4. Introduction to Operating System

Introduction

Operating system is a collection of programs. It is a software that supports a computer's basic functions. It acts as an interface between applications and computer hardware as shown below in the figure 4.1. Examples of operating systems are Windows, Linux, MacOS, etc.

Figure 4.1 : OS Position

Operating system also does the task of managing computer hardware. Operating system manage hardware by creating a uniform set of functions or operations that can be performed on various classes of devices (for instance read x bytes at address y from hard drive of any type, where x is the number of bytes read). These general functions rely on a layer of drivers that provide specific means to carry out operations on specific hardware. Drivers usually are provided by the manufacturers, but the OS must also provide a way to load and use drivers. OS must detect the device and select an appropriate driver if several are available.

In the figure 4.2 below, OS interface is a set of commands or actions that an operating system can perform and a way for a program or person to activate them.

Operating system is core software component of computer. It performs many functions and acts as an interface between computer and the outside world. A computer consists of several components including monitor, keyboard, mouse, and other parts. Operating system provides an interface to these parts through 'drivers'. This is why, when sometimes we install a new printer or other piece of hardware, our system will ask us to install more software called driver. A brief on device drivers is given in section 4.2.4, Device Management.

Operating system basically manages all internal components of computer. It was developed to better use the computer system.

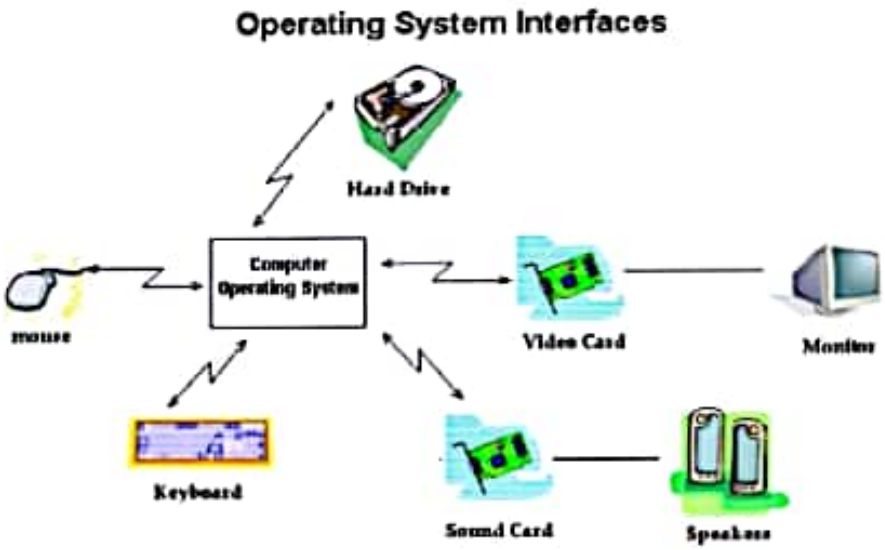


Figure 4.2 : Operating System Interfaces

4.1 Basic Resources Managed By Operating System

Following are the basic resources that are managed by operating system:

- Memory space (Memory management)
- CPU time (Process management)
- Disk space (File system management)
- Input-output device (Device management)

4.1.a Basics Of Memory Management

Any program needs to be loaded in memory (RAM) before it can actually execute. Random Access Memory (RAM) is a type of computer memory that can be accessed randomly, i.e. any byte of memory can be accessed without touching or traversing through the preceding bytes. RAM these days is available with various storage capabilities (256MB, 512MB, 1GB, 2GB, 4GB, 6GB and 8GB sizes).

Memory management deals with managing computer's primary memory. It decides the amount of memory to be allocated to a particular program. It also keeps track of free or unallocated as well as allocated memory. Memory management applies only to RAM.

The sole motto of memory management to be done by operating system is how best to utilize the available memory. In this process we should try to minimize fragmentation.

