector uses larger sector size. Each sector contains a header and trailer.

iv) The data on a hard drive is read by read-write heads. In standard, one head is reserved per surface, each on a separate arm., controlled by arm assembly.

v) The storage capacity of a disk is equal to the number of heads or number of bytes per sector.

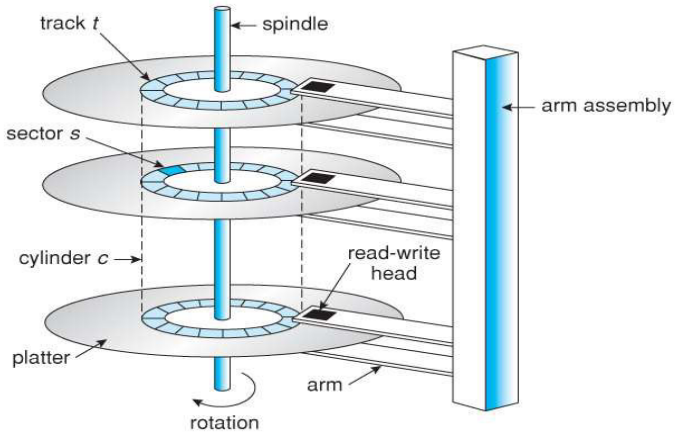


Fig.6.1 Disk Structure

CHECK YOUR PROGRESS-I

1. True or False
   i) Pen drive is a mass storage device
   ii) Track of disk surface is known as cylinder.
   iii) Hard disk platters are plastics.
2. What is a sector?
3. What is the data size of a sector?
4. What do you mean by the storage capacity of disk structure?

Space for learners:

6.5 DISK SCHEDULING ALGORITHM

Disk scheduling is a process where the operating system schedules the I/O requests and it is also known as I/O scheduling. Disk scheduling is important because to manage the multiple I/O requests, this may arrive from a different process. But only one I/O request can be served at a time by the disk controller. In this situation, other I/O request needs to wait in the waiting queue.

The types of disk scheduling algorithms are:

i) First Come First Serve (FCFS) Algorithm

ii) Shortest Seek Time First Scheduling

iii) SCAN Scheduling Algorithm

iv) C-Scan Scheduling Algorithm

Before discussing disk scheduling, you should learn the following parameters.

i) **Seek Time**: It is time to locate the disk arm to read or write data.

ii) **Rotational Latency:** It is the time taken by the sector to rotate itself into a position to access the read and write heads.

iii) **Transfer Time**: It is the time to transfer the information depending on the speed of the disk.

iv) Disk Access Time: It is the combination of seek time + rotational latency + transfer time.

v) **Disk Response Time:** It is the average request waiting time to perform its I/O operation.

6.5.1 FCFS Disk Scheduling Algorithm

The First Come First Serve (FCFS) is the simplest disk scheduling among all. In this algorithm, the first request is always served first in the disk queue. Though there is no starvation in the algorithm it does not provide the fastest service. Let understand the FCFS disk scheduling algorithm with the following examples. Let's you have following order of request. Request = {70,3, 2, 40,4,6,90} and the position of the read and write head is 5 and the total number of the track is 100.

Space for learners:

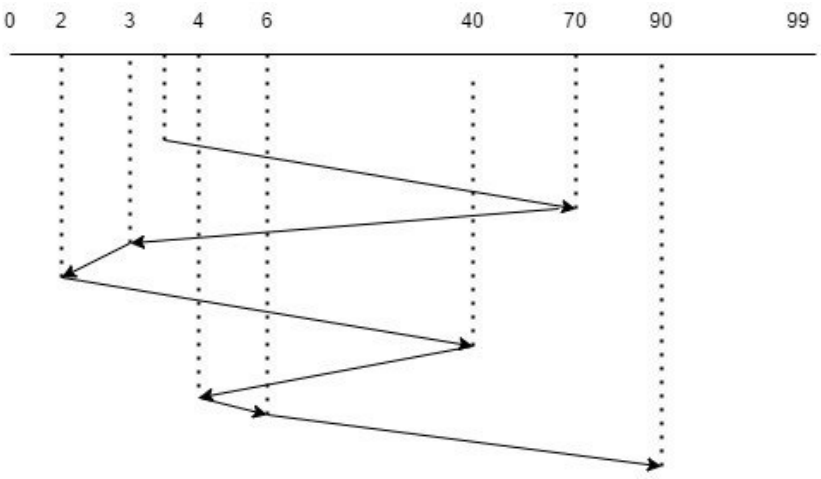


Fig. 6.2 FCFS Disk Scheduling

The current position of the read and write head is 5. So, we start at 5. As the algorithm is FCFS, the first move is towards 70. Then 3, 2, 40, 4, 6, and finally 90. So, the total cylinder move (seek time) is:

seek time = (70-5) + (70-3) + (3-2) + (40-2) + (40-4) + (6-4) + (90-6) = 303

In this algorithm, every request gets a reasonable chance. But it does not have any technique for optimization of seek time.

6.5.3 SSTF Disk Scheduling Algorithm

In Shortest Seek Time First (SSTF) algorithm, the request which has less seek time execute first. So, the average response time is decreased in SSTF. It increases the throughput of the scheduling. But there is a chance of starvation in SSTF. Let's understand the algorithm by taking the following examples. Let's you have a disk that contains 100 tracks (0-100). The requests are with the track numbers70, 3, 2, 40, 4, 90, respectively. The current position of the read/write head is 5. As we need the seek time for this algorithm, so the seek times of the heads are as follows.

i) If you will from 5 to 4, then it will give you the shortest seek time among all the requests and that is (5-4) = 1.

ii) Then from 4, the head will move 3.

iii) Then it will move to 2. From 2, it will move to 40, 70, and finally 90.

Space for learners:

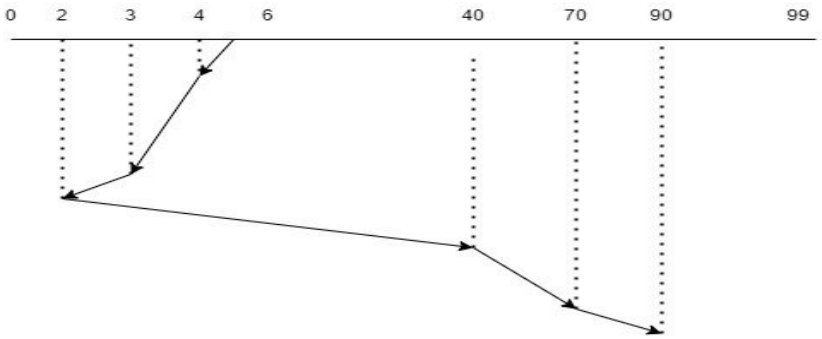


Fig. 6.3 SSTF Disk Scheduling

So, seek time = (5-2) + (90-2) = 91

6.5.3 SCAN Disk Scheduling Algorithm

In the SCAN disk scheduling algorithm, we move the head either to the smaller value or to the larger value. In the moving path, each request is addressed. When the disk arm reaches to end, it will move towards reverse and all the requests are addressed. Let's you have a disk that contains 100 tracks (0-100). The requests are with the track numbers 70, 3, 2, 40, 4, 90, respectively. The current position of the read/write head is 5.Let's the head will move towards the larger end and it will execute as follows.

i) As head movement is towards the larger value, so it will move in the right direction from 5. From 5 it will reach 40, 70, and then 90.

ii) From 90, it will move to 4, 3, and finally 2.

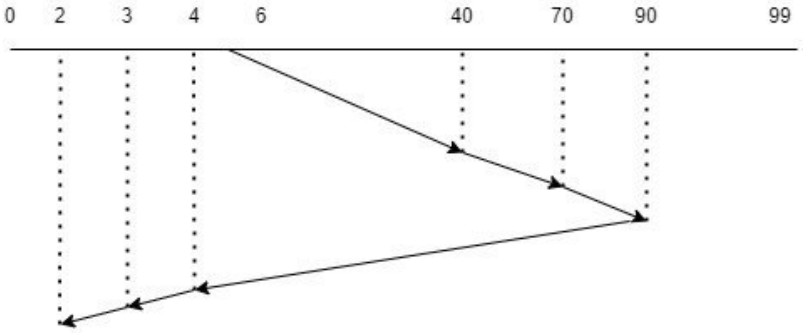


Fig. 6.4 SCAN Disk Scheduling

So, seek time = (90-5) + (90-2) = 173

Space for learners:

6.5.4 C- SCAN Disk Scheduling Algorithm

In the C-SCAN algorithm, the disk moves in a particular direction serving the requests to the end of the direction, and then comes back to the reverse direction until the end. From that end, only the arm starts moving with serving the remaining requests. Let's you have a disk that contains 100 tracks (0-100). The requests are with the track numbers 70, 3, 2, 40, 4, 90, respectively. The current position of the read/write head is 5.Let's the head will move towards the larger end and it will execute as follows.

i) Let's the arm move in the right direction. So its first move from 5 to 40, 70, the, 90 and finally it will go to 99.

ii) From 99, it will directly come to the reverse end, i.e. to the left side end 0.

iii) From 0 it will go to 2, 3, and 4.

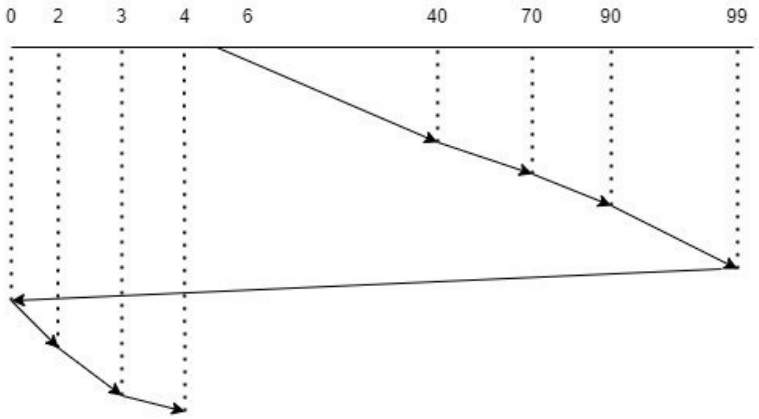


Fig. 6.5 C – SCAN Disk Scheduling

So, seek time = (99-5) + (99-0) + (4-0) = 19

CHECK YOUR PROGRESS-II

5. What is a FCFS disk scheduling?
6. Assuming that the disk head is located initially at 32, Find the number of disk moves required with FCFS if the disk queue of I/O block requests are 98, 37, 14, 124, 65, 67 .
7. Fill in the blanks
   i) The set of tracks that are at one arm position make up a _____.
   ii) The time taken to move the disk arm to the desired cylinder is called the _____

Space for learners:

6.6 SWAP SPACE MANAGEMENT

Space for learners:

Swapping in memory management means swapping of the process so that the maximum number of processes sharing the CPU. It is used multiprogramming. It is a memory management technique used to remove a process from the main memory to secondary memory and then bring it back to the main memory. These are known as swap out and swap in. The area on the disk where the swapped-out processes are stored is called swap space.

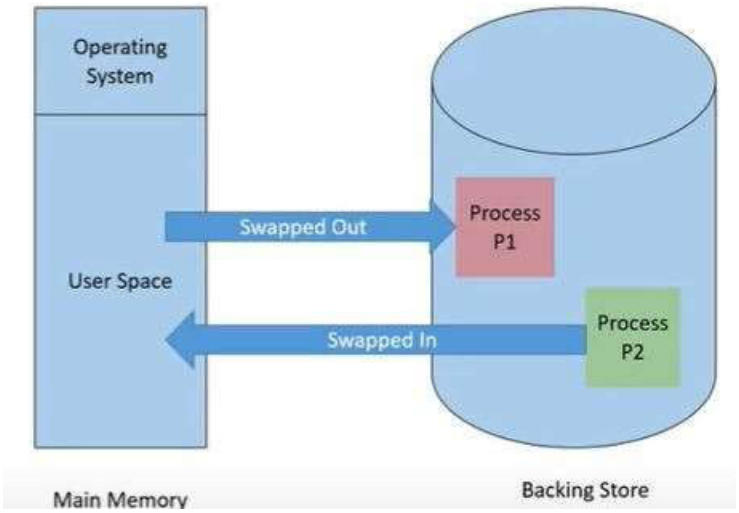


Fig. 6.6: Example of Swap space management

In Fig. 6.6, the process p1 is swapped out from the main memory to swap space, and process p2 is swap in the main memory. So this process is Swap in and out.

