OS LAB MANUAL

Compiled by: Lokanth Regmi

BACHLOR OF COMPUTER ENGINEERING

SIXTH SEMESTER

IOE Pulchowk Campus

Ex.No : 1.a **UNIX COMMANDS**

AIM :

To study and excute the commands in unix.

COMMAND :

Date Command:

This command is used to display the current data and time.

Syntax:

$date

$date +%ch

Options: -

a = Abbreviated weekday.

A = Full weekday.

b = Abbreviated month.

B = Full month.

c = Current day and time.

C = Display the century as a decimal number.

d = Day of the month.

D = Day in „mm/dd/yy" format

h = Abbreviated month day.

H = Display the hour.

L = Day of the year.

m = Month of the year.

M = Minute.

P = Display AM or PMS = Seconds

T = HH:MM:SS formatu = Week of the year.

y = Display the year in 2 digit.

Y = Display the full year.

Z = Time zone .To change the format :

Syntax :

$date „+%H-%M-%S"

Calendar Command :

This command is used to display the calendar of the year or the particular month ofcalendar year.

Syntax:

a. $cal <year>
b. $cal <month> <year>

Here the first syntax gives the entire calendar for given year & the second Syntax givesthe calendar of reserved month of that year.

Echo Command:

This command is used to print the arguments on the screen .

Syntax : $echo <text>

Multi line echo command :

To have the output in the same line , the following commands can be used.

Syntax : $echo <text\>text

To have the output in different line, the following command can be used.

Syntax : $echo "text

>line2

>line3"

Banner Command :

It is used to display the arguments in „#" symbol .

Syntax : $banner <arguments>

'who' Command :

It is used to display who are the users connected to our computer currently.

Syntax : $who – option"s

Options : -

H–Display the output with headers.

b–Display the last booting date or time or when the system was lastely rebooted.

'who am i' Command :

Display the details of the current working directory.

Syntax : $who am i

'tty' Command :

It will display the terminal name.

Syntax : $tty

'CLEAR' Command :

It is used to clear the screen.

Syntax : $clear

'MAN' Command :

It help us to know about the particular command and its options & working. It is like „help" command in windows .

Syntax : $man <command name>

MANIPULATION Command :

It is used to manipulate the screen.

Syntax : $tput <argument> **Arguments :**

1. Clear – to clear the screen.
2. Longname – Display the complete name of the terminal.
3. SMSO – background become white and foreground become black color.
4. ..rmso – background become black and foreground becomes white color.
5. Cop R C – Move to the cursor position to the specified location.
6. Cols – Display the number of columns in our terminals.

LIST Command :

It is used to list all the contents in the current working directory.

Syntax : $ ls – options <arguments>

If the command does not contain any argument means it is working in theCurrent directory.

Options :

a– used to list all the files including the hidden files.c– list all the files columnwise.

d- list all the directories.

m- list the files separated by commas.

p- list files include „/" to all the directories.

r- list the files in reverse alphabetical order.

f- list the files based on the list modification date.

x-list in column wise sorted order.

DIRECTORY RELATED COMMANDS :

1. Present Working Directory Command :

To print the complete path of the current working directory.

Syntax : $pwd

2. MKDIR Command :

To create or make a new directory in a current directory .

Syntax : $mkdir <directory name>

3. CD Command :

To change or move the directory to the mentioned directory .

Syntax : $cd <directory name.

4. RMDIR Command :

To remove a directory in the current directory & not the current directory itself.

Syntax : $rmdir <directory name>

FILE RELATED COMMANDS :

CREATE A FILE :

To create a new file in the current directory we use CAT command.

Syntax :

$cat > <filename.

The > symbol is redirectory we use cat command.

DISPLAY A FILE :

To display the content of file mentioned we use CAT command without „>" operator.

Syntax :

$cat <filename.

Options –s = to neglect the warning /error message.

COPYING CONTENTS :

To copy the content of one file with another. If file doesnot exist, a new file is createdand if the file exists with some data then it is overwritten.

Syntax :

$ cat <filename source> >> <destination filename>

$ cat <source filename> >> <destination filename> it is avoid overwriting.

$cp <source filename path > <destination filename path>

MOVE Command :

To completely move the contents from source file to destination file and to remove thesource file.

Syntax :

$ mv <source filename> <destination filename>

REMOVE Command :

To permanently remove the file we use this command .

Syntax :

$rm <filename>

Options : -

-n content of file with numbers included with blank lines.

Syntax :

$cat –n <filename>

SORTING A FILE :

To sort the contents in alphabetical order in reverse order.

Syntax :

$sort <filename >

Option : $ sort –r <filename>

COPYING CONTENTS FROM ONE FILE TO ANOTHER :

To copy the contents from source to destination file . so that both contents are same.

Syntax :

$cp <source filename> <destination filename>

WORD Command :

To list the content count of no of lines , words, characters .

