BASIC COMMANDS TO KNOW BEFORE SHELL SCRIPTING

What is Shell Scripting?

Shell scripts are a series of commands written in a specific scripting language that can be executed directly by the operating system's command interpreter, without the need for a compiler or other tools.

What is Bash?

Bash, short for "Bourne-Again SHell," is a popular command-line shell and scripting language used on Unix-based operating systems such as Linux and macOS.

FILE NAVINGATION COMMANDS IN UNIX

cd <FolderName> - To change directory
mkdir <FolderName> - To create directory
touch <fileName> - To create a file

STEPS TO RUN THE SCRIPT FILE

WHILE USING GIT BASH IN WINDOWS:

WAY 1 – TO RUN BASH SCRIPT:

bash <FileName>

WAY 2 - TO RUN SHELL SCRIPT:

sh <FileName>

SCRIPTS CAN BE RUN EITHER BY DIRECTLY OR BY SETTING THE PATH

METHOD -1 :SETTING THE PATH

Note:Changing path in wrong way results in crashing OS ,Please be Carefull

How to set Path ?

set path in .profile (path where the scripts are)

STEP - 1

nano ~/.profile

STEP - 2

#INSIDE .profile type the below line

export PATH="$PATH:$HOME/PATH_TO_THE_SCRIPTS"

To run the script :

Step - 1:

Give executable permission using following commands:

chmod +x <filename>

(or)

chmod 777 <filename>

Step – 2:

Simply Run the file using the command below

. / <fileName>

METHOD 2: DIRECTLY

Way – 1:

bash <FileName>

Way -2:

sh <FileName>

Note:

Here we are using bash and shell as interpreter at some points, Bash is written over the shell. Hence, note that all the codes which are interpreted by the shell was also interpreted by bash, But some of the syntax are not supported by shell. Therefore, it is better to use bash as interpreter at most.

Ex. No: 7

DATE:

FILE OPERATIONS

AIM:

To write a bash script program to perform file operations.

ALGORITHMS:

1. Start
2. Display the menu and get the choice.
3. If choice is 1, get the file name and create a file.
4. If choice is 2, get the file name and show the file.
5. If choice is 3, get source and destination file name and copy it.
6. If choice is 4, get the source and new file name and rename it.
7. If choice is 5, get the file name and append the text.
8. If choice is 6, get the source and new file name and create a short cut link.
9. If choice is 7, get the file name and count the words.
10. If choice is 8, the exit.

SOURCE CODE:

```
options=("Create_a_File" "Show_File_contents" "Copy_a_File" "Rename_a_File"
"Append_a_File" "Linking_a_file" "No_of_words_in_a_File" "Exit")

PS3="Enter your option: "
select opt in ${options[@]}
do

case $opt in

    ${options[0]})
        read -p "Enter file name : " name
        touch $name  ;;
    ${options[1]})
        read -p "Enter File name : " name
        cont=$(cat $name)
        echo $cont  ;;
    ${options[2]})
        read -p "Enter  File Name to copy: " s
        read -p "Enter File Name to save the copy : " a
```

```
        cp $s $a
        echo "Copied Successfully" ;;
    ${options[3]})
        read -p "Enter old File Name : " s
        read -p "Enter new File Name : " a
        mv $s $a
        echo "Renamed Successfully" ;;
    ${options[4]})
        read -p "Enter file Name to write : " f
        echo "Write the conftents (press Ctrl + D to stop writing )"
        cat >>$f ;;
    ${options[5]})
        read -p "Enter  File to Link  : " f1
        read -p "Enter New File to Link : " f2
        ln $f1 $f2
        ;;
    ${options[6]})
        read -p "Enter the File Name : " f
        w=$(cat $f |wc -w)
        echo "The File $f contains $w Words. ";;
    ${options[7]})
        echo "You are on Exit .."
        exit 0 ;;

esac

done
```

SAMPLE OUTPUT:

```
$ sh EX07_FILE_OPERATIONS.sh

1) Create_a_File        3) Copy_a_File          5) Append_a_File        7) No_of_words_in_a_File
2) Show_File_contents   4) Rename_a_File        6) Linking_a_file       8) Exit
Enter your option : 1
Enter file name : hello

Enter your option : 5
Enter file Name to write : hello
Write the contents
Hello World !!!

$ sh EX07_FILE_OPERATIONS.sh
1) Create_a_File        3) Copy_a_File          5) Append_a_File        7) No_of_words_in_a_File
2) Show_File_contents   4) Rename_a_File        6) Linking_a_file       8) Exit

Enter your option : 2
Enter File name : hello
Hello World !!!
```

```
Enter your option : 3
Enter  File Name to copy: hello
Enter File Name to save the copy : hello2
Copied Successfully

Enter your option : 2
Enter File name : hello2
Hello World !!!

Enter your option : 4
Enter old File Name : hello2
Enter new File Name : hello_world
Renamed Successfully

$ sh EX07_FILE_OPERATIONS.sh
1) Create_a_File          3) Copy_a_File          5) Append_a_File          7) No_of_words_in_a_File
2) Show_File_contents     4) Rename_a_File        6) Linking_a_file         8) Exit

Enter your option : 6
Enter  File to Link  : hello
Enter New File to Link : hello2

Enter your option : 7
Enter the File Name : hello
The File hello contains 3 Words.

Enter your option : 8
You are on Exit ..
```

RESULT:

Thus the above shell script for file operations was executed successfully.