Swap space management is another low-level task of the operating system because it deals with disk space. As mentioned above, the process is out from main memory to secondary memory, i.e., disk space. So, disk space is required in swap space management. But the disk space is slower than the memory access. So it will reduce system performance. So, the goal of swap space management is to introduce virtual memory for better throughput.

Swap spaces are variously used by different operating systems. The swap space may contain the entire system or process images that are currently not in use or loaded in RAM. So it is using paging techniques to manage the space. The size of swap space may vary from megabytes to gigabytes. The amount of swap space needed on a system can vary depending on the amount of physical memory, the

amount of virtual memory it is backing, and how the virtual memory is used.

Swap space can reside in two ways.

i) Normal File system

ii) Separate Disk Partition

In the Normal File system, swap space may create by using the normal file system routine. In this process, external fragmentation can increase the swapping times. Otherwise, swap space may create by using a raw partition. No presence of a file system is found here. Here, algorithms are used to increase the speed of the swapping rather than storage. So, it may increase the internal fragmentation of the system.

Space for learners:

6.7 RAID STRUCTURE

The performance of a single disk is less as compared to the multiple disks. The Redundant Arrays of Independent Disks (RAID) is a technique where multiple disks are combined to form a single disk that increases the performance of the system along with the data redundancy. Though the data redundancy takes extra space it increases the reliability of the system. If a disk fails and the data is backed up in another disk, then at the time of disk failure, the information will not lose. You can perform the disk operation. A RAID system is evaluated using the following parameters.

i) Reliability: It denotes the number of disk failures, but the RAID performance will not reduce.

ii) Availability: It denotes the available time, the system for actual use.

iii) Performance: It denotes the response time and throughput of the RAID system.

iv) Capacity: It denotes the overall capacity of the RAID in terms of the number of disks per block.

A RAID structure is transparent. It appears as a big disk for the user. It has the following levels.

i) RAID 0: In this type, the blocks are “striped” across disks without any mirror and parity. Minimum two blocks are present in the RAID 0. As blocks are stripped, the performance of the

RAID in increasing as compared to the normal disk. As the system is very simple, it can not be used for a complex system.

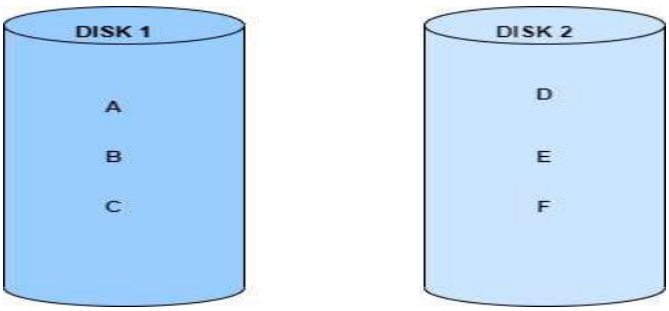


Fig. 6.7: RAID 0.

ii) RAID 1: In this level of RAID, the blocks are mirrors without stripe and parity. Here also, a minimum of two blocks are required for the implementation. Due to no striping and parity, the performance of the RAID 1 is more than the RAID 0. As blocks are mirrored, excellent redundancy is achieved in RAID 1.

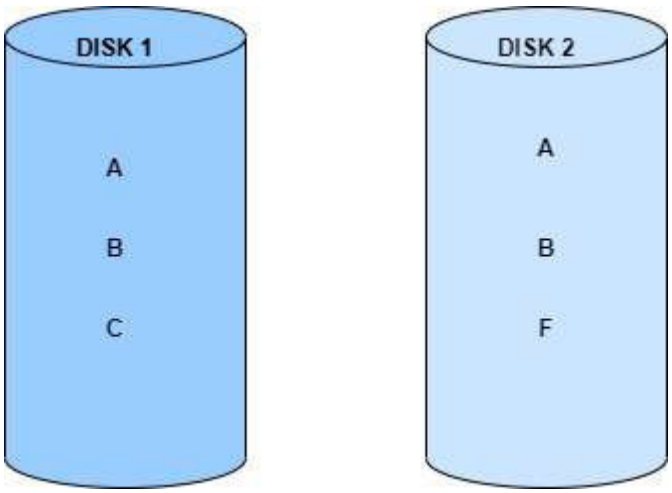


Fig. 6.8 RAID 1

iii) RAID 5: In this level of RAID, the blocks are striped and distributed parity is used in the RAID 5. Minimum 3 blocks are used in RAID 5. As the blocks are stripped, the performance of the RAID level 5 is increasing. The RAID 5 achieved good redundancy due to the distributed parity. In RAID 5, the best cost-effective option is provided using both performance and redundancy.

Space for learners:

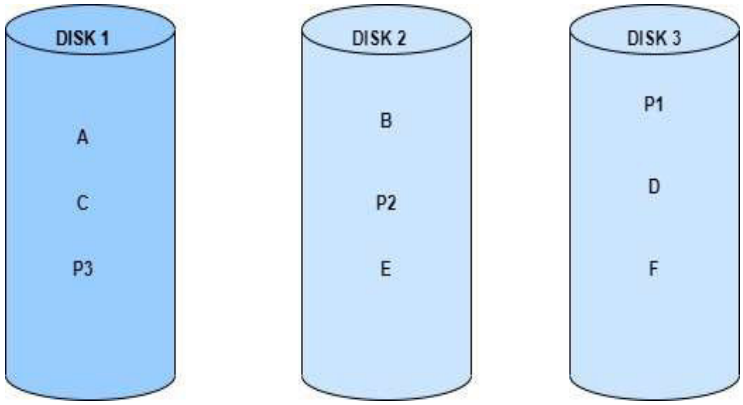


Fig. 6.9: RAID 5

iv) RAID 10: In this level of RAID, the blocks are striped and mirrored. This is also called a strip of the mirror. Minimum 4 blocks are used in RAID 5. Excellent performance and redundancy are observed in RAID 10 due to the stripped and mirror.

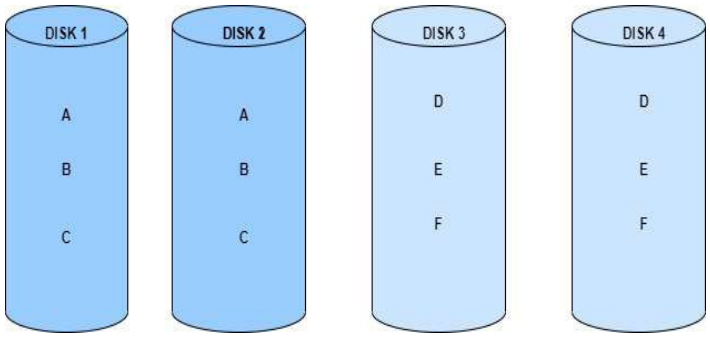


Fig. 6.10: RAID 10

CHECK YOUR PROGRESS-III

8. What are the level of RAID?
9. Can we use RAID 0 for complex system?
10. What is RAID 10?
11. Which RAID type doesn't use parity for data protection?
12. What is the minimum number of disks required for RAID1?

Space for learners:

6.8 STABLE STORAGE

Stable storage means no data loss even if the disk of the computer system fails. It is a computer data storage technology that guarantees atomicity for any read-write operation. To implement stable storage, replication of data on different devices is required. It will help to recover a copy of the data even if the data is removed from some devices.

The causes of the system or device failure are defined below.

i) System Crashes
ii) User Error
iii) Carelessness
iv) Sabotage (intentional corruption of data)
v) Statement Failure
vi) Application software errors
vii) Network Failure
viii) Media Failure
ix) Natural Physical Disasters

6.9 TERTIARY STORAGE STRUCTURE

Tertiary storage consists of high-capacity data archives using vast numbers of removable media, such as tapes or optical discs. Tertiary storage or tertiary memory is a level below secondary storage. It involves a robotic mechanism that will mount (insert) and dismount removable mass storage media into a storage device according to the system's demands; such data are often copied to secondary storage before use.

Figure 6.11: Tertiary storage platforms: (A) Quantum tape library, (B) BluRay optical jukebox [**Image Source**: https://www.sciencedirect.com/topics/computer-science/tertiary-storage]

Space for learners:

The main objective of tertiary storage is to provide hug storage at a low cost in terms of magnetic tapes, optical disks, and optical tapes. They are consisting of fixed storage drives and removable media units. The storage drives are fixed to the computer system but the removable media can be removed to increase the storage capacity by increasing the media units. When data on a media are accessed, the media unit is accessed from its normal location and one storage drive is chosen from the local computer. If there is a media unit in the storage system, the old storage system is unloaded and ejected so that the new media unit can load in the drive. Each storage drive handles the driver and unit efficiently.

Space for learners:

6.10 SUMMING UP

- The secondary storage is those where the memory is non-volatile. It means that the data will be intact with the device even if the device or system turns off.
- The secondary storage structure is auxiliary storage and is less expensive but has less speed than primary storage.
- The mass storage device is characterized by:
- Sustainable speed of the device
- Seek time of the device
- Cost of the device
- The capacity of the device.
- The disks are arranged in a 1-D array of blocks. These blocks are the storage unit of the disk structure which is known as the sector.
- The disks are in the form of platters that are covered with magnetic media.
- The hard disk platters are metal whereas the floppy disk platter is plastics.
- Every platter has two working surfaces and each working surface has some rings called tracks. The tracks which are in the same distance from the edge of the platter is known as a cylinder.

- Each track is further divided into sectors. The sector contains 512 bytes of data. Some sector uses larger sector size. Each sector contains a header and trailer.
- Disk scheduling is a process where the operating system schedules the I/O requests and it is also known as I/O scheduling.
- Disk scheduling is important because to manage the multiple I/O requests, this may arrive from a different process.
- The types of disk scheduling algorithms are:
    - First Come First Serve (FCFS) Algorithm
    - Shortest Seek Time First Scheduling
    - SCAN Scheduling Algorithm
    - C-Scan Scheduling Algorithm
- The First Come First Serve (FCFS) is the simplest disk scheduling among all. In this algorithm, the first request is always served first in the disk queue.
- In Shortest Seek Time First (SSTF) algorithm, the request which has less seek time execute first.
- In the SCAN disk scheduling algorithm, we move the head either to the smaller value or to the larger value. In the moving path, each request is addressed. When the disk arm reaches to end, it will move towards reverse and all the requests are addressed.
- In the C-SCAN algorithm, the disk moves in a particular direction serving the requests to the end of the direction, and then comes back to the reverse direction until the end. From that end, only the arm starts moving with serving the remaining requests.
- Swapping in memory management means swapping of the process so that the maximum number of processes sharing the CPU.
- Swap space can reside in two ways.
    - Normal File system
    - Separate Disk Partition

Space for learners:

- The Redundant Arrays of Independent Disks (RAID) is a technique where multiple disks are combined to form a single disk that increases the performance of the system along with the data redundancy.
- The RAID has the following levels, RAID 0, RAID 1, RAID 5, and RAID 10.
- Stable storage means no data loss even if the disk of the computer system fails. It is a computer data storage technology that guarantees atomicity for any read-write operation.
- Tertiary storage consists of high-capacity data archives using vast numbers of removable media, such as tapes or optical discs.

Space for learners:

6.11 ANSWERS TO CHECK YOUR PROGRESS

1. i) True ii) True iii) False
2. The modern disk structure contains tracks and each sector contains multiple sectors. The disks are arranged in a 1-D array of blocks. These blocks are the storage unit of the disk structure which is known as the sector.
3. The data size of a sector is 512 bytes.
4. The storage capacity of a disk is equal to the number of heads or number of bytes per sector.
5. The First Come First Serve (FCFS) is the simplest disk scheduling among all. In this algorithm, the first request is always served first in the disk queue. Though there is no starvation in the algorithm it does not provide the fastest service.
6. 321.
7. i) Cylinder ii) Seek Time
8. The levels of RAID are RAID 0, RAID 1, RAID 5, and RAID 10.
9. No
10. In RAID 10, the blocks are striped and mirrored. This is also called a strip of the mirror. Minimum 4 blocks are used in RAID 5. Excellent performance and redundancy are observed in RAID 10 due to the stripped and mirror.