I. Fragmentation

It occurs when there are many free small blocks in memory that are too small to satisfy any request. In computer storage, fragmentation is a phenomenon in which storage space is used inefficiently, resulting in to reduced capacity as well as performance. Fragmentation also leads to wastage of storage space. The term also refers to the wasted space itself.

There are three different types of fragmentation:

- External fragmentation
- Internal fragmentation
- Data fragmentation

It can be present in isolation or conjunction. When a computer program is finished with a partition, it can put the partition back to the computer. The size and amount of time for which a partition is held by a program varies. During its life span, a computer program can request and free many partitions of memory.

When a program starts, there are long and contiguous free memory areas. Over the time and with use, these long contiguous regions become fragmented into smaller contiguous areas. Eventually, it may become impossible for the program to request large partitions of memory.

- **Internal Fragmentation**

Internal fragmentation is the space wasted inside the allocated memory block because of the restriction on allowed sizes of allocated blocks. Allocated memory can be slightly larger than requested memory. The size difference is the memory internal to partition but would not be used. Internal fragmentation is difficult to reclaim. Design change is the best way to remove internal fragmentation. For example, in dynamic memory allocation, memory pools cut internal fragmentation by spreading the space overhead over a larger number of objects.

- **External Fragmentation**

External fragmentation happens when a dynamic memory allocation algorithm allocates some memory and a small piece is left over which cannot be used effectively. The amount of usable memory reduces drastically, if too much external fragmentation occurs. The total memory space available is enough to satisfy a request but is not contiguous and so, is wasted. Here the jargon 'external' is used to depict that the storage that is further unusable is outside the allocated regions.

For example, consider a situation wherein a program allocates three continuous blocks of memory and then frees the middle one. The memory allocator can use this free block of memory for future allocations. However, this free block cannot be used for allocation if the memory to be allocated is larger in size than the free block. External fragmentation also occurs in file systems as many files of different sizes are created, size changed and deleted. The effect is even worse if a file which is divided into many small pieces is deleted, because this leaves almost equal small regions of free space.

- **Data Fragmentation**

Data fragmentation occurs when a collection of data in memory is broken up into many pieces that are not close enough. It is typically the result of attempting to insert a large object into storage that has already suffered external fragmentation.

ii. Memory Management Techniques

- Single Contiguous Allocation
- Partitioned Memory Management Scheme (Refer Glossary for scheme definition)
- Paged Memory Management
- Segmented Memory Management

Single Contiguous Allocation

Single Contiguous Allocation is the simplest form of memory management. In this type of memory management, all memory (with an exception of small portion reserved for running the operating system) is available for a single program to run. So only one program is loaded in all available memory and so generally the rest of memory is wasted. Simplest example of this type of memory management is MS-DOS. Advantage of single contiguous memory allocation is, it supports fast sequential and direct access. Provides good performance and the number of disk seek required is minimal. Disadvantage of single contiguous memory allocation is fragmentation.

Figure 4.3 : Single Contiguous Allocation

Partitioned Memory Management Scheme

This type of memory management divides primary memory into multiple memory partitions, usually contiguous areas of memory. Memory management here consists of allocating a partition to a job or program when it starts and unallocating it, when job or program ends. A typical scenario of partitioned memory management is depicted in the figure 4.4 below. This type of memory management also needs some hardware support.

The figure 4.4 below shows multiple memory partitions. Each partition runs a process (Refer Glossary). Process 1, Process 2, etc. are the processes allocated to a particular partition in memory. Once the process is completed, partition gets empty. A is the base register address. Base register contains the lowest memory address a process may refer to. We can get length of a partition from Bounds register. Bounds register stores upper and lower bounds on addresses in a time sharing environment. Time sharing environment refers to concurrent use of computer by more than one user – users share the computer's time. Time sharing is the term used synonymously with multi-user.

