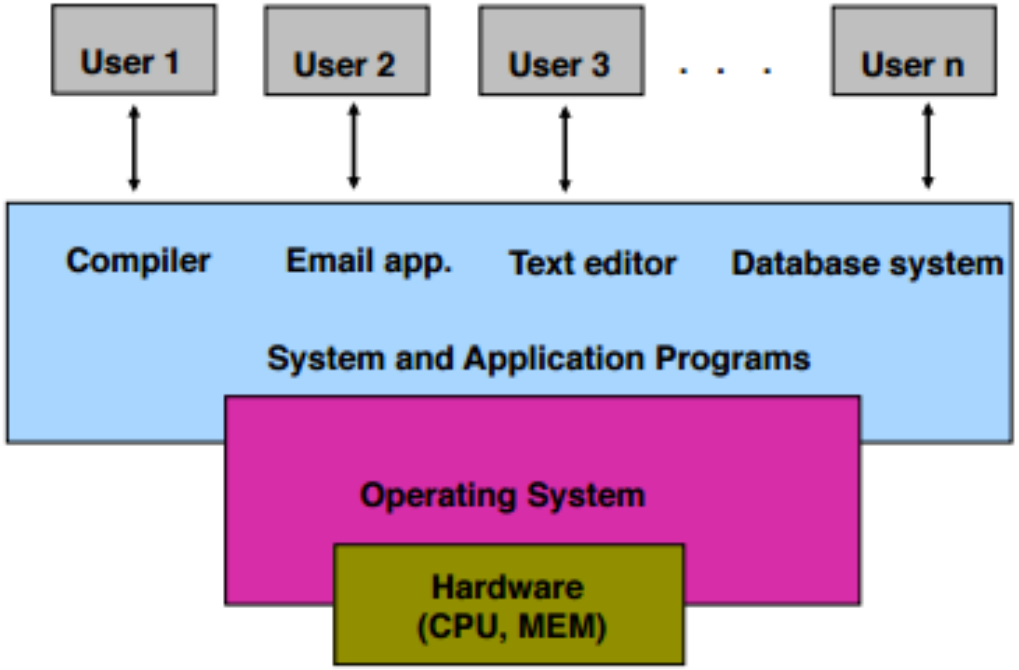
Basic concepts of operating systems

UNIT 2

1. Operating system definition
2. Functions of operating systems
3. Structure and components of operating systems
4. Types of operating systems
5. Resources management
   - **5.1.** Process management
   - **5.2.** Memory management
   - **5.3.** File management
   - **5.4.** I/O management
   - **5.5.** Secondary storage management
6. Licenses

1. Operating system definition

- An **operating system** is a program that acts as an **intermediary** between the user/programs of computer and hardware, which **manages system resources** in a safe and optimal way.



1. Operating system definition

- Responsibilities of the OS include:
  - Running programs.
  - Hiding the complexities of hardware from the user.
  - Managing between the hardware's resources which include the processors, memory, data storage and I/O devices.
  - Handling "interrupts" generated by the I/O controllers.
  - Access control.
  - Error detection.
- Goals of operating systems:
  - Comfort: The main goal of an operating system is to hide all of that detail of using the computer’s hardware.
  - Convenience: It is easier to compute with an operating system than without it.
  - Efficiency: Manage system resources in an efficient manner. This is particularly important for large, share and multi-user systems.

1. Operating system definition

- **Search information about the operating systems evolution**

2. Functions of operating systems

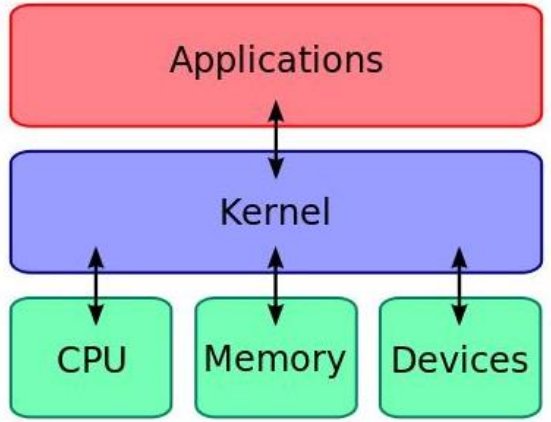
- Following are some of important functions of an operating system:
  - Memory management: Keeps tracks of primary memory, i.e., what parts of it are in use by whom, what part are not in use.
  - Processor management: Decides which process gets the processor, when and for how much time.
  - Device management: Decides which process gets the device, when and for how much time.
  - File management: A file system is normally organized into directories for easy navigation and usage. An operating system does the following activities for file management: Keeps track of information, location, uses or status and decides who gets the resources.

2. Functions of operating systems

- Following are some of important functions of an operating system:
  - Security: By means of password and similar other techniques, it prevents unauthorized access to programs and data.
  - Control over system performance: Recording delays between request for a service and response from the system.
  - Job accounting: Keeping track of time and resources used by various jobs and users.
  - Error detecting aids: Production of dumps, traces, error messages and other debugging and error detecting aids.
  - Coordination between other software and users: Coordination and assignment of compilers, interpreters, assemblers and other software to the various users of the computer systems.

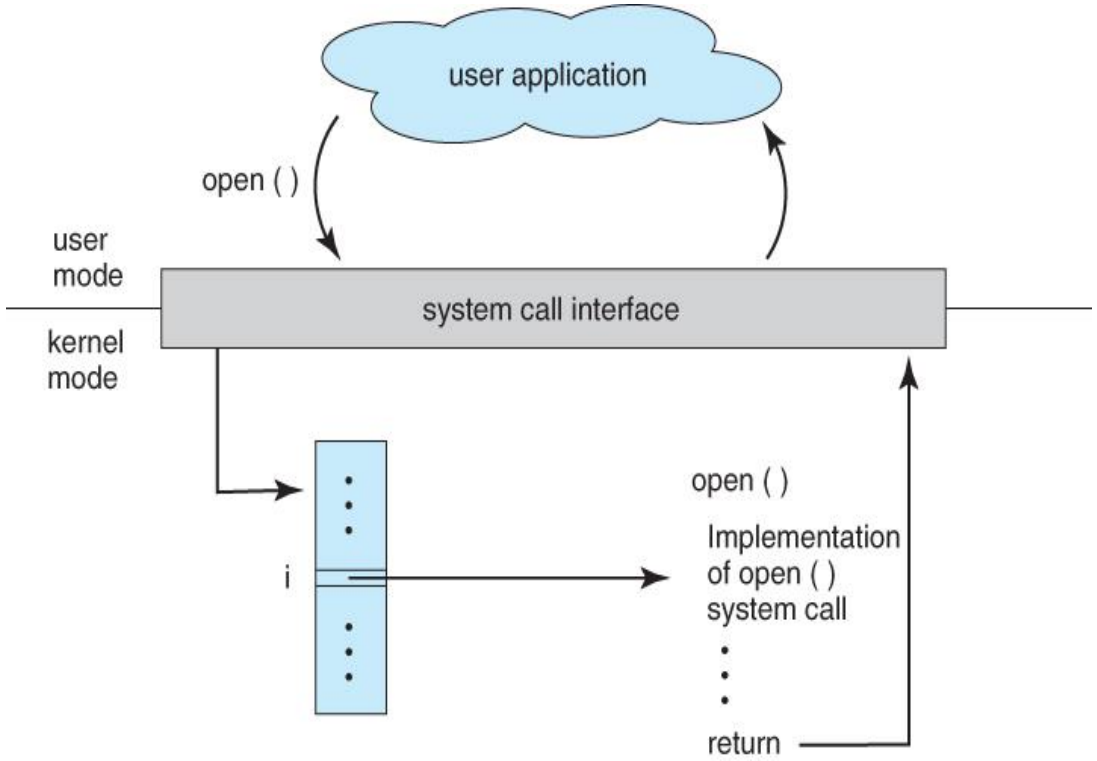
3. Structure and components of operating systems

- What are the main components of an Operating System?
  - **Kernel:** The kernel provides the most basic level of control over all of the computer's hardware devices. It manages memory access for programs in the RAM, it determines which programs get access to which hardware resources, it sets up or resets the CPU's operating states for optimal operation at all times, and it organizes the data with File Systems.
  - **Program execution:** The operating system provides an interface between an application program and the computer hardware, so that an application program can interact with the hardware
  - **Interrupts:** Interrupts are central to operating systems, as they provide an efficient way for the operating system to interact with and react to its environment. Even very basic computers support hardware interrupts, and allow the programmer to specify code which may be run when that event takes place.
  - **Memory management**
  - **Disk access and file systems**
  - **Networking & Security**
  - **Shell** that interprets user orders.
  - **Graphical user interface:** In most of the modern computer systems.