11. RAID 1.

12. 2.

6.12 POSSIBLE QUESTIONS

Short answer type questions:

1. What do you mean by mass storage?
2. What are the characteristics of mass storage?
3. Define the term cylinder and sector of disk structure.
4. Explain the term seek time and rotational latency.
5. Difference between FCFS and SSTF disk scheduling algorithm.
6. Consider a disk queue with request for input/output to block on cylinders 98, 183, 37, 122, 14, 124, 65, 67cin that order. Assume that the disk head is initially positioned at cylinder 53 and moving towards cylinder number 0. What is the total number of head movements using Shortest Seek Time First (SSTF) and SCAN algorithms?
7. What is swap space?
8. Why is the necessity of RAID structure?
9. What is stable storage?
10. What is a tertiary storage structure?

Long answer type questions

1. Explain the Disk scheduling algorithms with examples.
2. If the disk head is located initially at 32, find the number of disk moves required with FCFS, SSTF, SCAN, and C-SCAN if the disk queue of I/O blocks requests are 98, 37, 14, 124, 65, 67.
3. Explain about different RAID structure.

6.13REFERENCES &SUGGESTED READINGS

- Schaum's Outline of Operating Systems.
- Operating system concept 9E by Silberschatz, Publisher: Wiley.

Space for learners:

UNIT 7: SECURITY

Unit Structure:

7.1 Introduction
7.2 Unit Objectives
7.3 Security
  7.3.1 Security Goals
  7.3.2 Security Issues and Measures
7.4 Threats
  7.4.1 Programs Threats
  7.4.2 System Threats
  7.4.3 Network Threats
  7.4.4 Attack and Its Types
7.5 Cryptography
  7.5.1 Types of Cryptography
7.6 User Authentication
7.7 Security Defense Mechanisms
7.8 Protect Systems and Networks with Firewalling
7.9 Computer Security Classification
7.10 Summing Up
7.11 Answers to Check Your Progress
7.12 Possible Questions
7.13 References and Suggested Readings

Space for learners:

7.1 INTRODUCTION

Security is the state of being free from threat. One of the major mechanisms of ensuring security is encryption. Basically security is concerned with the unauthorized access of information. Security ensures safe sharing of software and hardware resources of a system. Authentication is very important in this regard. Security attacks mainly focus on the illegal use of confidential resources such as data files.

7.2 UNIT OBJECTIVES

After going through this unit, you will be able to

- explain the basic concepts of security, threat, attack

- discuss the various security goals
- explain the concept of user authentication
- discuss about cryptography, computer security classification

Space for learners:

7.3 SECURITY

Security is a mechanism which provides protection to the system resources. System resources can be both software and hardware like CPU, disk, memory, data etc. stored in the system. Basically security is used to prevent unauthorized access of these resources. It deals with the threats that are external to the systems.

Some of the basic incidents which can be termed as security violation are given below:

i) **Theft of Data**: If an unauthorized user tries to steal information then this can be termed as theft of data.

ii) **Unauthorized Modification of Data**: If an unauthorized user tries to alter the data then it is termed as unauthorized modification of data.

iii) **Unauthorized Destruction of Data**: If an unauthorized user tries to delete the data then it is termed as unauthorized destruction of data.

7.3.1 Security Goals

Security between intended sender and receiver can be achieved through following major security goals:

i) **Confidentiality**:

It is a service through which only the intended sender and the receiver will know the actual data.

ii) **Data Integrity**:

It is service through which only the authorized user can access or modify the actual data.

iii) **Nonrepudiation**:

It is a service through which no user can refuse the previous commitment after doing so.

iv) **Authentication**:

It is a service through which only the authorized user can access the system resources.

v) **Availability**:

The system resources need to be available for the authorized user when needed.

Space for learners:

7.3.2 Security Issues and Measures

Security of a system can be violated by threats and attacks. If a system is hacked by unauthorized user then there will be loss of confidentiality, integrity of the system resources.

Some of the important security issues are:

i)Loss of data

ii) Modification of data

iii) Misuse of data

To protect the system from these security issues following measures need to be taken:

i) Protection mechanisms to prevent modification and loss of data.

ii) Control system resource sharing among the users.

iii)Authentication of the valid user needs to be done before accessing the system resources.

iv) Cryptographic techniques need to be used to ensure secure communication between sender and receiver.

v) Security policies need to be introduced among the users.

vi) The site containing computer system should be physically secured from attacker.

vii) The operating system must protect itself from malicious attacks.

viii) Secure network communication must be established among the systems.

ix) Anti-Malware programs need to be used to protect the system.

x) To protect the system from network threats firewall is used.

CHECK YOUR PROGRESS-I

1. What are the major security goals?
2. What are the different security issues a system can have?

7.4 THREATS

Threat can potentially harm the system resources. This means alteration or hiding or destroying the actual content of a message, occupying hard drive space and illegal use of passwords.

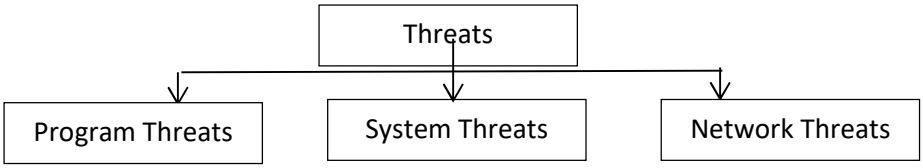


Fig.7.1: Types of Threats

7.4.1 Program Threats

When a program is created by a user is used by another user then misuse of the program may occur. If misuse of the program is happened then this event is termed as program threat. Some of the examples of program threats are *Trojan horse, Trap door, Bufferoverflow and Logic bomb*.

i) **Trojan horse**: This program sits ideally and transmits all the information to the attacker. Suppose if you login to a site using browser and if the Trojan horse is attached with the browser then the user id and password will be stored by the Trojan horse and it sends the user id and password to the attacker.

ii) **Trap door**: It is a program which is installed in a system and has some security hole in the code and due to this if the program performs illegal actions without the

Space for learners:

knowledge of the user, then it is called to have a Trap Door.

iii) **Buffer overflow**: Suppose a program is created by a user and that program is installed in another system, and this program consumes all the resources of the system where it is being installed. In this situation Buffer Overflow may occur.

iv) **Logic Bomb**: This situation is very hard to detect. Here, an installed program misbehaves when certain conditions met otherwise it works as a genuine program.

Space for learners:

7.4.2 System Threats

The misuse of Operating System (OS) and user files is termed as system threats. For example, mostly we install OS in C drive. There are many hidden folders in program files and few of the files we cannot even access. But these files can be accessed by the attackers by launching worms or viruses. Some of the examples of System Threats are Worms, Virus etc.

i) **Worms**: Worms create duplicate copies which contain malicious code that simply consume system resources and deny service of the user. This slows down the system.

ii) **Virus**: This can delete or alter the information available in a system. It contains small section of code embedded in a program. When this program is accessed by the user, the virus starts getting embedded in other files.

7.4.3 Network Threats

The misuse of user's confidential information while accessing the network without the knowledge of the user leads to network threats. Some of the examples of network threats are Port Scanning, Denial of Service etc.

i) **Port Scanning**: It is a mechanism through which unauthorized user can detect the system vulnerabilities to attack the system.

ii) **Denial of Service**: This prevents authorized access of a user. For example, if denial of service attacks in the browser's content setting then user may not be able to use the internet.

STOP TO CONSIDER

Program threat is concerned with the misuse of ones created program. System threat is concerned with OS of the system and network threat is concerned with the use of internet.

7.4.4 Attack

Attack is a kind of threat to a system from malicious users. It is of two types namely *Active Attack* and *Passive Attack*.

i) **Active Attack**: In this attack, the attacker tries to alter the content of the message. This type of attack can be easily detected so proper cure is needed. Here, attacker uses information to launch attack on the target. Example of active attack: *Masquerade, Replay, Modification of Messages, Denial of Service.*

ii) **Passive Attack**: In this attack, the attacker learns and uses information of the message and listens to the traffic to launch attack on the target. It is difficult to detect so prevention is better in case of passive attack. Example of passive attack: *Release of Message Content, Traffic Analysis.*

7.5 CRYPTOGRAPHY

It is a technique to hide the actual content from unauthorized user. It is the study and practice of different mechanisms for secure communication in the presence of unauthorized user in between the intended sender and receiver. Here we have two basic terminologies, Encryption and Decryption. Secure communication is done between sender and receiver with the help of cipher test.

Space for learners:

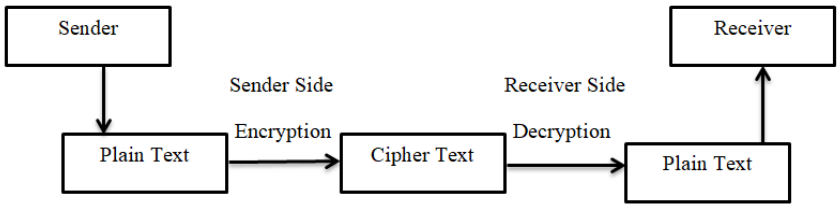


Fig. 7.2: Concept of Cryptography

Explanation of Fig. 7.2:

As we already know that cryptography ensures secure transmission of data between sender and receiver, so the sender will apply some encryption technique on the plain text that is the actual content and converts it to the cipher text. This cipher text will be sent to the receiver and at the receiver side, it will convert the cipher text to the plain text with the help of the encryption technique that has been used by the sender. The conversion of cipher text to plain text is termed as decryption.

Some of the important terminologies:

Cryptanalysis: The art of decoding the cipher text in order to hack without knowing the encryption technique is referred to as cryptanalysis. Cryptanalyst is the person who is always busy in cryptanalysis.

Cryptology: It is the combination of both cryptography and cryptanalysis.

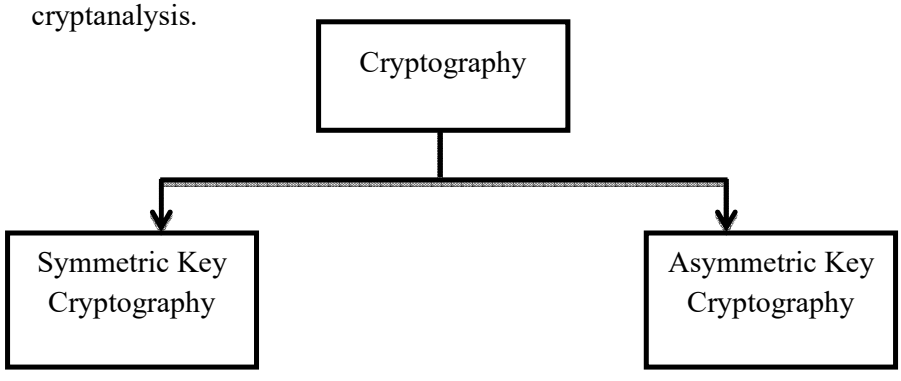


Fig. 7.3: Types of Cryptography

i) Symmetric Key Cryptography:

Here the sender and receiver of a message use the same key for both the encryption and decryption process respectively. Data Encryption System (DES) is one of the most popular symmetric key cryptography techniques.

Space for learners:

ii) Asymmetric Key Cryptography:

Here a pair of public and private key is used for both encryption and decryption process respectively. RSA algorithm is a good example of asymmetric key cryptography.

Space for learners:

7.6 USER AUTHENTICATION

Authentication is a process of identification. User authentication means identifying whether the user is a valid user or not. Suppose, you login to e-commerce site using the user id and password. The user id and password are the essential credentials to authenticate your identification. Normally in our personal computer we use to provide PIN or Fingerprint or Password to protect our system from unauthorized access.

