Experiment 1.a

Basics of Unix Commands.

Introduction to Unix

Aim: To study about the basics of UNIX

Unix: It is a multiuses operating system. Dereloped at AT & Bell industries, USA in 1969. Ken Thomson along with Dennis Ritchie dereloped it from MULTICS (Multiplexed Information and Computing Service) OS. By 1960, UNIX had been completely rewritten using C language.

Linux: It is similar to UNIX, which is created by Linus Torvalds. 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.

Structure of a Linux System

It consists of 3 parts: a) UNIX Kernel
b) Shells
c) Tools and Application

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

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

Experiment 1.b

Basic Unix Commands

Aim: To study of Basic Unix Commands and various UNIX editors such as vi, ed, ex and EMACS.

Content

Note: Syn → Syntax

a) date :- used to check the date and time.
Syn : $ date.

b) cal : used to display the calendar
Syn : $ cal 2 2009

c) echo - used to print the message on the screen.
Syn : $ echo "text"

d) ls - used to list the files. Your files are kept in a directory.
Syn : $ ls ls -s.

e) lp : used to take printouts.
Syn : $ lp filename.

f) man : used to provide manual help on every UNIX commands.
$ Syn : $ man unix command
$ man cat.

g) who & whoami : It displays data about all users who have logged into the system currently. The next command displays about current users only.
Syn: $who, $whoami

h) uptime : Tells you how long the computes has been running since its last reboot or powes off.
Syn: $uptime

i) uname: It displays the system information such as hardwase platform, system name and processor, OS types.
Syn: $uname -a

j) hostname : displays and set system host name
Syn: $hostname

k) bc: Stands for "best calculator".

File Manipulation Commands:

a) cat: this create, view and concentrate files

Creation
Syn: $cat>filename:

Viewing
Syn: $cat filename:
Add Text to an existing file: Syn $cat >> filename:

Concatenate
Syn: $cat file1 file2 > file3
$cat file1 file2 >> file3 (no ores working of file3)

b) grep: used to search a particulas word or pattern related to that word from the file. Syn: $grep search word filename:

c) rm - deletes a file from the file system
Syn: $rm filename:

d) touch - used to create a blank file
Syn: $touch file names

e) cp - copies the files or directories
Syn: $cpsource file destination file

g) cut - it cuts or pickup a given numbes of characters or fields of the file. Syn. $cut <option> <filename>.

h) head - displays 10 lines from the head (top) of a given file.
Syn: $head filename. To display the top 2 lines: Syn: $head -2 filename.

i) tail - displays last 10 lines of the file.
Syn: $tail filename.

j) chmod: - used to change the permissions of a file or directory.
Syn: $ch mod category operation permission file.
Where Category is the uses type.
Operation is used to assign or remove permission.
Permission is the type of permission
File: are used to assign or remove permission all.

k) wc: it counts the numbes of lines, words, characters in a specified files with the options as -l, -w, -c.

Syn: $wc -l filename
$wc -w filename
$wc -c filename

Experiment 1.c

Unix Editors

Aim: To study of various UNIX editors such as vi, ed, ex and EMACS

Concept: Editor is a program that allows users to see a portions a file on the screen and modify characters and lines by simply typing at the current position. UNIX supports variety of Editors. They are ed, ex, vi, EMACS

Vi - vi stands for "visual" vi is the most important and powerful editor, vi is a full screen editor that allows users to view and edit entire document at the same time. Vi editor was written in University of California at Berkley by Bill Joy, who is one of the co-founders of Sun Microsystems.

Features of Vi

It is easy to learn and has more powerful features. It works greats and is case sensitive. Vi has powerful fun undo functions and has 3 modes.

1. Command Mode 2. Insert Mode 3. Escape or ex mode

In command mode, no text is displayed on the screen.
In Insert mode, it permits users to edit insert or replace text
In escape mode, it displays commands at command line
Moving the cursor with the help of h, l, k, j, ft etc.

EMACS Editor

Motion Commands:

M-> Move to end of file.
M-< Move to beginning of file.
C-v Move forward a screen. M-v Move Backward a Screen
C-n Move to next line
C-p Move to previous line
C-a Move to the beginning of the line
C-e Move to the end of the line
C-f Move forward a character.
C-b Move backward a character
M-f Move forward a word
M-b Move backward a word.

Deletion Commands:

DEL	delete the previous character
C-d	delete the current character
M-DEL	delete the previous word.
M-d	delete the next word.
C-x DEL	delete the previous sentence
M-k	delete the rest of the current sentence
C-k	deletes the rest of the current line
C-xu	undo the lasted bit change.