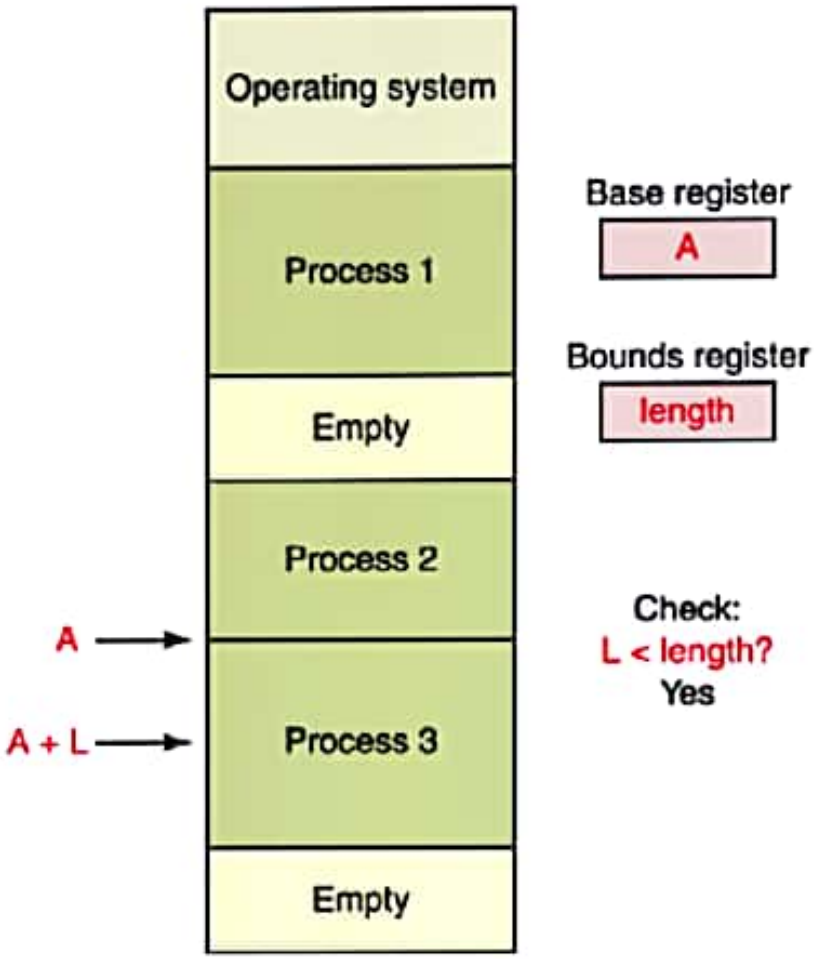


Figure 4.4 : Partitioned Memory Management

Partitioned memory management is further divided in to 2 types of partitioning schemes:

- Fixed memory partitioning (Static partitioning)

Main memory is divided into a number of static partitions when a computer system starts. A process may be loaded into a partition of equal or greater size. Advantage of such type of partitioning is that it is simple to implement and has a little operating system overhead. Disadvantages are inefficient use of memory due to internal fragmentation. Also maximum number of active processes is fixed.

- Variable memory partitioning (Dynamic partitioning)

Such partitions are created dynamically so that each process is loaded into a partition of exactly the same size as that process. Advantage of such type of partitioning is that there is no internal fragmentation. Also main memory is used more efficiently. Disadvantage is inefficient use of processor due to the need for compaction to counter external fragmentation.

For further details on types of partitioning please refer below link:

http://www.csbdu.in/econtent/Operating System/unit3.pdf

Paged Memory Management

This type of memory management divides computer's primary memory into fixed-size units known as page frames. The program's address space is divided into pages of the same size. Usually with this type of memory management, each job runs in its own address space. Advantage of paged memory management is that there is no external fragmentation. But on the contrary there is a small amount of internal fragmentation.

Below is the Illustration representing paged memory management. It depicts how logical pages address in memory can be mapped to physical address. The operating system uses base address as a measure to find addresses. Base address means starting address of a particular memory block. According to the program written, CPU generates address. In this case, address generated by CPU is called logical address. This address is added to base address so that it forms physical address. To translate a logical address into corresponding physical address, we need to divide the logical address into page number and offset. Offset refers to a value being added to base address to produce a second address.

