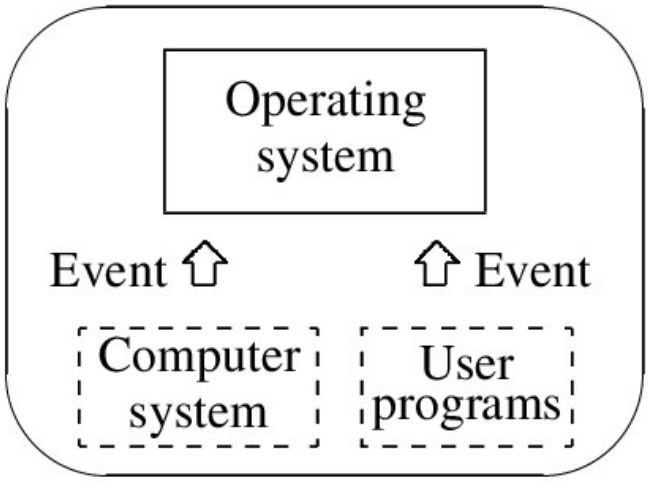
1. A) Define OS. What are the goals f an OS? Explain. (8marks)
   - An operating system (OS) is a collection of programs that achieve *effective utilization* of a computer system by providing
     - Convenient methods of using a computer
       - Saves users' time and boosts their productivity
     - Efficient use of the computer

- Two primary goals of an OS are
  - Efficient use of the computer's resources
    - To ensure cost-effectiveness of the computer
  - User convenience
    - A user should find it easy to use the computer
- These two goals sometimes conflict
  - Prompt service can be provided through exclusive use of a computer; however, efficient use requires sharing of a computer's resources among many users
  - An OS designer decides which of the two goals is more important under what conditions
    - That is why we have so many operating systems!
- The *computing environment* influences the notion of user convenience
  - The computing environment is comprised of
    - The computer system
    - Its interfaces with other systems
    - Nature of computations performed by its users
  - Computing environments change with the architecture and use of a computer, e.g. personal computing, online applications, embedded applications
  - Hence the notion of user convenience has several facets that depend on the computing environment
- User convenience has several facets
  - Fulfillment of a necessity

        * Use of programs and files
    - Good service
        * Speedy response
    - Ease of Use
        * User friendliness
    - New programming model
        * e.g., Concurrent programming
    - Web-oriented features
        * e.g., Web-enabled servers
    - Evolution
        * Addition of new features, use of new computers

1. B) List and Explain different computational structures of an OS.

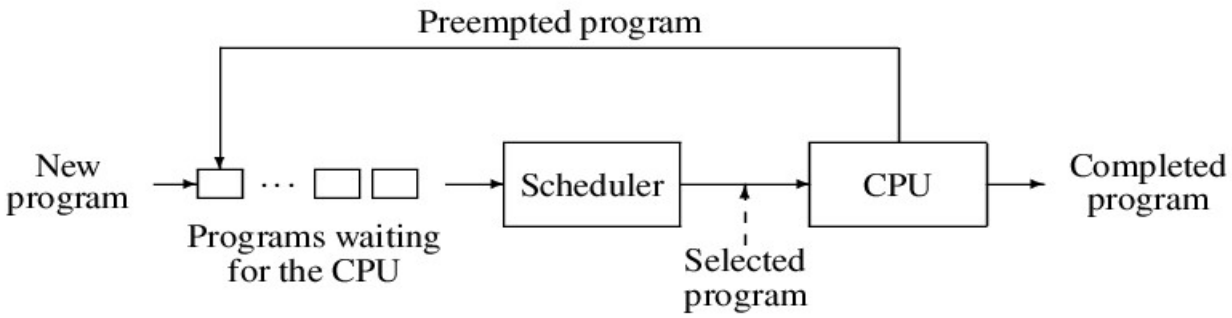


Computing environment

Task	When performed
Maintain information for security	While registering new users
Construct a list of resources in the system	During booting
Verify identity of a user	At login time
Initiate execution of programs	At user commands
Maintain information for protection	When users specify or change protection information
Check protection information and perform resource allocation	At user/program request
Maintain current status of all resources	Continually during OS operation
Maintain current status of all programs and perform scheduling	Continually during OS operation

- A *computational structure* is a configuration of several programs. Three typical computational structures are:
  - A single program
    - Execution of a program on a given set of data
  - A sequence of single programs
    - A program should be executed only if previous programs in the sequence executed successfully
    - Programs should pass their results to other programs through files
  - Co-executing programs
    - The OS must execute these programs at the same time
    - The OS must provide an interface between co-executing programs so that their results are available to one another
- Programs waiting for CPU attention are entered in a pool
- The scheduler picks one program from the pool for execution on the CPU
- The CPU may be forcibly taken away from a program. This action

  is called *pre-emption*
- A pre-empted program is again entered into the pool



- Criteria and techniques of resource management
  - Criterion: Resource utilization efficiency
    - Share the resources wherever possible
    - Avoid idling of resources
    - Speedy resource allocation and de-allocation is desirable
  - Technique 1: Partitioning of resources
    - A *resource partition* is a collection of resources
      - A resource partition is allocated to a process
      - Simplifies resources allocation and also speeds it up
  - Technique 2: Allocation from a pool
    - Resources are allocated individually when needed

2. A) What are different classes of Os? Explain them with their primary concerns

Batch Processing Systems

- *Batch: sequence* of user jobs formed for processing by the OS
- Batching kernel initiates processing of jobs without requiring computer operator's intervention
- Card readers and printers were a performance bottleneck in the 1960s
  - Virtual card readers and printers implemented through magnetic tapes were used to solve this problem
- *Control statements* used to protect against interference between jobs
- *Command interpreter* read a card when currently executing program in job wanted the next card

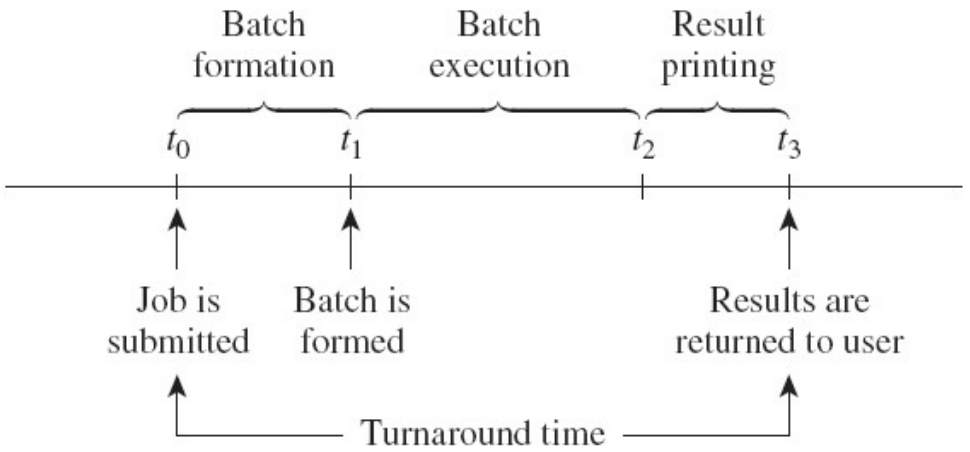


Figure 3.1 Turnaround time in a batch processing system.

```
// JOB ···                    → "Start of job" statement
// EXEC FORTRAN               → Execute the Fortran compiler
                  } Fortran
                    program
// EXEC                       → Execute just compiled program
                  } Data for
                    Fortran
                    program
/*                            → "End of data" statement
/&                            → "End of job" statement
```

Figure 3.2 Control statements in IBM 360/370 systems.