Search and Replace in EMACS

y Change the occurrance of the pattern
n Don't change the occurrence, but look for the others
q Don't change. Leave Query replace immediately
! Change this occurrence and all others in the file

Experiment 2

Programs using the following System calls of Unix Operating System fork, exec, getpid, exit, wait, close, stat, opendir, readdir

Aim: To write C Programs using the following System calls of Unix operating System fork, exec, getpid, exit, wait, close, stat, opendir, readdir

1. Program for System Calls of Unix Operating Systems (Opendir, readdir, closedir)

Algorithm

1. Start the program
2. Create struct dirent
3. Declare the variable buff and pointer dptr.
4. Get the directory name
5. Open the directory
6. Read the contents in directory and print it
7. Close the directory.

2 Program for System Calls of Unix Operating System (fork, getpid, exit)

Algorithm

1. Start the program
2. Declare the variables pid, pid1, pid2.
3. Call fork() System call to create process
4. If pid == -1, exit.
5. If pid != -1, get the process id using getpid()
6. Print the process id
7. Stop the program

Experiment 3.

Aim: To write
C Programs to Simulate UNIX commands like cp, ls, grep

Aim: To write C programs to simulate Unix commands like cp, ls, grep.

1. Program for simulation of cp unix commands.

Algorithm

1. Start the program
2. Declare the variables ch, *fp, sc = 0.
3. Open the file in read mode.
4. Get the characters.
5. If ch == " " Then increment sc value by one.
6. Print no. of spaces.
7. Close the file.

2. Program for Simulation of ls unix commands.

Algorithm:

1. Start the program
2. Open the directory with
3. Read the directory content and print it.
4. Close the directory.

3.

3. Program for Simulation of Grep Unix Commands.

Algorithm

1. Start the program
2. Declare the variables line[max], count=0, occurrences=0 and pointers *fp, *newline.
3. Open the file in read mode.
4. In while loop check fgets(fline, max, fp) != NULL
5. Increment count value.
6. Check newline = strchr(fline, "\n")
7. Print the count, line value and increment the occurrence value.
8. Stop the program.

Experiment 4

Simple Shell Programs:

Aim: To write Simple Shell programs by using conditional, branching and looping statements

1. Write a Shell Program to check the given number is even or odd.

Algorithm:

1. Start the program
2. Read the value of n.
3. Calculate "r= expr $n%2"
4. If the value of r equals 0 then print the number is even.
5. If the value of r not equals to 0 then print the number is odd.

2. Write a shell program to check the given year is leap year or not.

Algorithm:

1. Start the program
2. Read the value of year
3. Calculate "b=expr $y % 4"
4. If the value of b equals 0, then print the year is leap year.
5. If the value of b not equals to 0, then print the year is not a leap year.

3. Write a Shell program to find the factorial of a number.

Algorithm:

1. Start the program
2. Read the value of n.
3. Calculate 'i = expr $n-1'
4. If the value of i is greater than 1, then calculate 'n = expr $n * $i' and 'i = expr $i-1'.
5. Print the factorial of the given number.

4. Write a Shell Program to Swap the 2 integers.

Algorithm:

1. Start the program.
2. Read the value of a, b.
3. Calculate the swapping of 2 values by using a temporary variable temp
4. Print the value of a and b.

Experiment 5

CPU Scheduling Algorithms

Priority

Aim: To write a C program for implementation of Priority Scheduling algorithms.

Algorithm:

1. Inside the structure declare the variables.
2. Declare the variable i,j as integers, totwtime, and totttime is equal to zero.
3. Get the value of 'n' assign p and allocate the memory.
4. Inside the for loop get the value of burst time and prority.
5. Assign wtime as zero.
6. Check p[i].pri is greater than p[j].pri
7. Calculate the total of burst time and waiting time and assign as turnaround time.

8. Stop the program

Experiment 5.c

FCFS

Aim: To write a C program for implementation of FCFS and SJF algorithm.

Algorithm:

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

Experiment 5.d

Aim: To write a C program for implementation of SJF scheduling algorithms.

Algorithms

1. Inside the Structure declare the variables.
2. Declare the variable i, j as integers, totwtime and totttime is equal to zero.
3. Get the value of 'n' assign pid as 1 and get the value of p[i] btime
4. Assign p[0] wtime as zero and tot time as btime and inside the loop calculate wait time and turn around time.
5. Calculate total wait time and total turnaround time by dividing by total numbes of process.
6. Print total wait time and total turnaround time
7. Stop the program

Experiment 6

Producers Consumers Problem Using Semaphores.

Aim: To write a C program to implement the produces-consumes problem using semaphores.

Algorithm:

1. Start the program
2. Declare the required variables.
3. Initialize the buffer size and get maximum item you want to produce.
4. Get the option, which you want to do either produces, consumes or exit from the operation.

5. If you select the produces, check the buffer size if it is full the produces should not produce the item or otherwise produce the item and increase the value buffer size.

6. If you select the consumes, check the buffer size if it is empty, the consumes should not consume the item or otherwise consume the item and decrease the value of buffer size.

7. If you select exit, come out of the program.
8. Stop the program