For example, if B represents address 200, then the expressions, B+10 would signify the address 210. 10 in expression is the offset. To specify addresses using an offset is called relative addressing because the resulting address is relative to some other point. Offset is also known as displacement. In figure below, p represents page number and d is the offset. We use page number as index to page table and the entry gives corresponding frame number. 'f' in page table represents frame number. Frame number is concatenated with the offset to get a corresponding physical address.

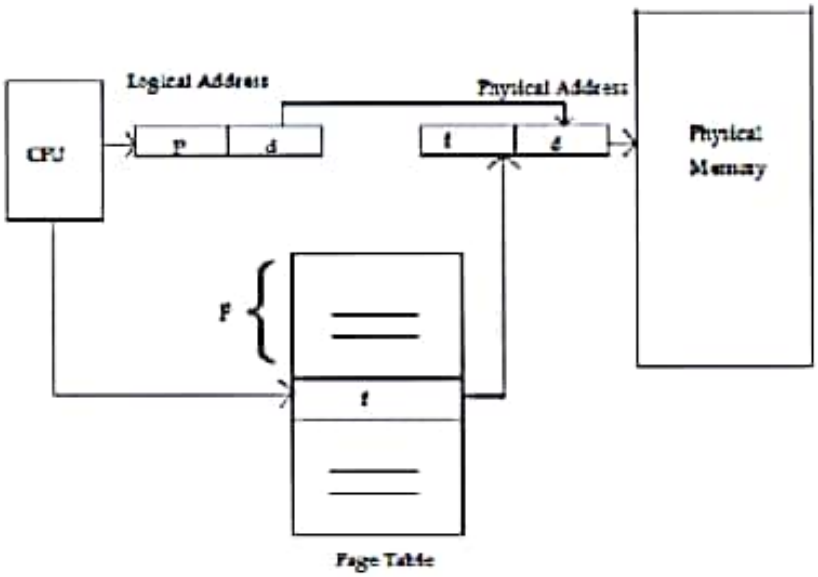


Figure 4.5 : Paged Memory Management

Segmented Memory Management

Segmented memory means, the division of computer's primary memory into segments or sections as shown in figure 4.6. In this type of memory management, there is reference to a memory location that includes a value . This value identifies a segment and an offset within that segment. We may create different segments for different program modules, or for different types of memory usage such as code and data segments. Certain memory segments may even be shared between programs. Segmented memory does not provide user's program

with a linear and contiguous (or continuous) address space. Linear address space is a memory addressing scheme used in processors where the whole memory can be accessed using a single address that fits in a single address or instruction.

Segments are areas of memory that usually correspond to a logical grouping of information such as code procedure or a data array (Refer Glossary). Segmentation allows better access protection than other memory management schemes because memory references are relative to a specific segment and hardware will not permit the application to reference memory not defined for that segment. It is possible to implement segmentation with or without paging.

Segmentation With Paging has the following advantages:

1. Paging eliminates external fragmentation. As a result it provides efficient use of main memory.
2. Segmentation which is visible to the programmer, includes strength of paging. It has the ability to handle growing data structures, support for sharing, and modularity, and protection.

To combine fragmentation and paging, a user's address space is broken into a number of segments. Each of these segments in turn is broken into fixed-size pages. If size of a segment is less than a page in length, then the segment occupies just one page.

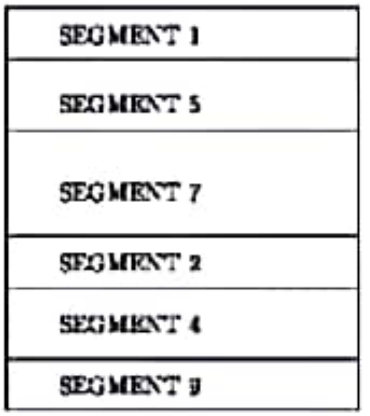


Figure 4.6 : Segmented Memory Management

4.1.b Process Management

CPU has multiple processes running at a time. CPU always has certain programs scheduled in a queue waiting for their turn to execute. These all programs must be scheduled by operating system.