Multiprogramming Systems

- **Provide efficient resource utilization in a noninteractive environment**
- **Uses DMA mode of I/O**
  - **Can perform I/O operations of some program(s) while using the CPU to execute some other program**
    - **Makes efficient use of both the CPU and I/O devices**
- **Turnaround time of a program is the appropriate measure of user service in these systems**

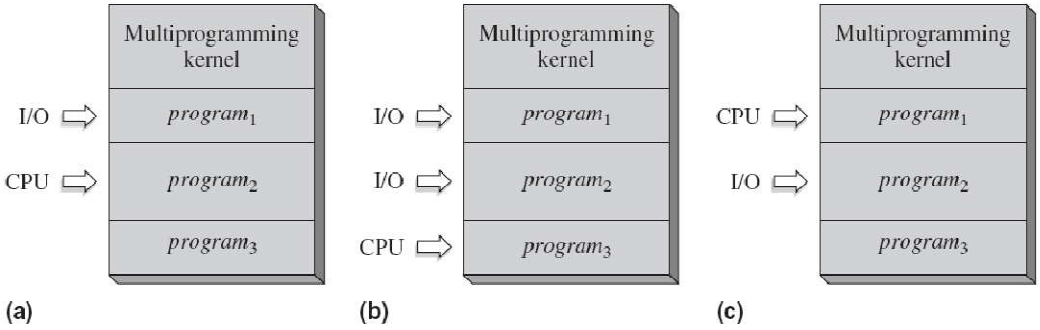


Figure 3.3 Operation of a multiprogramming system: (a) $program_2$ is in execution while $program_1$ is performing an I/O operation; (b) $program_2$ initiates an I/O operation, $program_3$ is scheduled; (c) $program_1$'s I/O operation completes and it is scheduled.

Table 3.4 Architectural Support for Multiprogramming

Feature	Description
DMA	The CPU initiates an I/O operation when an I/O instruction is executed. The DMA implements the data transfer involved in the I/O operation without involving the CPU and raises an I/O interrupt when the data transfer completes.
Memory protection	A program can access only the part of memory defined by contents of the *base register* and *size register*.
Kernel and user modes of CPU	Certain instructions, called *privileged instructions*, can be performed only when the CPU is in the kernel mode. A program interrupt is raised if a program tries to execute a privileged instruction when the CPU is in the user mode.

- An appropriate measure of performance of a multiprogramming OS is *throughput*
  - Ratio of the number of programs processed and the total time taken to process them
- OS keeps enough programs in memory at all times, so that CPU and I/O devices are not idle
  - *Degree of multiprogramming:* number of programs
  - Uses an appropriate *program mix* of *CPU-bound programs* and *I/O-bound programs*
  - Assigns appropriate priorities to CPU-bound and I/O-bound programs

2. B) Explain the terms a}efficiency b)system performance c)user service

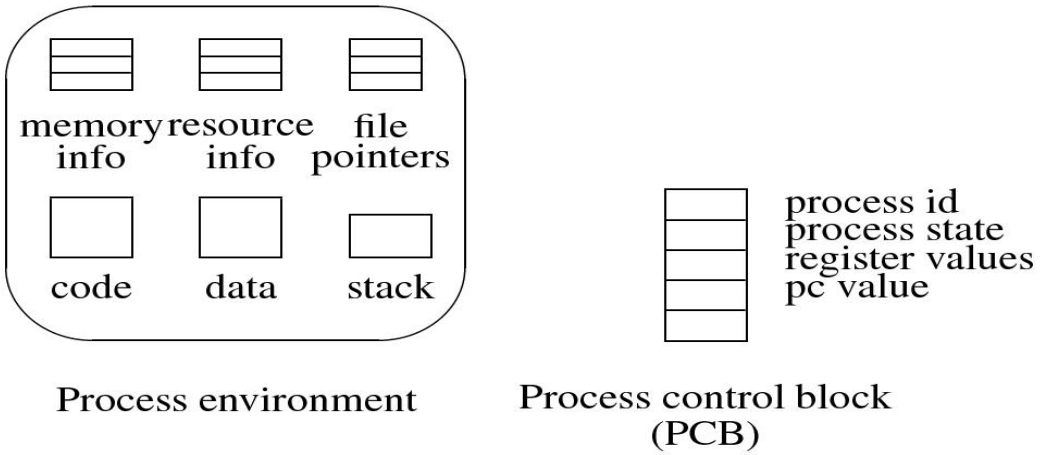
   An operating system must ensure efficient use of the fundamental computer system resources of memory, CPU, and I/O devices such as disks and printers. Poor efficiency can result if a program does not use a resource allocated to it, e.g.,

if memory or I/O devices allocated to a program remain idle. Such a situation may have a snowballing effect: Since the resource is allocated to a program, it is denied to other programs that need it. These programs cannot execute, hence resources allocated to them also remain idle. In addition, the OS itself consumes some CPU and memory resources during its own operation, and this consumption of resources constitutes an overhead that also reduces the resources available to user programs. To achieve good efficiency, the OS must minimize the waste of resources by programs and also minimize its own overhead.

Table 1.1 Facets of User Convenience Facet Examples Fulfillment of necessity Ability to execute programs, use the file system Good Service Speedy response to computational requests User friendly interfaces Easy-to-use commands, graphical user interface (GUI) New programming model Concurrent programming Web-oriented features Means to set up Web-enabled servers Evolution Add new features, use new computer technologies

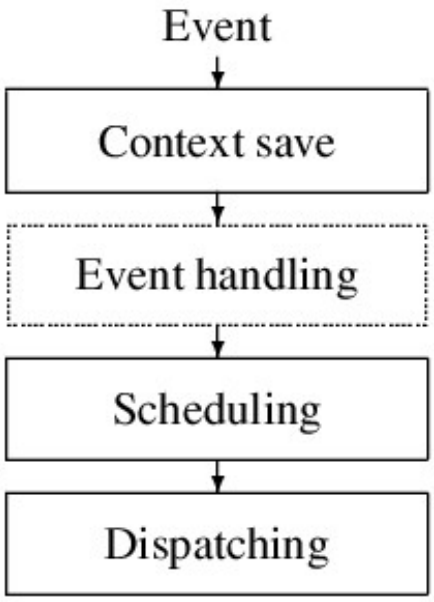
3. A) Explain with a neat sketch the view of a processor
   The kernel allocates resources to a process and schedules it for use of the CPU. Accordingly, the kernel's view of a process consists of two parts: • Code, data, and stack of the process, and information concerning memory and other resources, such as files, allocated to it. • Information concerning execution of a program, such as the process state, the CPU state including the stack pointer, and some other items of information described later in this section. These two parts of the kernel's view are contained in the process context and the process control block (PCB), respectively (see Figure 5.6). This arrangement enables different OS modules to access relevant process-related information conveniently and efficiently

   Process Context The process context consists of the following: 1. Address space of the process: The code, data, and stack components of the process 2. Memory allocation information : Information concerning memory areas allocated to a process. This information is used by the memory management unit (MMU) during operation of the process 3. Status of file processing activities: Information about files being used, such as current positions in the files.