3. Structure and components of operating systems

- A **System call** is the programmatic way in which a computer program requests a service from the kernel of the operating system it is executed on.
- Examples:
  - UNIX: fd = open (“myfile”, O_RDONLY);
  - JAVA: System.out.println(“Hello world!”) ;
  - Shell: mkdir myfolder

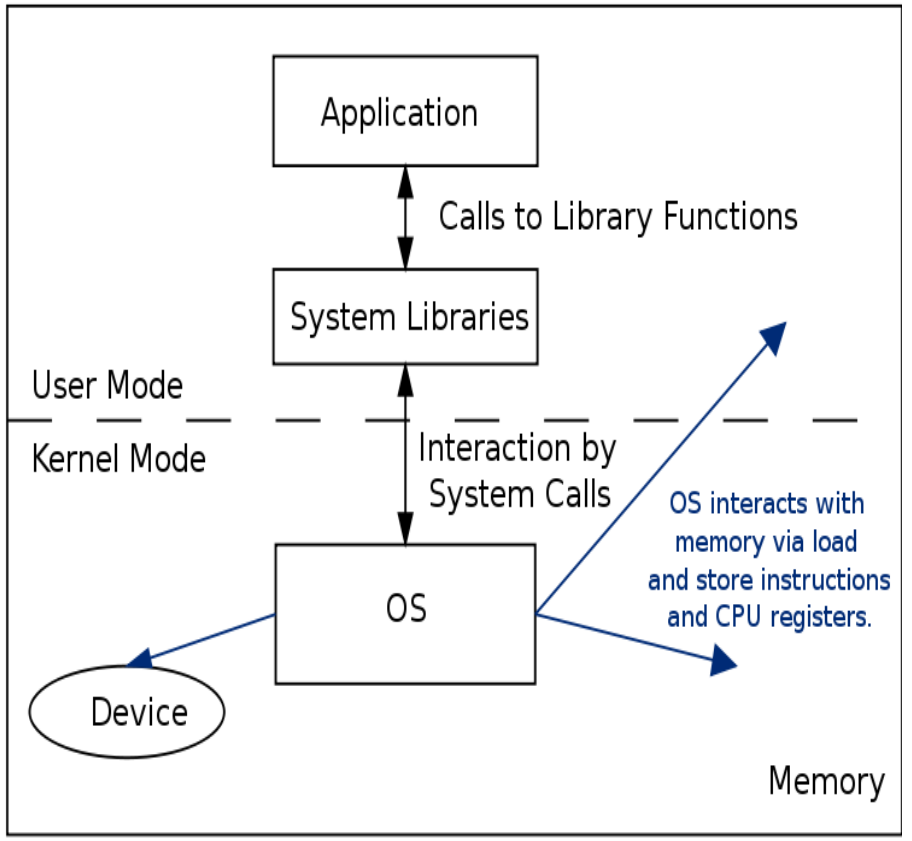


3. Structure and components of operating systems

- System calls can be roughly grouped into five major categories:
  - Process Control: load, execute, end, abort, create process, get/set process attributes, wait for time, wait event, signal event, allocate, free memory.
  - File management: create file, delete file, open, close, read, write, get/set file attributes.
  - Device Management: request device, release device, read, write, get/set device attributes, logically attach or detach devices.
  - Information Maintenance: get/set time or date, get/set system data, get/set process, file, or device attributes.
  - Communication: create, delete communication connection, send, receive messages, transfer status information, attach or detach remote devices.

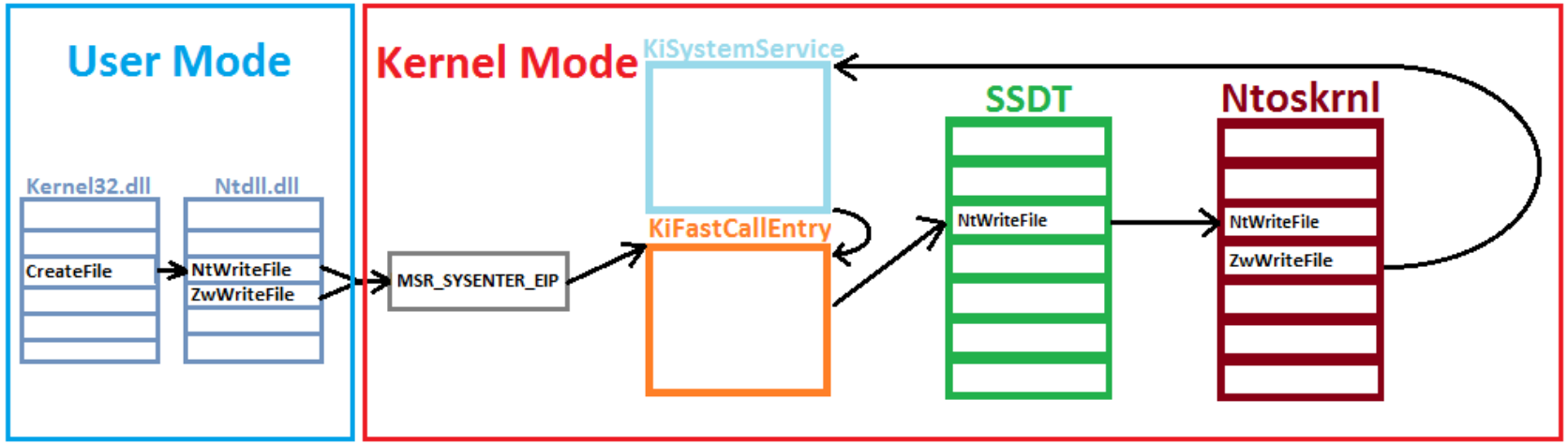
3. Structure and components of operating systems

- In any modern operating system, the CPU is actually spending time in two very distinct modes:
  1. Kernel mode: The executing code has complete and unrestricted access to the underlying hardware. It can execute any CPU instruction and reference any memory address.
  2. User mode: In user mode, the executing code has no ability to directly access hardware or reference memory.



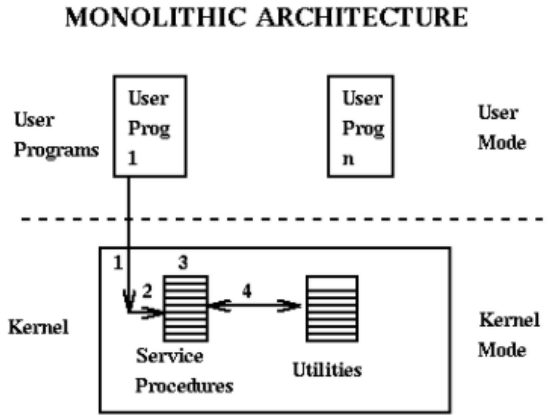
3. Structure and components of operating systems

- Most important functions in kernel mode:
  - Processor planning.
  - Virtual memory management.
  - File system management.
  - Call system management.
  - Create, destroy and synchronize processes.



4. Types of operating systems

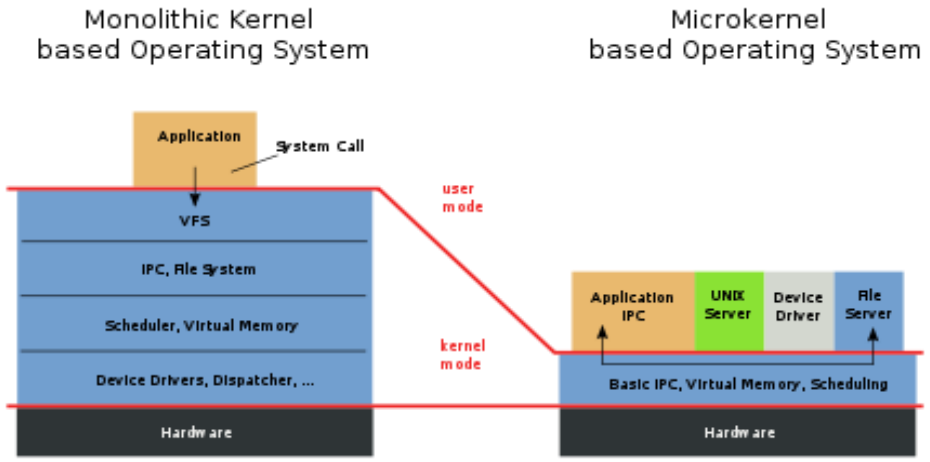
- When classifying operating systems, we can take into account different criteria, which can be combined:
  1. According to kernel organization:
     - A **monolithic system** where the components are organized haphazardly and any module can call any other module without any reservation. The structure is that there is no structure. The operating system is written as a collection of procedures, each of which can call any of the other ones whenever it needs to.
     - A **layered system** divide the operating system into a number of levels, each built on top of lower layers. With modularity, layers are selected such that each uses functions (operations) and services of only lower-level layers.

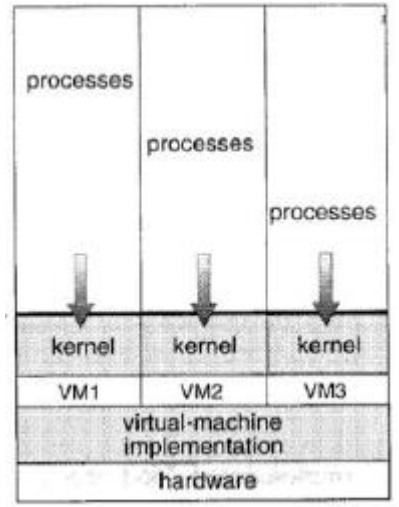


Layer	Function
5	The operator
4	User programs
3	Input/output management
2	Operator-process communication
1	Memory and drum management
0	Processor allocation and multiprogramming

4. Types of operating systems

1. According to kernel organization:
   - A **microkernel** consists on reducing the kernel to only basic process communication and I/O control and let other system services run in user space just like any other normal processes. These services are called servers and kept separate and run in different address spaces A microkernel running in kernel mode delivers the message to the appropriate server.
   - **Virtual machine** approach provides an interface that is identical to the underlying bare hardware. Each process is provided with a (virtual) copy of the underlying computer. The resources of the physical computer are shared to create the virtual machine. CPU scheduling can be used to share the CPU and to create the appearance that users have their own processors.





Virtual Machine.

4. Types of operating systems

2. According to the number of tasks it can perform:
   - A **single-tasking** operating system has only one running program.
   - A **multi-tasking** operating system allows more than one program to be running at the same time.
3. According to the number of users:
   - A **single-user** operating system is a system in which only one user can access the computer system at a time.
   - A **multi-user** operating system allows more than one user to access a computer system at one time.
4. According to the number of processors:
   - A **single-processor** contains only one processor.
   - A **multi-processor** are those who can use two or more central processing units (CPUs) within a single computer system. The term also refers to the ability of a system to support more than one processor or the ability to allocate tasks between them

4. Types of operating systems

5. According to response time:
   - In **time-sharing** in which each process is assigned a fraction of time to use the processor.
   - A **real-time** with strict timing constraints.
6. According to boot mode:
   - **Installable:** It needs to be installed in hard drive to boot.
   - **Live:** It is a complete bootable computer installation including operating system which runs directly from a CD-ROM or similar storage device into a computer's memory.
7. According to resources management:
   - **Centralized:** It can use resources allocated in only one computer.
   - **Network operating system:** Oriented to computer networking, to allow shared file and printer access among multiple computers in a network, to enable the sharing of data, users, groups, security, applications, and other networking functions.
   - **Distributed:** It consists of autonomous computers that are connected using a distribution middleware. They help in sharing different resources and capabilities to provide users with a single and integrated coherent network.