Syntax :

$wc<filename>

Options :

-c – to display no of characters.

-l – to display only the lines.

-w – to display the no of words.

LINE PRINTER :

To print the line through the printer, we use lp command.

Syntax :

$lp <filename>

PAGE Command :

This command is used to display the contents of the file page wise & next page can beviewed by pressing the enter key.

Syntax :

$pg <filename>

FILTERS AND PIPES

HEAD : It is used to display the top ten lines of file.

Syntax: $head<filename>

TAIL : This command is used to display the last ten lines of file.

Syntax: $tail<filename>

PAGE : This command shows the page by page a screenfull of information is displayed after which the page command displays a prompt and passes for the user to strike the enter key to continue scrolling.

Syntax: $ls –a\p

MORE : It also displays the file page by page .To continue scrolling with more command ,press the space bar key.

Syntax: $more<filename>

GREP : This command is used to search and print the specified patterns from the file.

Sy**ntax:** $grep [option] pattern <filename>

SORT : This command is used to sort the datas in some order.

Syntax: $sort<filename>

COMMUNICATION THROUGH UNIX COMMANDS

MESG

Description: The message command is used to give permission to other users to send message to your terminal.

Syntax: $mesg y

Command: **WRITE**

Description: This command is used to communicate with other users, who are logged in at the same time.

Syntax: $write <user name>

Command: **WALL**

Description: This command sends message to all users those who are logged in using theunix server.

Syntax: $wall <message>

Command: **MAIL**

Description: It refers to textual information, that can be transferred from one user toanother

Syntax: $mail <user name>

Command: **REPLY**

Description: It is used to send reply to specified user.Syntax: $reply<user name>

EX.NO :1.b **vi EDITOR COMMANDS**

DATE :

AIM :

To study the various commands operated in vi editor in UNIX.

DESCRIPTION :

The Vi editor is a visual editor used to create and edit text, files, documents and programs. It displays the content of files on the screen and allows a user to add, delete or change part of text . There are three modes available in the Vi editor , they are

1. Command mode
2. Input (or) insert mode.

Starting Vi :

The Vi editor is invoked by giving the following commands in UNIX prompt.

Syntax : $vi <filename> (or)

$vi

This command would open a display screen with 25 lines and with tilt (~) symbol at the start of each line. The first syntax would save the file in the filename mentioned and for the next the filename must be mentioned at the end.

Options :

1.vi +n <filename> - this would point at the nth line (cursor pos).

2.vi –n <filename> - This command is to make the file to read only to change from one mode to another press escape key.

INSERTING AND REPLACING COMMANDS :

To move editor from command node to edit mode, you have to press the <ESC> key. For inserting and replacing the following commands are used.

ESC a Command :

This command is used to move the edit mode and start to append after the current character.

Syntax : <ESC> a

ESC A COMMAND :

This command is also used to append the file , but this command append at the end ofcurrent line.

Syntax : <ESC> A

ESC i Command :

This command is used to insert the text before the current cursor position.

Syntax : <ESC> i

ESC I Command :

This command is used to insert at the beginning of the current line.

Syntax : <ESC> I

ESC o Command :

This command is insert a blank line below the current line & allow insertion of contents.

Syntax : <ESC> o

ESC O Command :

This command is used to insert a blank line above & allow insertion of contents.

Syntax : <ESC> O

ESC r Command :

This command is to replace the particular character with the given characters.

Syntax : <ESC> rx Where x is the new character.

ESC R Command :

This command is used to replace the particular text with a given text.

Syntax : <ESC> R text

<ESC> s Command :

This command replaces a single character with a group of character .

Syntax : <ESC> s

<ESC> S Command :

This command is used to replace a current line with group of characters.

Syntax : <ESC> S

CURSOR MOVEMENT IN vi :

<ESC> h:

This command is used to move to the previous character typed. It is used to move toleft of the text . It can also used to move character by character (or) a number of characters.

Syntax : <ESC> h – to move one character to left.

<ESC> nh – tomove „n" character to left.

<ESC> l :

This command is used to move to the right of the cursor (ie) to the next character. It canalso be used to move the cursor for a number of character.

Syntax : <ESC> l – single character to right.

<ESC> nl - „n" characters to right.

<ESC> j :

This command is used to move down a single line or a number of lines.

Syntax :

<ESC> j – single down movement.

<ESC> nj – „n" times down movement.

<ESC> k :

This command is used to move up a single line or a number of lines.

Syntax :

<ESC> k – single line above.

<ESC> nk – „n" lines above.

ENTER (OR) N ENTER :

This command will move the cursor to the starting of next lines or a group of linesmentioned.

Syntax :

<ESC> enter <ESC> n enter.

<ESC> + Command :

This command is used to move to the beginning of the next line.