Operating system must allocate resources to processes, enable processes to exchange and share information, enable synchronization among processes, and protect resources of each process from other processes. Synchronization means the coordination of events or processes so that a system operates in unison.

Process creation involves four principle events:

- System initialization or start up. When an OS is booted, typically several processes are created.
- A process is run to execute a process creation system call. Most of the times, a running process will issue system calls to create one or more new processes to help it do its

job. Creation of new processes is useful when the work to be done can be easily formulated as several related, but otherwise independent interacting processes.

- Request to create a new process is issued by user. In interactive systems, users can start a program by clicking or double clicking the icons or thumbnails or even by typing a command on command prompt.
- Batch job initiation. Users can submit batch jobs to the system (most of the times, remotely). When OS knows that it has the resources to run another job, a new process is created and the next job is run from the input queue.

Several processes are created when an operating system boots. Some of them are foreground processes while others are the background processes. Foreground processes interact with a user and perform work for users. In multiprocessing environment, the process that accepts input from the keyboard or other input device at a particular point in time is at times called the foreground process. This means that any process that actively interacts with user is a foreground process. Background processes are the processes which are not associated with a particular user, but instead have some specific function to be performed. For example email notification is an example of background processes. System clock displayed at the bottom right corner of status bar in windows is another example of background process.

Some of the common reasons for process termination are:

- User logs off
- A service request to terminate is executed by process
- Error and fault conditions
- Normal completion
- Time limit exceeded
- Memory unavailable
- I/O failure
- Fatal error, etc.

Figure 4.7 on next page contains great deal of information.

- Let us consider a running process 'P' that issues an input-output request
    - The process blocks.
    - Thereafter, at certain point of time later, a disk interrupt occurs and the driver detects that P's request is satisfied.
    - P is unblocked, i.e. the state of process P is changed from blocked to ready.
    - Later, at some point of time, the operating system looks for a ready job to run and picks the process/job P from the queue.
- A pre-emptive scheduler in figure 4.7 below has the dotted line 'Pre-empt'; where as a non-pre-emptive scheduler doesn't.
- The number of processes change for two arcs namely, create and terminate.
- Suspend and resume are a part of medium term scheduling
    - It is done on a bit longer time scale.

- It involves memory management as well.
- It is also known as two level scheduling.

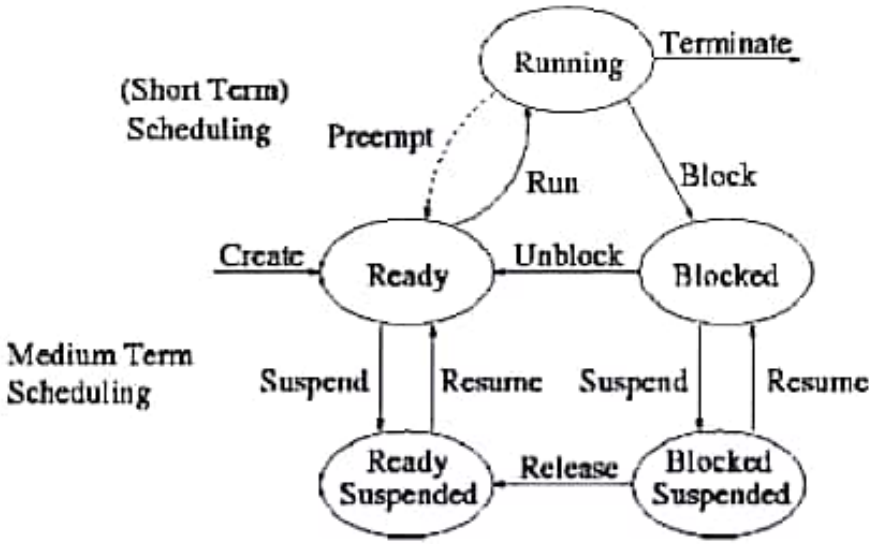
There are 2 types of scheduling:

1. **Pre-emptive scheduling:**

In a computer, tasks are assigned priorities. At times it is necessary to run a certain task that has high priority before another task (even if the task is in running state). Therefore, the running task is interrupted for some time, put to either blocked or suspended state and resumed later, when the priority task has finished its execution. This process is called pre-emptive scheduling.

2. **Non pre-emptive scheduling:**

In non-pre-emptive scheduling, a running task executes till it completes fully. It cannot be interrupted. That means when a process enters the running state, it cannot be deleted from the scheduler until it finishes its service time.



Unblock is done by another task (a.k.a wakeup, release, allocate, V)
Block a.k.a sleep, request, P

Figure 4.7 : Process Management

The figure 4.7 above shows various states of a process. Initially the process is created. Once created, it goes to the ready state. Ready state means the process is ready to execute. It has been loaded into the main memory and is waiting for execution on a CPU. There can be many ready processes waiting for execution at a given point of time. A queue of processes which are ready to execute gets created in memory. One of the processes from that queue is picked up for execution and its state gets changed to running. A running process can be blocked. Blocked state means a process is blocked on some event.

A process may be blocked due to various reasons such as when a particular process has exhausted, the CPU time allocated to it, it is waiting for an event to occur. Blocked process

can either move to ready state or can move to suspended state. In systems that support virtual memory, a process may be swapped out, that is, it would be removed from main memory and would be placed in virtual memory by the mid-term scheduler. This is called suspended state of a process. From here the process may be swapped back in to the ready state. Such state is called ready suspended state. Process that are blocked may also be swapped out. Such a state of process where a process is both swapped out and blocked is called blocked suspended state. Suspended processes can be sent back to ready state only once they are released. This cycle continues till a process finishes its execution i.e. terminated. A process may be terminated from the running state by completing its execution or can be killed explicitly. In either of these cases, we say that the process is terminated.

4.1.c File System Management

Computer contains numerous files. They need to be organized in a proper way so that we can keep track of those files. File retrieval should be easier as well as faster. File system management helps us achieve this.

File system manager is used by the operating system to do file system management. File system manager organizes and keeps track of all the files and directories on secondary storage media.

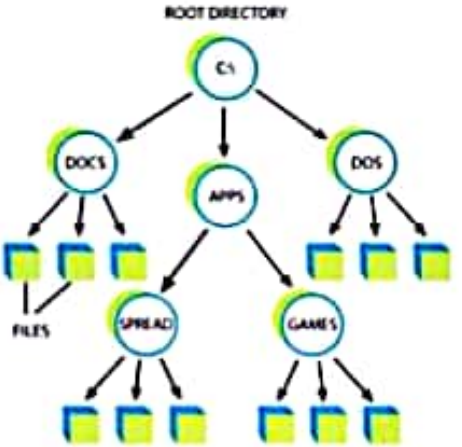


Figure 4.8 : File System Management

Figure 4.8 above shows a typical hierarchical file system structure. Operating system keeps track of following tasks for providing efficient file management:

- It is able to identify numerous files by giving unique names to them.
- It maintains a list to keep track of exact file location.
- Provide fast and simple algorithms to write and read files in co-operation with device manager.
- Grant and deny access rights on files to programs and users.
- Allocate and de-allocate files so that files can be processed in co-operation with process manager.
- Provide programs and users with simple commands for handling files.

On storage medium, file gets saved in blocks or sectors. To access these files, file manager and device manager works together. The device manager knows where to find each sector on disk, but only file manager has a list telling in what sectors either file is stored. This list is called File Allocation Table (FAT). FAT has been in use under DOS for a long time. Some of its alterations are still used by Win95 and Win98.