EX. NO : 8

DATE:

UNIX UTILITES

AIM:

To write a bash script program to perform the basic UNIX utilities.

ALGORITHM:

1. Start.
2. Display the menu and get the choice.
3. If choice is 1,get file name and create a file and write in it.
4. If choice is 2, get the file name and perform head command.
5. If choice is 3, get the file name and perform tail command.
6. If choice is 4, get the source and perform cut command.
7. If choice is 5, get the source and destination file name and copy it.
8. If choice is 6, get the source and destination file name and join it.
9. If choice is 7, get the file name and show the difference.
10. If choice is 8, the exit.

SOURCE CODE:

```
#!/bin/bash

options=("Create" "Head" "Tail" "Cut" "Paste" "Join" "Diff" "Exit")

PS3="Enter your option : "

select opt in ${options[@]}
do
  case $opt in
        ${options[0]})
          read -p "Enter file name : " f
          touch $f
          echo "Write something in file (Press Ctrl + D to stop writing)"
```

```
            cat >> $f ;;
        ${options[1]})
            read -p "Enter file name : " f
            head -1 $f ;;
        ${options[2]})
            read -p "Enter file name : " f
            tail -1 $f ;;
        ${options[3]})
            read -p "Enter file name : " f
            cut -c -2 $f ;;
        ${options[4]})
            read -p "Enter source file name : " f1
            read -p "Enter destination file name : " f2
            paste $f1 $f2 ;;
        ${options[5]})
            read -p "Enter file 1 name : " f1
            read -p "Enter file 2 name : " f2
            read -p "Enter destination file name : " df
            cat $f1 $f2 > $df ;;
        ${options[6]})
            read -p "Enter file 1 name : " f1
            read -p "Enter file 2 name : " f2
            diff $f1 $f2;;
        ${options[7]})
            echo "You are on Exit .."
            exit 0 ;;

    esac

done
```

SAMPLE OUTPUT:

1) Create
2) Head
3) Tail
4) Cut

5) Paste
6) Join
7) Diff
8) Exit
Enter your option : 1

Enter file name : file1.txt
Write something in file (Press Ctrl + D to stop writing)
Iam an Aspiring Data Engineer
Iam proud to be a student of annamalai university
Hard work always wins

Enter your option : 1
Enter file name : file2.txt
Write something in file (Press Ctrl + D to stop writing)
Ragupathy
jenly
Bella Swen
Iam an Aspiring Data Engineer

Enter your option : 3
Enter file name : file1.txt
Hard work always wins
Enter your option : 4
Enter file name : file1.txt
Ia
Ia
Ha

Enter your option : 5
Enter source file name : file1.txt
Enter destination file name : file2.txt
Iam an Aspiring Data Engineer Ragupathy
Iam proud to be a student of annamalai university jenly

Hard work always wins Bella Swen

Enter your option : 6
Enter file 1 name : file1.txt
Enter file 2 name : file2.txt
Enter destination file name : file3.txt4

Enter your option : 7
Enter file 1 name : file1.txt
Enter file 2 name : file3.txt

Enter your option : 7
Enter file 1 name : file1.txt
Enter file 2 name : file2.txt
1,3d0
< Iam an Aspiring Data Engineer
< Iam proud to be a student of annamalai university
< Hard work always wins
Enter your option : 8
You are on Exit ..

RESULT:

Thus the above shell script for basic UNIX utilities was executed successfull

Ex. No: 9

DATE:

SORTING OF 'N' NUMBERS USING AWK

AIM:

To write a shell script for arrange the numbers.

ALGORITHMS:

1. Start.
2. User is given a choice to perform ascending or descending order of sorting.
3. The choice is passed on to the Sorting function.
4. Get n number of elements from the user.
5. Sorting function uses Bubble sort algorithm.
6. Bubble Sort Algorithm:

```
for(i=0;i<n;i++)
for(j=0;j<n-1;j++)
    if(s[j]  '$v'  s[j+1])
    {
       t=s[j]
       s[j]=s[j+1]
       s[j+1]=t
    }
```

   Where '$v' is the variable that decides whether to sort in ascending or descending order.
7. Print the numbers.
8. Stop.