Syntax :

<ESC> + <ESC> n+

Syntax :

<ESC> $

<ESC> ^ :

This command is used to move to first character of first lines.

Syntax :

<ESC> ^

<ESC> b Command :

This command is used to move back to the previous word (or) a number of words.

Syntax :

<ESC> b <ESC>nb

<ESC> - Command :

This command is used to move to the beginning of the previous line.

Syntax :

<ESC> - <ESC> n-

<ESC> 0 :

This command will bring the cursor to the beginning of the same current line.

Syntax :

<ESC> 0

<ESC> $:

This command will bring the cursor to the end of the current line.

<ESC> e Command :

This command is used to move towards and replace the cursor at last character of theword (or) no of words.

Syntax :

<ESC> e <ESC>ne

<ESC> w Command :

This command is used to move forward by a single word or a group of words.

Syntax :

<ESC> w <ESC> nw

DELETING THE TEXT FROM Vi :

<ESC> x Command :

To delete a character to right of current cursor positions , this command is used.

Syntax :

<ESC> x <ESC> nx

<ESC> X Command :

To delete a character to left of current cursor positions , this command is used.

Syntax :

<ESC> X <ESC> nX

<ESC> dw Command :

This command is to delete a single word or number of words to right of current cursorposition.

Syntax :

<ESC> dw <ESC> ndw

db Command :

This command is to delete a single word to the left of the current cursor position.

Syntax :

<ESC> db <ESC> ndb

<ESC> d$ Command :

This command is used to delete the text from current cursor position to last character ofcurrent line.

Syntax : <ESC> d$

SAVING AND QUITING FROM vi :-

<ESC> w Command :

To save the given text present in the file.

Syntax : <ESC> : w

<ESC> q! Command :

To quit the given text without saving.

Syntax : <ESC> :q!

<ESC> wq Command :

This command quits the vi editor after saving the text in the mentioned file.

Syntax : <ESC> :wq

<ESC> x Command :

This command is same as „wq" command it saves and quit.

Syntax : <ESC> :x

<ESC> q Command :

This command would quit the window but it would ask for again to save the file.

Syntax : <ESC> : q

EX.NO :1.c **UNIX SHELL PROGRAMMING COMMANDS:**

AIM :

To study about the Unix Shell Programming Commands.

INTRODUCTION :

Shell programming is a group of commands grouped together under single filename.After logging onto the system a prompt for input appears which is generated by a CommandString Interpreter program called the shell. The shell interprets the input, takes appropriate action, and finally prompts for more input. The shell can be used either

interactively - enter commands at the command prompt, or as an interpreter to execute a shell script. Shell scripts are dynamically interpreted, NOT compiled.

Common Shells.

C-Shell - csh : The default on teaching systems Good for interactive systems Inferiorprogrammable features

Bourne Shell - bsh or sh - also restricted shell - bsh : Sophisticated pattern matchingand file name substitution

Korn Shell : Backwards compatible with Bourne Shell Regular expressionsubstitution emacs editing mode

Thomas C-Shell - tcsh : Based on C-Shell Additional ability to use emacs to edit the command line Word completion & spelling correction Identifying your shell.

SHELL KEYWORDS :

echo, read, if fi, else, case, esac, for , while , do , done, until , set, unset, readonly, shift,export, break, continue, exit, return, trap , wait, eval ,exec, ulimit , umask.

General things SHELL

The shbang line The "shbang" line is the very first line of the script and lets the kernel know what shell will be interpreting the lines in the script. The shbang line consists of a #! followed by the full pathname to the shell, and can be followed by options to control the behavior of the shell.

EXAMPLE

```
#!/bin/sh
```

Comments Comments are descriptive material preceded by a # sign. They are in effect untilthe end of a line and can be started anywhere on the line.

EXAMPLE

```
# this text is not
# interpreted by the shell
```

Wildcards There are some characters that are evaluated by the shell in a special way. They arecalled shell metacharacters or "wildcards." These characters are neither numbers nor letters. For example, the *, ?, and [ ] are used for filename expansion. The <, >,2>, >>, and | symbols are used for standard I/O redirection and pipes. To prevent these characters from being interpreted by the shell they must be quoted.

EXAMPLE

Filename expansion:

```
rm *; ls ??; cat file[1-3];
```

Quotes protect metacharacter:

```
echo "How are you?"
```

SHELL VARIABLES :

Shell variables change during the execution of the program .The C Shell offers acommand "Set" to assign a value to a variable.