- The process environment consists all information needed for accessing and controlling resources allocated to a process (it is also called the *process context*):
  - Address space of the process, i.e., code, data, and stack
  - Memory allocation information
  - Status of file processing activities, e.g., file pointers
  - Process interaction information
  - Resource information

  Miscellaneous information
- The PCB contains information needed for controlling operation of the process. Its fields are:
  - Process id
  - Ids of parent and child processes
  - Priority
  - Process state (defined later)
  - CPU state, which is comprised of the PSW and CPU registers
  - Event information
  - Signal information
  - PCB pointer (used for forming a list of PCBs)



- Occurrence of an event causes an interrupt
- The *context save* function saves the state of the process that was in operation
- Scheduling selects a process; dispatching switches the CPU to its execution

3. B) Define process state. Write a neat sketch and explain the fundamental states of a process.

   Process State:

   A *process state* is an indicator of the current activity of a process

   a. An OS designer defines process states to simplify functioning of the OS, e.g., to simplify scheduling

   b. Some sample process states

      i. A process is waiting for an I/O operation to complete: *blocked* state

      ii. The CPU is executing instructions of a process: *running* state

   c. The kernel keeps track of the state of a process and changes it as its activity changes

   d. Different operating systems may use different process states

   The fundamental process states are:

   e. Running

      i. A CPU is allocated to the process and is executing its instructions

f. Blocked

i. The process is waiting for a resource to be allocated or a specified event to occur

1. The process should not be scheduled until the awaited event occurs

g. Ready

i. Process is not blocked but it is not running

1. It can be considered for scheduling

h. Terminated

i. Operation of the process has completed

4. Operation of a process

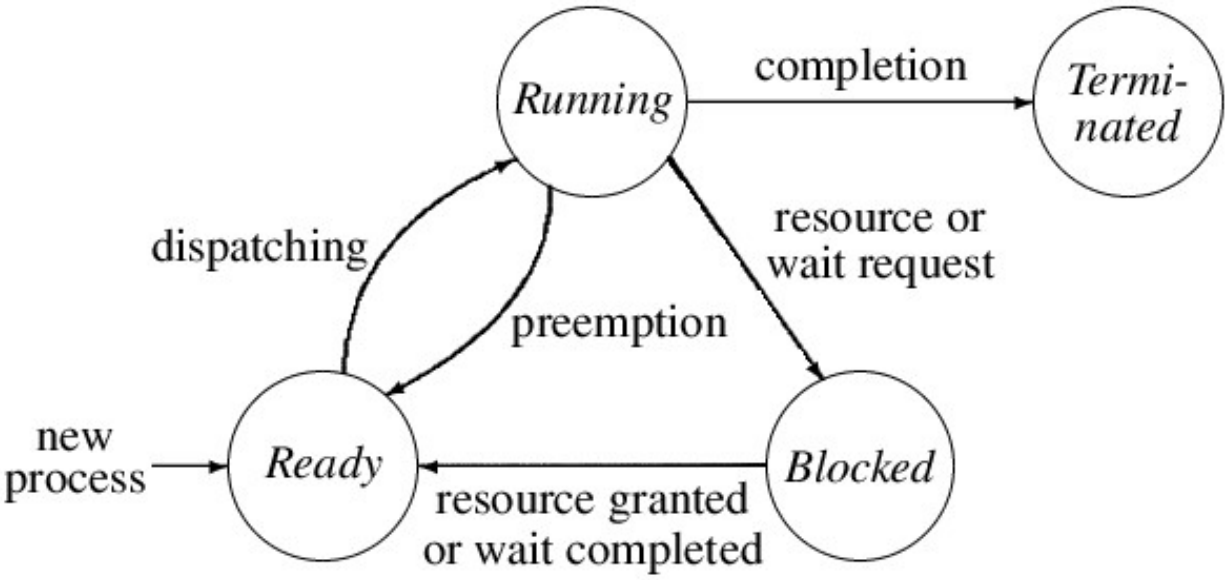
a. A process has a state

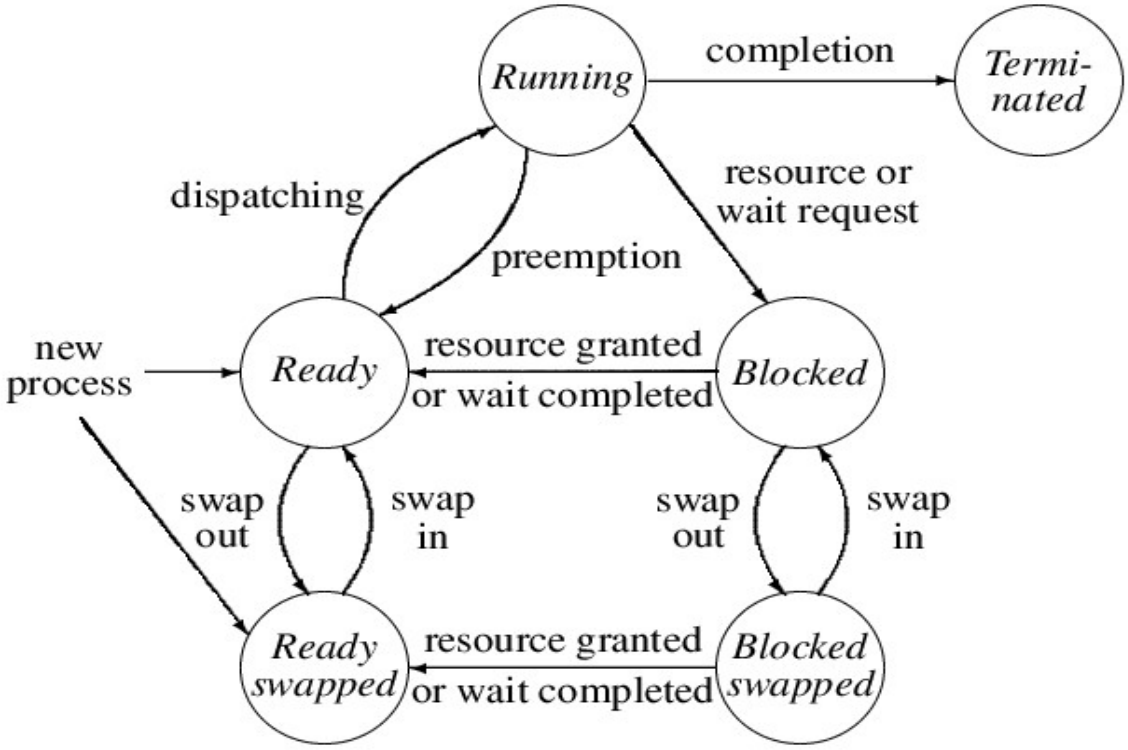
b. The state of a process changes when the nature of its activity changes

i. This change of state is called a *state transition*

ii. It is caused by an event

Several state transitions can occur before the process terminates





4. A)Perform FCFS and SRN Scheduling and compare

Process	P_1	P_2	P_3	P_4	P_5
Arrival time	0	2	3	5	9
Service time	3	3	2	5	3
Deadline	4	14	6	11	12

