12/2017

Ashutosh :-Operating system

Operating System

Short Notes

**1 OPERATING SYSTEM – OVERVIEW**

**Definition: -** An operating system is a program that acts as an interface between the user and the computer hardware and controls the execution of all kinds of programs

**Important Functions:**

1. **Memory Management: -** It is referring to management of Main Memory
2. **Processor Management: -** OS decides which process gets the processor when and for how much time. This function is called process scheduling. **Traffic controller** Keeps tracks of processor and status of process.
3. **Device Management: -** An OS manages device communication via their respective drivers. **I/O controller** Keeps tracks of all devices
4. **File Management: -** It is normally organized into directories for easy navigation and usage. **File System** Keeps track of information, location, uses, status.
5. **Other Important Activities: -**
6. **Security -** By means of password and similar other techniques, it prevents unauthorized access to programs and data.
7. **Control over system performance -** Recording delays between request for a service and response from the system.
8. **Job accounting -** Keeping track of time and resources used by various jobs and users.
9. **Error detecting aids -** Production of dumps, traces, error messages, and other debugging and error detecting aids.
10. **Coordination between other software and users -** Coordination and assignment of compilers, interpreters, assemblers and other software to the various users of the computer systems.

**2 OPERATING SYSTEM ─ TYPES**

**Batch Operating System**

User did not interact to the computer directly. User prepares his job on an off-line device like punch cards and submits it to the computer operator. jobs with similar needs are batched together and run as a group.

**Time-sharing Operating Systems**

Processor's time which is shared among multiple users simultaneously is termed as time-sharing.

**Distributed Operating System**

Distributed systems use multiple central processors to serve multiple real-time applications and multiple users.

**Network Operating System**

A Network Operating System runs on a server and provides the server the capability to manage data, users, groups, security, applications, and other networking functions.

**Real-Time Operating System**

A real-time system is defined as a data processing system in which the time interval required to process and respond to inputs is so small that it controls the environment.

1. **Hard real-time systems** guarantee that critical tasks complete on time. In hard real-time systems, secondary storage is limited or missing and the data is stored in ROM. In these systems, virtual memory is almost never found.
2. **Soft real-time systems** are less restrictive. A critical real-time task gets priority over other tasks and retains the priority until it completes. Soft real-time systems have limited utility than hard real-time systems. For example, multimedia, virtual reality, Advanced Scientific Projects like undersea exploration and planetary rovers, etc.

**3 OPERATING SYSTEM ─ SERVICES**

1. **Program Execution**
2. **I/O Operation**
3. **File System Manipulation**
4. **Communication**
5. **Error Handling**
6. **Resource Management**
7. **Operating System**
8. **Protection**

**4 OPERATING SYSTEM ─ PROPERTIES**

1. **Batch Processing: -** an Operating System collects the programs and data together in a batch before processing starts.
2. **Multitasking: -** Multitasking is when multiple jobs are executed by the CPU simultaneously by switching between them.
3. **Multiprogramming: -** When two or more programs reside in memory at the same time, is referred as multiprogramming.
4. **Interactivity: -** ability of users to interact with a computer system.
5. **Real-Time Systems: -** are usually dedicated, embedded systems.
6. **Distributed Environment: -** refers to multiple independent CPUs or processors in a computer system.
7. **Spooling: -** is an acronym for simultaneous peripheral operations on line. Spooling refers to putting data of various I/O jobs in a buffer. Provide waiting station where data can rest while slower device catches up. Provide parallel computation. Uses a disk as a very large buffer. Capable of overlapping I/O operation.

**5 OPERATING SYSTEM ─ PROCESSES**

1. **Process: -** A process is defined as an entity which represents the basic unit of work to be implemented in the system. it can be divided into four sections
   1. **Stack: -** The Process Stack contains the temporary data such as method/function parameters, return address, and local variables.
   2. **Heap: -** This is a dynamically allocated memory to a process during its runtime.
   3. **Text: -** This includes the current activity represented by the value of Program Counter and the contents of the processor's registers.
   4. **Data: -** This section contains the global and static variables.
2. **Program: -** is a piece of code which may be a single line or millions of lines.
3. **Process Life Cycle: -** When a process executes, it passes through different states.
   1. **Start: -** This is the initial state when a process is first started/created.
   2. **Ready: -** The process is waiting to be assigned to a processor.
   3. **Running: -** processor executes its instructions.
   4. **Waiting: -** if it needs to wait for a resource, such as waiting for user input
   5. **Terminated or Exit: -** when finishes its execution
4. **Process Control Block (PCB): -** Is a data structure maintained by the OS for every process.
   1. **Process State: -** The current state of the process i.e., whether it is ready, running, waiting, or whatever**.**
   2. **Process privileges: -**This is required to allow/disallow access to system resources.
   3. **Process ID: -** Unique identification for each of the process in the operating system.
   4. **Pointer A pointer: -** to parent process.
   5. **Program Counter: -** is a pointer to the address of the next instruction to be executed in process.
   6. **CPU registers: -** Various registers where process need to be stored for execution.
   7. **CPU Scheduling Information: -** Process priority and other scheduling information which is required to schedule the process.
   8. **Memory management information: -** This includes the information of page table, memory limits, Segment table depending on memory used by the operating system.
   9. **Accounting information: -** This includes the amount of CPU used for process execution, time limits, execution ID etc.
   10. **IO status information: -** This includes a list of I/O devices allocated to the process

**6 OPERATING SYSTEM ─ PROCESS SCHEDULING**

**Definition**

The process scheduling is the activity of the process manager. Process scheduling is an essential part of a Multiprogramming operating systems

**Process Scheduling Queues**

The OS maintains a separate queue for each of the process states and PCBs of all processes in the same execution state are placed in the same queue.

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

**Two-State Process Model**

Two-state process model refers to running and non-running states.

1. **Running: -** When a new process is created, it enters into the system as in the running state.
2. **Not Running: -** Processes that are not running are kept in queue, waiting for their turn to execute.

Each entry in the queue is a pointer to a particular process. Queue is implemented by using linked list. Use of dispatcher is as follows. When a process is interrupted, that process is transferred in the waiting queue. If the process has completed or aborted, the process is discarded. In either case, the dispatcher then selects a process from the queue to execute.

**Schedulers**

main task is to select the jobs to be submitted into the system and to decide which process to run. Schedulers are of three types: I. Long-Term Scheduler, II. Short-Term Scheduler, III. Medium-Term Scheduler.

**Long-Term Scheduler**

It is also called a job scheduler. A long-term scheduler determines which programs are admitted to the system for processing. It selects processes from the queue and loads them into memory for execution. The primary objective of the job scheduler is to provide a balanced mix of jobs, such as I/O bound and processor bound. When a process changes the state from new to ready, then there is use of long-term scheduler.

**Short-Term Scheduler**

It is also called as CPU scheduler. It is the change of ready state to running state of the process. Short-term schedulers, also known as dispatchers, make the decision of which process to execute next.

**Medium-Term Scheduler**

Medium-term scheduling is a part of swapping. A suspended processes cannot make any progress towards completion. In this condition, to remove the process from memory and make space for other processes, the suspended process is moved to the secondary storage. This process is called swapping, and the process is said to be swapped out or rolled out.

**Context Switch**

A context switch is the mechanism to store and restore the state or context of a CPU in Process Control block so that a process execution can be resumed from the same point at a later time. When the process is switched, the following information is stored for later use.

* Program Counter
* Scheduling information
* Base and limit register value
* Currently used register
* Changed State
* I/O State information
* Accounting information.

**Comparison among Schedulers**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.N.** | **Long-Term Scheduler** | **Short-Term Scheduler** | **Medium-Term Scheduler** |
| 1 | It is a job scheduler | It is a CPU scheduler | It is a process swapping scheduler. |
| 2 | Speed is lesser than short term scheduler | Speed is fastest among other two | Speed is in between both short and long term scheduler. |
| 3 | It controls the degree of multiprogramming | It provides lesser control over degree of multiprogramming | It reduces the degree of multiprogramming. |
| 4 | It is almost absent or minimal in time sharing system | It is also minimal in time sharing system | It is a part of Time sharing systems. |
| 5 | It selects processes from pool and loads them into memory for execution | It selects those processes which are ready to execute | It can re-introduce the process into memory and execution can be continued. |

**7 OPERATING SYSTEM ─ SCHEDULING ALGORITHMS**

Non-preemptive algorithms are designed so that once a process enters the running state, it cannot be preempted until it completes its allotted time,

whereas the preemptive scheduling is based on priority where a scheduler may preempt a low priority running process anytime when a high priority process enters into a ready state.

