Exp-1	Basic UNIX Commands

Aim: To Practice a Basic UNIX Commands using Terminal.

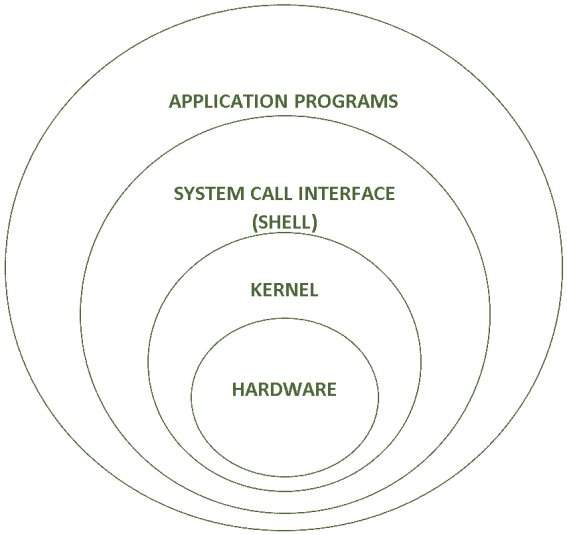
Description:

UNIX: It is a multi-user operating system. Developed at AT&T Bell Industries, USA in 1969. Ken Thomson along with Dennis Ritchie developed it from MULTICS(Multiplexed Information and Computing Service) OS. By1980, UNIX had been completely rewritten using C language.

LINUX :It is similar to UNIX, which is created by Linus Torualds. All UNIX commands works in Linux. Linux is a open source software. The main feature of Linux is coexisting with other OS such as windows and UNIX.

STRUCTUREOFALINUXSYSTEM: It consists of three parts.

a) UNIX Kernel b)Shells c)Tools and Applications



UNIX Architecture

UNIXKERNEL: Kernel is the core of the UNIX OS.It controls all tasks, schedule all Processes and carries out all the functions of OS. Decides when one program stops and another starts.

SHELL: Shell is the command interpreter in the UNIX OS. It accepts command from the user and analyses and interprets them

Basic Unix Commands:

Result:

Exp-2	Programs for UNIX operating systemcalls

AIM: To write programs using following UNIX operating systemcalls Fork, exec, getpid,exit,wait, close, stst,opendir and readdir.

Description:

The interface between a process and an operating system is provided by system calls. In general, system calls are available as assembly language instructions. They are also included in the manuals used by the assembly level programmers.

System calls in Unix are used for file system control, process control, interprocess communication etc. Access to the Unix kernel is only available through these system calls. Generally, system calls are similar to function calls, the only difference is that they remove the control from the user process.

System Calls:

Result:

Exp-3	Simulation of UNIX commands like cp, ls, grep, etc

Aim: To Simulate the following UNIX commands like cp, ls, grep,etc.

Description: UNIX commands can be simulated in the high level language using Unix system calls and APIs available in the language. In addition, programming language construct can also be used to avail the UNIX commands.

Algorithm:

Step 1: Include necessary header files for manipulating directory.
Step 2: Declare and initialize required objects.
Step 3: Read the directory name form the user.
Step 4: Open the directory using opendir() system call and report error if the directory is not available.
Step 5: Read the entry available in the directory.
Step 6: Display the directory entry ie., name of the file or sub directory.
Step 7: Repeat the step 6 and 7 until all the entries were read.

Simulation Programs on cp,ls,grep:

Result:

Exp-4	Simulate the following CPU scheduling algorithms a)RoundRobin b)SJF c)FCFS d)Priority

Aim: To Simulate the following CPU scheduling algorithms

a) RoundRobin b)SJF c)FCFS d)Priority

A) Write a program to implement Round Robin CPU scheduling algorithm.

Algorithm:

Step 1:Inside the structure declare the variables.
Step 2: Declare the variable i,j as integer, totwtime and tottime is equal to zero.
Step 3: Get the value of,„n‟‟assign p and allocate the memory.
Step4:Inside the for loop get the value of burst time and priority and read the time quantum.
Step 5:Assign wtime as zero.
Step 6: Check p[i].pri is greater than p[j].pri .
Step 7: Calculate the total of burst time and waiting time and assign as turn around time.
Step 8: Stop the program.

Source Code:

Input and Output:

Result:

(B) Write a program to implement SJF CPU scheduling algorithm.

Algorithm:

Step 1:Inside the structure declare the variables.
Step 2: Declare the variable i,j as integer,totwtime and totttime is equal to zero.
Step 3: Get thevalue of„n‟‟assign pid asI andgetthe value ofp[i].btime.
Step 4:Assign p[0]wtime as zero and tottime as btime and inside theloop calculate waittime and turnaroundtime.
Step 5: Calculate total waittime and total turnaround time by dividing bytotal number of process.
Step 6: Print total waittime and total turnaroundtime.
Step 7: Stop the program.