Arrives	Service	Time
P_1	0	3
P_2	2	3
P_3	3	2
P_4	5	5
P_5	9	3

	FCFS					SRN				
	Completed			Processes	Sche-	Completed			Processes	Sche-
Time	id	ta	w	in system	duled	id	ta	w	in system	duled
0	–	–	–	$\{P_1\}$	P_1	–	–	–	$\{P_1\}$	P_1
3	P_1	3	1.00	$\{P_2, P_3\}$	P_2	P_1	3	1.00	$\{P_2, P_3\}$	P_3
5						P_3	2	1.00	$\{P_2, P_4\}$	P_2
6	P_2	4	1.33	$\{P_3, P_4\}$	P_3					
8	P_3	5	2.50	$\{P_4\}$	P_4	P_2	6	2.00	$\{P_4\}$	P_4
13	P_4	8	1.60	$\{P_5\}$	P_5	P_4	8	1.60	$\{P_5\}$	P_5
16	P_5	7	2.33	$\{\}$	–	P_5	7	2.33	$\{\}$	–

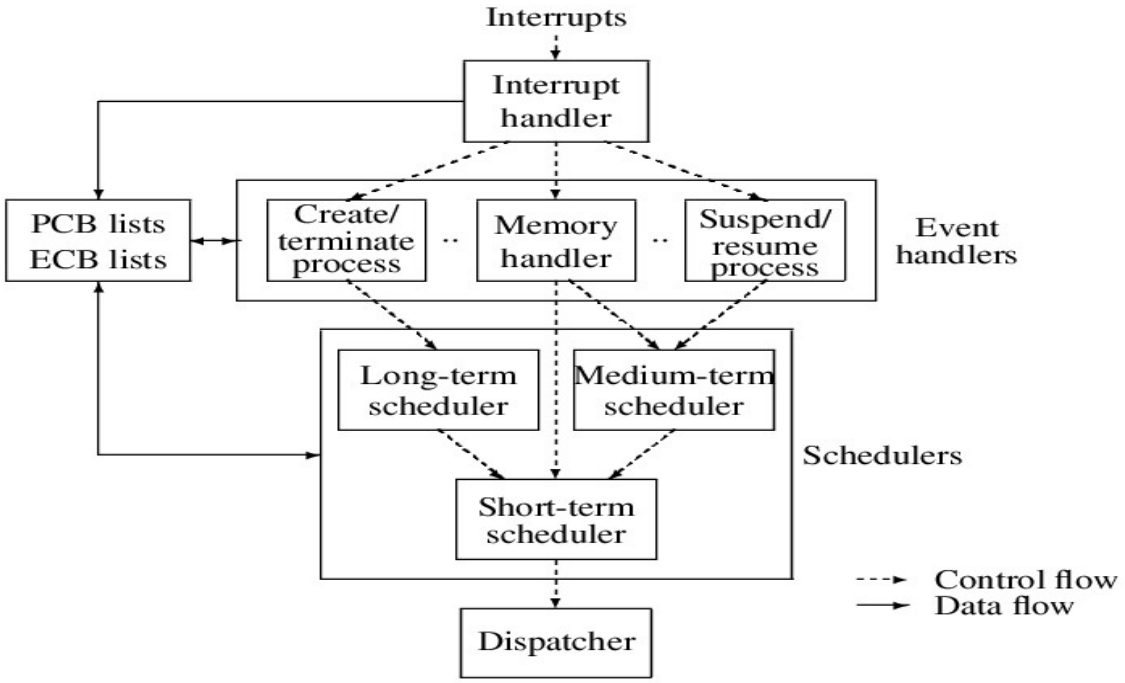
$\overline{ta} = 5.40$ seconds
$\overline{w} = 1.75$

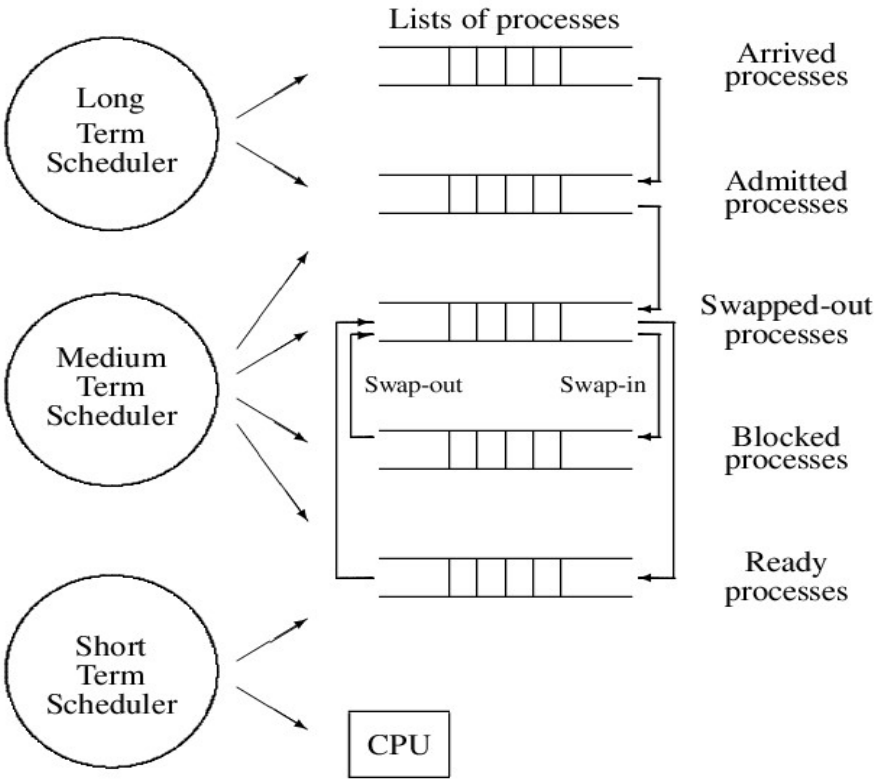
$\overline{ta} = 5.20$ seconds
$\overline{w} = 1.59$

4. B) Long, medium, and short-term scheduling

   A single scheduler cannot provide the desired combination of performance and user service, so an OS uses three schedulers

   a. Long-term scheduler
      i. Decides when to admit an arrived process
         1. Uses nature of a process, availability of resources to decide
   b. Medium-term scheduler
      i. Performs swapping
         1. Maintains a sufficient number of processes in memory
   c. Short-term scheduler
      i. Decides which ready process should operate on the CPU





- An event handler passes control to the long- or medium-term scheduler
- These schedulers pass control to the short-term scheduler
- The medium term scheduler swaps *blocked* and *ready* processes an changes their states accordingly

5. A)Compare contiguous and noncontiguous memory allocation

BASIS THE COMPARISON	CONTIGUOUS MEMORY ALLOCATION	NONCONTIGUOUS MEMORY ALLOCATION
Basic	Allocates consecutive blocks of memory to a process.	Allocates separate blocks of memory to a process.
Overheads	Contiguous memory	Noncontiguous memory