User of a system can be authenticated using the following mechanisms:

i) **Using password**

Users normally have their own user id and password to access the system resources. Password is a combination of special characters, numbers, and alphabets. Now a days, One Time Password (OTP) is also popularly used to access resources. Normally OTP is sent to the registered mobile number or email id.

ii) **Using physical object**

Users normally use to withdraw cash in Automated Teller Machine (ATM) with the help of a unique card which is registered with the user only with a secured PIN.

iii) **Using biometric**

User authentication using biometric method is the most safeguard mechanism to protect the resources. Here physical characteristics of user like fingerprint, voice, and retina are used for authentication purpose as these are very difficult to forge. Now a days, in most of the offices or organizations punching machine is used to record the employees attendance.

CHECK YOUR PROGRESS-II

Fill-in the blanks:

3. _____ is a mechanism through which unauthorized user can detect the system vulnerabilities to attack the system.

4. The art of decoding the cipher text in order to hack without knowing the encryption technique is referred to as _____.

Space for learners:

7.7 SECURITY DEFENSE MECHANISMS

To ensure security of the system different defense mechanisms need to be followed. Following are some of the efficient defense mechanism normally used to protect the system against malicious attacks:

i) **Encryption**: Plain text is converted to cipher text (encrypted text) to hide the actual content of the message to safeguard it from attacker.

ii) **Digital Signature**: User digitally or electronically signs the data and sends it to the intended receiver. And the receiver verifies the signature before accessing it for security reasons.

iii) **Data Integrity**: There is a check value embedded with the message and this check value is known by the receiver and sender. Whenever sender sends message to the receiver this check value is appended with the message before sending it and after that the receiver matches the check value after receiving the message with the check value that receiver already has. If the check value matches then receiver accepts the message and if the check value does not match then receiver assumes that modification has been made so simply discards it.

iv) **Access Control**: This ensures that the user has the right to access the system resources.

v) **Authentication**: Only the authorized users can establish secure communication and share resources.

vi) **Traffic Padding**: Here some extra bits are added with the actual content of the message for encryption.

Space for learners:

7.8 PROTECT SYSTEMS AND NETWORKS WITH FIREWALLING

Firewall is a network security mechanism to protect the system from network threat. Basically, it monitors the incoming and outgoing packets that consists data/information in the network and based on the security rules it accepts or rejects the packets. It creates an interface between internal network and incoming traffic from the external network to protect the system from malicious threats. It can be both software and hardware. A software firewall is a program installed in the system. It regulates traffic through port numbers and applications. On the other hand, hardware firewall is a equipment installed between the network and gateway.

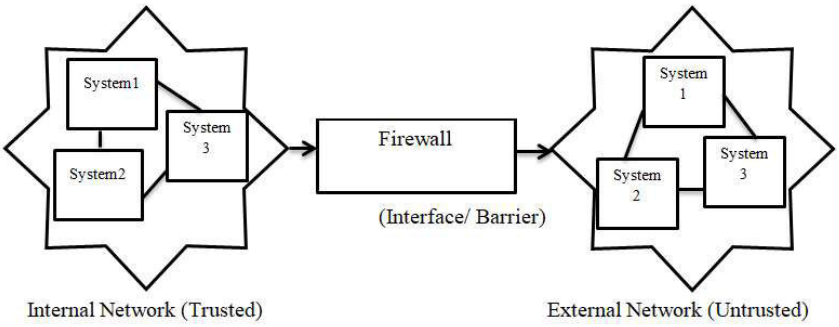


Fig. 7.4: Illustration diagram of Firewall

Some features of good firewall:

i) All the authorized incoming and outgoing packets must pass through the firewall.

ii) Firewall must be strong enough to reject unauthorized incoming packets.

Some limitations of firewall:

i) Inside a network if a malicious user tries to launch attack then firewall cannot protect the network.

ii) There are some applications where we need to disable the firewall. By doing so, there will be no control in the incoming and outgoing traffic.

iii) Firewall does not analyze the content of the packet so if a packet contains malicious content sent from an authorized user then it cannot discard the packet. This cause a threat to the system.

There are two types of firewall namely *Host based firewall* and *Network based firewall*.

i) **Host based firewalls**: These are installed on the network, which control incoming and outgoing packets. Host based firewall is a software application and it comes with the operating system. It provides protection to the internal network. Host based firewall protects systems from malicious attacks and unauthorized access.

ii) **Network based firewalls**: It operates on the network. These firewalls filter all the incoming and outgoing traffic across the network. It protects the internal network from the unauthorized access of the third party. A network based firewall might have two or more Network Interface Cards (NICs).

Working of Firewall:

Internet is untrusted network where different computers or systems are connected together to share resources/ information. Firewall maintains access list where authorized and unauthorized system's details are stored.

Example 1: Suppose we have 3 users A, B and C. In the access list of the firewall only user A and B have the access rights but user C does not have the access right. When A wants to access information through internet from B then B sends the information to A. Though user C hacks the connection and sends another reply to user A, then A simply discards it as C is not in the access list of the firewall. It method is named as ***Packet Filtering***.

Example 2: Suppose user A is currently visiting a specific site. Firewall of user A has the record of the user name and the visiting site in a conversation list. If an attacker hacks this connection and sends unwanted data to the user A, then firewall rejects these data as

Space for learners:

it already knows the visiting site from the conversation list. This method of protection is termed as ***Stateful Inspection***.

Example 3: Suppose user A is connected to the internet through a different user B. And user A requests some information from the internet via user B. User B passes the request to the internet. This way user B is hiding user A from the attackers available in the internet. This method of protection is termed as ***Proxy Firewall***.

So, if you want to protect your system from malicious threat then never disable the firewall.

Space for learners:

7.9 COMPUTER SECURITY CLASSIFICATION

Computer security means protection of system resources from unauthorized access. It prevents modification and deletion of data from malicious users. It restricts unauthorized users. Computer security is mainly concerned with three goals i.e. confidentiality, integrity and availability. These points are already discussed above. Computer security is important to safeguard the system resources from virus and worms. It protects crucial information of users.

According to the U.S. Department of Defense Trusted Computer System's Evaluation Criteria there are four security classifications available for computer: A, B, C, and D[4]. In the following table brief description of each classification is given.

Table 7.1: Security classes

Security Classes	Description
A(Highest Level)	This classification uses formal design specifications and verification techniques to grant access to user for secure communication or resource sharing.
B	This classification provides mandatory protection system. It is of three types. i) B1: This maintains the security label of objects in the system. Label is used to make decision to control the access. ii) B2: This extends the sensitivity labels to

	each system resource. iii) B3: This allows creating user groups for access control to grant access or restrict access to other object available in the system.
C	This classification provides protection and user accountability using audit capabilities. It is of two types. i) C1: This incorporates controls to protect the user's resources. Example: UNIX versions are mostly Cl class. ii) C2: This adds an individual-level access control to the capabilities of a Cl level system.
D (Lowest Level)	This classification is used for systems that have failed to meet the requirements of any of the other security classes. For example, MS-DOS and Windows 3.1 are in division D

Space for learners:

7.10 SUMMING UP

- Security is concerned with the unauthorized access of the system resources.
- Security mainly focuses on the confidentiality, integrity and availability of the system resources.
- Threats like virus and worms are used by the attacker to hack a system. To prevent this hacking we need to use some secure communication strategies like encryption, digital signature etc.
- Threats are of three types: program threat, system threat and network threat.
- There are two types of security attack namely active attack and passive attack.
- Cryptography is a mechanism of secret writing. It is of two types i.e. symmetric key cryptography and asymmetric key cryptography.
- Firewall is one of the major mechanisms to establish security.

7.11 ANSWERS TO CHECK YOUR PROGRESS

1. The major security goals are: Confidentiality, Data Integrity, Nonrepudiation, Authentication, Availability.

2. The different security issues a system can have, are: Loss of data, Modification of data, Misuse of data.

3. Port Scanning

4. Cryptanalysis

7.12 POSSIBLE QUESTIONS

1. What do you mean by security?
2. What is packet filtering?
3. Compare stateful inspection and proxy firewall.
4. What do you mean be cryptography? How it is used to secure systems?
5. State the difference between virus and worms.
6. What are the classes of computer security? Explain.
7. Define attack. Explain the different types of attack.
8. Define threat.
9. Give some examples of program, system and network threats.
10. What are the major security goals?
11. Explain about firewall with a suitable diagram.
12. How firewall works?
13. Define access control.
14. How Trojan horse works?

7.13 REFERENCES & SUGGESTED READINGS

- www.geeksforgeeks.org
- www.javatpoint.com
- www.tutorialspoint.com
- www.easyengineeringclasses.com

Space for learners:

UNIT 8: DISTRIBUTED OPERATING SYSTEM

Unit Structure:

8.1 Introduction
8.2 Unit Objectives
8.3 Advantages of Distributed Operating System
- 8.3.1 Sharing of Resources
- 8.3.2 Scalability
- 8.3.3 Computation Speedup
- 8.3.4 Reliability
- 8.3.5 Communication

8.4 Types of Network Based Operating System
- 8.4.1 Client Server Network
- 8.4.2 Peer to Peer Network

8.5 Network Structure
- 8.5.1 Local Area Networks
- 8.5.2 Wide Area Network (WAN)

8.6 Communication Structure
- 8.6.1 Name Resolution
- 8.6.2 Routing Strategies
- 8.6.3 Packet Strategies
- 8.6.4 Connection Strategies
- 8.6.5 Contention

8.7 Communication Protocols
8.8 Design Issues
8.9 Distributed File System
- 8.9.1 Naming and Transparency
- 8.9.2 Remote File Access
- 8.9.3 Stateful versus Stateless service
- 8.9.4 File Replication
- 8.9.5 Andrew File System (AFS)
- 8.9.6 Google File System (GFS)

8.10 Summing Up
8.11 Answers to Check Your Progress
8.12Possible Questions
8.13 References & Suggested Readings

Space for learners:

8.1 INTRODUCTION

Distributed Operating Systems are the systems where the processors are interdependent via some communication network in a loosely coupled environment. Processors in a distributed system have distinct names such as host, site, nodes, computers, system etc. Each of the nodes has their own resources such as memory, system clock, kernel etc. For a specific processor, the resources of all other processors appear to be remote. They communicate with each other with the help of different communication media such as telephone lines or high speed buses.

The Structure of a Distributed System is as shown below:

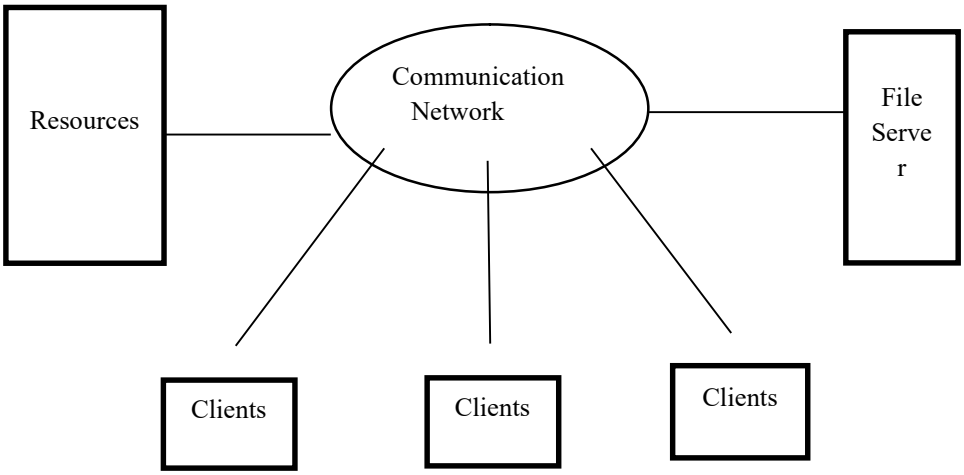


Figure 8.1: A Distributed System

Space for learners:

8.2 UNIT OBJECTIVES

After going through this unit you will be able to:

- Learn the basics of Distributed Operating System
- Architecture of the system
- Technologies used for setting up the system
- Protocols used for communication
- Issues in designing a DS
- A brief description about distributed file system
- Some popular DFS

8.3 ADVANTAGES OF DISTRIBUTED OPERATING SYSTEM

Different objectives can be there for using a distributed operating system: Sharing of Resources, Reliability, computation speed up and communication are some of them.