For example:

```
% set myname= Fred
% set myname = "Fred Bloggs"
% set age=20
```

A $ sign operator is used to recall the variable values.For example:

```
% echo $myname will display Fred Bloggs on the screen
A @ sign can be used to assign the integer constant values.
```

For example:

```
%@myage=20
%@age1=10
%@age2=20
%@age=$age1+$age2
%echo $age
```

List variables

```
% set programming_languages= (C LISP)
% echo $programming _languages


C LISP
% set files=*.*
% set colors=(red blue green)
% echo
 $colors[2]blue
 % set colors=($colors yellow)/add to list
```

Local variables Local variables are in scope for the current shell. When a script ends, they are no longer available; i.e., they go out of scope. Local variables are set andassigned values.

EXAMPLE

```
variable_name=value
name="John Doe" x=5
```

Global variables Global variables are called environment variables. They are set for the currently running shell and any process spawned from that shell. They go out of scope whenthe script ends.

EXAMPLE

```
VARIABLE_NAME=value
export VARIABLE_NAMEPATH=/bin:/usr/bin:.
```

Extracting values from variables To extract the value from variables, a dollar sign is used.

EXAMPLE

```
echo
$variable_name
echo $name
echo $PATH
```

Rules : -

1.A variable name is any combination of alphabets, digits and anunderscore („-„);

2. No commas or blanks are allowed within a variable name.
3. The first character of a variable name must either be an alphabet or anunderscore.
4. Variables names should be of any reasonable length.

5. Variables name are case sensitive . That is , Name, NAME, name,NAme, are all different variables.

EXPRESSION Command :

To perform all arithematic operations .

Syntax :

Var = „expr$value1" + $ value2"

Arithmetic The Bourne shell does not support arithmetic. UNIX/Linux commands must be used to perform calculations.

EXAMPLE

```
n=`expr 5
+ 5`echo
$n
```

Operators The Bourne shell uses the built-in test command operators to test numbers andstrings.

EXAMPLE

Equality:

= *string*

!= *string*

-eq *number*

-ne *number*

Logical:

-a *and*

-o *or*

! *not*

Logical:

AND&&
OR ||

Relational:

-gt *greater than*

-ge *greater than, equal to*

-lt *less than*

-le *less than, equal to*

Arithmetic :

+, -, *, /, %

Arguments (positional parameters) Arguments can be passed to a script from the command line.Positional parameters are used to receive their values from within the script.

EXAMPLE

At the command line:

$ scriptname arg1 arg2
arg3 ...In a script:

echo $1 $2 $3 *Positional parameters*

echo $* *All the positional paramters*

echo $# *The number of positional parameters*

READ Statement :

To get the input from the user.

Syntax :

read x y

(no need of commas between variables)

ECHO Statement :

Similar to the output statement. To print output to the screen, the echo command is used.Wildcards must be escaped with either a backslash or matching quotes.

Syntax :

Echo “String” (or) echo $ b(for variable).

EXAMPLE

echo "What is your name?"

Reading user input The read command takes a line of input from the user and assigns it toa variable(s) on the right-hand side. The read command can accept muliple variable names.

CONDITIONAL STATEMENTS :

The if construct is followed by a command. If an expression is to be tested, it is enclosed insquare brackets. The then keyword is placed after the closing parenthesis. An if must end with a fi.

Syntax :

1. if: This is used to check a condition and if it satisfies the condition if thendoes the next action , if not it goes to the else part.
2. if…else

Syntax :

If cp $ source $ target

Then

Echo File copied successfully

Else

Echo Failed to copy the file.

3. nested if

here sequence of condition are checked and the corresponding performed accordingly.

Syntax :

if condition

then

command

if condition

then

command

else

command

fi

fi ----------

4. LOOPS

There are three types of loops: while, until and for. The while loop is followed by a command or an expression enclosed in square brackets, a do keyword, a block of statements, and terminated withthe done keyword. As long as the expression is true, the body of statements between do and done will be executed.

The until loop is just like the while loop, except the body of the loop will be executed as long asthe expression is false.

The for loop used to iterate through a list of words, processing a word and then shifting it off, toprocess the next word. When all words have been shifted from the list, it ends. The for loop is followed by a variable name, the in keyword, and a list of words then a block of statements, and terminates with the done keyword.

The loop control commands are break and continue.

EXAMPLE

while
command
do

block of statements

done

while [
expression]do

block of statements

done

until command

do

block of statements

statementsdone

for variable in word1 word2 word3 ...

do

block of

done

---

until [expression]

do

block of

statementsdone

---

until control

commanddo

commands

done

Break Statement :

This command is used to jump out of the loop instantly, without waiting to get the control command.

ARRAYS

(positional parameters) The Bourne shell does support an array, but a word list can be created by using positional parameters. A list of words follows the built-in set command, andthe words are accessed by position. Up to nine positions are allowed.The built-in shift command shifts off the first word on the left-hand side of the list. The individual words are accessed by position values starting at 1.