4. Types of operating systems

- **Exercise:**

According to…	Kernel	Response time	Boot mode	Number of users	Number of processors	Number of tasks
Unix						
Solaris						
Ubuntu Live						
Merlin						
MS-DOS						
Windows XP						
Suse						

5. Resources management

5.1. Process management

5.2. Memory management

5.3. File management

5.4. I/O management

5.5. Secondary storage management

5.1 Process management

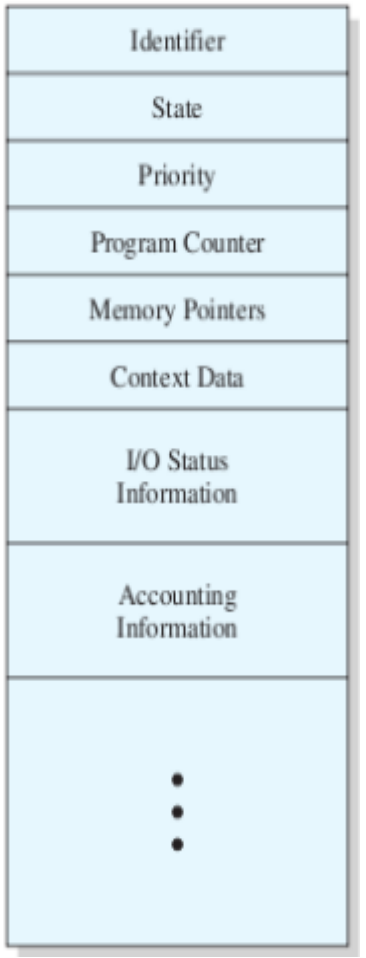
- A process is an instance of a program in execution.
- A program by itself is not a process; a program is a passive entity, such as a file containing a list of instructions stored on disks (often called an executable)
- However, a process is an active entity, with a program counter specifying the next instruction to execute and a set of associated resources.
- A program becomes a process when an executable file is loaded into memory.
- Although two processes may be associated with the same program, they are nevertheless considered two separate execution sequences.
- For instance, several users may be running different copies of mail program, or the same user may invoke many copies of web browser program. Each of these is a separate process, and although the text sections are equivalent, the data, heap and stack section may vary.

5.1 Process management

- Current operating system can load multiple programs into memory and execute them concurrently.
- There are four principal events that cause processes to be created:
  1. System initialization.
  2. Execution of a process creation system call by a running process.
  3. An user request to create a new process.
  4. Initiation of a batch job.
- The operating system is responsible for:
  - Create, execute, resume, suspend and delete processes.
  - Concurrent execution of processes, avoiding deadlock
  - Communication between processes

5.1 Process management

- In operating system each process is represented by a **process control block (PCB)**. A process table stores one entry for PCB element the OS needs to manage processes. PCB contains information like:
  - Identifier: A unique identifier associated with this process, to distinguish it from all other processes.
  - State: If the process is currently executing, it is in the running state.
  - Priority: Priority level relative to other processes.
  - Program counter: The address of the next instruction in the program to be executed.
  - Memory pointers: Includes pointers to the program code and data associated with this process, plus any memory blocks shared with other processes.
  - Context data: These are data that are present in registers in the processor while the process is executing.
  - I/O status information: Includes outstanding I/O requests, I/O devices (e.g., tape drives) assigned to this process, a list of files in use by the process, and so on.
  - Accounting information: May include the amount of processor time and clock time used, time limits, account numbers, and so on.



5.1 Process management

- Each process may be in one of the following states:

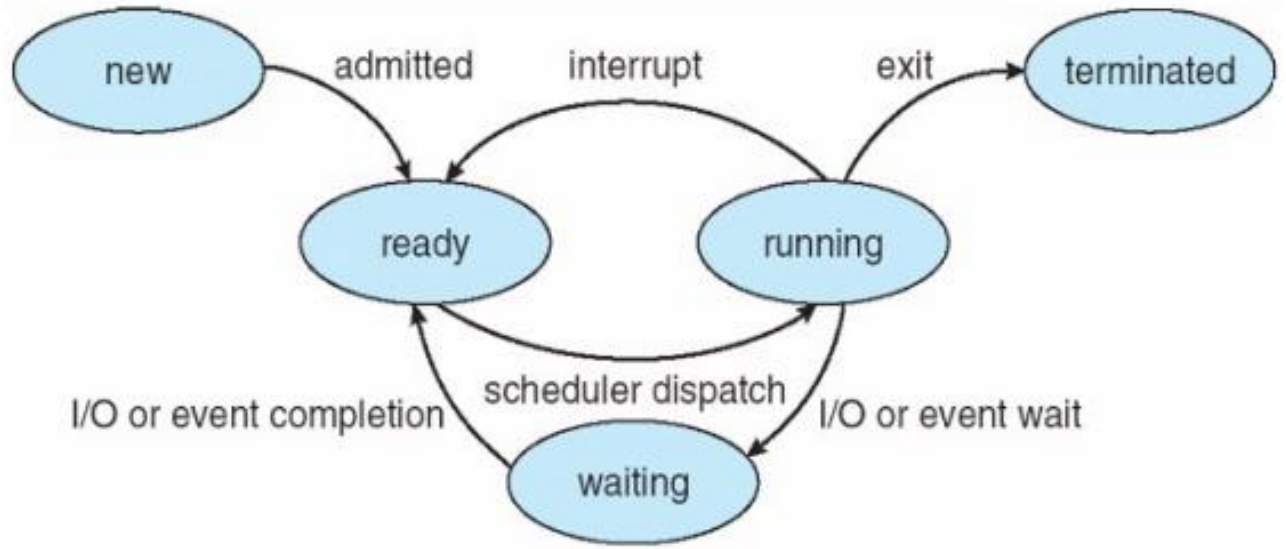


Fig1.3: Process state Transition diagram

5.1 Process management

- Each state means:
  - New: The process is being created.
  - Running: Instructions are being executed.
  - Waiting: The process is waiting for some event to occur (such as I/O completion or reception of a signal)
  - Ready: The process is waiting to be assigned to a processor.
  - Terminated: The process has finished execution.

5.1 Process management

- A process needs to be loaded into the main memory before its execution.
- All the suspended processes are stored into secondary or virtual memory.
- The process **scheduler** is a part of the operating system that decides which process runs at a certain point in time.
- The **dispatcher** is the module that gives control of the CPU to the process selected by the short-term scheduler.
- The scheduler maintains a queue of processes and the dispatcher handles the actual context switch.

5.1.1 Scheduling algorithms

- In a multiprogramming system, frequently multiple process competes for the CPU at the same time.
- When two or more process are simultaneously in the ready state a choice has to be made which process is to run next. As we said before, this part of the OS is called scheduler and the algorithm is called scheduling algorithm.
- Scheduling algorithms can be divided into two categories with respect to how they deal with clock interrupts:

1. **Preemptive** scheduling algorithm picks a process and lets it run for a maximum of some fixed time. If it is still running at the end of the time interval, it is suspended and the scheduler picks another process to run (if one is available).
2. **Non preemptive** scheduling algorithm picks a process to run and then just lets it run until it blocks (either on I/O or waiting for another process) or until it voluntarily releases the CPU.

5.1.1 Scheduling algorithms

- Many criteria have been suggested for comparison of CPU scheduling algorithms.
  - CPU utilization: We have to keep the CPU as busy as possible. It may range from 0 to 100%. In a real system it should range from 40 – 90 %
  - Throughput: It is the measure of work in terms of number of process completed per unit time. For long process this rate may be 1 process per hour, for short transaction, throughput may be 10 process per second.
  - Waiting time: The sum of periods waiting in the ready queue.
  - Turnaround Time: Time a process takes from submission to completion (including waiting in ready queue, execution on the CPU and doing I/O).
  - Response time: In interactive system the turnaround time is not the best criteria. Response time is the amount of time it takes to start responding, not the time taken to output that response.

5.1.1 Scheduling algorithms

FCFS (First Come First Serve)

- Also called **FIFO** (First In First Out).
- Processes are assigned the CPU in the order they request it.
- When the first job enters the system, it is started immediately and allowed to run as long as it wants to.
- As other jobs come in, they are put onto the end of the queue.
- When the running process blocks, the first process on the queue is run next.
- Non-preemptive.

5.1.1 Scheduling algorithms

FCFS (First Come First Serve). Example

Process	Arrival time	Burst time
1	1	3
2	3	5
3	5	4
4	6	2

		Cycles														
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Processes	1	→	1	2	**3**											
	2			→		1	2	3	4	**5**						
	3					→					1	2	3	**4**		
	4						→								1	**2**

Average waiting time = 0+1+4+7=12/4=3

Average turnaround time= 3+6+8+9=26/4=6,5

5.1.1 Scheduling algorithms

SJF (Shortest Job First)

- When several equally important jobs are sitting in the queue waiting to be started, the scheduler picks the shortest jobs first.
- If the Burst Time is equal, use FCFS.
- When the first job enters the system, it is started immediately and allowed to run as long as
- The SJF is either preemptive or non preemptive. Preemptive SJF scheduling is sometimes called Shortest Remaining Time First (SRT), that we will study later.

5.1.1 Scheduling algorithms

SJF (Shortest Job First). Non-preemptive example

Process	Arrival time	Burst time
1	1	3
2	3	5
3	5	4
4	6	2

		Cycles														
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Processes	1	→	1	2	3											
	2			→		1	2	3	4	5						
	3					→							1	2	3	4
	4						→				1	2				

Average waiting time = 0+1+6+3=10/4=2.5

Average turnaround time= 3+6+10+5=24/4=6

5.1.1 Scheduling algorithms

SJF (Shortest Job First) Preemptive or SRT (Shortest Remaining Time) example.