**First Come, First Served (FCFS)**

* Jobs are executed on first come, first served basis.
* It is a non-preemptive scheduling algorithm.
* Easy to understand and implement.
* Its implementation is based on FIFO queue.
* Poor in performance, as average wait time is high

**Shortest Job Next (SJN)**

* This is also known as shortest job first, or SJF.
* This is a non-preemptive scheduling algorithm.
* Best approach to minimize waiting time.
* Easy to implement in Batch systems where required CPU time is known in advance.
* Impossible to implement in interactive systems where the required CPU time is not known.
* The processer should know in advance how much time a process will take.

**Priority Based Scheduling**

* Priority scheduling is a non-preemptive algorithm.
* Each process is assigned a priority. Process with highest priority is to be executed first and so on.
* Processes with same priority are executed on first come first served basis.
* Priority can be decided based on memory requirements, time requirements or any other resource requirement.

**Shortest Remaining Time**

* Shortest remaining time (SRT) is the preemptive version of the SJN algorithm.
* The processor is allocated to the job closest to completion but it can be preempted by a newer ready job with shorter time to completion.
* Impossible to implement in interactive systems where required CPU time is not known.
* It is often used in batch environments where short jobs need to be given preference.

**Round Robin Scheduling**

* Round Robin is a preemptive process scheduling algorithm.
* Each process is provided a fix time to execute; it is called a **quantum**.
* Once a process is executed for a given time period, it is preempted and other process executes for a given time period.
* Context switching is used to save states of preempted processes.

**Multiple-Level Queues Scheduling**

Multiple-level queues are not an independent scheduling algorithm. They make use of other existing algorithms to group and schedule jobs with common characteristics.

 Multiple queues are maintained for processes with common characteristics.

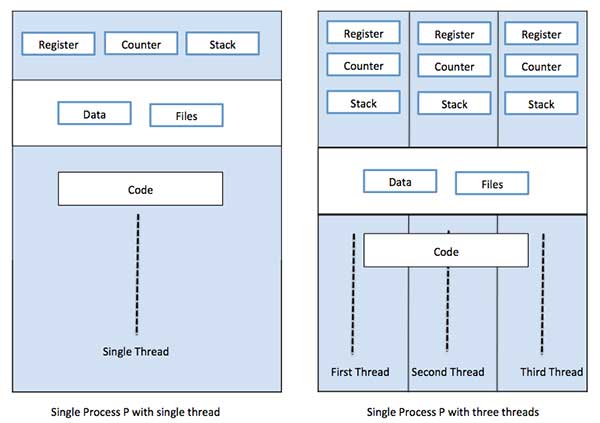
 Each queue can have its own scheduling algorithms.

 Priorities are assigned to each queue.

**8 OPERATING SYSTEM ─ MULTITHREADING**

**What is a Thread?**

A thread is a flow of execution through the process code, with its own program counter that keeps track of which instruction to execute next, system registers which hold its current working variables, and a stack which contains the execution history. A thread is also called a lightweight process.



**Difference between Process and Thread**

|  |  |  |
| --- | --- | --- |
| **S.N.** | **Process** | **Thread** |
| 1 | Process is heavy weight or resource intensive. | Thread is light weight, taking lesser resources than a process. |
| 2 | Process switching needs interaction with operating system. | Thread switching does not need to interact with operating system. |
| 3 | In multiple processing environments, each process executes the same code but has its own memory and file resources. | All threads can share same set of open files, child processes. |
| 4 | If one process is blocked, then no other process can execute until the first process is unblocked. | While one thread is blocked and waiting, a second thread in the same task can run. |
| 5 | Multiple processes without using threads use more resources. | Multiple threaded processes use fewer resources. |
| 6 | In multiple processes each process operates independently of the others. | One thread can read, write or change another thread's data. |

**Advantages of Thread**

* Threads minimize the context switching time.
* Use of threads provides concurrency within a process.
* Efficient communication.
* It is more economical to create and context switch threads.
* Threads allow utilization of multiprocessor architectures to a greater scale and efficiency.

**Types of Thread**

1. **User Level Threads (User managed threads)**

The thread management kernel is not aware of the existence of threads. The thread library contains code for creating and destroying threads, for passing message and data between threads, for scheduling thread execution and for saving and restoring thread contexts. The application starts with a single thread.

1. **Kernel Level Threads (Operating System managed threads)**

Thread management is done by the Kernel. Kernel threads are supported directly by the operating system. The Kernel performs thread creation, scheduling and management in Kernel space. Kernel threads are generally slower to create and manage than the user threads.

**Multithreading Models**

Some operating system provide a combined user level thread and Kernel level thread.

1. **Many-to-Many Model** multiplexes any number of user threads onto an equal or smaller number of kernel threads.
2. **Many-to-One Model** maps many user level threads to one Kernel-level thread.
3. **One-to-One Model** one-to-one relationship of user-level thread to the kernel-level thread

**Difference between User-Level & Kernel-Level Thread**

|  |  |  |
| --- | --- | --- |
| **S.N.** | **User-Level Threads** | **Kernel-Level Thread** |
| 1 | User-level threads are faster to create and manage. | Kernel-level threads are slower to create and manage. |
| 2 | Implementation is by a thread library at the user level. | Operating system supports creation of Kernel threads. |
| 3 | User-level thread is generic and can run on any operating system. | Kernel-level thread is specific to the operating system. |
| 4 | Multi-threaded applications cannot take advantage of multiprocessing. | Kernel routines themselves can be multithreaded. |

**9 OPERATING SYSTEM ─ MEMORY MANAGEMENT**

**1. Process Address Space**

It is the set of logical addresses that a process references in its code.

1. **Symbolic addresses**: - The addresses used in a source code. The variable names, constants, and instruction labels are the basic elements of the symbolic address space.
2. **Relative addresses: -** At the time of compilation, a compiler converts symbolic addresses into relative addresses.
3. **Physical addresses: -** The loader generates these addresses at the time when a program is loaded into main memory.

Virtual and physical addresses are the same in compile-time and load-time address-binding schemes. Virtual and physical addresses differ in execution-time address-binding scheme.

The set of all logical addresses generated by a program is referred to as a logical address space. The set of all physical addresses corresponding to these logical addresses is referred to as a physical address space.

**2. Static vs Dynamic Loading**

The choice between Static or Dynamic Loading is to be made at the time of computer program being developed.

At the time of loading, with **static loading**, the absolute program (and data) is loaded into memory in order for execution to start.

If you are using dynamic loading, dynamic routines of the library are stored on a disk in relocatable form and are loaded into memory only when they are needed by the program.

**3. Static vs Dynamic Linking**

**Static linking** is used, the linker combines all other modules needed by a program into a single executable program to avoid any runtime dependency.

**Dynamic linking** is used, it is not required to link the actual module or library with the program, rather a reference to the dynamic module is provided at the time of compilation and linking.

**4. Swapping**

Swapping is a mechanism in which a process can be swapped temporarily out of main memory (or move) to secondary storage (disk) and make that memory available to other processes. Swapping is also known as a technique for memory compaction. **Swap Out** Main Memory to Secondary Memory, **Swap In** Secondary Memory to Main Memory.

**5. Memory Allocation**

Main memory usually has two partitions:

* **Low Memory --** Operating system resides in this memory.
* **High Memory --** User processes are held in high memory.

Operating system uses the following memory allocation mechanism.

* **Single-partition allocation: -** relocation-register scheme is used to protect user processes from each other, and from changing operating-system code and data. Relocation register contains value of smallest physical address whereas limit register contains range of logical addresses. Each logical address must be less than the limit register.
* **Multiple-partition allocation: -** main memory is divided into a number of fixed-sized partitions where each partition should contain only one process.

**6. Fragmentation**

As processes are loaded and removed from memory, the free memory space is broken into little pieces. It happens after sometimes that processes cannot be allocated to memory blocks considering their small size and memory blocks remains unused. This problem is known as Fragmentation.