EXAMPLE

```
set word1 word2 word3
echo $1 $2 $3      Displays word1, word2, and
word3set apples peaches plums
shift                                  Shifts off apples
echo $1                                Displays first element of the list
echo $2                                Displays second element of the list
echo $*                                Displays all elements of the list
```

Command substitution To assign the output of a UNIX/Linux command to a variable, or usethe output of a command in a string, backquotes are used.

EXAMPLE

```
variable_name=`com
mand`echo
$variable_name
now=`date`
echo $now
echo "Today is `date`"
```

FILE TESTING

The Bourne shell uses the test command to evaluate conditional expressions and has a built-in set of options for testing attributes of files, such as whether it is a directory, a plain file (not a directory),a readable file, and so forth.

EXAMPLE

-d *File is a directory*

-f *File exists and is not a directory*

–r *Current user can read the file*

–s *File is of nonzero size*

–w *Current user can write to the file*

–x *Current user can execute the file*

```
#!/bin/sh
1if [ –f
 file ]
 then
    echo file exists
 fi
```

```
2if [ –d
 file ]
 then
    echo file is a directory
 fi

3if [ -s
 file ]
 then
    echo file is not of zero length
 fi

4if [ -r file -a -w
 file ]then
    echo file is readable and writable
 fi
```

EXECUTION OF SHELL SCRIPT :

1.By using change mode command2.$ chmod u + x sum.sh

3.$ sum.s hor

$ sh sum.sh

SHELL PROGRAMMING

Ex.No :2a CONCATENATION OF TWO STRINGS

Date:

Aim:

To write a shell program to concatenate two strings.

Algorithm:

Step1: Enter into the vi editor and go to the insert mode for entering the codeStep2: Read the first string.

Step3: Read the second string

Step4: Concatenate the two strings

Step5: Enter into the escape mode for the execution of the result and verify the output

Program:

```
echo "enter the first
string"read str1
echo "enter the second
string"read str2
echo "the concatenated string is" $str1$str2
```

Sample I/P:

```
Enter first string:
Hello Enter first
string: World
```

Sample O/P:

The concatenated string is HelloWorld

Ex.No. :2b COMPARISON OF TWO STRINGS

Aim:

To write a shell program to compare the two strings.

Algorithm:

Step1: Enter into the vi editor and go to the insert mode for entering the codeStep2: Read the first string.

Step3: Read the second string

Step4: Compare the two strings using the if loop

Step5: If the condition satisfies then print that two strings are equal else print twostrings are not equal.

Step6: Enter into the escape mode for the execution of the result and verify the output

Program:

```
echo "enter the first string"read str1
echo "enter the second string"read str2
if [ $str1 = $str2 ]then
echo "strings are equal"else
echo "strings are unequal"fi
```

Sample I/P:1

Enter first string: hai

Enter second string: hai

Sample O/P:1

The two strings are equal

Sample I/P:2

Enter first string: hai

Enter second string: cse

Sample O/P:2

The two strings are not equal

Ex.No:2c MAXIMUM OF THREE NUMBERS

Aim:

To write a shell program to find greatest of three numbers.

Algorithm:

Step1: Declare the three variables.

Step2: Check if A is greater than B and
C.Step3: If so print A is greater.

Step4: Else check if B is greater than
C.Step5: If so print B is greater.

Step6: Else print C is greater.

Program:

```
echo "enter
A"read a

echo "enter
B"read b

echo "enter
C"read c

if [ $a -gt $b -a $a -gt
$c ]then

echo "A is greater"

elif [ $b -gt $a -a $b -gt
$c ]then

echo "B is
greater"else

echo "C is
greater"fi
```

Sample I/P:

```
Enter
A:23
Enter
B:45
Enter
C:67
```

Sample O/P:

```
C is greater
```

Ex.No:2d FIBONACCI SERIES

Aim:

To write a shell program to generate fibonacci series.

Algorithm :

Step 1 : Initialise a to 0 and b to 1. Step 2 : Print the values of 'a' and 'b'.

Step 3 : Add the values of 'a' and 'b'. Store the added value in variable 'c'.Step 4 : Print the value of 'c'.

Step 5 : Initialise 'a' to 'b' and 'b' to 'c'.

Step 6 : Repeat the steps 3,4,5 till the value of 'a' is less than 10.

Program Do your self

Ex.No:2e ARITHMETIC OPERATIONS USING CASE

Aim:

To write a shell program to perform the arithmetic operations using case

Algorithm :

Step 1 : Read the input variables and assign the value

Step 2 : Print the various arithmetic operations which we are going to perform

Step 3 : Using the case operator assign the various functions for the arithmetic operators.

Step 4 : Check the values for all the corresponding operations.

Step 5 : Print the result and stop the execution.

Program Do your self

SYSTEM CALLS

Ex.No:3a PROCESS CREATION

AIM:

To write a program to create a process in UNIX.

ALGORITHM:

STEP 1: Start the program. STEP 2: Declare pid as integer.

STEP 3: Create the process using Fork command.