When a new job arrives its total time is compared to the current process remaining time. If the new job needs less time to finish than the current process, the current process is suspended and the new job is started. This scheme allows new short jobs to get good service

Process	Arrival time	Burst time
1	1	3
2	3	5
3	5	2
4	6	4

		Cycles														
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Processes	1	→	3	2	1											
	2			→		5			4	3	2	1				
	3					→	2	1								
	4						→						4	3	2	1

Average waiting time = 0+3+0+5=8/4=2

Average turnaround time= 3+8+2+9=22/4=5.5

5.1.1 Scheduling algorithms

Round-Robin

- In the Round-Robin scheduling, processes are dispatched in a FIFO manner, but are given a limited amount of CPU time called **quantum**.
- If a process does not complete before its CPU-time expires, the CPU is preempted and given to the next process waiting in a queue. The preempted process is then placed at the back of the ready list.
- If the process has blocked or finished before the quantum has elapsed, the CPU switching is done.
- Round-Robin scheduling is preemptive, therefore it is effective in timesharing environments in which the system needs to guarantee reasonable response times for interactive users.
- The most interesting issue with round robin scheme is the length of the quantum. Setting the quantum too short causes too many context switches and lower the CPU efficiency. On the other hand, setting the quantum too long may cause poor response time and approximates FCFS.

5.1.1 Scheduling algorithms

Round-Robin. Example (quantum = 2)

Process	Arrival time	Burst time
1	1	3
2	3	5
3	5	2
4	6	4

Average waiting time = 2+5+1+5=13/4= 3,25
Average turnaround time = 5+10+3+9=27/4=6,75

		Cycles														
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Processes	1	→	1	2			3									
	2			→	1	2				3	4			5		
	3					→		1	2							
	4						→					1	2		3	4
Queue		1		2	1	1 3	3 2 4	2 4	2 4	4	4	2	2	4		

5.1.1 Scheduling algorithms

Priority Scheduling

- A priority is associated with each process, and the CPU is allocated to the process with the highest priority. Equal priority processes are scheduled in the FCFS order.
- To prevent high priority process from running indefinitely, the scheduler may decrease the priority of the currently running process at each clock interrupt. If this causes its priority to drop below that of the next highest process, a process switch occurs.
- Each process may be assigned a maximum time quantum that is allowed to run. When this quantum is used up, the next highest priority process is given a chance to run.

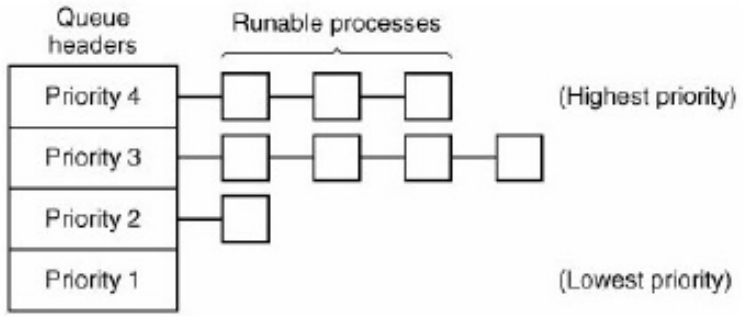


Fig: A scheduling algo. with four Priority classes

5.2 Memory management

- In its simpler forms, this involves providing ways to allocate portions of memory to programs at their request, and freeing it for reuse when no longer needed.
- In a uniprogramming system, main memory is divided into two parts:
  - One part for the OS.
  - One part for the program currently being executed.
- In a multiprogramming system, the user part of the memory must be further subdivided to accommodate multiple processes. The task of subdivision is carried out dynamically by the operating system module called **memory manager**.
- The memory manager functions can dynamically allocate, manipulate, and release memory.

5.2.1 Memory management techniques

- Two major schemes for memory management:
  - **Contiguous allocation:** It means that each logical object is placed in a set of memory locations with strictly consecutive addresses.
  - **Non-contiguous allocation:** It implies that a single logical object may be placed in non-consecutive sets of memory locations. Paging and Segmentation are the two mechanisms that are used to mange non-contiguous memory allocation, as we will study later.
- Two possibilities for contiguous allocation: Fixed partitioning and variable partitioning.

5.2.1 Memory management techniques

Contiguous allocation: Fixed partitioning

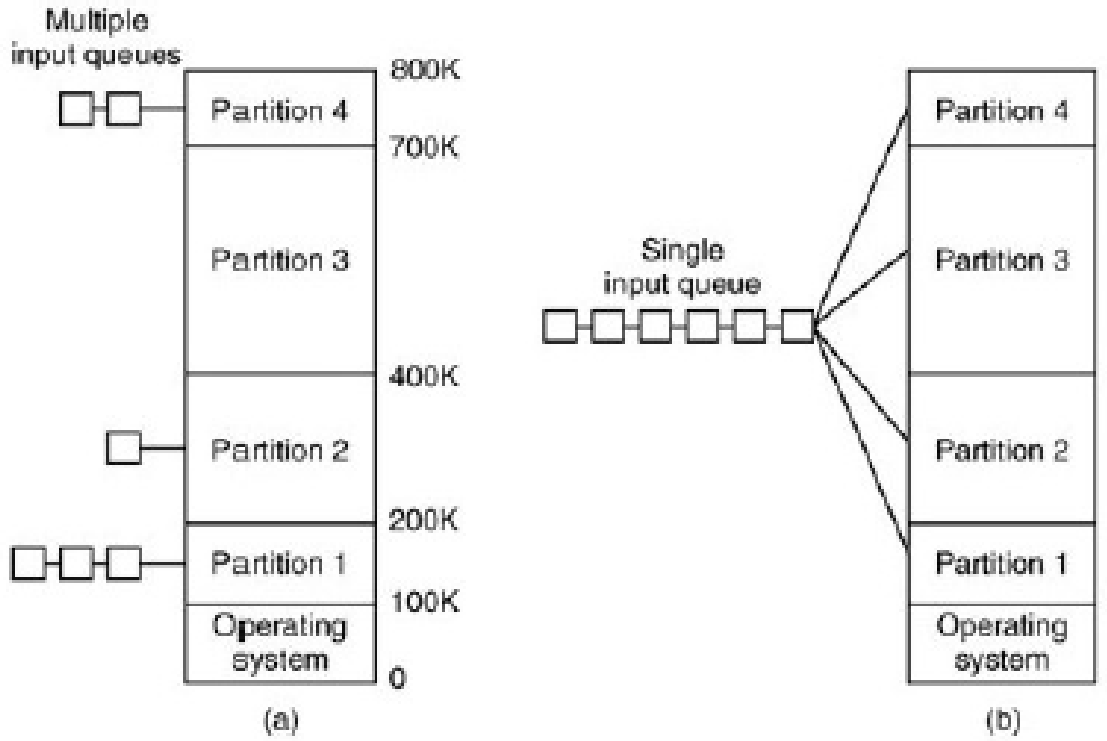
- Main memory is divided into a number of static partitions at system generation time. A process may be loaded into a partition of equal or greater size.
- It is simple to implement, but very inefficient due to internal fragmentation. This phenomenon means that there is wasted space internal to a partition due to the fact that the block of data loaded is smaller than the partition.
- Two possibilities:

a) Equal size partitioning

b) Unequal size Partition

5.2.1 Memory management techniques

Contiguous allocation: Fixed partitioning



(a) Fixed memory partitions with separate input queues for each partition.
(b) Fixed memory partitions with a single input queue.

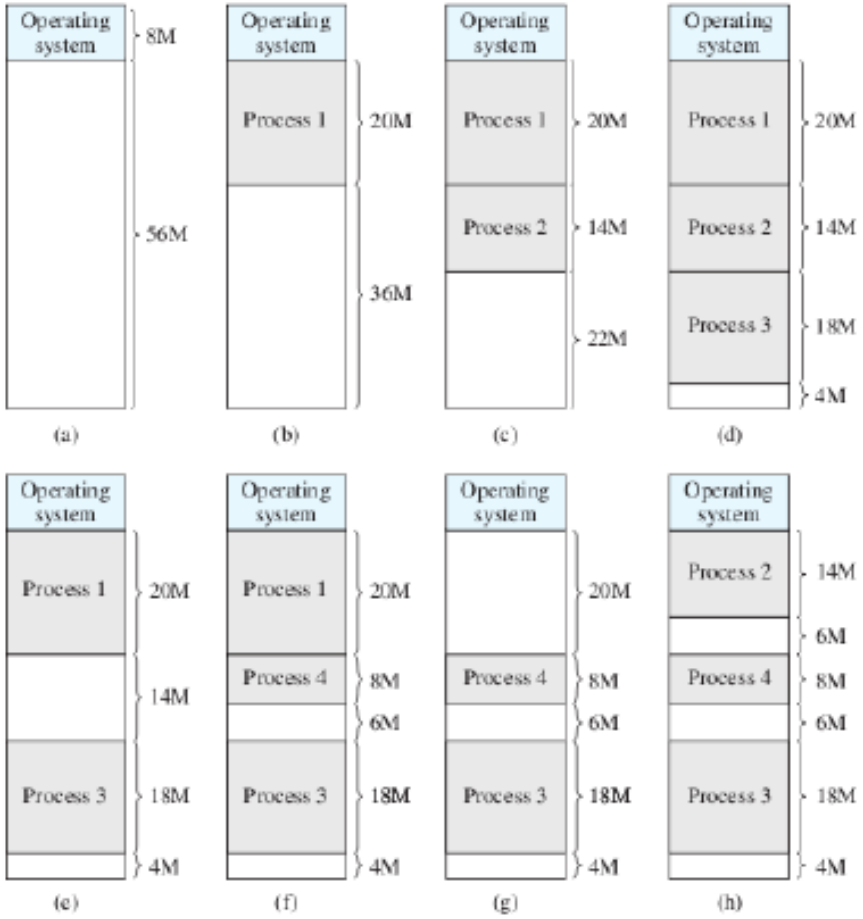
5.2.1 Memory management techniques

Contiguous allocation: Variable partitioning

- The partitions are of variable length and number.
- When a process is brought into main memory, it is allocated exactly as much memory as it requires and no more.
- Eventually it leads to a situation in which there are a lot of small holes in memory and it becomes more and more fragmented, and memory utilization declines.
- This phenomenon is referred to as external fragmentation**.**
- One technique for overcoming external fragmentation is compaction: From time to time, the operating system shifts the processes so that they are contiguous and so that all of the free memory is together in one block.
- This last one may well be sufficient to load in an additional process. The difficulty with compaction is that it is a time consuming procedure and wasteful of processor time.