Source Code:

Input and Output:

Result:

(C)Write a program to implement FCFS CPU scheduling algorithm

Algorithm:

Step 1: Inside the structure declare the variables.
Step 2: Declare the variable i,j as integer,totwtime and totttime is equal to zero.
Step 3: Get the value of „n" assign pid as I and get the value of p[i].btime.
Step 4: Assign p[0] wtime as zero and tot time as btime and inside the loop calculate wait time and turnaround time.
Step 5: Calculate total wait time and total turnaround time by dividing by total number of process.
Step 6: Print total wait time and total turnaround time.
Step 7: Stop the program.

Source Code:

Input and Output:

Result:

(d)Write a program to implement Priority CPU scheduling algorithm

Algorithm:

Step 1: Inside the structure declare the variables.
Step 2: Declare the variable i,j as integer,totwtime and totttime is equal to zero.
Step 3: Get the value of „n" assign pid as I and get the value of p[i].btime.
Step 4: Assign p[0] wtime as zero and tot time as btime and inside the loop calculate wait time and turnaround time.
Step 5: Calculate total wait time and total turnaround time by dividing by total number of process.
Step 6: Print total wait time and total turnaround time.
Step 7: Stop the program.

Source Code:

Input and Output:

Result:

Exp-5	Dynamic priority scheduling algorithm.

Aim: To implement the dynamic priority scheduling algorithm.

Algorithm:

Step 1:Inside the structure declare the variables.
Step 2: Declare the variable i,j as integer,totwtime and totttime is equal to zero.
Step 3: Get the value of,„n‟‟assign pid as I and get the value of p[i].btime.
Step 4:Assign p[0]wtime as zero and tottime as btime and inside the loop calculate wait time and turnaround time.
Step 5: Calculate total wait time and total turnaround time by dividing by total number of process.
Step 6: Print total waittime and total turnaround time.
Step 7: Stop the program.

Source Code:

Input and Output:

Result:

Exp-6	Round Robin Scheduling Algorithm.

Aim: To simulate the Round Robin scheduling algorithm is to assume that there are five jobs with different weights ranging from 1to5. Implement round robin algorithm with time slice equivalent to weight.

Algorithm:

Step 1:Inside the structure declare the variables.
Step 2: Declare the variable i,j as integer, totwtime and tottime is equal to zero.
Step 3: Get the value of,„n‟assign p and allocate the memory.
Step4:Inside the for loop get the value of burst time and priority and read the time quantum.
Step 5:Assign wtime as zero.
Step 6: Check p[i].pri is greater than p[j].pri .
Step 7: Calculate the total of burst time and waitingtime and assignas turnaroundtime.
Step 8: Stop the program.

Source Code:

Input and Output:

Result:

Exp-7	Priority Scheduling Algorithm.

Aim: To implement priority scheduling algorithm. While executing, no process should wait for more than10seconds.If waiting time is more than 10seconds,that process has to be executed foratleast1second before waiting again.

Algorithm:

Step 1:Inside the structure declare the variables.
Step 2: Declare the variable i,j as integer, totwtime and totttime is equal to zero.
Step 3: Get thevalue of„n"assign pid asI and get the value ofp[i].btime.
Step 4:Assign p[0]wtime as zero and tottime asbtime and inside the loop calculate waittime and turnaroundtime.
Step 5: Calculate total waittime and total turnaround time by dividing by total number of process.
Step 6: Print total waittime and total turnaroundtime.
Step 7: Stop the program

Source Code:

Input and Output:

Result:

Exp-8	Semaphore and Monitors

Aim: Control the number of ports opened by the operating system with

a) Semaphore b)Monitors.

a) Semaphore Algorithm:

Step 1: Start the program.
Step 2: Use wait() and signal()
Step 3: Find the ports opened
Step 4: Display the ports
Step 5: Stop the program

SourceCode:

Input and output

Result:

b)Monitors Algorithm:

Step 1: Start the program.
Step 2: Start the monitor
Step 3: Find the ports opened
Step 4: Display the ports
Step 5: Stop the program

SourceCode:

Input and output

Result:

Exp-9	Shared Memory

Aim: To simulate how parent and child processes use sharedmemory and addressspace.

Algorithm:

Step 1: Start the program.
Step 2: Create a parent process
Step 3: Create a child process
Step 4: Let parent process share memory
Step 5: Stop the program

Source Code:

Input and output:

Result:

Exp-10	Sleeping barber problem

Aim: To simulate sleeping barberproblem.