STEP 4: Check pid is less than 0 then print error else if pid is equal to 0 then execute command else parent process wait for child process.

STEP 5: Stop the program.

SAMPLE OUTPUT:

```
$cc pc.c
$a.out
parent process$ child process
$ps
```

```
PID   CLS  PRI   TTY   TIME  COMD5913
          TS    70    pts022    0:00     ksh
6229      TS    59    pts022    0:00     ps
```

Ex.No:3b EXECUTING A COMMAND

AIM:

To write a program for executing a command.

ALGORITHM:

STEP 1: Start the program.

STEP 2: Execute the command in the shell program using exec ls.

STEP 3: Stop the execution.

PROGRAM:

```
echo Program for executing UNIX command using shell programming
echo Welcome

ps
exec wc e
```

SAMPLE OUTPUT:

```
$ sh exec.sh
program for executing UNIX command using shell
programmingWelcome
```

PID CLS PRI TTY TIME

COMD958 TS 70 pts001

0:00 ksh

971 TS 70 pts001 0:00 sh

972 TS 59 pts001 0:00 ps

3 41 81 e

Ex.No:3c SLEEP COMMAND

AIM:

To create child with sleep command.

ALGORITHM:

STEP 1: Start the program.

STEP 2: Create process using fork and assign into a variable.

STEP 3: If the value of variable is < zero print not create and > 0 process create andelse print child create.

STEP 4: Create child with sleep of

2.STEP 5: Stop the program.

<<Program do your self>>

OUTPUT:

$ cc sleep.c

$ a.out

parent process$ this is child process

Ex.No:3d SLEEP COMMAND USING GETPID

AIM:

To create child with sleep command using getpid.

ALGORITHM:

STEP 1: Start the execution and create a process using fork() command.

STEP 2: Make the parent process to sleep for 10 seconds.

STEP 3:In the child process print it pid and it corresponding pid.

STEP 4: Make the child process to sleep for 5 seconds.

STEP 5: Again print it pid and it parent pid.

STEP 6: After making the sleep for the parent process for 10 seconds print it pid.

STEP 7: Stop the execution.

<<*Program do your self*>>

OUTPUT:

$ cc sleepid.c

$ a.out

parent

process

child

process

child process id is 12691

its parent process id is

12690child process after

sleep=5 child process id is

12691

its parent process id is

12690child process after

sleep=10 child id is

12690

parent id is

11383parent

terminates

$

Ex.No:3e SIGNAL HANDLING

AIM:

To write a program for signal handling in UNIX.

ALGORITHM:

STEP 1:start the program

STEP 2:Read the value of

pid.

STEP 3:Kill the command surely using kill-9 pid.

STEP 4:Stop the program.

PROGRAM:

```
echo program for performing KILL
operationsps
echo enter the
pidread pid
kill-9 $pid
echo
finished
```

OUTPUT:

```
$sh kill.sh
program for performing KILL operations

 PID   CLS  PRI   TTY          TIME   COMD
858    TS    70    pts001       0:00   ksh
858    TS    70    pts001       0:00   sh
858    TS    59    pts001       0:00   ps
enter the pid
872
killed
$sh kill.sh
```

program for performing KILL operations

PID CLS PRI TTY TIME

COMD858 TS 70 pts001

0:00 ksh

858 TS 70 pts001 0:00 sh

858 TS 59 pts001 0:00 ps

enter the

pid876

UX:kill(kill.sh):ERROR: no such proccess

$sh kill.sh

program for performing KILL operations

PID CLS PRI TTY TIME COMD

858 TS 70 pts001 0:00 ksh

858 TS 70 pts001 0:00 sh

858 TS 59 pts001 0:00 ps

enter the pid

858

finished

Ex.No:3f WAIT COMMAND

AIM:

To perform wait command using c program.

ALGORITHM:

STEP 1: Start the execution

STEP 2: Create process using fork and assign it to a variable

STEP 3: Check for the condition pid is equal to 0

STEP 4: If it is true print the value of i and terminate the child process

STEP 5: If it is not a parent process has to wait until the child terminate

STEP 6: Stop the execution

<<Program do your self>>

ExNo:4a READING FROM A FILE

Date:

AIM:

To create the file,read data from the file,update the file.

ALGORITHM:

1. Get the data from the user.
2. 2.Open a file.

3.Read from the file.4.Close the file.

PROGRAM:

```
#include<stdi
o.h>int main()
{
        char
        str[100];
        FILE *fp;
        fp=fopen("file1.dat","r"
        );while(!feof(fp))
        {
                fscanf(fp,"%s",str);
                printf(" %s ",str);
        }
         fclose(fp);
}
```

OUTPUT:

$ vi read1.c

$gcc read1.c

$ ./a.out

hai this is a program to read the content of the file.