* **External fragmentation: -** Total memory space is enough to satisfy a request or to reside a process in it, but it is not contiguous, so it cannot be used. External fragmentation can be reduced by compaction or shuffle memory contents to place all free memory together in one large block.
* **Internal fragmentation: -** Memory block assigned to process is bigger. Some portion of memory is left unused, as it cannot be used by another process. Internal fragmentation can be reduced by effectively assigning the smallest partition but large enough for the process.

**7. Paging**

A computer can address more memory than the amount physically installed on the system. This extra memory is actually called **virtual memory** and it is a section of a hard that's set up to emulate the computer's RAM. Paging technique plays an important role in implementing virtual memory.

Paging is a memory management technique in which process address space is broken into blocks of the same size called **pages**. The size of the process is measured in the number of pages.

main memory is divided into small fixed-sized blocks of (physical) memory called **frames.** Size of a frame is kept the same as that of a page to have optimum utilization of the main memory and to avoid external fragmentation.

**Address Translation**

Page address is called **logical address** and represented by **page number** and the **offset**.

**Logical Address = Page number + page offset**

Frame address is called **physical address** and represented by a **frame number** and the **offset.**

Physical Address = Frame number + page offset

A data structure called page map table is used to keep track of the relation between a page of a process to a frame in physical memory.

**8. Segmentation**

Segmentation is a memory management technique in which each job is divided into several segments of different sizes, one for each module that contains pieces that perform related functions.

Segmentation memory management works very similar to paging but here segments are of variable-length where as in paging pages are of fixed size.

The operating system maintains a segment map table for every process and a list of free memory blocks along with segment numbers, their size and corresponding memory locations in main memory.

**10 OPERATING SYSTEM ─ VIRTUAL MEMORY**

A computer can address more memory than the amount physically installed on the system. This extra memory is actually called virtual memory and it is a section of a hard disk that's set up to emulate the computer's RAM.

Virtual memory is commonly implemented by demand paging. It can also be implemented in a segmentation system. Demand segmentation can also be used to provide virtual memory.

**1. Demand Paging**

A demand paging system is quite similar to a paging system with swapping where processes reside in secondary memory and pages are loaded only on demand, not in advance.

if the program references a page which is not available in the main memory because it was swapped out a little ago, the processor treats this invalid memory reference as a **page fault** and transfers control from the program to the operating system to demand the page back into the memory.

**2. Page Replacement Algorithm**

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. Paging happens whenever a page fault occurs and a free page cannot be used for allocation purpose accounting to reason that pages are not available or the number of free pages is lower than required pages.

A page replacement algorithm looks at the limited information about accessing the pages provided by hardware, and tries to select which pages should be replaced to minimize the total number of page misses, while balancing it with the costs of primary storage and processor time of the algorithm itself.

**3. Reference String**

The string of memory references is called reference string. Reference strings are generated artificially or by tracing a given system and recording the address of each memory reference. The latter choice produces a large number of data, where we note two things.

 For a given page size, we need to consider only the page number, not the entire address.

 If we have a reference to a page p, then any immediately following references to page p will never cause a page fault. Page p will be in memory after the first reference; the immediately following references will not fault.

**4. First In First Out (FIFO) Algorithm**

 Oldest page in main memory is the one which will be selected for replacement.

 Easy to implement, keep a list, replace pages from the tail and add new pages at the head.

**5. Optimal Page Algorithm**

 An optimal page-replacement algorithm has the lowest page-fault rate of all algorithms. An optimal page-replacement algorithm exists, and has been called OPT or MIN.

 Replace the page that will not be used for the longest period of time. Use the time when a page is to be used.

**6. Least Recently Used (LRU) Algorithm**

 Page which has not been used for the longest time in main memory is the one which will be selected for replacement.

 Easy to implement, keep a list, replace pages by looking back into time.

**7. Page Buffering Algorithm**

 To get a process start quickly, keep a pool of free frames.

 On page fault, select a page to be replaced.

 Write the new page in the frame of free pool, mark the page table and restart the process.

 Now write the dirty page out of disk and place the frame holding replaced page in free pool.

**8. Least Frequently Used (LFU) Algorithm**

The page with the smallest count is the one which will be selected for replacement.

This algorithm suffers from the situation in which a page is used heavily during the initial phase of a process, but then is never used again.

**9. Most Frequently Used (MFU) Algorithm**

This algorithm is based on the argument that the page with the smallest count was probably just brought in and has yet to be used.

**11 OPERATING SYSTEM ─ I/O HARDWARE**

I/O devices can be divided into two categories:

* **Block devices: -** A block device is one with which the driver communicates by sending entire blocks of data.
* **Character devices: -** A character device is one with which the driver communicates by sending and receiving single characters (bytes, octets).

**1. Device Controllers**

Device drivers are software modules that can be plugged into an OS to handle a particular device.

The Device Controller works like an interface between a device and a device driver.

**2. Synchronous vs Asynchronous I/O**

* **Synchronous I/O —** In this scheme CPU execution waits while I/O proceeds
* **Asynchronous I/O —** I/O proceeds concurrently with CPU execution

**3. Communication to I/O Devices**

There are three approaches available to communicate.

* **Special Instruction I/O: -** This uses CPU instructions that are specifically made for controlling I/O devices
* **Memory-mapped I/O: -** When using memory-mapped I/O, the same address space is shared by memory and I/O devices.
* **Direct memory access (DMA): -** CPU grants I/O module authority to read from or write to memory without involvement.

**4. Polling vs Interrupts I/O**

**Polling I/O: -** Polling is the simplest way for an I/O device to communicate with the processor the processor. The process of periodically checking status of the device to see if it is time for the next I/O operation, is called polling.

**Interrupts I/O: -** An alternative scheme for dealing with I/O is the interrupt-driven method. An interrupt is a signal to the microprocessor from a device that requires attention.

**12 OPERATING SYSTEM ─ I/O SOFTWARE**

**I/O software is often organized in the following layers:**

* **User Level Libraries:** This provides simple interface to the user program to perform input and output.
* **Kernel Level Modules:** This provides device driver to interact with the device controller and device independent I/O modules used by the device drivers.
* **Hardware:** This layer includes actual hardware and hardware controller which interact with the device drivers and makes hardware alive.

**1. Device Drivers**

Device drivers are software modules that can be plugged into an OS to handle a particular device. Device drivers are software modules that can be plugged into an OS to handle a particular device. Operating System takes help from device drivers to handle all I/O devices.

**device driver handles** a request is as follows: Suppose a request comes to read a block N. If the driver is idle at the time a request arrives, it starts carrying out the request immediately. Otherwise, if the driver is already busy with some other request, it places the new request in the queue of pending requests.

**2. Interrupt handlers**

It also known as an interrupt service routine or ISR. It is piece of software (device driver) or more specifically a callback function in an OS. Its execution is triggered by the reception of an interrupt.

The interrupt mechanism accepts an address (a number that selects a specific interrupt handling routine/function from a small set). In most architectures, this address is an offset stored in a table called the **interrupt vector table**. This vector contains the memory addresses of specialized interrupt handlers.

**3. Device-Independent I/O Software**

The basic function is to perform the I/O functions that are common to all devices and to provide a uniform interface to the user-level software.

**4. User-Space I/O Software**

These are the libraries which provide richer and simplified interface to access the functionality of the kernel or ultimately interactive with the device drivers.

**5. Kernel I/O Subsystem**

* **Scheduling -** Kernel schedules a set of I/O requests to determine a good order in which to execute them. When an application issues a blocking I/O system call, the request is placed on the queue for that device. The Kernel I/O scheduler rearranges the order of the queue to improve the overall system efficiency and the average response time experienced by the applications.
* **Buffering -** Kernel I/O Subsystem maintains a memory area known as **buffer** that stores data while they are transferred between two devices or between a device with an application operation.
* **Caching -** Kernel maintains cache memory which is region of fast memory that holds copies of data. Access to the cached copy is more efficient than access to the original.
* **Spooling and Device Reservation -** A spool is a buffer that holds output for a device. In some operating systems, spooling is managed by a system daemon process. In other operating systems, it is handled by an in kernel thread.
* **Error Handling -** An operating system that uses protected memory can guard against many kinds of hardware and application errors.

**13 OPERATING SYSTEM ─ FILE SYSTEM**

**1. File**

A file is a named collection of related information that is recorded on secondary storage. a file is a sequence of bits, bytes, lines or records whose meaning is defined by the files creator and user.