Algorithm:

Step 1: Start the program.
Step 2: Use semaphore and mutex
Step 3: If there are customers barber is busy cutting the hair
Step 4: Else the barber is in sleeping mode
Step 5: Stop the program

Source Code:

Input and output

Result:

Exp-11	Dining Philosopher's problem

Aim: To simulate dining philosopher's problem.

Algorithm:

Step 1: Start the program.
Step 2: Use semaphore and monitor.
Step 3: If all the philosophers are hungry then display deadlock.
Step 4: If both of the chopsticks are free the philosopher can start eating.
Step 5:Otherwise display the philosopher is in thinking state.
Step 6: stop the program.

Source Code:

Input and Output

Result:

Exp-12	Producer and Consumer problem

Aim: To simulate producer and consumer problem using threads.

Algorithm:

Step 1: Start the program.
Step 2: Create a producer as parent

Step 3: Create consumer as child

Step 4: If producer produces a thread then display thread is created.

Step 5:If consumer uses the thread produced display thread is consumed

Step 6: Stop the program

Source Code:

Input and Output:

Result:

Exp-13	**Memory Allocation Methods** **a) First fit b) Worst fit c) Best fit**

Aim: To implement the following memory allocation methods for f ixed partition
a)Firstfit b)Worstfit c)Bestfit

A) Write a program to implement First fit memory allocation method.

Source Code:

Input and Output:

Result:

B) Write a program to implement Worst fit memory allocation method.

Source Code:

Input and Output:

Result:

C) Write a program to implement Best fit memory allocation method

Source Code:

Input and Output

Result:

Exp-14	Page Replacement Algorithms

Aim: To simulate the following page replacement algorithms

a) FIFO b)LRU c)LFU etc.,

a) Write a program to implement LRU page replacement algorithm.

Algorithm:

Step 1: Create a queue to hold all pages in memory

Step 2: When the page is required replace the page at the head of the queue

Step 3: Now the new page is inserted at the tail of the queue

Step 4: When the page fault occurs replace page present at the head of the queue

Step 5: Stop the program

Source Code:

Input and Output:

Result:

b. Write a program to implement LRU page replacement algorithm.

Algorithm:

Step 1: Create a queue to hold all pages in memory

Step 2: When the page is required replace the page at the head of the queue

Step 3: Now the new page is inserted at the tail of the queue

Step 4: Create a stack

Step 5: When the page fault occurs replace page present at the bottom of the stack

Source Code:

Input and Output:

Result:

c. Write a program to implement LFU page replacement algorithm

Algorithm:

Step 1: Create a queue to hold all pages in memory

Step 2: When the page is required replace the page at the head of the queue

Step 3: Now the new page is inserted at the tail of the queue

Step 4: Create a stack

Step 5: When the page fault occurs replace page present at the bottom of the stack

Source Code:

Input and Output:

Result:

Exp-15	Paging Technique of Memory Management

Aim: To simulate Paging Technique of memorymanagement

Algorithm:

Step 1: Read all the necessary input from the keyboard.

Step 2: Pages - Logical memory is broken into fixed - sized blocks.

Step 3: Frames – Physical memory is broken into fixed – sized blocks.

Step 4: Calculate the physical address using the following

Physical address = (Frame number * Frame size) + offset

Step 5: Display the physical address.

Step 6: Stop the process.

Source Code:

Input and Output:

Result:

Exp-16	Bankers Algorithm for Dead Lock avoidance and prevention

Aim: To simulate Bankers Algorithm for DeadLock avoidance and prevention

Algorithm:

Step-1: Start the program.

Step- 2: Get the values of resources and processes.

Step-3: Get the avail value.

Step-4: After allocation find the need value.

Step-5: Check whether its possible to allocate.

Step-6: If it is possible then the system is in safe state.

Step-7: Else system is not in safety state.

Step-8: If the new request comes then check that the system is in safety or not if we allow the request.

Step-9: stop the program.

Source Code:

Input and Output:

Result:

Exp-17	File Allocation Strategies a) Sequential b)Indexed c)Linked

Aim: To simulate the following file allocation strategies

b) Sequential b)Indexed c)Linked

a. Write a program to simulate Sequential file allocation Strategy

Algorithm:

Step 1: Start the program.

Step 2: Get the number of memory partition and their sizes.

Step 3: Get the number of processes and values of block size for each process.

Step 4: First fit algorithm searches the entire memory block until a hole which is big enough is encountered. It allocates that memory block for the requesting process.

Step 5: Best-fit algorithm searches the memory blocks for the smallest hole which can be allocated to requesting process and allocates if.

Step 6: Worst fit algorithm searches the memory blocks for the largest hole and allocates it to the process.

Step 7: Analyze all the three memory management techniques and display the best algorithm which utilizes the memory resources effectively and efficiently.

Step 8: Stop the program.