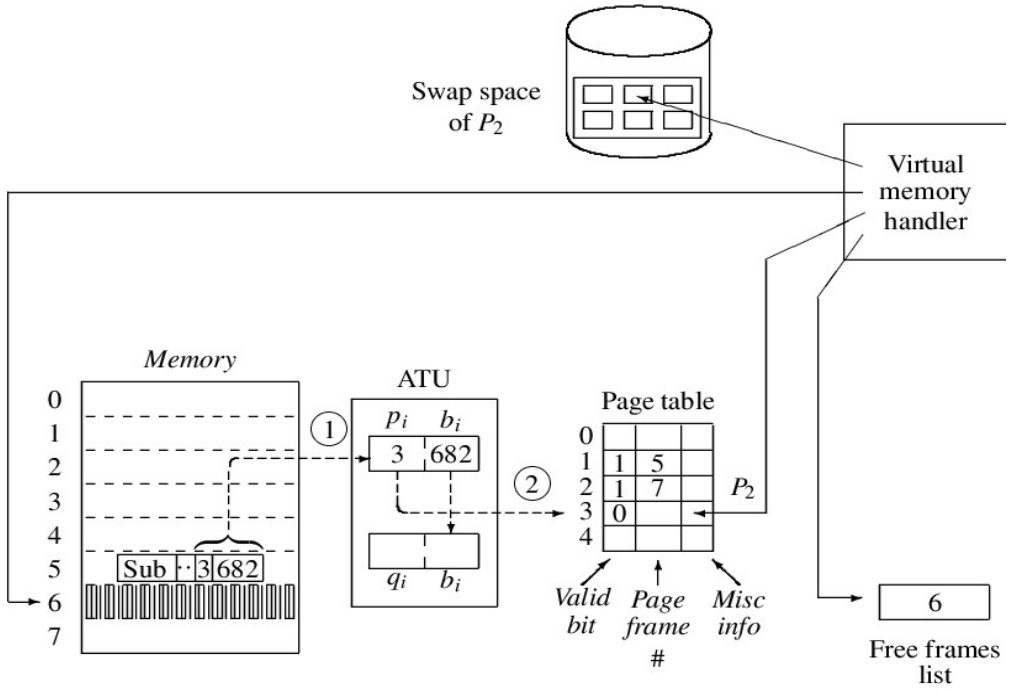
BASIS THE COMPARISON	CONTIGUOUS MEMORY ALLOCATION	NONCONTIGUOUS MEMORY ALLOCATION
	allocation does not have the overhead of address translation while execution of a process.	allocation has overhead of address translation while execution of a process.
Execution rate	A process executes fatser in contiguous memory allocation	A process executes quite slower comparatively in noncontiguous memory allocation.
Solution	The memory space must be divided into the fixed-sized partition and each partition is allocated to a single process only.	Divide the process into several blocks and place them in different parts of the memory according to the availability of memory space available.
Table	A table is maintained by operating system which maintains the list of available and occupied	A table has to be maintained for each process that carries the base addresses of each block which has been acquired by a

BASIS THE COMPARISON	CONTIGUOUS MEMORY ALLOCATION	NONCONTIGUOUS MEMORY ALLOCATION
	artition in the memory space	process in memory.

5. B)define
   1) Internal and external fragmentation
      Both the internal and external classification affects data accessing speed of the system. They have a basic difference between them i.e. **Internal fragmentation**occurs when fixed sized memory blocks are allocated to the process without concerning about the size of the process, and **External fragmentation** occurs when the processes are allocated memory dynamically.
   2) Paging and segmentation
      Paging is used to get a large linear address space without having to buy more physical memory.**Segmentation** allows programs and data to be broken up into logically independent address spaces and to aid sharing and protection.
   3) **Logical and physical address**
      Address uniquely identifies a location in the memory. We have two types of addresses that are logical address and physical address. The logical address is a virtual address and can be viewed by the user. The user can't view the physical address directly. The logical address is used like a reference, to access the physical address. The fundamental difference between logical and physical address is that **logical address** is generated by CPU during a program execution whereas, the **physical address** refers to a location in the memory unit.
   4) Page and page frame
      The "page" is the smallest unit of memory, managed by the computer's operating system, either in a physical form or virtual. The "page frame", refers to a "page of memory".

6. A)demand loading of a page

- Memory commitment would be high if the entire address space of a process is kept in memory, hence
  - *Only some pages* of a process are present in memory
  - Other pages are loaded in memory when needed; this action is called *demand loading* of pages
    * The logical address space of a process is stored in the *swap space*
    * The MMU raises an interrupt called *page fault* if the page to be accessed does not exist in memory
    * The *VM handler*, which is the software component of the virtual memory, *loads the required page from the swap space* into an empty page frame



- Reference to page 3 causes a page fault because its valid bit is 0
- The VM handler loads page 3 in an empty page frame and updates its entry in the page table

3 operations to support demand paging : Page-in, page-out and page replacement

- Page-in
  * A page is loaded in memory when a reference to it causes a page fault

- Page-out
    - A page is removed from memory to free a page frame
    - If it is a dirty page, it is copied into the swap space
- Page replacement
    - A page-out operation is performed to free a page frame
    - A page-in operation is performed into the same page frame

– Effective memory access time of logical address = (***pr_1*** **x 2 x access time of memory) +**

(1 – *pr_1*) (access time of memory + Time required to load the page + 2 x access time of memory)

where,

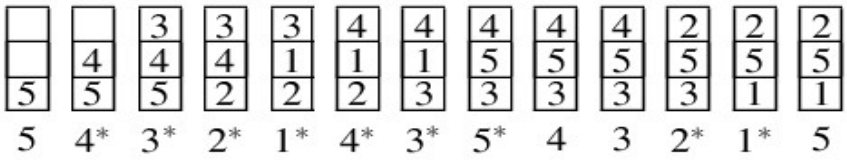
pr_1 is the probability that the page **page #** is already in memory. It is called ***hit ratio***.

1 – *pr_1* is miss ratio

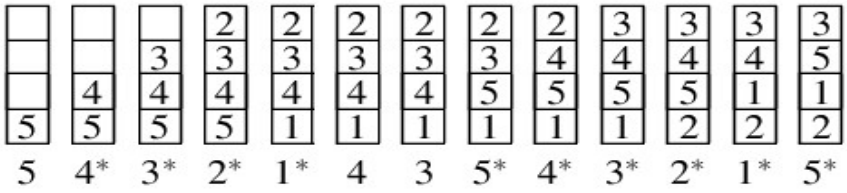
2 x access time of memory – one to access the entry of page # in the process's page table for address translation, and the other to access the operand from memory using effective address translation