**2. File Structure**

A File Structure should be according to a required format that the operating system can understand.

**3. File Type**

File type refers to the ability of the operating system to distinguish different types of file such as text files source files and binary files etc.

**Ordinary files**

* These are the files that contain user information.
* These may have text, databases or executable program.
* The user can apply various operations on such files like add, modify, delete or even remove the entire file.

**Directory files**

* These files contain list of file names and other information related to these files.

**Special files**

* These files are also known as device files.
* These files represent physical device like disks, terminals, printers, networks, tape drive etc.

These files are of two types:

1. **Character special files -** data is handled character by character as in case of terminals or printers.
2. **Block special files -** data is handled in blocks as in the case of disks and tapes.

**4. File Access Mechanisms**

File access mechanism refers to the manner in which the records of a file may be accessed. There are several ways to access files:

**1 Sequential access: -**

* A sequential access is that in which the records are accessed in some sequence.
* This access method is the most primitive one.

**2 Direct/Random access**

* Random access file organization provides, accessing the records directly.
* Each record has its own address on the file with by the help of which it can be directly accessed for reading or writing.
* The records need not be in any sequence within the file and they need not be in adjacent locations on the storage medium.

**3 Indexed sequential access**

* This mechanism is built up on base of sequential access.
* An index is created for each file which contains pointers to various blocks.
* Index is searched sequentially and its pointer is used to access the file directly.

**5. Space Allocation**

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 a contiguous address space on disk.
* Assigned disk address is in linear order.
* Easy to implement.
* External fragmentation is a major issue with this type of allocation technique.

**2. Linked Allocation**

* Each file carries a list of links to disk blocks.
* Directory contains link / pointer to first block of a file.
* No external fragmentation
* Effectively used in sequential access file.
* Inefficient in case of direct access file.

**3. Indexed Allocation**

* Provides solutions to problems of contiguous and linked allocation.
* An index block is created having all pointers to files.
* Each file has its own index block which stores the addresses of disk space occupied by the file.
* Directory contains the addresses of index blocks of files.

**14 OPERATING SYSTEM ─ SECURITY**

Security refers to providing a protection system to computer system resources.

**1. Authentication**

Authentication refers to identifying each user of the system and associating the executing programs with those users.

1. **Username / Password**
2. **User card/key**
3. **User attribute - fingerprint/ eye retina pattern/ signature**

**2. One Time passwords**

A unique password is required every time user tries to login into the system.

* **Random numbers -** Users are provided cards having numbers printed along with corresponding alphabets. System asks for numbers corresponding to few alphabets randomly chosen.
* **Secret key -** User are provided a hardware device which can create a secret id mapped with user id. System asks for such secret id which is to be generated every time prior to login.
* **Network password -** Some commercial applications send one-time passwords to user on registered mobile/ email which is required to be entered prior to login.

**3. Program Threats**

Operating system's processes and kernel do the designated task as instructed. If a user program made these process do malicious tasks, then it is known as **Program Threats**. Following is the list of some well-known program threats.

* **Trojan Horse -** Such program traps user login credentials and stores them to send to malicious user who can later on login to computer and can access system resources.
* **Trap Door -** If a program which is designed to work as required, have a security hole in its code and perform illegal action without knowledge of user then it is called to have a trap door.
* **Logic Bomb -** Logic bomb is a situation when a program misbehaves only when certain conditions met otherwise it works as a genuine program. It is harder to detect.
* **Virus -** Virus as name suggest can replicate themselves on computer system. They are highly dangerous and can modify/delete user files, crash systems. A virus is generally a small code embedded in a program. As user accesses the program, the virus starts getting embedded in other files/ programs and can make system unusable for user.

**4. System Threats**

System threats refers to misuse of system services and network connections to put user in trouble. System threats can be used to launch program threats on a complete network called as **program attack.** some well-known system threats.

* **Worm -** is a process which can choked down a system performance by using system resources to extreme levels. A Worm process generates its multiple copies where each copy uses system resources, prevents all other processes to get required resources. Worms processes can even shut down an entire network.
* **Port Scanning -** is a mechanism or means by which a hacker can detects system vulnerabilities to make an attack on the system.
* **Denial of Service -** attacks normally prevents user to make legitimate use of the system. For example, a user may not be able to use internet if denial of service attacks browser's content settings.

**5. Computer Security Classifications**

**I. Type A: -** Highest Level. Uses formal design specifications and verification techniques. Grants a high degree of assurance of process security.

**II. Type B: -** Provides mandatory protection system. Have all the properties of a class C2 system. Attaches a sensitivity label to each object. It is of three types.

* **B1 -** Maintains the security label of **each object in the system**. Label is used for making decisions to access control.
* **B2 -** Extends the sensitivity labels to **each system resource**, such as storage objects, supports covert channels and auditing of events.
* **B3 -** Allows creating **lists or user groups for access control** to grant access or revoke access to a given named object.

**III. Type C: -** Provides protection and user accountability using audit capabilities. It is of two types.

* **C1 -** Incorporates controls so that users can protect their private information and keep other users from accidentally reading / deleting their data. UNIX versions are mostly Cl class.
* **C2 -** Adds an individual-level access control to the capabilities of a Cl level system

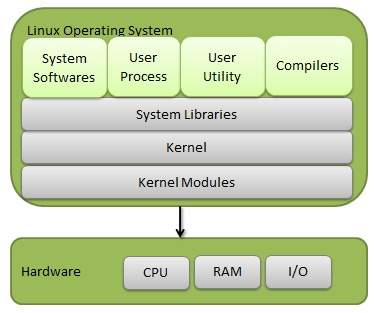
**IV. Type D: -**Lowest level. Minimum protection. MS-DOS, Window 3.1 fall in this category.

**15 OPERATING SYSTEM ─ LINUX**

Linux is one of popular version of UNIX operating System. It is open source as its source code is freely available. It is free to use. Linux was designed considering UNIX compatibility. Its functionality list is quite similar to that of UNIX.

**Components of Linux System**

Linux Operating System has primarily three components

**Kernel -** is the core part of Linux. It is responsible for all major activities of this operating system. It consists of various modules and it interacts directly with the underlying hardware. Kernel provides the required abstraction to hide low level hardware details to system or application programs.

**System Library -** System libraries are special functions or programs using which application programs or system utilities accesses Kernel's features. These libraries implement most of the functionalities of the operating system and do not requires kernel module's code access rights.

**System Utility -** System Utility programs are responsible to do specialized, individual level tasks.

**Kernel Mode vs User Mode**

Kernel component code executes in a special privileged mode called **kernel mode** with full access to all resources of the computer.

Support code which is not required to run in kernel mode is in System Library. User programs and other system programs works in **User Mode** which has no access to system hardware and kernel code.

**Basic Features**

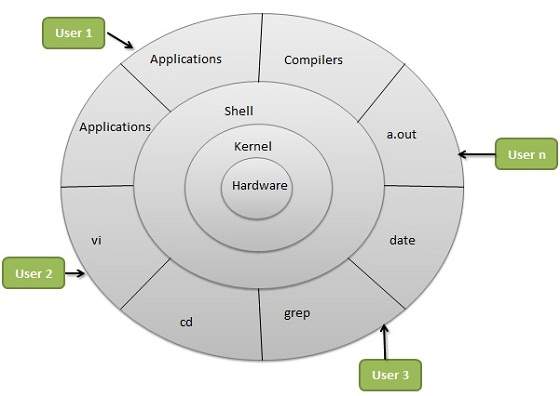
Following are some of the important features of Linux Operating System.

* **Portable -** Portability means software can work on different types of hardware in same way. Linux kernel and application programs supports their installation on any kind of hardware platform.
* **Open Source -** Linux source code is freely available and it is community based development project. Multiple teams work in collaboration to enhance the capability of Linux operating system and it is continuously evolving.
* **Multi-User -** Linux is a multiuser system means multiple users can access system resources like memory/ ram/ application programs at same time.
* **Multiprogramming -** Linux is a multiprogramming system means multiple applications can run at same time.
* **Hierarchical File System -** Linux provides a standard file structure in which system files/ user files are arranged.
* **Shell -** Linux provides a special interpreter program which can be used to execute commands of the operating system. It can be used to do various types of operations, call application programs. etc.
* **Security -** Linux provides user security using authentication features like password protection/ controlled access to specific files/ encryption of data.