Source Code:

Input and Output:

Result:

b. Write a program to simulate Linked file allocation Strategy

Algorithm:

Step 1: Create a queue to hold all pages in memory

Step 2: When the page is required replace the page at the head of the queue

Step 3: Now the new page is inserted at the tail of the queue

Step 4: Create a stack

Step 5: When the page fault occurs replace page present at the bottom of the stack

Step 6: Stop the allocation.

Source Code:

Input and Output:

Result:

c. Write a program to simulate Indexed file allocation Strategy

Algorithm:

Step 1: Start.

Step 2: Let n be the size of the buffer

Step 3: check if there are any producer

Step 4: if yes check whether the buffer is full

Step 5: If not the producer item is stored in the buffer

Step 6: If the buffer is full the producer has to wait

Step 7: Check there is any consumer. If yes check whether the buffer is empty

Step 8: If no, the consumer consumes them from the buffer

Step 9: If the buffer is empty, the consumer has to wait.

Step 10: Repeat checking for the producer and consumer till required

Step 11: Terminate the process.

Source Code:

Input and Output:

Result:

Exp-18	**File Organization Techniques** **a) Single level directory b) Two level c) Hierarchical d) DAG**

Aim: To simulate all File OrganizationTechniques

a)Single level directory b)Two level c)Hierarchical d)DAG

a. Write a program to implement Single level directory File Organization Technique

Algorithm:

Step 1: Start the program.

Step 2: Get the name of the directory.

Step 3: get the number of files.

Step 4: Get the name of the each file.

Step 5: Now each file is in the form of filled circle

Step 6: Every file is connected with the given directory

Step 7: Display the connected graph along with name using graphics

Step 8: Stop the program.

Source Code:

Input and Output:

Result:

b. Write a program to implement Two level directory File Organization Technique

Algorithm:

Step 1: Start the program.

Step 2: Get the name of the directories.

Step 3: get the number of files.

Step 4: Get the name of the each file.

Step 5: Now each file is in the form of fill circle

Step 6: Every file is connected with respective directory

Step 7: Display the connected graph along with name using graphics

Step 8: Stop the program.

Source Code:

Input and Output:

Result:

c. Write a program to implement Hierarchical level directory File Organization Technique

Algorithm:

Step 1: Start the program.

Step 2: Get the name of the directories.

Step 3: get the number of files.

Step 4: Get the name of the each file.

Step 5: Now each file is in the form of fill circle

Step 6: Every file is connected with respective directory

Step 7: Display the connected graph along with name in hierarchical way using graphics

Step 8: Stop the program.

Source Code:

Input and Output:

Result:

d. Write a program to implement DAG File Organization Technique

Algorithm:

Step 1: Start the program.

Step 2: Get the name of the directories.

Step 3: get the number of files.

Step 4: Get the name of the each file.

Step 5: Now each file is in the form of fill circle

Step 6: Every file is connected with respective directory

Step 7: Display the connected graph along with name using graphics

Step 8: Stop the program.

Source Code:

Input and Output:

Result:

Viva Questions

1. Define Operating Systems and discuss its role from different perspectives.
2. Explain fundamental difference between i) N/w OS and distributed OS ii) web basedand embedded computing.
3. What do you mean by cooperating process? Describe its four advantages.
4. What are the different categories of system programs? Explain.
5. List out different services of Operating Systems and explain each service.
6. Explain the concept of virtual machines. Bring out its advantages.
7. Distinguish among following terminologies i) Multiprogramming systems ii) Multitasking Systems iii) Multiprocessor systems.
8. What is distributed operating system? What are the advantages of distributed operating system?
9. What are system calls? Explain different categories of system calls with example?
10. What are the 3 main purposes of an Operating System?
11. Explain the concept of virtual machines.
12. Explain the distinguishing features ofi). Real time system ii) Multiprocessor system
13. What is the purpose of command interpreter? Why is it usually separate from theKernel?
14. What is an Operating System? Explain considering different possible views.
15. What is operating system? What are functions of operating system?
16. What are multiprocessor systems? Give advantages.
17. What is the main difficulty that a programmer must overcome in writing anoperating system for real time environment?
18. Define spooling and the need for it. Explain its working with necessary diagrams.
19. Explain the following terms and their working with diagrama) Buffering b) Spooling c) Time sharing d) Distributed system e) Real-time
20. Compare tightly coupled systems with loosely coupled systems.
21. Describe differences between symmetric and asymmetric multiprocessing. What arethree advantages and one disadvantage of multiprocessor systems?
22. Explain distinguished features ofi) Time Sharing System ii) Parallel Processing
23. Write a brief note on different operating system structures
24. Explain different sub components of an operating system.
25. Bring out the requirements ofi) Real time operating systems (ii) Distributed operating systems
26. Justify the statement “Operating System can be viewed as a government, resourceallocator and a control program”.