ExNo:4b WRITING INTO A FILE

Date

AIM:

To write a C program to write the data into a file.

ALGORITHM:

Step1.Get the data from the user.Step2.Open a file.

Step3.Write the data from the file. Step4.Get the data and update the file.

PROGRAM:

```
#include<stdi
o.h>int main()
{
        char
        str[100];
        FILE *fp;
        printf("Enter the string");
        gets(str);
        fp=fopen("file1.dat","w
        +");while(!feof(fp))
        {
                fscanf(fp,"%s",str);
        }
```

```
        fprintf(fp,"%s",str);
    }
```

OUTPUT:

```
$ gcc write.c
$ ./a.out
Enter the string: os lab
$vi file1.dat

os lab
```

ExNo:4c FILE CREATION

Date:

AIM:

To write a C program to create a file.

ALGORITHM:

Step1:Start the program.

Step2:Create the file using create function and assign a variable to it.

Step3:If the value of the variable is less then print file cannot be created ,otherwise print file is created.

Step4:Stop the program.

PROGRAM:

```
void main()
{
    int id;
    if(id=creat("z.txt",0)==
    -1)
```

```
        {
                printf("cannot create the
                file");exit(1);
        }
                else
                        {
                                printf("file is
                                created");exit(1);
                        }

  }
```

OUTPUT:

```
$ cc fc.c
$ a.out
file is created $
```

Ex. No:5a IMPLEMENTATION OF ls COMMAND

AIM:

To write a C program to simulate the operation of "ls" command in Unix.

ALGORITHM:

1. Check if the number of command line arguments is less than 2. If yes,Print error and exit.
2. Check if the second argument (i.e. directory to be listed) is valid or not. If not thenexit.
3. Print the content of the directory till it becomes NULL.
4. Close the directory entry file.

PROGRAM:

```
#include<stdio.h>
#include<sys/types.h>
#include<dirent.h>
main(int argc, char
*argv[])
```

```
{
        DIR *dp;
        struct dirent
         *dirp;
         if(argc<2)
        {
                printf("\n You have provided only 1
                argument\n");exit(1);
         }
          if((dp=opendir(argv[1]))==NULL)
         {
              printf("\nCannot open %s
              file!\n",argv[1]);exit(1);
          }
          while((dirp=readdir(dp))!=NULL
          )printf("%s\n",dirp->d_name);
    closedir(dp);
}
```

OUTPUT:

```
[root@lab1cab01 orbit]# gcc
lsdemo.c[root@lab1cab01 orbit]#
./a.out


You have provided only 1 argument
[root@lab1cab01 orbit]# ./a.out
/xyz/foo
Cannot open /xyz/foo file!
[root@lab1cab01 orbit]# ./a.out /root
[root@lab1cab01 orbit]# ./a.out
/tmp/orbit
.
..
sem.c
first.c
best.c
```

a.out
lsdem
o.c
roundrobin.c
bestop.doc
firstop.doc
roundrobin.d
oc
[root@lab1cab01 orbit]#

ExNo:6a FIRST COME FIRST SERVE

AIM:

To write a C program to implement the CPU scheduling algorithm for FIRST COME FIRST SERVE.

PROBLEM DESCRIPTION:

Cpu scheduler will decide which process should be given the CPU for its execution.For this it use different algorithm to choose among the process. one among that algorithm is fcfs algorithm.

In this algorithm the process which arrive first is given the cpu after finishing its request only it will allow cpu to execute other process.

ALGORITHM:

Step1: Create the number of process.

Step2: Get the ID and Service time for each process.

Step3: Initially, Waiting time of first process is zero and Total time for the firstprocess is the starting time of that process.

Step4: Calculate the Total time and Processing time for the remaining processes.Step5: Waiting time of one process is the Total time of the previous process.

Step6: Total time of process is calculated by adding Waiting time and Service time.Step7: Total waiting time is calculated by adding the waiting time for lack process.Step8: Total turn around time is calculated by adding all total time of each process.

Step9: Calculate Average waiting time by dividing the total waiting time by totalnumber of process.

Step10: Calculate Average turn around time by dividing the total time by thenumber of process.

Step11: Display the result.

ExNo:6b SHORTEST JOB FIRST

AI

M:

To write a C program to implement the CPU scheduling algorithm for Shortest job first.

PROBLEM DESCRIPTION:

Cpu scheduler will decide which process should be given the CPU for its execution. For this it use different algorithm to choose among the process. one among that algorithm is sjf algorithm.

In this algorithm the process which has less service time given the cpu after finishing its request only it will allow cpu to execute next other process.

ALGORITHM:

Step1:Get the number of process.

Step2:Get the id and service time for each process.

Step3:Initially the waiting time of first short process as 0 and total time of firstshort is process the service time of that process.

Step4:Calculate the total time and waiting time of remaining process.