Below are the different ways of allocating files:

- **Contiguous file allocation:**

In this type of file allocation, at the time of file creation, a single set of blocks is allocated to a file. Each file is stored contiguously in sectors, one sector after another. The advantage is that the File Allocation Table has a single entry for each file, indicating the start sector, the length, and the name. Moreover, it is also easy to get a single block because its address can be easily calculated. The disadvantage may be that it can be difficult to get a sufficiently large block of contiguous blocks. Contiguous file allocation is now a days only used for tapes and recordable CDs.

- **Non-contiguous file allocation:**

With this type of allocation, all blocks of a file can be distributed all over the storage medium. The File Allocation Table (FAT) lists not only all files, but also has an entry for each sector the file occupies. As all information is stored in the FAT, and there is no assumption on the distribution of the file, this method of allocation, at times, is also known as FAT.

The advantage is that it is very easy to get a single block, because each block has its entry in the FAT. Also, it is a very simple allocation method where no overhead is produced and no search method for free blocks is needed. The disadvantage is that FAT can grow to an enormous size, which can slow down the system. Compaction would be needed time to time. This type of file allocation is used for disk allocation in MS-DOS.

- **Chained allocation:**

With chained file allocation only the first block of file gets an entry in the FAT. A block has sectors in it. Of these, the first sector has got data as well as a pointer at its end that points to the next sector to be accessed after it (or indicates that it was the last). That means each sector (if it is not a last sector) has got a pointer at its end pointing to the next sector that should be accessed. The advantage again is that the FAT has a single entry for each file, indicating position of the first sector and file name. There is no need to store the files contiguously.

The disadvantage of chained allocation is that it takes much time to retrieve a single block because that information is neither stored anywhere nor can it be calculated. If we want to access a particular sector, all preceding sectors have to be read in order one after another. This way, the information about location of the next block is identified. Unix i-node is an example of this type of allocation. i-nodes are data structures (Refer Glossary) of a file system used to store all the important properties of each file: owner's group id and user id, access permission on that file, size of the file and more. Unix i-node is an operating system which has i-nodes as its data structure.

- **Indexed allocation:**

With indexed file allocation also only the first block of file gets an entry in the FAT. In the first sector, no data is stored. It has only pointers to where the file is on storage medium. That is why the first block is known as the index block. It just contains the pointers or index to file location on a storage medium. Here also the FAT will only contain a single entry for each file, indicating file name and position of the first sector. It is easy to retrieve a single block because the information about its location is stored in the first block.

The disadvantage of Indexed allocation is that for each file an additional sector is needed to keep track of the location of a file. A very small file also always occupies at least two blocks, where as the data could have easily fit in one block. This results in secondary storage space wastage. Indexed allocation is implemented in all UNIX systems. It is reliable as well as fast. Also, now-a-days, the wastage of storage space does not matter much.
The main concern of file system management is to provide a strategy that lets the FAT not to grow too large and that makes it possible to retrieve a special sector of file such that the storage space is not wasted much.

4.1.d Device Management

Operating system also deals with device management, specially input and output devices. In a multi-user system, there are shared devices like printer, scanner, etc. Management of these devices by sending them appropriate commands is the major task of operating system.
A software routine which knows how to deal with each device is called a "driver" and operating system requires drivers for the peripherals attached to computer. When a new peripheral is added, device driver is installed in to the operating system.
Operating system also deals with the access time of these devices. It helps make the device access fast and in the most efficient way possible.
Major concern of operating system in device management is to prevent dead lock situation.