8.3.1 Sharing of Resources

The nodes those are connected to a distributed network can take the privilege of using resources from other nodes connected to the network. Resource may be either hardware (such as printer, scanner etc.) or software (such as text, audio or video files).If a node at site A needs a scanner, it can access it from some other site B. In the same way, a node at site C can access files remotely from some node in site D. Information processing can also be done remotely with the help of distributed database.

8.3.2 Scalability

Scalability of a system can be measured in different dimensions. It can be in terms of size that means extension of number of users or resources in the existing network. It can be administratively scalable, where you can expand the number of organizations of the system. Or the scalability can be in terms of geographical area where the organizations connected to the system are far apart from one another. A group of users can work on a project by geographical scalability. They can share files of the project they are working. They can make use of RPC (remote procedural call) and with the help of remote login; they can edit the code written by some other user.

8.3.3 Computation Speedup

If it is possible to split a computation into smaller parts then each of these parts can be executed parallelly in different nodes of the system and then recombine the sub-parts at the end of the execution. This can speed up the computation to a great extent. Furthermore, if a site is heavily loaded with processes, then some of the processes can be transferred to moderately loaded sites in the network. This technique is called load-sharing.

Space for learners:

8.3.4 Reliability

Reliability indicates the ability of the system being capable of functioning perfectly even after the failure of one or more components. For example, if a user has booked a railway ticket, and before the changes are permanently stored in the database, the system crashes, in such a scenario it is expected that changes made by the user should persist.

If the system is build up with multiple general-purpose computers, then working of the system will not be affected even when any of the computers fails. But in case, each of the machines of a system is responsible for performing some decisive task, then failure of one machine may lead to shut down of the whole system. In general, reliability depends upon redundancy. If one node shuts down suddenly, then there should be some other node which is carrying all the data and information of the previous one.

8.3.5 Communication

The sites that are connected in a distributed system can communicate with one another with the help of message passing. In a standalone system, message passing is done within the cooperating processes. This idea has been expanded in distributed system to send messages among users of different sites. Other functionalities such as email, remote procedural calls, file transfer etc. can also be incorporated in a distributed system. Communication can be of different types such as unicast, multicast and broadcast. In case of unicast communication one host will communicate with any other host in the system, in case of multicast, one host communicates with a number of hosts in the system and in broadcast communication, one host communicates with every single host present in the system at the same time.

Corporate sectors are also benefited from the distributed system. By replacing the mainframe with a distributed system, they can get ample number of resources from geographically dispersed areas, enhanced functionality in a minimal cost, and better user interface and less maintenance cost.

Space for learners:

STOP TO CONSIDER

A distributed operating system is a set of loosely coupled systems connected via some communication network. Each of the systems has their own memory and processor. Resource sharing is possible in a distributed system even for geographically separated systems. A distributed system provides scalability in different aspects like geographical area, size or administrative scalability. Computation can be performed by dividing a problem into sub-problems. Corporate sectors can replace mainframe with a distributed system to reduce cost.

CHECK YOUR PROGRESS-I

Fill-in the blanks:

1. RPC stands for _____ .

2. _____ indicates the ability of the system being capable of functioning perfectly even after the failure of one or more components.

3. Network operating system can be broadly divided into Client Server Network and _____ .

Space for learners:

8.4 TYPES OF NETWORK BASED OPERATING SYSTEM

A network based operating system is a group of computers having individual operating systems connected through a common network. The network works as a boss for the whole system. It groups the standalone computers and synchronizes their activities.

Network operating system can be broadly divided into two types:

Client server network and Peer to Peer network

8.4.1 Client Server Network

These types of networks are considered to be the most common type of network operating system. The client-server concept changes slightly depending on the context it is using. One of them is the thin client computing, where the clients are light-weight computers

having a small amount of memory. Only the graphical user interface is installed in the client machine and all other services are accessed from the server. The client machine acts as a terminal to have access on the operating system which is actually running on the Server. One example of thin client computing is the 80486 system having windows XP.

A wider sense of client server concept is used in Authentication Server Based Network. Each of the machines in the system has an account through which they authenticate themselves before using the resources from the server. A server is a powerful computer that is used for storing the information and resources to be accessed by the systems of the network. There are different security groups for different machines and depending on that, the user is given the access. The server provides security to the entire network and acts as resource manager. High cost software needs to be installed in the server in order to ensure smooth performance of the network. This type of network is termed as domain and the server is termed as domain controller in the Microsoft Network (MSN)

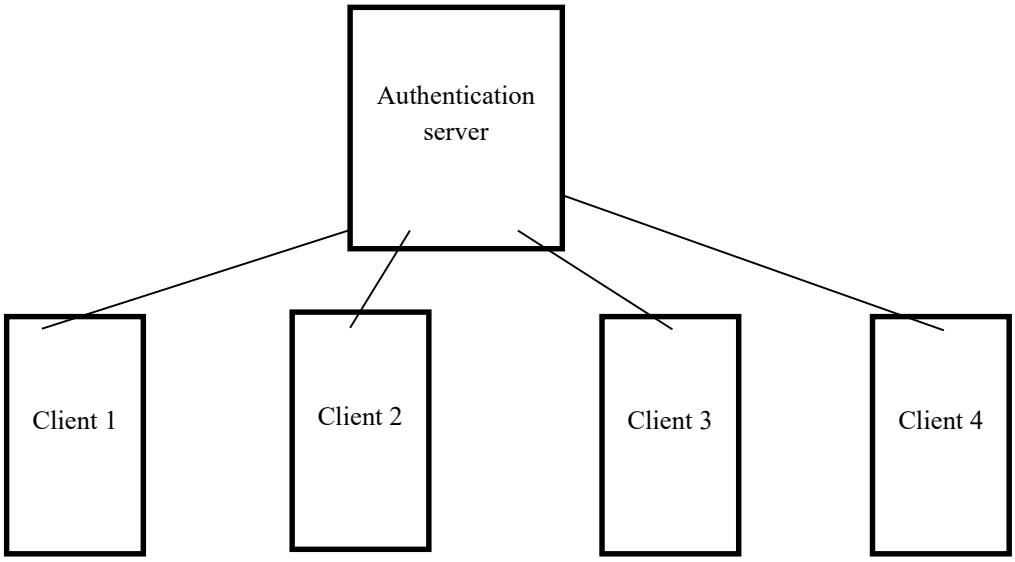


Figure 8.2: A client-server network

This architecture is mostly used in Organizations like Universities, colleges, banks and hospitals that need a dedicated server. It provides the facility of combining different parts of the network and gives concurrency transparency to its users, which means the same file can be accessed concurrently by different users leaving the database in a consistent state. These types of networks are useful for large organizations because of centralized security and management.

Space for learners:

Space for learners:

8.4.2 Peer to Peer Network

There is no authentication server present in peer-to-peer network but, the network may contain other type of servers like file server, fax server, remote access server and so on. Each of the computers in the network performs the services of both client and server. If any of the systems want to access data from other system, security is provided either by creating local account for the user or making the resources password protected. This type of architecture is suitable for small companies or in a home network. But if the network becomes bigger, managing of resources and creating local account becomes inconvenient. A user has to create hundreds of local accounts or he may have to remember all the passwords of different resources. On the other hand, in a client server network, user can access the network just by entering the network password and can avail the resources based on the permission given to him. The administrator will assign permissions for different resources making the task of the user much simpler.

Because of the absence of a dedicated server, compromised security has become the main pitfall of this architecture. On the other hand, absence of a server can reduce the cost of setting up a peer to peer network to a great extent.

The workgroups are less expensive as compared to client-server network for the following reasons:

- The operating system that runs on a server is costly as compared to general operating systems.
- More complex hardware are required for the server software
- System administrator needs to take extra load for maintenance of the tasks performed by the server.

Space for learners:

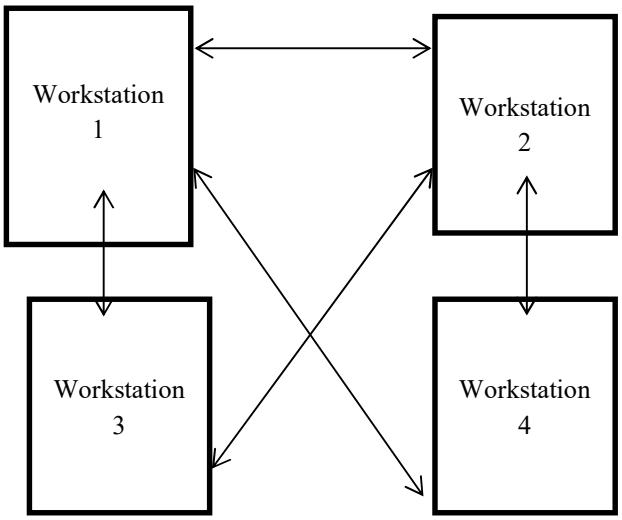


Figure 8.3: Peer to Peer Network

8.5 NETWORK STRUCTURE

There are two types of networks LAN and WAN in a communication network. They differ mainly in the geographical area they cover. LAN (Local Area Network) as the name says, it covers a small geographical area such as a department, an institution etc. whereas, WAN (Wide Area Network) covers a large geographical area such as a country. It combines two or more LANs by using independent processors.

8.5.1 Local Area Networks

The Local Area Network has come into light in early 1970s. It has been developed as a substitute for mainframe computer. Most of the organizations prefer small computers each with its own application, instead of having a mainframe computer. Such an environment is more reasonable and user friendly as all the workstations have access to both software and hardware resources. And it is quite obvious that an organization needs to share a lot of information among the workstations, thus bringing LAN into focus.

As the LAN is meant for a small geographical area, such as in institutions, colleges, University Departments etc., the workstations of a LAN are adjacent to each other as compared to those of WAN.

Because of this, the communication speed and rate of error are comparatively low. High-priced cables such as coaxial cable or fiber optic cables are used to achieve high speed and reliability. But if the LAN is covering a long distance, the cost of the network may increase with increasing cable length. Additionally, repeaters need to be added to boost the signal.

Most common way to construct a LAN is Ethernet cables, defined by IEEE standard 802.3. Speed of Ethernet cable ranges from 10mbps to 1gbps. The 802.3 standard defines the physical layer and MAC (Medium Access Control) sub-layers of OSI protocol. It is available in different versions.

802.3a (10Base2) uses thick coaxial cable with a bandwidth of 10mbps and maximum possible segment length 200m. Another variation of Ethernet cable IEEE 802.3i (10 Base T) uses Unshielded Twisted Pair Cable as a transmission media with a maximum speed of 10MBPS. Whereas the IEEE 802.3u (100BaseT) runs at a speed of 100mbps. Another variation called FDDI(Fiber Distributed Data Interface) is used by the LANs that extend up to 200kms. It uses fiber optics cable and provides a maximum speed of 100MBPS.

STOP TO CONSIDER

A communication network is built up with two types of networks: LAN and WAN. A LAN is preferred for small geographical areas. The Characteristics of the transmission Media used to set up a LAN are defined by IEEE standards. Variations in IEEE standard are available to provide different communication Speed. A LAN can be either wired or wireless. On the other hand, a WAN is preferred over a large geographical area. Technologies used for WAN connection are: leased line, dial-up connection, DSL, satellite communication etc.