**Architecture**

The following illustration shows the architecture of a Linux system:

The architecture of a Linux System consists of the following layers:

**Hardware layer -** Hardware consists of all peripheral devices (RAM/ HDD/ CPU etc.).

**Kernel –** It is the core component of Operating System, interacts directly with hardware, provides low level services to upper layer components.

**Shell -** An interface to kernel, hiding complexity of kernel's functions from users. The shell takes commands from the user and executes kernel's functions.

**Utilities -** Utility programs that provide the user most of the functionalities of an operating systems.

**Linux Commands**

|  |  |  |
| --- | --- | --- |
| File Commands | | |
| 1 | ls | Directory listing |
| 2 | ls –al | Formatted listing with hidden files |
| 3 | ls –lt | Sorting the Formatted listing by time modification |
| 4 | cd dir | Change directory to dir |
| 5 | Cd | Change to home directory |
| 6 | Pwd | Show current working directory |
| 7 | mkdir dir | Creating a directory dir |
| 8 | cat >file | Places the standard input into the file |
| 9 | more file | Output the contents of the file |
| 10 | head file | Output the first 10 lines of the file |
| 11 | tail file | Output the last 10 lines of the file |
| 12 | tail -f file | Output the contents of file as it grows,starting with the last 10 lines |
| 13 | touch file | Create or update file |
| 14 | rm file | Deleting the file |
| 15 | rm -r dir | Deleting the directory |
| 16 | rm -f file | Force to remove the file |
| 17 | rm -rf dir | Force to remove the directory dir |
| 18 | cp file1 file2 | Copy the contents of file1 to file2 |
| 19 | cp -r dir1 dir2 | Copy dir1 to dir2;create dir2 if not present |
| 20 | mv file1 file2 | Rename or move file1 to file2,if file2 is an existing directory |
| 21 | ln -s file link | Create symbolic link link to file |

|  |  |  |
| --- | --- | --- |
| Process management | | |
| 1 | ps | To display the currently working processes |
| 2 | top | Display all running process |
| 3 | kill pid | Kill the process with given pid |
| 4 | killall proc | Kill all the process named proc |
| 5 | pkill pattern | Will kill all processes matching the pattern |
| 6 | bg | List stopped or background jobs, resume a stopped job in the background |
| 7 | Fg | Brings the most recent job to foreground |
| 8 | fg n | Brings job n to the foreground |

|  |  |  |
| --- | --- | --- |
| File permission | | |
| 1 | chmod octal file | Change the permission of file to octal, which can be found separately for user, group, world by adding,  • 4-read(r)  • 2-write(w)  • 1-execute(x) |

|  |  |  |
| --- | --- | --- |
| Searching | | |
| 1 | grep pattern file | Search for pattern in file |
| 2 | grep -r pattern dir | Search recursively for pattern in dir |
| 3 | command | grep pattern | Search pattern in the output of a command |
| 4 | locate file | Find all instances of file |
| 5 | find -name filename | Searches in the current directory (represented by a period) and below it, for files and directories with names starting with filename |
| 6 | pgrep pattern | Searches for all the named processes, that matches with the pattern and, by default, returns their ID |

|  |  |  |
| --- | --- | --- |
| System Info | | |
| 1 | date | Show the current date and time |
| 2 | cal | Show this month's calender |
| 3 | uptime | 3. uptime Show current uptime |
| 4 | w | Display who is on line |
| 5 | whoami | Who you are logged in as |
| 6 | finger user | Display information about user |
| 7 | uname –a | Show kernel information |
| 8 | cat /proc/cpuinfo | Cpu information |
| 9 | cat proc/meminfo | Memory information |
| 10 | man command | Show the manual for command |
| 11 | df | Show the disk usage |
| 12 | du | Show directory space usage |
| 13 | free | Show memory and swap usage |
| 14 | whereis app | Show possible locations of app |
| 15 | which app | Show which applications will be run by default |

|  |  |  |
| --- | --- | --- |
| Compression | | |
| 1 | tar cf file.tar file | Create tar named file.tar containing file |
| 2 | tar xf file.tar | Extract the files from file.tar |
| 3 | tar czf file.tar.gz files | Create a tar with Gzip compression |
| 4 | tar xzf file.tar.gz | Extract a tar using Gzip |
| 5 | tar cjf file.tar.bz2 | Create tar with Bzip2 compression |
| 6 | tar xjf file.tar.bz2 | Extract a tar using Bzip2 |
| 7 | gzip file | Compresses file and renames it to file.gz |
| 8 | gzip -d file.gz | Decompresses file.gz back to file |

|  |  |  |
| --- | --- | --- |
| Network | | |
| 1 | ping host | Ping host and output results |
| 2 | whois domain | Get whois information for domains |
| 3 | dig domain | Get DNS information for domain |
| 4 | dig -x host | Reverse lookup host |
| 5 | wget file | Download file |
| 6 | wget -c file | Continue a stopped download |

|  |  |  |
| --- | --- | --- |
| Shortcuts | | |
| 1 | ctrl+c | Halts the current command |
| 2 | ctrl+z | Stops the current command, resume with fg in the foreground or bg in the background |
| 3 | ctrl+d | Logout the current session, similar to exit |
| 4 | ctrl+w | Erases one word in the current line |
| 5 | ctrl+u | Erases the whole line |
| 6 | ctrl+r | Type to bring up a recent command |
| 7 | !! | Repeats the last command |
| 8 | exit | Logout the current session |