5.2.1 Memory management techniques

Contiguous allocation: Variable partitioning



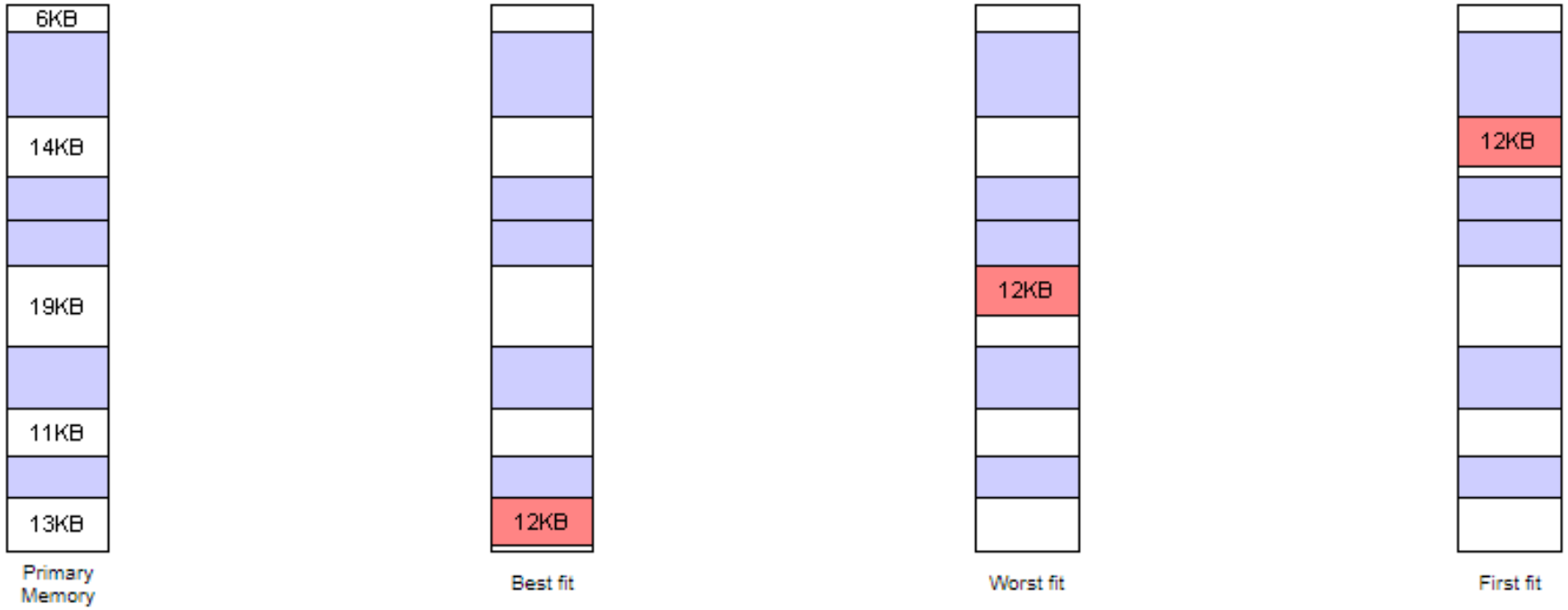
5.2.1 Memory management techniques

Contiguous allocation:

- No matter fixed or variable partitioning, various different strategies are used to allocate space to processes competing for memory.
  - Best fit: The allocator places a process in the smallest block of unallocated memory in which it will fit.
  - Worst fit: The memory manager places a process in the largest block of unallocated memory available.
  - First fit: There may be many holes in the memory, so the operating system, to reduce the amount of time it spends analyzing the available spaces, begins at the start of primary memory and allocates memory from the first hole it encounters large enough to satisfy the request.

5.2.1 Memory management techniques

Contiguous allocation: Example of the different strategies



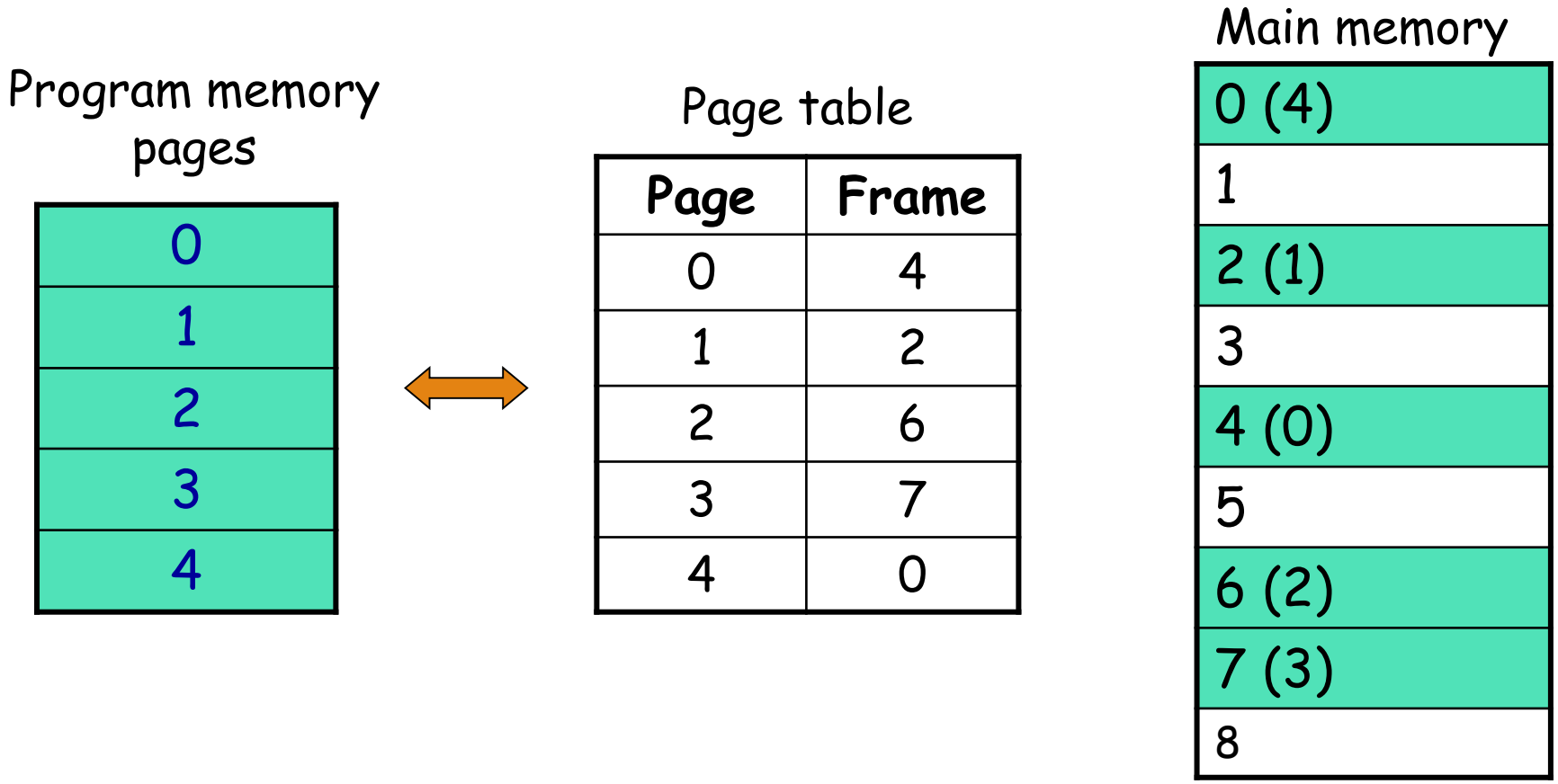
5.2.1 Memory management techniques

Non-contiguous Memory allocation

- Fragmentation is a main problem of contiguous memory allocation. We have seen a method called compaction to resolve this problem, but inefficient.
- An alternative approach is divide the size of new process P into two chunks of 1KB and 2KB to be able to load them into two holes at different places:
  1. If the chunks have to be of same size for all processes ready for the execution then the memory management scheme is called **Paging.**
  2. If the chunks have to be of different size in which process image is divided into logical segments (stack, code, data, etc.) of different sizes then this method is called **Segmentation.**
  3. On the other hand, **Segmented Paging** combines both techniques. There are segments divided at the same time into pages.
- This is the basis of **Virtual Memory,** which can work with only some chunks in the main memory and the remaining on the disk, bringing them into main memory only when required.