Space for learners:

Space for learners:

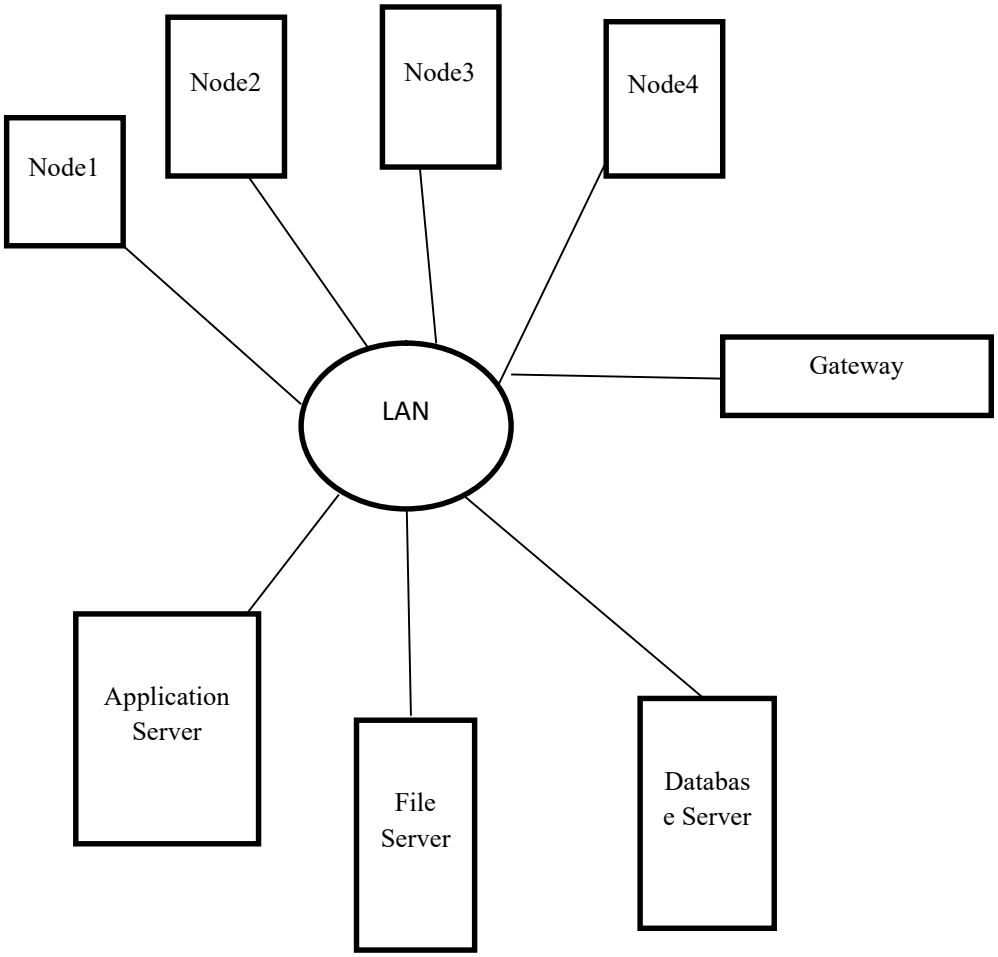


Figure 8.4: A LAN Network

As shown in Figure 8.4, nodes in a LAN are of different specifications from mainframe to personal digital assistant (PDA) which are connected to each other by different topologies. The network contain servers (File Server, Database Server, Application Server, Proxy server), peripheral devices (Printer, database Files), gateways to connect to different networks and Repeaters to amplify the signal.

In addition to cable connection, Local Area Network comes with wireless facility also. Wireless LANs use radio frequencies instead of cable. It is based on IEEE standard 802.11 released in the year 1997 with a speed of 2Mbps. Different variations have evolved over years, 802.11ax being the latest of all with a maximum possible data rate of 1.8 Gpbs.

8.5.2 Wide Area Network (WAN)

Space for learners:

The Wide Area Network was emerged in late 1960s. Idea was to connect workstations from a large geographical area, share resources, and perform confidential task in an economic way. LANs which are situated in different geographical area and are a part of same organization for example SBI(State Bank of India), can easily connect through WAN. Advanced Research Project Agency Network (ARPANET) was the first wide area network based on TCP/IP protocol and packet switching scheme. ARPANET initially connected five educational institutions including University of California and Los Angeles. With time, the researchers assembled the "network of networks" to give birth to modern day Internet. Internet is the largest WAN in today's world that establishes connections between LANs and MANs.

Some of the technologies used for setting up a WAN connection are:

Leased Line: These are dedicated lines that provide continuous data flow and can connect two or more LANs and MANs. Leased lines use fiber optic cable for high speed data and bandwidth.

Dial up Connection: Dial up connections use Public Switch Telephone Network (PSTN) for establishing an Internet connection. The telephone line is connected to a modem that converts the digital signals of the computer to analog signals used by the telephone lines. With times, this has become outdated because of its low data transfer rate and dependency on telephone lines to use the internet. That means the user is unable to use the telephone and the internet service at the same time.

Digital Subscriber Line: Digital Subscriber Line uses twisted pair cable for data transmission. These cables are generally used for telephone lines. By splitting the frequency, an uninterrupted service is provided to both phone calls and the internet. The technology behind it is as such: a variation of OFDM(Orthogonal Frequency Division Multiplexing) called discrete multitone is used to divide the bandwidth in parallel paths so that at the same time, both phone calls and data transmission can be carried out. This is operated by the DSL modem connected to the telephone lines.

Satellite Communication: Satellite communication has become popular over the years for wireless internet accessibility. A satellite communication comes in a range of Low Earth Orbit(LEO) and

Medium Earth Orbit (MEO). LEOs are mainly found in 1800 to 2100 miles above the earth and MEOs are found in 9000 to 10000 miles above the earth. Users can connect their laptops, cell phones and personal digital assistants to wireless network using the satellite communication.

Point to point link is used to connect the WAN with MANs and LANs. The WAN contains a networking device called packet switch that contains memory, Processor and input/output interfaces to connect with another packet switch. The message is stored in the memory, before being forwarded. Router is another networking device that takes the decision on which path; the arrived packet should be forwarded. There are two different routing mechanisms: static and dynamic. The static protocols take the decision based on the network topology, but it can't detect a link failure and modify the pre-defined route. On the other hand, dynamic protocols such as OSPF and RIP are capable of detecting link failure and accordingly update the route in the routing table dynamically.

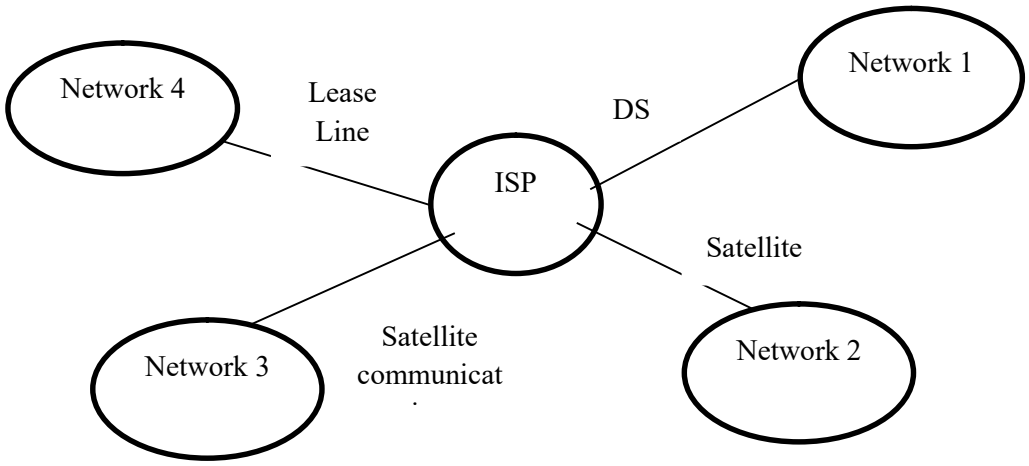


Figure 8.5: A WAN network

8.6 COMMUNICATION STRUCTURE

Communication Structure means the internal working of a distributed system. Here five issues need to be dealt with.

Space for learners:

8.6.1 Name Resolution

Space for learners:

In a distributed system, if a process in host A wants to communicate with a process in host B, they must know the address of each other. Therefore, every process is recognized by two parameters <host name, id> where the hostname is the unique name assigned to a particular host and id is the process within that host. Here comes the problem of Name resolution. Human beings communicate with the help of names, whereas computers find it convenient to use numbers for simplicity and better speed. Therefore, some mechanism needed to be introduced to convert the host names into numbers or ids so that the networking hardware can identify the receiver.

In the early days of Internet, each host used to maintain a data file which contains the host name and corresponding address of all the computers in the network. Whenever a system is added or removed from the network, every host needs to update their data file. With the growing number of networks, it has become cumbersome to update the file every time. Here comes the concept of Domain Name System(DNS). The protocols like TCP and IP convert the host names to IP addresses using the procedure called name resolution. The DNS is a distributed database system based on client/server architecture that is used for converting Host Name to IP addresses. A Domain Name (mostly called as Domain) is a name that is associated with an IP address. There are various kinds of Domain are available such as com for commercial sites, org for nonprofit organizations, country specific domain such as .in .uk .un etc. All these come under top level domain name where .org is a Generic Top Level Domain (GTLD) and .uk .un are Country code Top Level Domain (CcTLD).

When a host requests for an IP address, sayYahoo.com, the query is resolved in reverse order. The operating system will search the IP address in its local cache, in case it is not found, the query will be sent to resolver server. If the IP is not present in the cache of resolver server, then it will send it to the next level server i.e. the root server. Root server will send the resolver server to Top Level Domain Server for .com domain. The resolver server will then direct the query to authoritative server for the second level domain i.e. yahoo.com. The authoritative server will provide the resolver with the IP address. The resolver will provide the host with the IP address of Yahoo.com. The resolver server keeps the address in its cache so

that, next time when the request comes for yahoo.com, it can directly provide with the IP address.

8.6.2Routing Strategies

There may be different routing strategies for sending a message from host A to another host B. Routing tables are maintained to send packets through the most reliable path. Routing table contain the information like network id, subnet mask, next hop and minimum number of hops to reach the destination. Based on the network condition, the best possible route can be updated time to time. Most common routing strategies are: fix routing, virtual routing and dynamic routing.

- Fixed Routing: Here the optimal path is chosen between two hosts. The path is not updated later on, unless some hardware failure damages it permanently. Therefore, even if the path has a heavy traffic compared to the other paths, there is no option to change the path and adopt some lightly loaded path.
- Virtual Routing: In virtual routing a dedicated path is given for a particular session. Two hosts A and B can use different routes for different session.
- Dynamic Routing: The route is assigned dynamically before starting a communication between two sites. Messages may be sent through different paths. Dynamic routing algorithms such as RIP and OSPF permit the routers to share routing information with nearby routers in order to get the optimal path. Since the path is decided dynamically; out of order packets may arrive at the destination. Therefore a sequence number is added to each of the outgoing packets so that at the receiver side, they can be reassembled. Though dynamic routing is complex to set up, it is suitable for huge networks.

STOP TO CONSIDER

In order to reduce collision in the network, routing strategies are introduced. They can help sending a packet from source to destination through an optimal path. With advancement in technologies, dynamic routing algorithms are introduced that can update the path by collecting information from nearby routers.

Space for learners:

It is possible to use both fixed and dynamic routing in the same system. The sender may send a message to the gateway using a fixed routing scheme whereas the gateway uses dynamic routing to transfer the message to the destination network.

Space for learners:

8.6.3 Packet Strategies

The data can be of variable length. To simplify the communication, a message is divided into fixed length unit named as packet, segment, frame etc. The transmission can be connectionless or connection-oriented. In case of connectionless transmission (such as in case of UDP), sender don't get any information whether the packet has reached the destination or not. If a message is divided into multiple packets, a connection is established to ensure reliability. In a connection oriented transmission, receiver sends an acknowledgement to the sender for the packets received. An acknowledgement can be either single or cumulative.

8.6.4 Connection Strategies

Three main strategies for establishing a connection are: packet switching, message switching and circuit switching.