6. B)

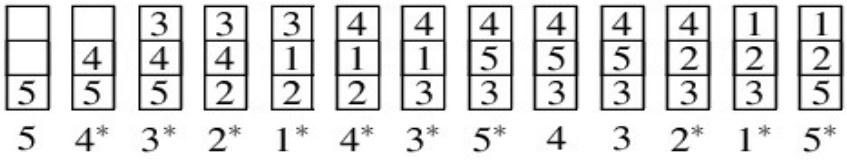
FIFO $alloc_i = 3$



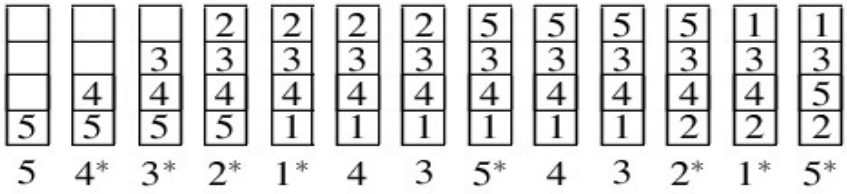
$alloc_i = 4$



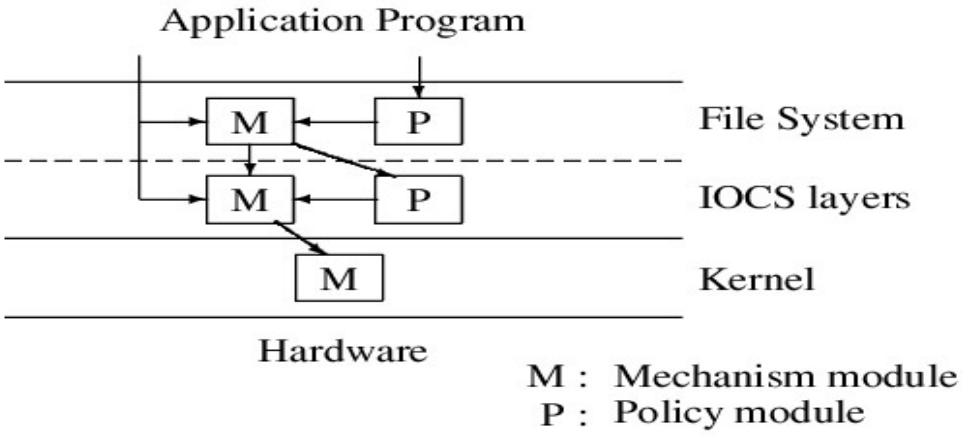
LRU $alloc_i = 3$



$alloc_i = 4$

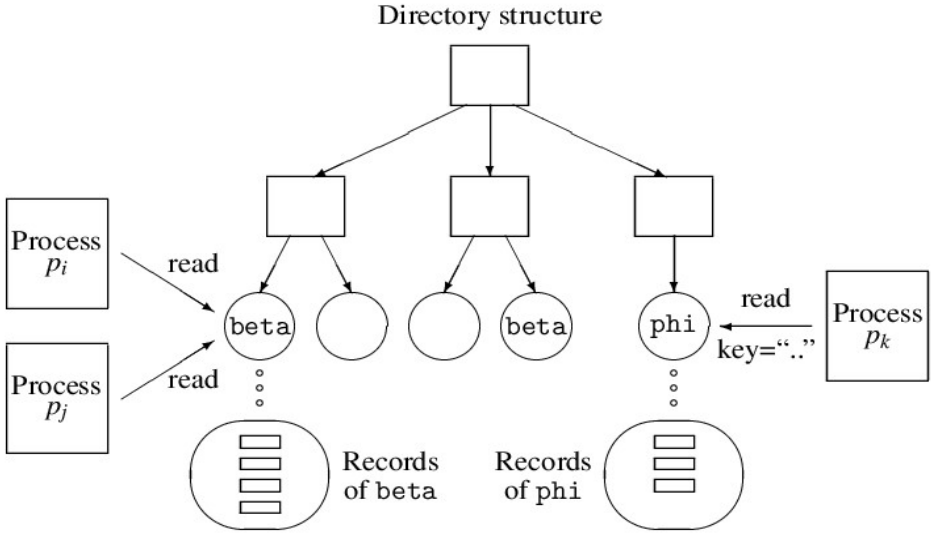


7. A) Explain file system and IOCS with necessary sketches



- A file system is structured into a hierarchy of layers
  - File system layer
    - Deals with files as objects
      - Creates files, allocates disk space to them

        - Implements file sharing and prevents unauthorized access
        - Prevents damage due to crashes
    - Input-Output Control System (IOCS) layer
        * Implements file system operations, ensuring
            - Efficient access to records in files
            - Efficient operation of I/O devices
- A Directory
    - Groups a set of files
        * It contains an entry for each of these files
        * The entry contains information useful for accessing the file
    - Users have different directories
        * Helps to separate files of different users
        * Provides file naming freedom
    - Directories are organized in a hierarchical structure
        * Permits a user to organize her files logically, e.g., according to activities



- Two files named beta exist. The file system must open the correct one when a process executes open (beta, ..)
- Records may be organized differently in different files
- Comments on the previous slide
    - Two files named beta exist in the file system
        - Thus users enjoy file naming freedom
    - Processes P_i and P_j share one of these files
        - The rules of sharing are determined by *file sharing semantics*
    - Files beta and phi have different organizations and are accessed differently
        - File beta is a sequential file; its records are read in a sequence
        - File phi is a direct file; its records can be read in random order

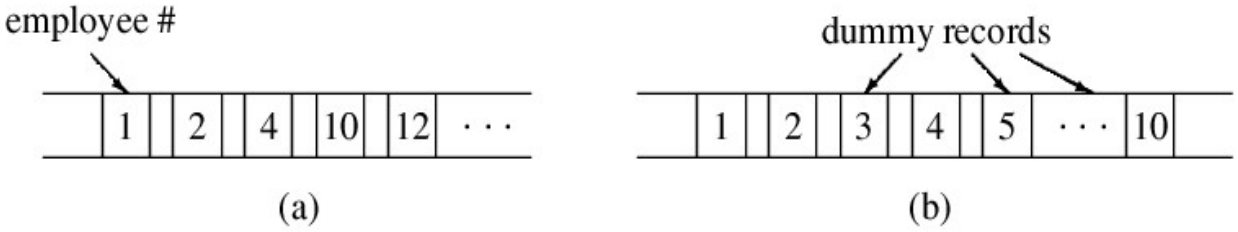
7. B)Explain fundamental file organizations

Record access pattern

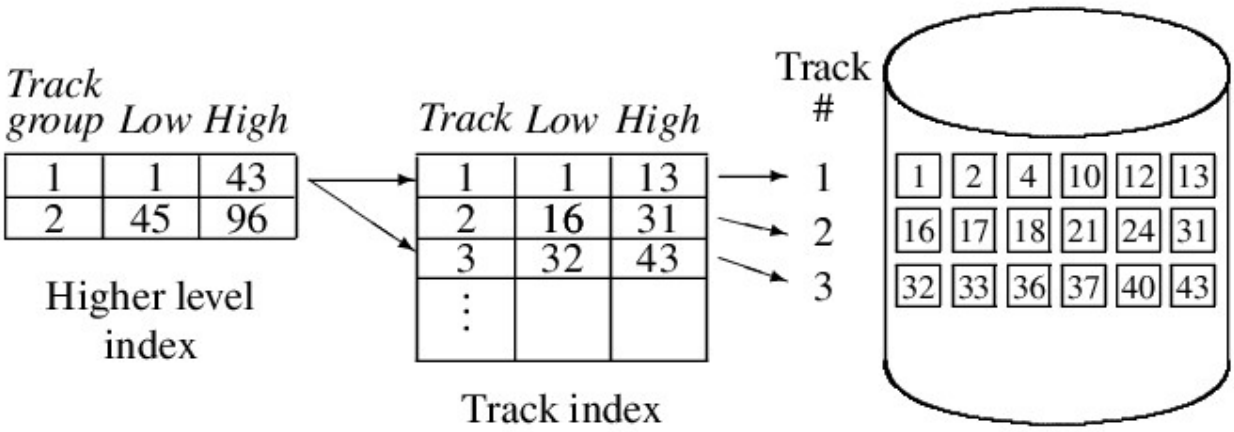
a. Definition
    i. The order in which a process accesses records in a file
b. Two fundamental record access patterns in applications
    i. Sequential access
        1. A process always performs an operation on the *next* record
        2. Suits batched operations; e.g., payroll
    ii. Random access
        1. A process may access *any* record in a file
c. File system should effectively support the record access pattern of an application

ile organization

d. Method of organizing and accessing data in a file
e. Sequential file organization
    i. Records in the file are accessed sequentially
        1. This file organization is independent of characteristics of I/O devices
f. Direct file organization
    i. Records are accessed randomly—a record is found using an address calculation formula applied to the value in its key field
        1. Files tend to be device dependent; may contain dummy records
g. Index sequential file organization
    i. File has a hierarchical organization. To locate a record, a region in the file is first located and then searched sequentially
    ii.