Dead Lock

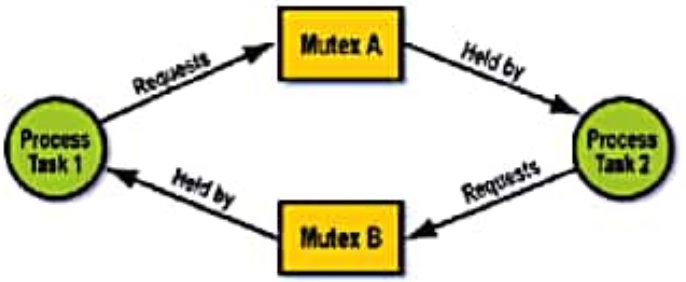


Figure 4.9 : Dead Lock

Above diagram depicts a typical dead lock situation that serially used devices can have. Dead lock is a situation where each set of processes is waiting for an event that only other process in the set can cause.
For example, in the above diagram, Process Task 1 requests a mutex or lock on object which is already held by Process Task 2. In turn, Process Task 2 needs a resource to complete its process which is already held by Process Task 1. So neither of the processes are releasing their resources or mutex and waiting for another process to release a mutex so that they can get it to complete its pending task. This situation would cause an indefinite wait for processes leading to a situation known as dead lock.
Mutex is the short form for Mutual Exclusion Object. A mutex is a logical unit. It is a program object that allows multiple program threads to share the same resource, but not simultaneously. File access is an example of mutex. A mutex with unique name is created when the program starts. Mutex must be locked by a thread that needs the resource. When data is no longer needed, the mutex is set to unlock.

OS helps in dealing with dead lock situation up to a certain extent. Below are some of the strategies for dealing with deadlock:

- Ostrich algorithm, i.e. ignoring the problem altogether. Ostrich algorithm is a strategy of ignoring potential problems on the basis that they may be exceedingly rare. Ostrich algorithm assumes that ignoring the problem would be more cost-effective as compared to allowing the problem to occur and then attempt for its prevention. It may occur very infrequently. This can be followed if cost of detection/prevention is not worth the time and cost spent.
- Detection and Recovery.
- Avoiding deadlock by careful resource allocation.
- Prevention of deadlock by structurally negating one of the four necessary conditions.

4.2 Resource Sharing Management

The primary task of operating system as discussed above is to keep track of resources like memory, CPU utilization, storage device, and input output devices, to grant resource requests, to mediate conflicting requests from different programs, etc.

4.2.1 Multiplexing

Multiplexing is a method by which multiple analogue data signals/digital data streams are combined into one signal over a shared medium. Multiplexing is used to share an expensive resource and thus help reduce the cost.

Communication channel is used for transmission of a multiplexed signal. This channel may be a physical transmission medium. Multiplexing basically works on the principle of division of signals. It divides the capacity of the high level communication channel into several low level logical channels. Each of these low level logical channel maps to each message signal or data stream to be transferred. A reverse of this process is known as de-multiplexing. De-multiplexing can extract the original channels or signals on the receiver side.

A device that performs the task of multiplexing is called a multiplexer (MUX), and a device that performs the task of de-multiplexing is known as a de-multiplexer (DEMUX).

Resource management basically includes multiplexing the resources in two ways:

Time Multiplexing

When a particular resource is time multiplexed, different programs or users get their turn to use that resource one after another. Turns for resource usage are decided as per the predefined operating system algorithm. Printer is one of the best examples of in-time multiplexing. In a network printer, different users get their turn to use the printer one after another based on time.

Space Multiplexing

When a particular resource is space multiplexed, the resource is shared equally in the available space. Time period for which a particular resource is used does not matter anymore. That is each one gets a part of the resource. Hard disk, main memory, etc are some of the examples of space multiplexing. These resources are most of the times divided equally between all the users.

To summarize, operating system is one of major programs, without which it would be almost impossible for us to access the computer as we are accessing it these days. Without operating system, computer would be nothing but a monitor executing binary and machine language data.

Glossary

Term used	Description
Scheme	Scheme in document refers to techniques.
Process	Job or a program that executes in memory.
Data array	Array means an ordered list of items. When data is arranged in an ordered way, it is called data array.
Data Structure	Data structure is a particular way of storing and organizing data in a computer so that it can be used efficiently