5.2.1 Memory management techniques

Non-contiguous Memory allocation. Paging example

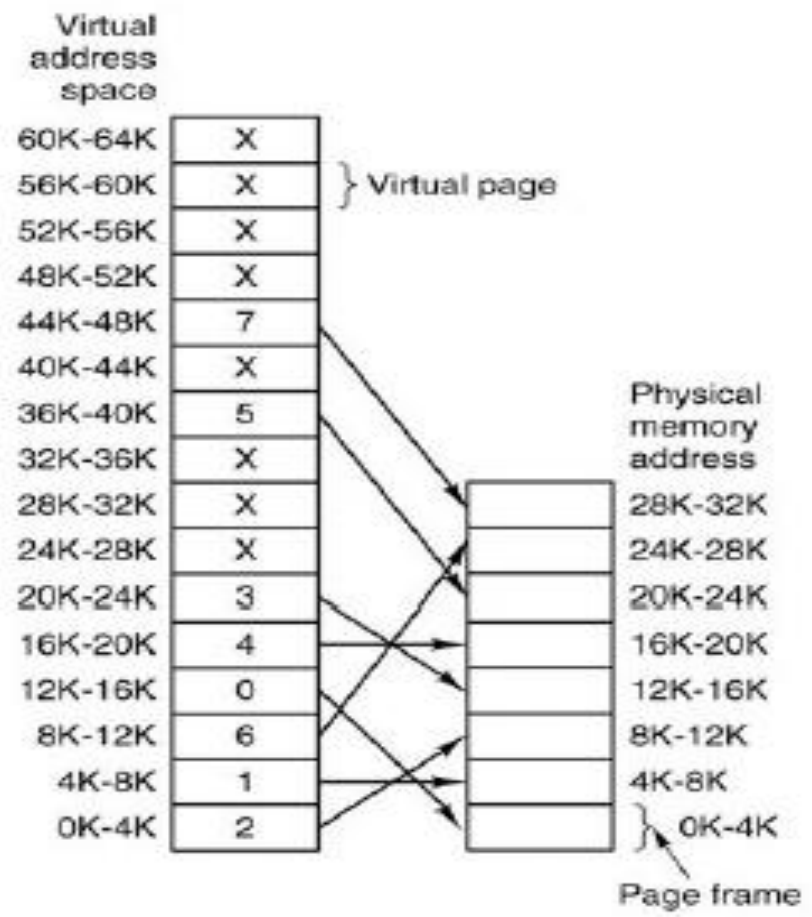


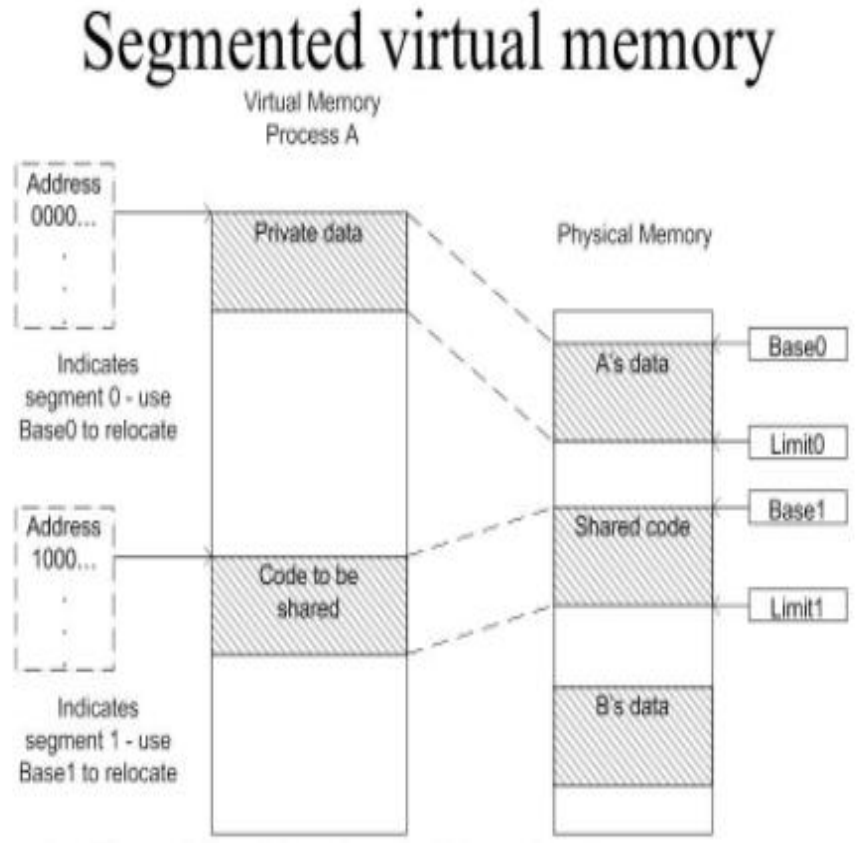
5.2.2 Virtual memory

- The basic idea behind virtual memory is that the combined size of the program, data, and stack may exceed the amount of physical memory available for it.
- As we said before, the operating system keeps those parts of the program currently in use in main memory, and the rest on the disk.
- It maps memory addresses used by a program, called virtual addresses, into physical addresses in computer memory.
- The operating system manages virtual address spaces and the assignment of real memory to virtual memory. Address translation hardware in the CPU, often referred to as a Memory Management Unit or MMU, automatically translates virtual addresses to physical addresses
- The primary benefits of virtual memory include freeing applications from having to manage a shared memory space, increased security due to memory isolation, and being able to conceptually use more memory than might be physically available.
- Virtual memory uses paging, segmentation and segmented paging.

5.2.2 Virtual memory

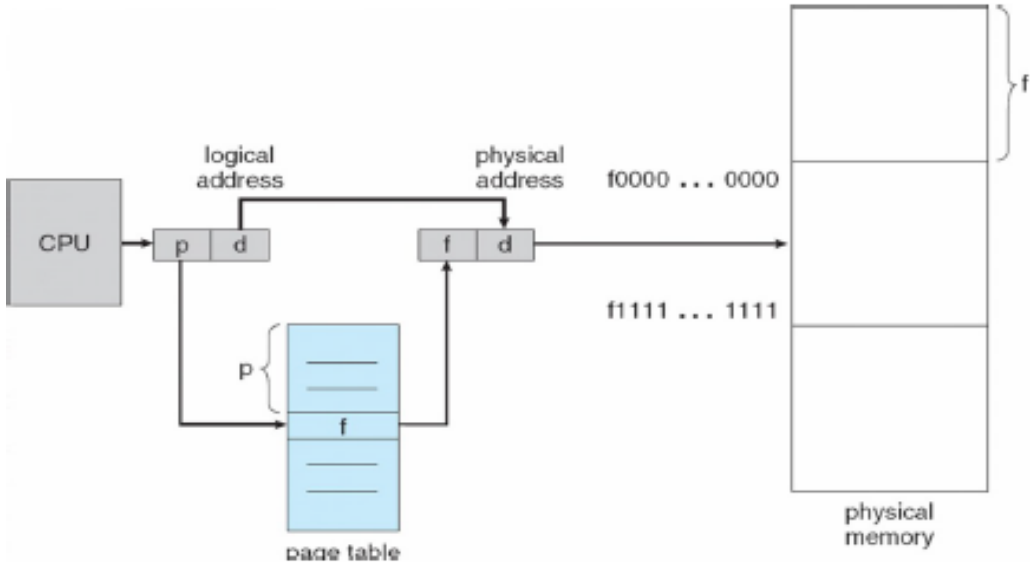
Paging and segmentation examples





5.2.2 Virtual memory

- Paging is the most common technique, in which a page fault is a trap to the software raised by the hardware when a program accesses a page that is mapped in the virtual address space, but not loaded in physical memory.
- Alike non-contiguous memory without virtual memory, the basic method for implementing paging involves breaking physical memory into fixed size block called frames and breaking logical memory into blocks of the same size called pages.
- Every address generated by the CPU is divided into two parts: a page number (p) and a page offset (d)



5.2.2 Virtual memory

- When a page fault occurs, the operating system has to choose a page to remove from memory to make room for the page that has to be brought in.
- The page replacement is done by swapping the required pages from backup storage to main memory and vice-versa.
- Page replacement algorithms are the techniques using which an operating system decides which memory pages to swap out, write to disk when a page of memory needs to be allocated.
- Research about **FIFO**, **Optimal** and **LRU** algorithms and how they work.

5.2.3 File management

- From the user point of view one of the most important part of the operating system is file system. The file system provides the resource abstraction typically associated with secondary storage. The file system permits users to create data collections, called files.
- A file is a named collection of related information that is recorded on secondary storage such as magnetic disks, magnetic tapes and optical disks.
- In general, a file is a sequence of bits, bytes, lines or records whose meaning is defined by the files creator and user.
- To organize and keep track of files, file systems normally have directories or folders, which, in many systems, are themselves files.
- On the other hand, a hard disk drive can be divided in to several storage units called partitions.
- In addition, a storage area that could be accessed using a single file system that the computer can recognize is referred to as a volume.

5.2.3 File management

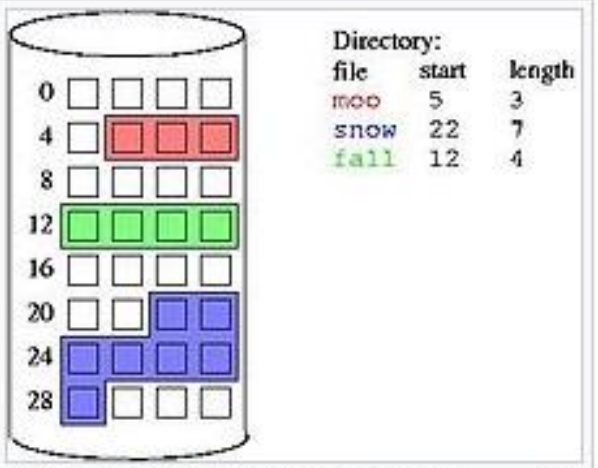
- **Exercise:** Search the most common file systems in Windows and Linux. Compare the maximum file and volume size they allow.

5.2.3 File management

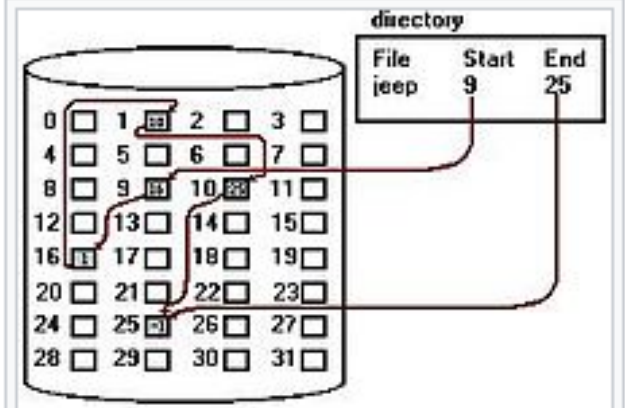
- A block device is one that stores information in fixed-size blocks, each one with its own address. Common block sizes range from 512 bytes to 4096 KB.
- When a file is used then the stored information in the file must be accessed and read into the memory of a computer system.
- Taking it into account, files are allocated disk spaces by operating system. Operating systems deploy following three main ways to allocate disk space to files:

1. Contiguous Allocation: Each file occupies contiguous blocks on the disk. The location of a file is defined by the disk address of the first block and its length.
2. Linked Allocation: Each file is a linked list of disk blocks. The directory contains a pointer to the first and (optionally the last) block of the file.
3. Indexed Allocation: Linked allocation does not support random access of files, since each block can only be found from the previous. Indexed allocation solves this problem by bringing all the pointers together into an index block.