- **Circuit Switching:** If two nodes want to communicate, a dedicated path is assigned to them. The Path cannot be used by other processes for the entire duration of the communication even if it is idle for some time. One of the examples of circuit switching is telephone network. Once user A calls another user B, others parties can't use that line until they hang up. Because of a dedicated communication channel, data rate is guaranteed in a circuit switching network but more bandwidth is needed to set up a circuit.
- **Message Switching:** There is no direct connection between sender and receiver in a message switching network. Instead, the intermediate nodes or switches receives the message and transfers it to the next hop. Each message contains a header that carries the information like source and destination addresses, Error Checking code and expiry date. Each hop needs to have sufficient resource to retransmit the message to the next hop in the network. In case, enough resource is not available the message is stored for an indefinite period. This

process is called store and forward. More than one message from various senders can be sent over the same link. Though message switching is better than packet switching strategy, it is not suitable for real-time data transfer since the processing takes place in each of the intermediate nodes, making the overall process slow. Also each of the intermediate nodes should have a large storage capacity to store the entire message, since the message can be of various lengths.

- **Packet Switching:** The message is divided into variable length data units called packets. Each of the packets contains control information and payload. The packets are free to follow different paths depending on the traffic of the network. On receiving side, the packets that belong to the same file are reassembled. Some of the advantages of this network are: it ensures reliability as the receiver can detect the missing packet. If a link is down, the packets can take another link, thus making it fault tolerant. Transmission latency is minimal. It makes best use of the network bandwidth therefore packet switching is the most commonly used connection strategy.

Space for learners:

8.6.5Contention

In a communication network, it is possible that more than two sites are transmitting messages simultaneously over the same link (for example in a mesh topology). If collision happens, there should be some mechanism to discard the message and inform the sender, so that the message can be retransmitted. If no mechanism is designed to avoid collision, it may be repeated resulting degradation of the system performance. Some of the techniques for collision detection are discussed below:

- CSMA/CD (Carrier Sense Multiple Access/Collision Detection): This is a media-access control (MAC) protocol used mainly in Ethernet LANs. Each of the stations in a network, sense the channel before starting the transmission. In case, the channel is busy, it will abstain from transmitting and continues to sense the channel. If the channel is free, the station will start sending. Suppose station A and station B starts transmitting at the same time, then both the signals will

collide. As the stations receive the collision signal, they will stop transmitting. This is called collision detection. Each of the stations will again try after some random amount of time. If no collision is detected during the transmission, the sender will complete the transmission.

As the number of nodes increases in a network, possibility of collision also increases, resulting in performance degradation. One solution to this problem is to limit the number of hosts in a network. Adding more nodes in a congested network may result in bad throughput.

- Token Passing: This is an access-control protocol implemented in a Ring topology. A token is a small message that contains a pre-defined bit pattern. The token is passed in the network in either clock-wise or anti-clock wise direction. When a node wants to transmit, it removes the token from the network and start transmitting. No other node is allowed to transmit without having the token. Once the node finishes its transmission, it releases the token allowing other nodes in the network to transmit data. If the token is lost, the system adopts an election algorithm to select a specific site for generating the token.

  A token-passing protocol can give constant performance. As the number of hops increase in the network, average waiting time may increase but it will still perform better than that of an Ethernet network. However, for a small network, LAN is preferable.

Space for learners:

8.7 COMMUNICATION PROTOCOLS

The communication network must have some set of rules for establishing connections, transferring packets, error detection, selecting the shortest path and so on. To deal with all these issues, the whole process is divided into some layers. Each of the underlying layers performs their assigned duties and sends the message to the upper layer. A message sent by a host, passes through all of the layers and then enters to the recipients system. Each layer is bound to follow specific protocols while communicating. The International Standards Organization (ISO) has defined seven layers as described below:

- Physical Layer: This layer is responsible for transmitting the message in the form of bits. Bit rate (the number of bits transmitted per second) is also defined by physical layer. The physical and logical structure of the network (known as network topology) is defined in this layer. Along with that, physical layer also defines the mode of transmission i.e. duplex, half duplex and full duplex.
- Data-link Layer: Data link layer divides the bit streams sent by the physical layer into some data units called frames. It is responsible for detecting lost or damaged frames and adds mechanisms for retransmission of the frames. If two or more devices are connected to the same link, which device will get access of the link is determined by data link layer. Flow control is also performed in this layer.
- Network Layer: Network layer is responsible for source to destination delivery of the packets along with assigning logical address and selecting routes for outgoing packets.
- Transport Layer: Transport layer delivers the packet received from network layer to the correct process within that host. This is called process-to-process delivery. This layer provides connections to the packets, performs end to end flow and error control.
- Session Layer: This layer is responsible for identifying the mode of communication (half duplex, duplex, full duplex) in a particular session. This is called network dialog control. The session layer adds check points or synchronization point to a data stream so that in case of any failure, only the part of the message after the check point can be retransmitted.
- Presentation Layer: The presentation layer is responsible for translation of the messages from one format to another. As different computers use different encoding systems, message transferred in one format needs to be translated to some other format. This task is performed in the presentation layer. It also performs encryption and decryption of data. Another responsibility of this layer is the compression of data stream which is particularly important for multimedia data such as video, audio etc.

Space for learners:

- Application Layer: This layer is responsible for direct interaction with the user. It allows a user for remote access to a system, and controlling files in the remote system. It deals with email and the organization of distributed databases.

Figure 8.6 explains the OSI protocol stack. In sender side, the message travels from presentation layer to the physical layer, each of the layers adding their own header with the message. Once the message is converted to a bit stream, it is transferred to the receiver; through some transmission media for example fiber optics or twisted pair cable in case of wired medium and radio wave or microwave in case of wireless media. On the receiving side, the message travels from physical layer to application layer, removing the corresponding headers in each layer. Logically each of the layers in sender side communicates with each corresponding layer in the receiver side as the protocols defined by a particular layer can be understood by that specific layer only.

A message needs to cross one or more router before reaching the intended destination. Each of the routers needs the IP address of that message in order to direct it into the correct route and thus unpacking the message up to the network layer. For this reason, the physical, data link and network layers are known as hardware layers.

Space for learners:

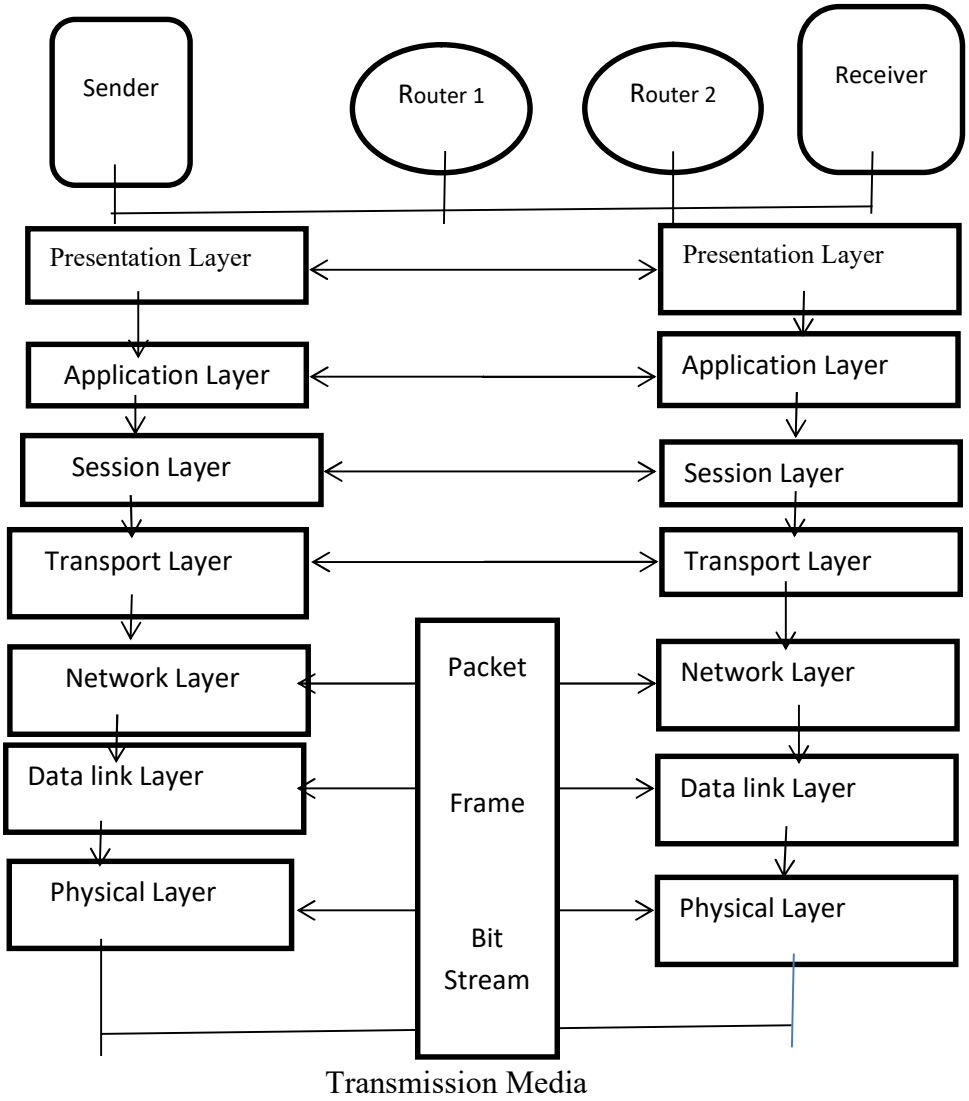


Figure 8.6: OSI protocol stack

The most widely used protocol stack is the TCP/IP protocol stack. It has fewer layers as that of OSI model. This model is more reliable as compared to OSI model. Application layer of TCP/IP model which is a combination of session layer, presentation layer and application layer, uses different protocols like HTTP, FTP, SMTP, SNMP etc. Transport layer has two protocols UDP (User Datagram Protocol) which is a connection less protocol and TCP (Transmission control protocol) which is a connection-oriented protocol. Next lower layer is the Internet Layer, the basis of which is the IP(Internet Protocol) which routes the IP packets. There is no dedicated physical link in TCP/IP model, allowing the data packets to travel in any physical path.

Space for learners:

8.8 DESIGN ISSUES

The users of a distributed system should feel like they are working in a traditional centralized system. Transparency is one of the key design issues in a distributed system. It can be measured in different parameters. Location transparency hides the details of storage location of the resources. Replication transparency hides the number of copies present for the same data. Concurrent transparency allows multiple users to access the same file concurrently without their knowledge. Parallelism transparency allows parallel execution of the activities without user's knowledge. User mobility is another kind of transparency where a user is allowed to login to the system from any machine.

Fault tolerance is another important issue. A system should be able to tolerate different kinds of failure like machine failure, crash of storage devices, and link failure to some extent. However, the performance of the system will reduce because of the failures. A system is not fault tolerant if it stops working with the breakdown of some of its components.

Another important aspect is the scalability. Scalability means how well the system will work as it grows in terms of resources and number of systems. Scalability can be measured in three dimensions

- Size scalability
- Geographical scalability
- Administrative scalability.

A size scalable system should function properly when the number of components increases. The system may experience high traffic in a specific day. The existing database may not be able to handle the traffic, which brings the need for adding more databases or more servers to the system. If the system is truly scalable, adding more resources should not reduce system's performance and it should not get slower. If the system is based on centralized data, centralized service and algorithms, size scalability may have to deal with different issues.

In case of centralized server, where a single server is responsible for the implementation of different services, congestion may result with growing number of users and applications. But it is unavoidable to

Space for learners:

use centralized server in some confidential situations like banking, medical, administration etc. where a single server is used to store all the sensitive information and separating it by special networking devices from the rest of the network. The problem persists in case of centralized data also. If the Domain Name Service (DNS) is implemented in a single database, each request over the internet would be forwarded to that particular database causing a severe congestion on the link. At last, the centralized algorithm is also a bad idea. A large distributed system has tremendous number of messages routed in different links. The idea here is to collect all the routing information and redirect it to a single machine and the algorithm computes the best suitable path. The information may create heavy traffic in a part of the network. To deal with this issue, decentralized idea has come up where no single machine stores all the routing information and they make decision only based on local information.

Stop to Consider

Design issues of a distributed system bring the issue of scalability, fault tolerance and transparency into focus. A scalable system should work perfectly even it grows in terms of size, geographical area or administrative organization. A fault tolerant system should not be able to tolerate faults to some extent. The term transparency means the underlying protocols should be hidden from the end-user.