**Unix Command**

|  |  |
| --- | --- |
| **[Unix](https://en.wikipedia.org/wiki/Unix" \o "Unix)**[**command-line interface**](https://en.wikipedia.org/wiki/Command-line_interface)**programs and**[**shell builtins**](https://en.wikipedia.org/wiki/Shell_builtin) | |
| **File system** | * [cat](https://en.wikipedia.org/wiki/Cat_(Unix)), [chmod](https://en.wikipedia.org/wiki/Chmod), [chown](https://en.wikipedia.org/wiki/Chown), [chgrp](https://en.wikipedia.org/wiki/Chgrp), [cksum](https://en.wikipedia.org/wiki/Cksum), [cmp](https://en.wikipedia.org/wiki/Cmp_(Unix)), [cp](https://en.wikipedia.org/wiki/Cp_(Unix)), [dd](https://en.wikipedia.org/wiki/Dd_(Unix)), [du](https://en.wikipedia.org/wiki/Du_(Unix)), [df](https://en.wikipedia.org/wiki/Df_(Unix)), [file](https://en.wikipedia.org/wiki/File_(command)), [fuser](https://en.wikipedia.org/wiki/Fuser_(Unix)), [ln](https://en.wikipedia.org/wiki/Ln_(Unix)), [ls](https://en.wikipedia.org/wiki/Ls), [mkdir](https://en.wikipedia.org/wiki/Mkdir), [mv](https://en.wikipedia.org/wiki/Mv), [pax](https://en.wikipedia.org/wiki/Pax_(Unix)), [pwd](https://en.wikipedia.org/wiki/Pwd), [rm](https://en.wikipedia.org/wiki/Rm_(Unix)), [rmdir](https://en.wikipedia.org/wiki/Rmdir), [split](https://en.wikipedia.org/wiki/Split_(Unix)), [tee](https://en.wikipedia.org/wiki/Tee_(command)), [touch](https://en.wikipedia.org/wiki/Touch_(Unix)), type, [umask](https://en.wikipedia.org/wiki/Umask). |
| **Processes** | * [at](https://en.wikipedia.org/wiki/At_(Unix)), [bg](https://en.wikipedia.org/wiki/Job_control_(Unix)#Commands), [crontab](https://en.wikipedia.org/wiki/Cron), [fg](https://en.wikipedia.org/wiki/Job_control_(Unix)#Commands), [kill](https://en.wikipedia.org/wiki/Kill_(command)), [nice](https://en.wikipedia.org/wiki/Nice_(Unix)), [ps](https://en.wikipedia.org/wiki/Ps_(Unix)), [time](https://en.wikipedia.org/wiki/Time_(Unix)). |
| **User environment** | * [env](https://en.wikipedia.org/wiki/Env), [exit](https://en.wikipedia.org/wiki/Exit_(command)), [logname](https://en.wikipedia.org/wiki/Logname), [mesg](https://en.wikipedia.org/wiki/Mesg), [talk](https://en.wikipedia.org/wiki/Talk_(software)), [tput](https://en.wikipedia.org/wiki/Tput), [uname](https://en.wikipedia.org/wiki/Uname), [who](https://en.wikipedia.org/wiki/Who_(Unix)), [write](https://en.wikipedia.org/wiki/Write_(Unix)). |
| **Text processing** | * [awk](https://en.wikipedia.org/wiki/AWK), [basename](https://en.wikipedia.org/wiki/Basename), [comm](https://en.wikipedia.org/wiki/Comm), [csplit](https://en.wikipedia.org/wiki/Csplit), [cut](https://en.wikipedia.org/wiki/Cut_(Unix)), [diff](https://en.wikipedia.org/wiki/Diff_utility), [dirname](https://en.wikipedia.org/wiki/Dirname), [ed](https://en.wikipedia.org/wiki/Ed_(text_editor)), [ex](https://en.wikipedia.org/wiki/Ex_(text_editor)), [fold](https://en.wikipedia.org/wiki/Fold_(Unix)), [head](https://en.wikipedia.org/wiki/Head_(Unix)), [iconv](https://en.wikipedia.org/wiki/Iconv), [join](https://en.wikipedia.org/wiki/Join_(Unix)), [m4](https://en.wikipedia.org/wiki/M4_(computer_language)), [more](https://en.wikipedia.org/wiki/More_(command)), [nl](https://en.wikipedia.org/wiki/Nl_(Unix)), [paste](https://en.wikipedia.org/wiki/Paste_(Unix)), [printf](https://en.wikipedia.org/wiki/Printf_format_string), [sed](https://en.wikipedia.org/wiki/Sed), [sort](https://en.wikipedia.org/wiki/Sort_(Unix)), [strings](https://en.wikipedia.org/wiki/Strings_(Unix)), [tail](https://en.wikipedia.org/wiki/Tail_(Unix)), [tr](https://en.wikipedia.org/wiki/Tr_(Unix)), [uniq](https://en.wikipedia.org/wiki/Uniq), [vi](https://en.wikipedia.org/wiki/Vi), [wc](https://en.wikipedia.org/wiki/Wc_(Unix)), [xargs](https://en.wikipedia.org/wiki/Xargs). |
| [**Shell builtins**](https://en.wikipedia.org/wiki/Shell_builtin) | * [alias](https://en.wikipedia.org/wiki/Alias_(command)), [cd](https://en.wikipedia.org/wiki/Cd_(command)), [echo](https://en.wikipedia.org/wiki/Echo_(command)), [test](https://en.wikipedia.org/wiki/Test_(Unix)), [unset](https://en.wikipedia.org/wiki/Environment_variable#unset_command), [wait](https://en.wikipedia.org/wiki/Wait_(command)). |
| **Searching** | * [find](https://en.wikipedia.org/wiki/Find_(Unix)), [grep](https://en.wikipedia.org/wiki/Grep). |
| **Documentation** | * [man](https://en.wikipedia.org/wiki/Man_page) |
| [**Software development**](https://en.wikipedia.org/wiki/Software_development) | * [ar](https://en.wikipedia.org/wiki/Ar_(Unix)), [ctags](https://en.wikipedia.org/wiki/Ctags), [lex](https://en.wikipedia.org/wiki/Lex_(software)), [make](https://en.wikipedia.org/wiki/Make_(software)), [nm](https://en.wikipedia.org/wiki/Nm_(Unix)), [strip](https://en.wikipedia.org/wiki/Strip_(Unix)), [yacc](https://en.wikipedia.org/wiki/Yacc), |
| **Miscellaneous** | * [bc](https://en.wikipedia.org/wiki/Bc_(programming_language)), [cal](https://en.wikipedia.org/wiki/Cal_(Unix)), [expr](https://en.wikipedia.org/wiki/Expr), [lp](https://en.wikipedia.org/wiki/System_V_printing_system), [od](https://en.wikipedia.org/wiki/Od_(Unix)), [sleep](https://en.wikipedia.org/wiki/Sleep_(Unix)), [true and false](https://en.wikipedia.org/wiki/True_and_false_(commands)) |