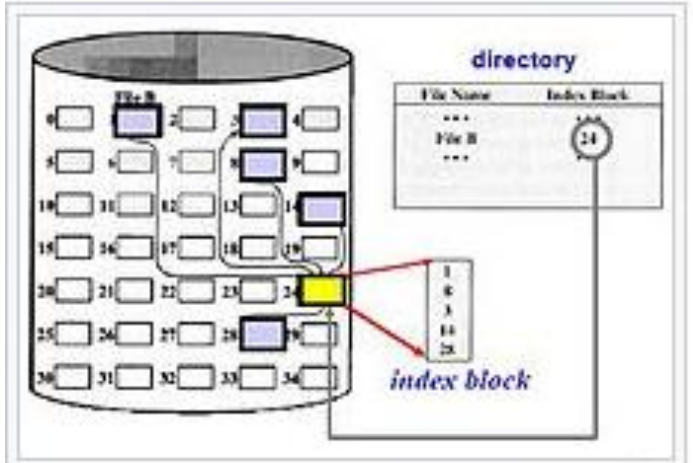
5.2.3 File management



A file allocated disk space using contiguous allocation method



Example of a file which has been allocated disk space using linked allocation method.



a file allocated disk space using Indexed allocation Method

5.2.4 I/O management

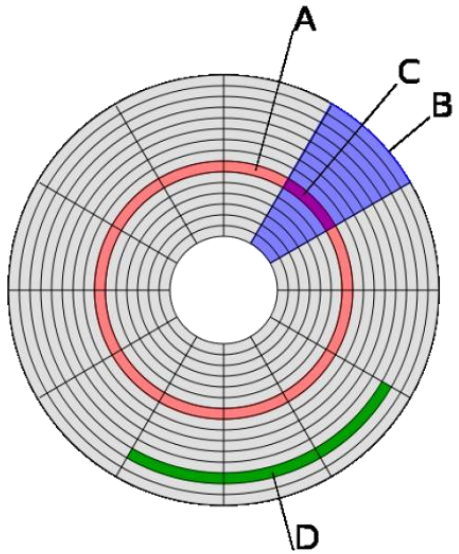
- One of the important jobs of an operating system is to hide hardware details to the user.
- Most important functions related to I/O management in operating systems are:
  - Planning I/O requests to improve performance.
  - Error management.
  - Adapt the speed of devices.
- A device driver is a computer program that operates or controls a particular type of device that is attached to a computer.
- A driver provides a software interface to hardware devices, enabling operating systems and other computer programs to access hardware functions without needing to know precise details of the hardware being used.

5.2.4 I/O management

- There are three fundamentally different ways to do I/O:
  - Programmed I/O: The CPU manually check if there are any I/O requests available periodically. If there isn't, it keeps executing its normal workflow. If there is, it handles the IO request instead.
  - Interrupt-Driven I/O: The CPU does not need to manually check for IO requests. When there is an I/O request available, the CPU is immediately notified using interrupts, and the request is immediately handled using a interrupt service routines.
  - DMA is a method allowing devices (typically has very slow I/O speeds) to access main memory without needing the CPU to explicitly handle the requests.

5.2.5 Secondary storage management

- Secondary Storage is usually:
  - Anything outside of “primary memory”
  - Storage that does not permit direct instruction execution or data fetch by load/store instructions.
  - It’s large
  - It’s cheap
  - It’s non-volatile
  - It’s slow



Hard Drive Structure:

A = track
B = sector
C = sector of a track
D = cluster

5.2.5 Secondary storage management

- Secondary Storage manager is mainly responsible for:
  - Managing free disk space. Operating systems mainly use the following three methods:
    - Bitmap: Storage devices are divided up into allocation units, perhaps as small as few words or kilobytes. Corresponding to each allocation unit is a bit in the bitmap, which is 0 if the unit is free and 1 if its occupied (or vice versa)
    - Linked list: Another way of keeping track of secondary memory is to maintain a linked list of allocated and free memory segments.
    - Indexed: An index block contains pointers to many other blocks. Better for random access, it may need multiple index blocks.

5.2.5 Secondary storage management

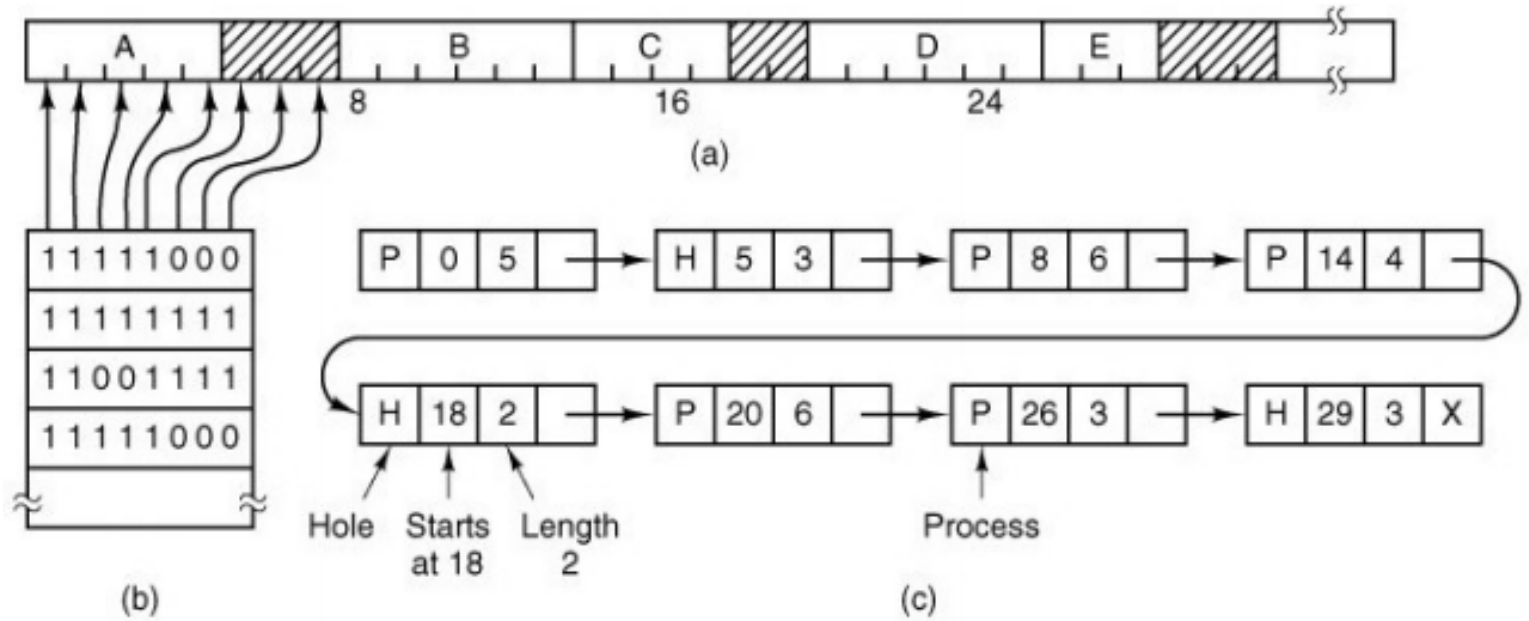
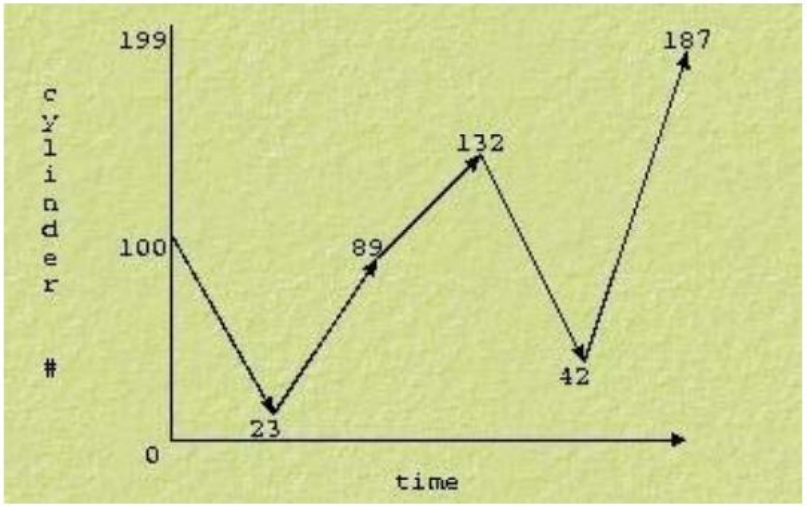


Fig:(a) A part of memory with five processes and three holes. The tick marks show the memory allocation units. The shaded regions (0 in the bitmap) are free. (b) The corresponding bitmap. (c) The same information as a list.

5.2.5 Secondary storage management

- Secondary Storage manager is mainly responsible for:
  - Scheduling disk access requests to achieve good efficiency. Disk scheduling is similar to process scheduling. Some of the disk scheduling algorithms are described below. We are going to use the same example with head initially at 100
    - **FCFS:** Simplest, perform operations in order requested.