Geographical scalability means whatever be the distance between the user and the resources, the user should be able to access them efficiently. Adding new nodes to the system should not slow down the transmission rate. The synchronous communication approach is suitable for small geographical area networks such as LANs. But in case of WAN, where two processes are geographically far apart, successful implementation of inter-process communication with synchronous communication is not an easy task.

Administrative scalability means even if an organization spans in many administratively independent domains, it should still be easily manageable. Achieving an administrative scalability is the toughest of all since it includes some non-technical issues such as policies of an organization and cooperation of humans. Security issues are also involved here. If a new domain is built up, all the other domains

Space for learners:

need to protect themselves from the new domain. The new domain may only have the read access to the files of other domains. Similarly, in order to protect itself from malicious attacks, the new domain may restrict its accessibility to the foreign code such as java applets in a web browser.

Space for learners:

8.9 DISTRIBUTED FILE SYSTEM

The nodes of a DFS(Distributed File System) can have remote access to the files of the system; i.e. the clients and servers are scattered over the network. Unlike the local file system, a distributed file system has multiple copies of storage over different servers. The DFS is implemented in a number of ways: in some systems servers run on particular machines, while some machines work as both client and server. The DFS may be implemented in the Operating System itself or as distinct software that connects the traditional operating systems with the file system.

Features of DFS are:

8.9.1 Naming and Transparency

The users of a computer system deal with a file with the help of file name which is actually a logical representation. The operating system then locates the particular data blocks (for the file) stored in the disk. Once the file is referred by the user with a textual name, that name is converted to some numerical value which in turn is mapped to the blocks of disk. This mapping hides the storage details of the file from the user. This is called abstraction. The DFS provide file replication along with abstraction. Multiple copies of files are stored in different systems in case of a DFS. When a file name is referred, the mapping will come up with a set of the replicas of that file. But the user is unaware of the existence of multiple copies.

8.9.1.1 Naming Structure

Two important aspects related to naming are *location transparency* where the physical location of a file is hidden from the user and *location independence* where the physical location of a file can be changed without changing its name. Location independence is a dynamic mapping property as the same file name is mapped into more than one location at different times, whereas location

transparency is a static property. For these systems location migration is not possible i.e the location of a file can't be changed automatically. However, manual changing of files within machines is possible.

Space for learners:

8.9.1.2 Naming Schemes

Three naming schemes are there for a DFS. In the first scheme, a file is identified by its host name and local name which differentiates it from other files in the system. This scheme does not provide location transparency or location independence. Local and remote files can be accessed with the help of same file operation. In a DFS each of the traditional file system is treated as one of its components. In the first approach, some provisions are there for remote access of these component units.

In the second approach; the remote directory can be attached with local directories giving the appearance of a coherent directory tree. An implementation of this approach is the Network File System (NFS).

In the third approach, a global name structure spans all the independent components. The file structure composed here is same as that of a traditional file structure. But this approach is difficult to implement because of some special machine specific files like device files and binary directories that exist in UNIX environment.

The NFS directory scheme is the most difficult scheme to implement. The reason being, any remote directory can be attached by any machine in any of the local directories. If the server is facing some issues, the directories added by some computer may not be available in the global structure. An accreditation scheme is used to decide which machine will add directories at what time. It may happen so that the same client will be able to access a directory in one machine while it will be denied access for that in some other machine.

8.9.1.3 Implementation Techniques

In order to manage the mapping easily, sets of files are aggregated to some component units and mapping is done on these components. The textual files are mapped to low level file identifies that helps in finding the component into which the file belongs. Structured names

are used to implement low level identifiers. They are string of bits containing two parts: first part is for the component unit and second part is for the file within the unit. To maintain the uniqueness of a file, sufficient bits are used to ensure that the name used for the file is not being used by any other file. Another way to maintain uniqueness is to add a timestamp with the name.

Space for learners:

8.9.2 Remote File Access

If a user wants remote access to a file, the server that is storing the file is identified by the naming scheme. The data transfer procedure for remote files is similar to that of traditional disk-access method. In case of traditional disk-access method, caching is used to reduce the input/output of disk, whereas in remote file access, caching helps both in reducing disk input/output and network traffic. When a request for a disk-access comes, it is first checked in the cache, if it's not present then it has to be brought from the server. A cache mapping technique (least recently used) is applied to store the files in the cache so that next time when an access request comes, there would not be any need to go to the server, thus reducing the network traffic. One master copy resides in the server and replicas of that copy exists in different caches. As a file in cache is modified, that modification should be reflected in the master copy to maintain the consistency. The concept of demand paging is also implemented exactly same as that of traditional file systems, except that the backing store is a remote server rather than a local disk.

Consistency is another issue in file access. A client machine may want to check whether the cached copy of data is same with the master copy. For verifying the data, two approaches are used: in client-initiated approach, the client will start a validity check by contacting the server and checking whether the replica of the file is consistent with the master copy. The validity check may be done on every access or on first access of the file. In server initiated approach, the server reacts to the inconsistencies. The server is notified when the same file is opened in conflicting mode (Read-write, write-write) by two different clients. It can disable caching for that particular file to avoid inconsistency.

Space for learners:

Stop to Consider

Unlike the conventional file system, a distributed file system has files from different geographical areas. The storage details of the file are hidden from the end-user by a method called abstraction. The DFS also ensures the features like location transparency and location independence. For remote access of a file, caching is used to reduce the network traffic and latency. Consistency is maintained in a DFS with the help two approaches: client initiated approach and server initiated approach.

8.9.3 Stateful Versus Stateless Service

Stateful service is a connection-oriented service. Here a client first gives open ( ) command before accessing the file. The server stores the file in its memory and sends a unique connection-identifier to the file. The same identifier is used for all the file-access during a particular session. An example of stateful service is AFS(Andrew File System).In stateless services, each request can recognize the file in the memory along with the read/write access of the file. There is no need of establishing and terminating a connection here. There is no concept of session and each file request is considered as individual request. NFS (Network File System) is a stateless service.

The performance is better in case of a stateful service as it can store the files in its cache memory thus reducing the disk access which can't be done by stateless service. Again a server in the stateful service knows where a file is open, thus it can directly read the next blocks of the file in case of sequential access. In case of a failure, the server of a stateful protocol losses its states and a recovery protocol is needed to restore the state. A stateless protocol does not face these problems since all the requests are self-contained.

8.9.4 File Replication

Replication is the process of keeping multiple copies of the same data in different nodes of a distributed system. The reasons behind data replication are as follows: it provides better availability. The

system can work even if one or more nodes fail. Replication reduces the latency of a file access, as the file is kept in a short distance from the user. Since the file read operation is a non-conflicting one, the read query request for multiple hops can be performed from different replicas thus increasing the throughput of the overall system.

One main problem associated with replication is consistency. Since file replication is transparent to the user, changes made in one copy should be reflected to all the other copies. Consistency models are broadly divided into client centric consistency model and data-centric consistency model. In client-centric consistency, data may not be updated in all the servers parallelly rather; they are propagated from one server to another. Therefore, some of the processes may have to work with the old data which results in compromising the consistency. But lower consistency may give server availability and increased throughput. Some of the models that come under client-centric consistency are: eventual consistency; where data is updated at the end, leading to inconsistent data in some servers. This model suffers from lost update problem. Next model is the monotonic read; here, if a process reads a data item x, the successive processes will get either the same value or the latest value of x. Another model is monotonic write, where a sequence is maintained for all the write operations of a process and value is updated based on that sequence. Next to this model is the read your write model, where the write operation performed by a process is reflected in the server where a read operation is being performed on the same file. Next comes the write follow read model where a process reads the value before performing the write operation.

In data-centric consistency models an updated query is immediately reflected in all the other servers. Therefore many update operation are being performed at the same time. Different models are there in data-centric consistency model. The strongest of all is the external consistency. Any process that reads the value of a data-item x, will get the last updated value of the write operation. Google Spanner Distributed Database uses this consistency model. In situation-dependent consistency, the execution order of two processes is reflected in the same order into all the servers.

Some of the applications of distributed file systems are:

Space for learners:

8.9.5Andrew File System (AFS)

AFS was designed for the operating systems like BSD (Berkeley Software Distribution), UNIX and Mach. An AFS is made up of the structural elements called cells. Cells consist of servers and client machines. Servers and clients belong to a particular cell. The users can have accounts in more than one cell. The first cell, that a user logs in, is called home cell and all the other cells are named as foreign cells. Irrespective of the location of the user, the path of a file in the AFS tree will always be same. File access permissions such as read, write and update are set by the access control lists. AFS follows stateless protocol, therefore servers and clients don't store the file access information. AFS runs on TCP/IP. The Remote Procedural Call (RPC) that is designed for AFS performs the communication between client and server irrespective of their geographical area. Caching is applied to reduce the network load. For each file access request, the respective file is searched in the cache. If the file is not present then it will be accessed from the server. In case there is any modification in the cached copy of the file, the file is propagated back to the server. The frequently accessed files are stored in "working set" of the cache, thus can be accessed directly from the cache reducing the latency and network load. Generally the size of cache memory is 100MB.

8.9.6Google File System (GFS)

It was developed in the year 2003 to meet the growing demand of data processing systems. The system is scalable and can support a large number of clients without degradation of performance. It supports fault tolerance while implemented in inexpensive hardware. Like other Distributed File System, GFS ensures reliability transparency and availability. In addition to these, some specific design goals included in GFS are: a huge amount of data can be stored redundantly in inexpensive computers. It is capable of processing huge number of requests.

GFS is based on cluster based architecture. A cluster consists of one master node that manages Meta data, a number of chunk-servers that store the files in chunks and a number of clients. The Meta data includes the information about who can have access to the file, mapping the files to the memory chunks and determining the current location of the chunks. GFS is a stateful system therefore it manages

Space for learners:

the state information of all the clients. The client sends file access request to the master node. In response to the request, the master node sends Meta data to the clients. The client can then directly contact the chunk server for the file. Fault tolerance feature is also implemented in the system by keeping three replicas of the same files. If a chunk server is down, then the master server can redirect the client to one of the replicas. In case the master is down, any of the chunk servers can act as a master by keeping a Meta data list.

Space for learners:

8.10 SUMMING UP

- A Distributed System is a collection of independent computer systems that have their own memory and processor.
- They may be of different specifications from single microprocessor to general purpose computers connected through some communication medium like twisted pair cables, fiber optic cables and satellite communication.
- They can be arranged either in client-server architecture or in a peer to peer network.
- The internal working of a distributed systems deals with different issues like name resolution, routing strategies, collision avoidance techniques, connection strategies and it should solve contention problem and ensure security.
- Each message travels through the layers of networking models before reaching the destination. Implementation details of a distributed system should be transparent to the user. S/he should feel like working in a conventional system having no difference between a remote file and a local file.
- The system should be scalable in different aspects like size, administrative organization and geographical area. If some part of the system fails, the performance of overall system should not degrade.
- A DFS is a file system consisting of geographically dispersed clients and servers. A user of a DFS is unaware of the location of a file it wants to access. A local and remote file appears same to the user.

- Different DFS have been designed based on technical needs. Each of which have their own tradeoffs and advantages. Some of them are useful in small networks while some are designed for large distributed systems.

Space for learners:

8.11 ANSWERS TO CHECK YOUR PROGRESS

1. Remote Procedural Call
2. Reliability
3. Peer to Peer Network

8.12POSSIBLE QUESTIONS

1. What do you mean by a Distributed System? Explain the advantages of a distributed system.
2. Describe the Architecture of Client-Server system and Peer-to-Peer system
3. Briefly discuss the technologies used in LAN and WAN network
4. Describe the concept of Domain Name System. How does it help in name resolution?
5. Describe the methods used for collision resolution
6. How does a Routing strategy help in transmitting a file in a distributed system?
7. Discuss some of the fundamental differences between Andrew File System (AFS) and Google File System (GFS).

8.13 REFERENCES & SUGGESTED READINGS

- Distributed Systems: Principles and Paradigms Second Edition Andrew S Tanenbaum, Maarten Van Steen
- Operating System Concepts Seven Edition SILBERSCHATZ GALVIN GANGE