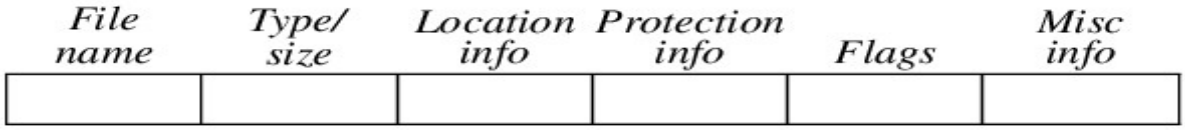


- In sequential files, an operation is always performed on the *next* record
- In direct files, an address calculation formula computes location of a record from its key value
- Dummy records have to exist for the formula to work

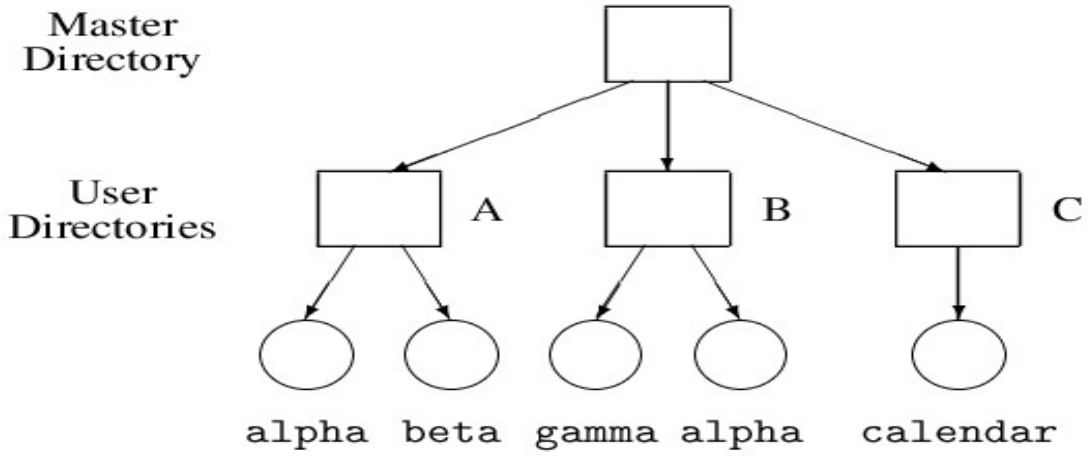


- The higher level index is searched to find a group of tracks where a record *may* exist
- The track index is searched to find the track which may contain the record
- The track is searched to check whether the record exists

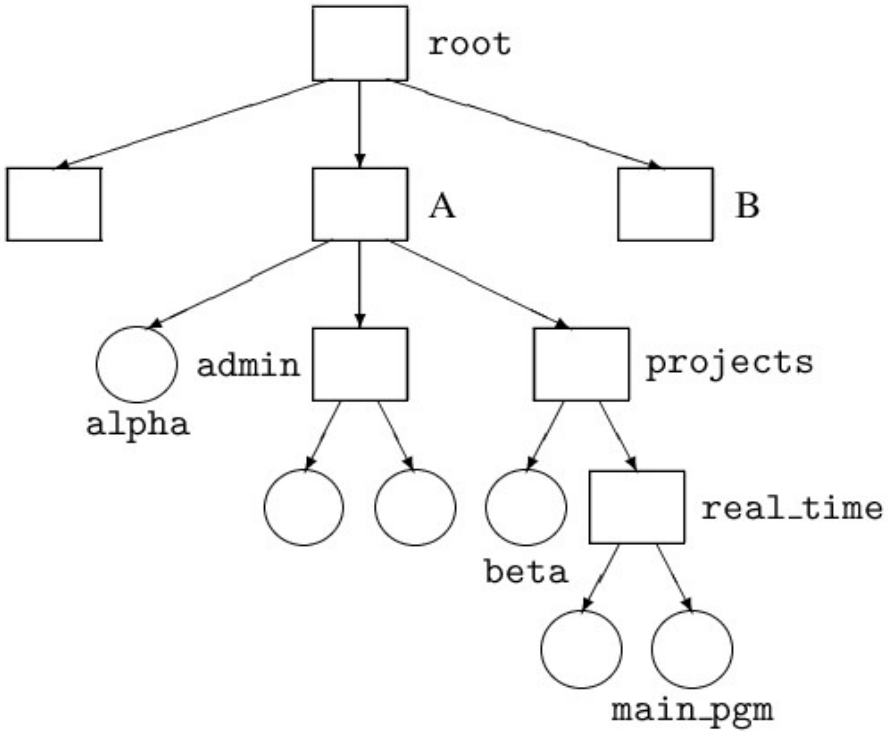
8. A) what is directory? Explain directory fields.



- *Location info* indicates where records of the file are located
- *Protection info* indicates who can access the file and in what manner
- *Flags* indicate whether the file is itself a directory, etc.

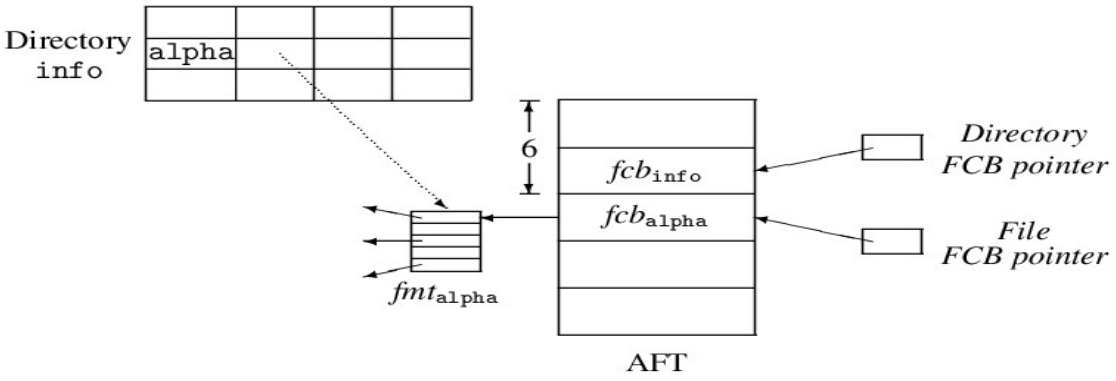


- A separate user directory exists for each user. This arrangement provides file naming freedom
- A user can access files of other users by going to that user's directory through the master directory
- Generalization provides several benefits
    - Directory as a file
        - A directory can exist in another directory
        - It leads to a directory hierarchy
        - User can form groups and subgroups of files
    - Generalized syntax for accessing files
        - A *path name* permits any file in the directory hierarchy to be accessed (subject to access privileges); e.g., ../projects/alpha
- Accessing a file
    - Each user has a *home directory* and a *current directory*
    - A file is accessed using absolute or relative path names