|  |  |  |
| --- | --- | --- |
| **Name** | **Category** | **Description** |
| [qalter](https://en.wikipedia.org/w/index.php?title=Qalter&action=edit&redlink=1) | Batch utilities | Alter batch job |
| [qdel](https://en.wikipedia.org/w/index.php?title=Qdel&action=edit&redlink=1) | Batch utilities | Delete batch jobs |
| [qhold](https://en.wikipedia.org/w/index.php?title=Qhold&action=edit&redlink=1) | Batch utilities | Hold batch jobs |
| [qmove](https://en.wikipedia.org/w/index.php?title=Qmove&action=edit&redlink=1) | Batch utilities | Move batch jobs |
| [qmsg](https://en.wikipedia.org/w/index.php?title=Qmsg&action=edit&redlink=1) | Batch utilities | Send message to batch jobs |
| [qrerun](https://en.wikipedia.org/w/index.php?title=Qrerun&action=edit&redlink=1) | Batch utilities | Rerun batch jobs |
| [qrls](https://en.wikipedia.org/w/index.php?title=Qrls&action=edit&redlink=1) | Batch utilities | Release batch jobs |
| [qselect](https://en.wikipedia.org/w/index.php?title=Qselect&action=edit&redlink=1) | Batch utilities | Select batch jobs |
| [qsig](https://en.wikipedia.org/w/index.php?title=Qsig&action=edit&redlink=1) | Batch utilities | Signal batch jobs |
| [qstat](https://en.wikipedia.org/w/index.php?title=Qstat_(Unix)&action=edit&redlink=1) | Batch utilities | Show status of batch jobs |
| [qsub](https://en.wikipedia.org/wiki/Qsub) | Batch utilities | Submit a script |
| cc/c99 | C programming | Compile standard C programs |
| [cflow](https://en.wikipedia.org/wiki/Cflow) | C programming | [Generate a C-language call graph](https://en.wikipedia.org/wiki/Call_graph) |
| [ctags](https://en.wikipedia.org/wiki/Ctags) | C programming | Create a tags file |
| [cxref](https://en.wikipedia.org/w/index.php?title=Cxref&action=edit&redlink=1) | C programming | [Generate a C-language program cross-reference table](https://en.wikipedia.org/wiki/C_(programming_language)) |
| [lex](https://en.wikipedia.org/wiki/Lex_programming_tool) | C programming | [Generate programs for lexical tasks](https://en.wikipedia.org/wiki/Lexical_analyzer) |
| [nm](https://en.wikipedia.org/wiki/Nm_(Unix)) | C programming | [Write the name list of an object file](https://en.wikipedia.org/wiki/Object_file) |
| [strings](https://en.wikipedia.org/wiki/Strings_(Unix)) | C programming | Find printable strings in files |
| [strip](https://en.wikipedia.org/wiki/Strip_(Unix)) | C programming | Remove unnecessary information from executable files |
| [yacc](https://en.wikipedia.org/wiki/Yacc) | C programming | [Yet another compiler compiler](https://en.wikipedia.org/wiki/Compiler) |
| [basename](https://en.wikipedia.org/wiki/Basename) | Filesystem | Return non-directory portion of a pathname; see also dirname |
| [cat](https://en.wikipedia.org/wiki/Cat_(Unix)) | Filesystem | Concatenate and print files |
| [cd](https://en.wikipedia.org/wiki/Cd_(command)) | Filesystem | Change the working directory |
| [chgrp](https://en.wikipedia.org/wiki/Chgrp) | Filesystem | Change the file group ownership |
| [chmod](https://en.wikipedia.org/wiki/Chmod) | Filesystem | Change the file modes/attributes/permissions |
| [chown](https://en.wikipedia.org/wiki/Chown) | Filesystem | Change the file ownership |
| [cksum](https://en.wikipedia.org/wiki/Cksum) | Filesystem | [Write file checksums and sizes](https://en.wikipedia.org/wiki/Checksum) |
| [cmp](https://en.wikipedia.org/wiki/Cmp_(Unix)) | Filesystem | Compare two files; see also diff |
| [compress](https://en.wikipedia.org/wiki/Compress) | Filesystem | Compress data |
| [cp](https://en.wikipedia.org/wiki/Cp_(Unix)) | Filesystem | Copy files |
| [dd](https://en.wikipedia.org/wiki/Dd_(Unix)) | Filesystem | Convert and copy a file |
| [df](https://en.wikipedia.org/wiki/Df_(Unix)) | Filesystem | Report free disk space |
| [dirname](https://en.wikipedia.org/wiki/Dirname) | Filesystem | Return the directory portion of a pathname; see also basename |
| [du](https://en.wikipedia.org/wiki/Du_(Unix)) | Filesystem | Estimate file space usage |
| [file](https://en.wikipedia.org/wiki/File_(command)) | Filesystem | Determine file type |
| [find](https://en.wikipedia.org/wiki/Find_(Unix)) | Filesystem | Find files |
| [link](https://en.wikipedia.org/wiki/Link_(Unix)) | Filesystem | Create a hard link to a file |
| [ln](https://en.wikipedia.org/wiki/Ln_(Unix)) | Filesystem | Link files |
| [ls](https://en.wikipedia.org/wiki/Ls) | Filesystem | List directory contents |
| [mkdir](https://en.wikipedia.org/wiki/Mkdir) | Filesystem | Make directories |
| [mkfifo](https://en.wikipedia.org/wiki/Mkfifo) | Filesystem | [Make FIFO special files](https://en.wikipedia.org/wiki/FIFO_(computing_and_electronics)) |
| [mv](https://en.wikipedia.org/wiki/Mv) | Filesystem | Move or rename files |
| [pathchk](https://en.wikipedia.org/w/index.php?title=Pathchk&action=edit&redlink=1) | Filesystem | Check pathnames |
| [pwd](https://en.wikipedia.org/wiki/Pwd) | Filesystem | print working directory - Return working directory name |
| [rm](https://en.wikipedia.org/wiki/Rm_(Unix)) | Filesystem | Remove directory entries |
| [rmdir](https://en.wikipedia.org/wiki/Rmdir) | Filesystem | Remove directories, if they are empty. |
| [touch](https://en.wikipedia.org/wiki/Touch_(Unix)) | Filesystem | Change file access and modification times |
| [unlink](https://en.wikipedia.org/wiki/Unlink_(Unix)) | Filesystem | Call the unlink function |
| [fort77](https://en.wikipedia.org/wiki/Fort77) | FORTRAN77 | [FORTRAN compiler](https://en.wikipedia.org/wiki/FORTRAN) |
| [alias](https://en.wikipedia.org/wiki/Alias_(command)) | Misc | Define or display aliases |
| [ar](https://en.wikipedia.org/wiki/Ar_(Unix)) | Misc | [Create and maintain library archives](https://en.wikipedia.org/wiki/Library_(computing)) |
| [bc](https://en.wikipedia.org/wiki/Bc_(programming_language)) | Misc | [Arbitrary-precision arithmetic language](https://en.wikipedia.org/wiki/Arbitrary-precision_arithmetic) |
| [cal](https://en.wikipedia.org/wiki/Cal_(Unix)) | Misc | Print a calendar |
| [crontab](https://en.wikipedia.org/wiki/Crontab) | Misc | Schedule periodic background work |
| [date](https://en.wikipedia.org/wiki/Unix_time#Command_line) | Misc | Display the date and time |
| [env](https://en.wikipedia.org/wiki/Env_(shell)) | Misc | Set the environment for command invocation |
| [fc](https://en.wikipedia.org/wiki/Fc_(Unix)) | Misc | Process the command history list |
| [gencat](https://en.wikipedia.org/w/index.php?title=Gencat&action=edit&redlink=1) | Misc | Generate a formatted message catalog |
| [getconf](https://en.wikipedia.org/w/index.php?title=Getconf&action=edit&redlink=1) | Misc | Get configuration values |
| [grep](https://en.wikipedia.org/wiki/Grep) | Misc | Search text for a pattern |
| [hash](https://en.wikipedia.org/wiki/Hash_(Unix)) | Misc | hash database access method |
| [id](https://en.wikipedia.org/wiki/Id_(Unix)) | Misc | Return user identity |
| [ipcrm](https://en.wikipedia.org/wiki/Ipcrm) | Misc | Remove a message queue, semaphore set, or shared memory segment identifier |
| [ipcs](https://en.wikipedia.org/wiki/Ipcs) | Misc | Report interprocess communication facilities status |
| [locale](https://en.wikipedia.org/w/index.php?title=Locale_(Unix)&action=edit&redlink=1) | Misc | Get locale-specific information |
| [localedef](https://en.wikipedia.org/w/index.php?title=Localedef&action=edit&redlink=1) | Misc | Define locale environment |
| [logname](https://en.wikipedia.org/wiki/Logname) | Misc | Return the user's login name |
| [m4](https://en.wikipedia.org/wiki/M4_(computer_language)) | Misc | Macro processor |
| [mailx](https://en.wikipedia.org/wiki/Mailx) | Misc | Process messages |
| [man](https://en.wikipedia.org/wiki/Man_page) | Misc | Display system documentation |
| [mesg](https://en.wikipedia.org/wiki/Mesg) | Misc | Permit or deny messages |
| [newgrp](https://en.wikipedia.org/w/index.php?title=Newgrp&action=edit&redlink=1) | Misc | [Change to a new group (functionaliy similar to sg[1])](https://en.wikipedia.org/wiki/List_of_Unix_commands#cite_note-1) |
| [od](https://en.wikipedia.org/wiki/Od_(Unix)) | Misc | Dump files in various formats |
| [pax](https://en.wikipedia.org/wiki/Pax_(Unix)) | Misc | Portable archive interchange |
| [split](https://en.wikipedia.org/wiki/Split_(Unix)) | Misc | Split files into pieces |
| stty | Misc | Set the options for a terminal |
| [tabs](https://en.wikipedia.org/w/index.php?title=Tabs_(Unix)&action=edit&redlink=1) | Misc | Set terminal tabs |
| [talk](https://en.wikipedia.org/wiki/Talk_(software)) | Misc | Talk to another user |
| [tput](https://en.wikipedia.org/wiki/Tput) | Misc | [Change terminal characteristics](https://en.wikipedia.org/wiki/Computer_terminal) |
| [tty](https://en.wikipedia.org/wiki/Tty_(unix)) | Misc | [Return user's terminal name](https://en.wikipedia.org/wiki/Computer_terminal) |
| [type](https://en.wikipedia.org/wiki/Type_(Unix)) | Misc | Displays how a name would be interpreted if used as a command |
| [ulimit](https://en.wikipedia.org/w/index.php?title=Ulimit&action=edit&redlink=1) | Misc | Set or report file size limit |