23, 89, 132, 42, 187

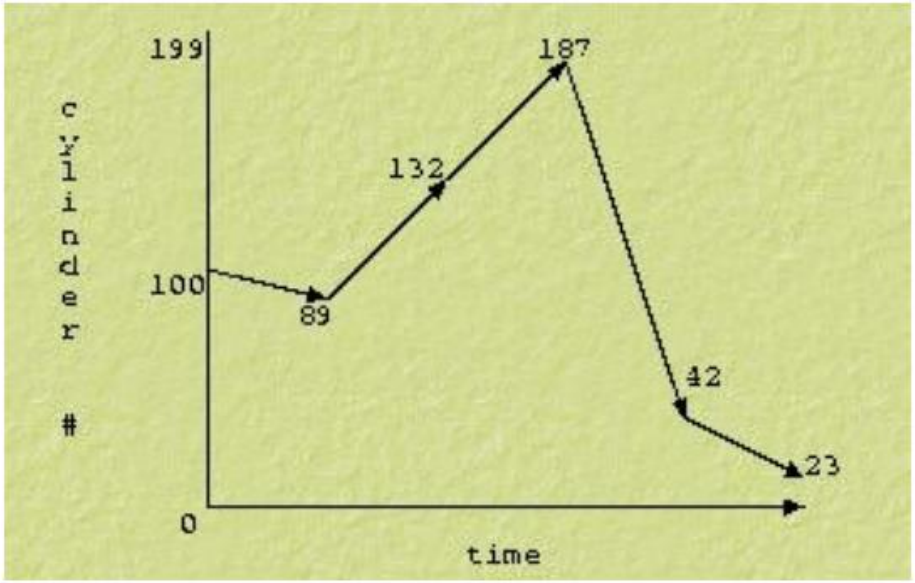


77+66+43+90+145=421

5.2.5 Secondary storage management

- **SSTF:** Select the disk I/O request that requires the least movement of the disk arm from its current position, regardless of direction

23, 89, 132, 42, 187



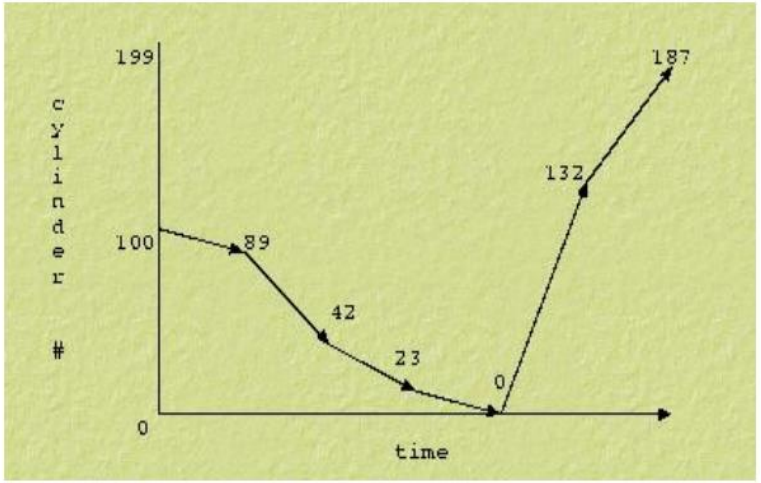
11+43+55+145+19=273

5.2.5 Secondary storage management

- **SCAN:** Go from the outside to the inside servicing requests and then back from the outside to the inside servicing requests. Sometimes called the elevator algorithm.
- **C-SCAN:** Moves inwards servicing requests until it reaches the innermost cylinder; then jumps to the outside cylinder of the disk without servicing any requests.

SCAN

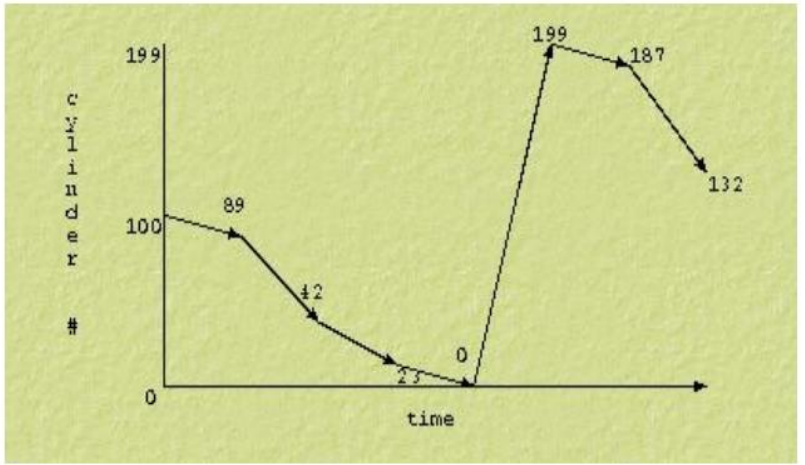
23, 89, 132, 42, 187



11+47+19+23+132+55=287

C-SCAN

23, 89, 132, 42, 187



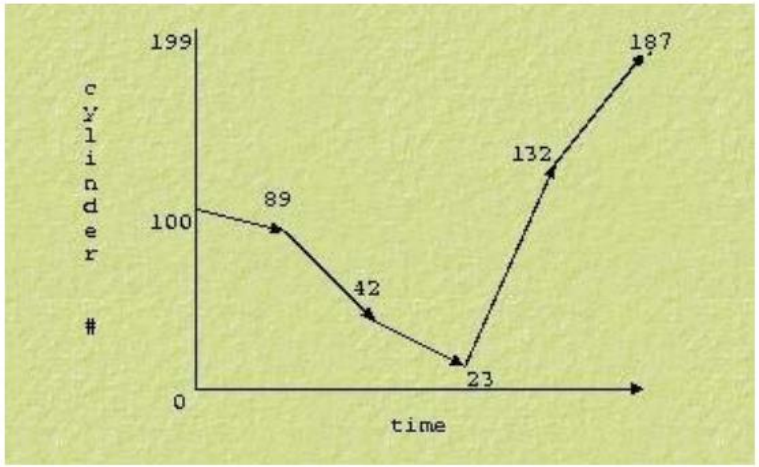
11+47+19+23+199+12+55=366

5.2.5 Secondary storage management

- **LOOK and C-LOOK:** Like SCAN and C-SCAN, but stops moving inwards (or outwards) when no more requests in that direction exist.

LOOK

23, 89, 132, 42, 187



11+47+19+109+55=241

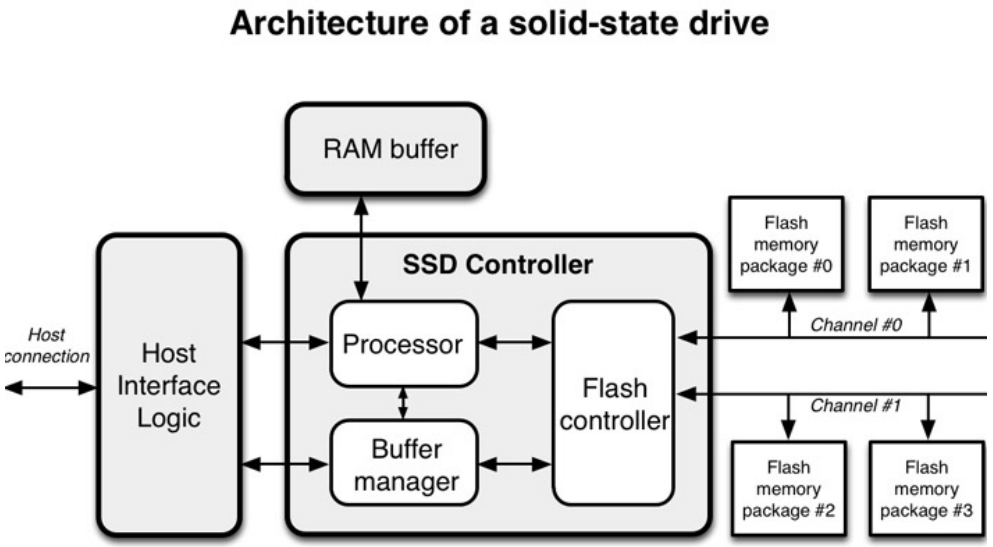
How would you represent C-LOOK algorithm for the same requests?
Calculate the total head movement

5.2.5 Secondary storage management

- **Given a hard disk of 200 tracks (Track 1-200). Write down the track numbers the disk head will travel for each disk-scheduling algorithm with the following sequence of disk track requests: 98, 183, 37, 122, 14, 124, 65 y 67. The disk head is currently at track 53. Which disk-scheduling algorithm (FIFO, SSTF, SCAN, C-SCAN, LOOK, C-LOOK) is most effective?**

5.2.5 Secondary storage management

- Scheduling disk access requests to achieve good efficiency. The previous methods are related to hard disks.
  - New techniques have been developed to deal with **SSD disks**. Further info: https://pdfs.semanticscholar.org/e4ed/6ff2363ad43e39000830c830cac6f2748cdd.pdf



6. Licenses

- A software license is a legal instrument (usually by way of contract law, with or without printed material) governing the use or redistribution of software.
- A typical software license grants the permission to use one or more copies of produce.
- The license also defines the responsibilities of the parties entering into the license agreement and may impose restrictions on how the software can be used.
- There are two main types:
  - **Proprietary:** This feature of proprietary software licenses means that certain rights regarding the software are reserved by the software publisher. The source code is usually a closely guarded secret.
  - **Free and open-source:** These two terms are sometimes used interchangeably. Free software is restrictive software that is accessible for use at no cost in monetary form. Open-source software is software of computer with source code made accessible with a permit in which the license holder gives the rights to change or study and circulates the product to anybody and for any reason.

6. Licenses

- **What's the meaning of public domain software?**
- **List one popular example of public domain software.**

6. Licenses

- Some common free software licenses based on copyleft are:
  - The **GNU General Public License (GNU GPL or GPL)** is a widely used free software license, which guarantees end users the freedom to run, study, share and modify the software. The GPL is a copyleft license, which means that derivative work can only be distributed under the same license terms.
  - The **GNU Lesser General Public License (LGPL)** allows developers and companies to use and integrate software released under the LGPL into their own (even proprietary) software without being required by the terms of a strong copyleft license to release the source code of their own components.
  - The **GNU Affero General Public License (AGPL)** is a modified version of the ordinary GNU GPL version. It has one added requirement: if you run a modified program on a server and let other users communicate with it there, your server must also allow them to download the source code corresponding to the modified version running there.
- Operating systems under GPL and LPGL: https://en.wikipedia.org/wiki/Comparison_of_open-source_operating_systems
- Software under AGPL: https://en.wikipedia.org/wiki/List_of_software_under_the_GNU_AGPL

6. Licenses

- In proprietary software, an **end-user license agreement (EULA)** or software license agreement is the contract between the licensor and purchaser, establishing the purchaser's right to use the software.
- An **Original Equipment Manufacturer (OEM)** license is the cheapest type of license to acquire, but also the most restrictive. An OEM license is bound to the hardware that it is installed on. In other words, the license lives and dies with the computer. If a computer with an OEM license is disposed of then, the license must be legally be disposed of as well and cannot be transferred to another computer.
- An **Open License Agreement (OPL)** is designed for between 5 and 250 computers and usually contains a single installation disk and a single product key. The difference however, is that the product key may be used repeatedly until the number of acquired licenses have been reached.