8. B)File system actions when a file is opened

- Perform *path name resolution*
  - For each component in the path name, locate the correct directory or file
    * Use two pointers called *Directory FCB pointer* and *File FCB pointer*
  - Handle path names passing through mount points
    * A file should be allocated disk space in its own file system
- Retain sufficient information to perform a *close* operation on the file
  - Close may have to update the file's entry in the parent directory
  - It may cause changes in the parent directory's entry in ancestor directories



- Build FCB for file info. Copy information from its directory entry
- Build FCB for file alpha. Copy information from its directory entry
- Put address of info's FCB in alpha's FCB
-

9. A) Define message passing . illustrate the implementation of message passing

Process P_i

```
...
send (pj, <message>);
...
```

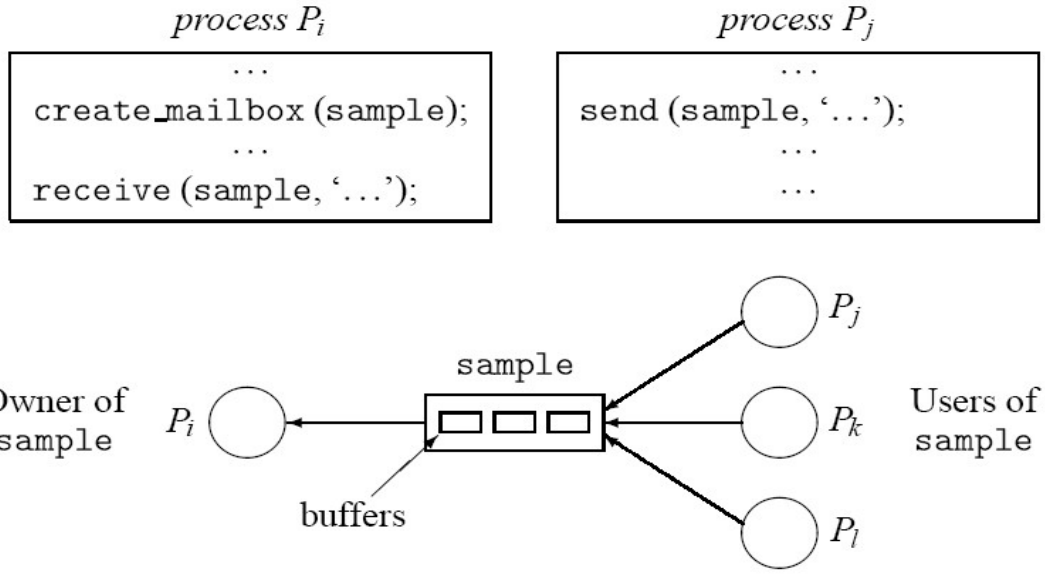
Process P_j

```
...
receive (Pi, msg_area);
...
```

- The sender process names the destination process and provides the message
- The destination process specifies an area in which the message should be put

- Issues in message passing
- Naming of processes
    - Direct and indirect naming
    - Direct: Process names are specified in send / receive commands
    - Indirect: Process names are inferred by the kernel
    - Symmetric and asymmetric naming
    - Symmetric: Both sender and receiver processes specify each other's names
    - Asymmetric: Receiving process does not specify name of a sender process
- Method for transferring messages
    - Process executing a receive command is blocked until a message is delivered
    - The sender process may or may not be blocked until delivery
    - Synchronous message passing: sender is blocked
    - Simplifies message passing, saves memory
    - Asynchronous message passing: sender is not blocked
    - Order in which messages are delivered
    - Handling of exceptions like non-existing processes, undeliverable messages
- Kernel responsibilities
    - Buffering of messages
    - Kernel builds an *interprocess message control block* (IMCB)
    - IMCB is stored in an appropriate data structure
    - The message may be stored in IMCB or separately
    - Blocking and unblocking of processes
    - May be performed using *event control blocks* (ECBs)

9. B)mailbox



- A mailbox has a name and is a repository of messages
- Processes connect to a mailbox and send / receive messages from it
- Benefits of using a mailbox
  - Anonimity of receiver
    - A sender process need not know identity of receiver process
  - Classification of messages
    - A process can use numerous mailboxes, one for each kind of message

```
repeat
    while receive (book, flags1, msg_area1) returns a message
        while receive (cancel, flags2, msg_area2) returns a message
            process the cancellation;
        process the booking;
    if receive (enquire, flags3, msg_area3) returns a message then
        while receive (cancel, flags2, msg_area2) returns a message
            process the cancellation;
        process the enquiry;
forever
```

- An airline reservations process uses three mailboxes— *book, enquire, receive*
- This way it can process cancellations followed by bookings followed by queries

10. A)Define deadlock. Explain deadlock handling approaches.
    - A deadlock is a situation in which a set of processes face indefinite waits
    - A deadlock results in reduction of concurrency and parallelism
    - Both response times to applications and system performance suffer

        * Definition
    - A deadlock involving a set of processes *D* is a situation in which
        - Every process P_i in D is blocked on some event e_i
        - Event e_i can be caused only by actions of some process(es) in *D*
        * Various kinds of deadlocks can arise in an OS, e.g.,
    - Synchronization deadlocks
        - Processes wait for one another to perform expected actions
    - Message communication deadlocks
        - Processes wait for messages from one another
    - Resource deadlocks
        - Processes wait for requested resources to be allocated to them
        * An OS handles only resource deadlocks, if at all
        * Events related to resource allocation
    - Resource request
        - A process requests a resource through a system call
        - The process is blocked if the resource cannot be allocated to it
    - Resource allocation to a process
        - If the process was blocked, its state is changed to *ready*
        - The process becomes a *holder* of the resource
        - The resource state is changed to 'allocated'
    - Resource release
        - A process releases a resource through a system call
        - The OS allocates the resource to a process blocked for it, if any
          Otherwise, it changes the state of the resource to 'free

- A resource deadlock can arise only if the following conditions hold simultaneously
    – Non-shareable resources
        * Processes need exclusive access to resources
    – Hold-and-wait
        * A process continues to hold resources allocated to it while it waits for other resources
    – No preemption
        * OS cannot preempt a resource from one process and allocate it to another process
    – Circular waits
        * A circular chain of hold-and-wait conditions exists

10. B)Deadlock prevention approaches;
    Use resource allocation policies that make deadlocks impossible
    a. How to design a deadlock prevention approach?
        i. Consider the conditions for deadlock
        ii. Ensure that they cannot hold simultaneously; i.e., make sure that one of them cannot arise
    b. A simple policy

i. Allocate all resources required by a process together. Hence the hold-and-wait condition is never satisfied
    1. This policy is expensive in practice as resources may be requested much before they are actually needed by a process