Step5:Waiting time of one process is the total time of the previous process.

Step6:Total time of process is calculated by adding the waiting time and servicetime of each process.

Step7:Total waiting time calculated by adding the waiting time of each process.Step8:Total turn around time calculated by adding all total time of each process.

Step9:calculate average waiting time by dividing the total waiting time by totalnumber of process.

Step10:Calculate average turn around time by dividing the total waiting time bytotal number of process.

Step11:Display the result.

Ex.No :7a ROUND ROBIN

Date:

AIM :

To write a C program to simulate the CPU scheduling algorithm for round robin

PROBLEM DESCRIPTION:

CPU scheduler will decide which process should be given the CPU for its execution .For this it use different algorithm to choose among the process .one among that algorithm is Round robin algorithm.

In this algorithm we are assigning some time slice .The process is allocated according to the time slice ,if the process service time is less than the time slice then process itself will release the CPU voluntarily .The scheduler will then proceed to the next process in the ready queue .If the CPU burst of the currently running process is longer than time quantum ,the timerwill go off and will cause an interrupt to the operating system .A context switch will be executed and the process will be put at the tail of the ready queue.

ALGORITHM:

Step 1: Initialize all the structure elements

Step 2: Receive inputs from the user to fill process id,burst time and arrival time.Step 3: Calculate the waiting time for all the process id.

i) The waiting time for first instance of a process is calculated as:a[i].waittime=count + a[i].arrivt

ii) The waiting time for the rest of the instances of the process iscalculated as:

a) If the time quantum is greater than the remaining burst time thenwaiting time is calculated as:

a[i].waittime=count + tq

b) Else if the time quantum is greater than the remaining bursttime then waiting time is calculated as:

a[i].waittime=count - remaining burst time

Step 4: Calculate the average waiting time and average turnaround time

Step 5: Print the results of the step 4.

ExNo:7b PRIORITY SCHEDULING:

AIM: To write a C program to implement CPU scheduling algorithm for priority scheduling.

PROBLEM DESCRIPTION:

Cpu scheduler will decide which process should be given the CPU for its execution.For this it use different algorithm to choose among the process. one among that algorithm is fcfs algorithm.

In this algorithm the process which arrive first is given the cpu after finishing its request only it will allow cpu to execute other process.

ALGORITHM:

Step1:Get the number of process,burst time and priority.Step2:Using for loopi=0 to n-1 do step 1 to 6.

Step3:If i=0,wait time=0,T[0]=b[0];

Step4:T[i]=T[i-1]+b[i] and wt[i]=T[i]-b[i].

Step5: Total waiting time is calculated by adding the waiting time for lack process.Step6: Total turn around time is calculated by adding all total time of each process.

Step7: Calculate Average waiting time by dividing the total waiting time by total numberofprocess.

Step8: Calculate Average turn around time by dividing the total time by the number ofprocess.

Step9: Display the result.

ExNo:8 PRODUCER CONSUMER PROBLEM USING SEMAPHORE

AIM:

To write a C program to implement the Producer & consumer Problem (Semaphore)

ALGORITHM:

Step 1: The Semaphore mutex, full & empty are initialized.

Step 2: In the case of producer process

i) Produce an item in to temporary variable.
ii) If there is empty space in the buffer check the mutex value for enter into the criticalsection.
iii) If the mutex value is 0, allow the producer to add value in the temporary variable to thebuffer.

Step 3: In the case of consumer process

i) It should wait if the buffer is empty
ii) If there is any item in the buffer check for mutex value, if the mutex==0, remove itemfrom buffer.
iii) Signal the mutex value and reduce the empty value by 1.
iv) Consume the item.

Step 4: Print the result

Ex.No:9 MEMORY MANAGEMENT SCHEME- PAGING

AIM:

To write a C program to implement memory management using paging technique.

ALGORITHM:

Step1 : Start the program.

Step2 : Read the base address, page size, number of pages and memory unit.

Step3 : If the memory limit is less than the base address display the memorylimit is less than limit.

Step4 : Create the page table with the number of pages and page address.

Step5 : Read the page number and displacement value.

Step6 : If the page number and displacement value is valid, add the displacementvalue with the address corresponding to the page number and display theresult.

Step7 : Display the page is not found or displacement should be less than pagesize.

Step8 : Stop the program.

Ex.No:10 MEMORY MANAGEMENT SCHEME-SEGMENTATION

AIM:

To write a C program to implement memory management using segmentation

ALGORITHM:

Step1 : Start the program.

Step2 : Read the base address, number of segments, size of each segment, memory limit.

Step3 : If memory address is less than the base address display "invalid memorylimit".

Step4 : Create the segment table with the segment number and segment address and display it.

Step5 : Read the segment number and displacement.

Step6 : If the segment number and displacement is valid compute the real address and display the same.

Step7 : Stop the program.