| [umask](https://en.wikipedia.org/wiki/Umask) | Misc | Get or set the file mode creation mask |
| [unalias](https://en.wikipedia.org/wiki/Unalias) | Misc | Remove alias definitions |
| [uname](https://en.wikipedia.org/wiki/Uname) | Misc | Return system name |
| [uncompress](https://en.wikipedia.org/wiki/Uncompress) | Misc | Expand compressed data |
| [write](https://en.wikipedia.org/wiki/Write_(Unix)) | Misc | Write to another user's terminal |
| [uucp](https://en.wikipedia.org/wiki/Uucp) | Network | System-to-system copy |
| [uudecode](https://en.wikipedia.org/wiki/Uudecode) | Network | Decode a binary file |
| [uuencode](https://en.wikipedia.org/wiki/Uuencode) | Network | Encode a binary file |
| [uustat](https://en.wikipedia.org/wiki/Uustat) | Network | [uucp status inquiry and job control](https://en.wikipedia.org/wiki/Uucp) |
| [at](https://en.wikipedia.org/wiki/At_(Unix)) | Process management | Execute commands at a later time |
| [batch](https://en.wikipedia.org/wiki/Batch_(Unix)) | Process management | Schedule commands to be executed in a batch queue |
| [bg](https://en.wikipedia.org/wiki/Bg_(Unix)) | Process management | Run jobs in the background |
| [fg](https://en.wikipedia.org/wiki/Fg_(Unix)) | Process management | Run jobs in the foreground |
| [fuser](https://en.wikipedia.org/wiki/Fuser_(Unix)) | Process management | [List process IDs of all processes that have one or more files open](https://en.wikipedia.org/wiki/Process_identifier) |
| [jobs](https://en.wikipedia.org/w/index.php?title=Jobs_(Unix)&action=edit&redlink=1) | Process management | Display status of jobs in the current session |
| [kill](https://en.wikipedia.org/wiki/Kill_(Unix)) | Process management | Terminate or signal processes |
| [nice](https://en.wikipedia.org/wiki/Nice_(Unix)) | Process management | Invoke a utility with an altered nice value |
| [nohup](https://en.wikipedia.org/wiki/Nohup) | Process management | [Invoke a utility immune to hangups](https://en.wikipedia.org/wiki/SIGHUP) |
| [ps](https://en.wikipedia.org/wiki/Ps_(Unix)) | Process management | Report process status |
| [renice](https://en.wikipedia.org/wiki/Renice) | Process management | [Set nice values of running processes](https://en.wikipedia.org/w/index.php?title=Nice_values&action=edit&redlink=1) |
| [time](https://en.wikipedia.org/wiki/Time_(Unix)) | Process management | Time a simple command |
| [uux](https://en.wikipedia.org/w/index.php?title=Uux_(Unix)&action=edit&redlink=1) | Process management | Remote command execution |
| [wait](https://en.wikipedia.org/wiki/Wait_(command)) | Process management | Await process completion |
| [make](https://en.wikipedia.org/wiki/Make_(software)) | Programming | Maintain, update, and regenerate groups of programs |
| [admin](https://en.wikipedia.org/w/index.php?title=Admin_(Unix)&action=edit&redlink=1) | [SCCS](https://en.wikipedia.org/wiki/Source_Code_Control_System) | [Create and administer SCCS files](https://en.wikipedia.org/wiki/Source_Code_Control_System) |
| [delta](https://en.wikipedia.org/w/index.php?title=Delta_(Unix)&action=edit&redlink=1) | SCCS | Make a delta (change) to an SCCS file |
| [get](https://en.wikipedia.org/w/index.php?title=Get_(Unix)&action=edit&redlink=1) | SCCS | Get a version of an SCCS file |
| [prs](https://en.wikipedia.org/w/index.php?title=Prs&action=edit&redlink=1) | SCCS | Print an SCCS file |
| [rmdel](https://en.wikipedia.org/w/index.php?title=Rmdel&action=edit&redlink=1) | SCCS | Remove a delta from an SCCS file |
| [sact](https://en.wikipedia.org/w/index.php?title=Sact_(Unix)&action=edit&redlink=1) | SCCS | Print current SCCS file-editing activity |
| [sccs](https://en.wikipedia.org/wiki/Source_Code_Control_System) | [SCCS](https://en.wikipedia.org/wiki/Source_Code_Control_System) | Front end for the SCCS subsystem |
| [unget](https://en.wikipedia.org/w/index.php?title=Unget&action=edit&redlink=1) | SCCS | Undo a previous get of an SCCS file |
| [val](https://en.wikipedia.org/w/index.php?title=Val_(Unix)&action=edit&redlink=1) | SCCS | Validate SCCS files |
| what | [SCCS](https://en.wikipedia.org/wiki/Source_Code_Control_System) | Identify SCCS files |
| [command](https://en.wikipedia.org/w/index.php?title=Command_(Unix)&action=edit&redlink=1) | Shell programming | Execute a simple command |
| [echo](https://en.wikipedia.org/wiki/Echo_(command)) | Shell programming | Write arguments to standard output |
| [expr](https://en.wikipedia.org/wiki/Expr) | Shell programming | Evaluate arguments as an expression |
| [FALSE](https://en.wikipedia.org/wiki/False_(Unix)) | Shell programming | Return false value |
| [getopts](https://en.wikipedia.org/wiki/Getopts) | Shell programming | Parse utility options |
| [logger](https://en.wikipedia.org/w/index.php?title=Logger_(Unix)&action=edit&redlink=1) | Shell programming | Log messages |
| [printf](https://en.wikipedia.org/wiki/Printf_(Unix)) | Shell programming | Write formatted output |
| [read](https://en.wikipedia.org/wiki/Read_(Unix)) | Shell programming | Read a line from standard input |
| [sh](https://en.wikipedia.org/wiki/Bourne_shell) | Shell programming | [Shell, the standard command language interpreter](https://en.wikipedia.org/wiki/Unix_shell) |
| [sleep](https://en.wikipedia.org/wiki/Sleep_(Unix)) | Shell programming | Suspend execution for an interval |
| [tee](https://en.wikipedia.org/wiki/Tee_(command)) | Shell programming | [Duplicate the standard output](https://en.wikipedia.org/wiki/Standard_streams) |
| [test](https://en.wikipedia.org/wiki/Test_(Unix)) | Shell programming | [Evaluate expression](https://en.wikipedia.org/wiki/Expression_(computer_science)) |
| [TRUE](https://en.wikipedia.org/wiki/True_(Unix)) | Shell programming | Return true value |
| [xargs](https://en.wikipedia.org/wiki/Xargs) | Shell programming | Construct argument lists and invoke utility |
| [who](https://en.wikipedia.org/wiki/Who_(Unix)) | System administration | Display who is on the system |
| [asa](https://en.wikipedia.org/w/index.php?title=Asa_(Unix)&action=edit&redlink=1) | Text processing | Interpret carriage-control characters |
| [awk](https://en.wikipedia.org/wiki/AWK) | Text processing | Pattern scanning and processing language |
| [comm](https://en.wikipedia.org/wiki/Comm) | Text processing | Select or reject lines common to two files |
| [csplit](https://en.wikipedia.org/wiki/Csplit) | Text processing | Split files based on context |
| [cut](https://en.wikipedia.org/wiki/Cut_(Unix)) | Text processing | Cut out selected fields of each line of a file |
| [diff](https://en.wikipedia.org/wiki/Diff) | Text processing | Compare two files; see also cmp |
| [ed](https://en.wikipedia.org/wiki/Ed_(text_editor)) | Text processing | The standard text editor |
| [ex](https://en.wikipedia.org/wiki/Ex_(editor)) | Text processing | Text editor |
| [expand](https://en.wikipedia.org/wiki/Expand_(Unix)) | Text processing | Convert tabs to spaces |
| [fold](https://en.wikipedia.org/wiki/Fold_(Unix)) | Text processing | Filter for folding lines |
| [head](https://en.wikipedia.org/wiki/Head_(Unix)) | Text processing | Copy the first part of files |
| [iconv](https://en.wikipedia.org/wiki/Iconv) | Text processing | Codeset conversion |
| [join](https://en.wikipedia.org/wiki/Join_(Unix)) | Text processing | Merges two sorted text files based on the presence of a common field |
| [lp](https://en.wikipedia.org/wiki/Lp_(Unix)) | Text processing | Send files to a printer |
| [more](https://en.wikipedia.org/wiki/More_(command)) | Text processing | Display files on a page-by-page basis |
| [nl](https://en.wikipedia.org/wiki/Nl_(Unix)) | Text processing | Line numbering filter |
| [paste](https://en.wikipedia.org/wiki/Paste_(Unix)) | Text processing | Merge corresponding or subsequent lines of files |
| [patch](https://en.wikipedia.org/wiki/Patch_(Unix)) | Text processing | Apply changes to files |
| [pr](https://en.wikipedia.org/wiki/Pr_(Unix)) | Text processing | Print files |
| [sed](https://en.wikipedia.org/wiki/Sed) | Text processing | Stream editor |
| [sort](https://en.wikipedia.org/wiki/Sort_(Unix)) | Text processing | Sort, merge, or sequence check text files |
| [tail](https://en.wikipedia.org/wiki/Tail_(Unix)) | Text processing | Copy the last part of a file |
| [tr](https://en.wikipedia.org/wiki/Tr_(Unix)) | Text processing | Translate characters |
| [tsort](https://en.wikipedia.org/wiki/Tsort_(Unix)) | Text processing | Topological sort |
| [unexpand](https://en.wikipedia.org/wiki/Unexpand) | Text processing | Convert spaces to tabs |
| [uniq](https://en.wikipedia.org/wiki/Uniq) | Text processing | Report or filter out repeated lines in a file |
| [vi](https://en.wikipedia.org/wiki/Vi) | Text processing | Screen-oriented (visual) display editor |
| [wc](https://en.wikipedia.org/wiki/Wc_(Unix)) | Text processing | Line, word and byte or character count |
| [zcat](https://en.wikipedia.org/wiki/Zcat) | Text processing | Expand and concatenate data |