SOURCE CODE:

```
#!/bin/sh
Sorting(){
v=$1
   awk 'BEGIN{
      printf("\nEnter no of Data : ");
      getline n;
      printf("\nEnter %d elements\n",n)
      for(i=0;i<n;i++)
         getline s[i];
```

```
        for(i=0;i<n;i++){
            for(j=0;j<n-1;j++)
            {
                if(s[j] '$v's[j+1])
                {
                    t=s[j]
                    s[j]=s[j+1]
                    s[j+1]=t
                }
            }
        }
        printf("\nSorted Elements =>\n",n)
        for(i=0;i<n;i++)
            printf("%d  ",s[i]);
        printf("\n")


    }'
}

ch=1
until [ $ch -eq 0 ]
do
echo "1.Ascending"
echo "2.Descending"
echo "0.exit"
read -p "Your Option : " ch
case $ch in
    1)Sorting ">";;
    2)Sorting "<";;
    0)echo "you are on exit";;
esac

done
```

SAMPLE INPUT OUTPUT:

```
1.Ascending
2.Descending
0.exit

Your Option : 1
Enter no of Data : 5
Enter 5 elements
9
2
```

```
5
1
4

Elements after sorting =>
1  2  4  5  9
1.Ascending
2.Descending
0.exit

Your Option : 2
Enter no of Data : 5
Enter 5 elements
2
4
1
5
9

Elements after sorting=>
9  5  4  2  1

1.Ascending
2.Descending
0.exit
Your Option : 0

you are on exit
```

RESULT:

Thus the above shell script for sorting was executed successfully.

Ex. No: 10

DATE:

CALCULATE $^{N}C_{R}$ VALUES USING RECURSION

AIM:

To write a shell script performs $^{n}C_{r}$ calculation using recursion.

ALGORITHMS:

1. Start.
2. Read values for n and r.
3. Pass the parameters n, r, (n-r) to the user defined factorial function and store the returned values in nf, rf, nrf respectively.
4. Apply the following $_{n}C_{r}$ formula: res=nf\(rf*nrf)
5. Print res.
6. Stop.

SOURCE CODE:

```
#!/bin/sh
factorial(){
x=$1
i=1
fact=1

until [ $i -gt $x ]
do
   fact=`expr $fact \* $i`
   i=`expr $i + 1`
done

}

read -p "Enter n value : " n
read -p "Enter r value : " r

#n fact
factorial $n
nf=$fact

#r fact
factorial $r
```

```
rf=$fact

#n-r fact
factorial `expr $n - $r `
n_r=$fact

#r! * (n-r)!
rn_rf=`expr $rf \* $n_r `

ncr=`expr $nf / $rn_rf `

echo "The Combination of $n C $r is :  $ncr "
```

SAMPLE INPUT OUTPUT:

- INPUT 1:

  Enter n value : 20
  Enter r value : 5
  The Combination of 20 C 5 is : 15504

- INPUT 2:

  Enter n value : 3
  Enter r value : 3
  The Combination of 3 C 3 is: 1

RESULT:

Thus the shell script for above program was written and verified.

Ex. No: 11

DATE:

DISPLAY THE NUMBERS BETWEEN 1 AD 9999 IN WORDS

AIM:

To write a Shell script displays the numbers between 1 and 9999 in words.

ALGORITHMS:

1. Start.
2. Read the number.
3. Separate the number and depending upon the position display the value in words.
4. Stop.

SOURCE CODE:

```
#!bin/sh

printDigit(){
d=$1
   w=$2
   case $d in
      1) echo "one $w";;
      2) echo "two $w";;
      3) echo "three $w";;
      4) echo "four $w";;
      5) echo "five $w";;
      6) echo "six $w";;
      7) echo "seven $w";;
      8) echo "eight $w";;
      9) echo "nine $w";;
   esac
}
PrintGreaterThan20(){
   d=$1

   case $d in
      2) echo "twenty ";;
      3) echo "thirty ";;
      4) echo "forty ";;
      5) echo "fifty ";;
      6) echo "sixty ";;
      7) echo "seventy ";;
      8) echo "eighty ";;
      9) echo "ninety ";;
   esac
```

```
}
PrintTenToNineTeen(){
   d=$1
   case $d in
      10) echo "ten";;
      11) echo "eleven";;
      12) echo "twelve";;
      13) echo "thirteen";;
      14) echo "fourteen";;
      15) echo "fifteen";;
      16) echo "sixteen";;
      17) echo "seventeen";;
      18) echo "eighteen";;
      19) echo "nineteen";;
esac
}
read -p "Enter any number between 1-9999 : " n
num=$n

Quo=`expr $num / 1000`
num=`expr $num % 1000`
printDigit $Quo "thousand"

Quo=`expr $num / 100`
num=`expr $num % 100`
printDigit $Quo "hundred"
if [ $num -gt 0 ] && [ $n -gt 99 ]; then
   echo "and"
fi

if [ $num -gt 19 ]; then
   Quo=`expr $num / 10`
   num1=`expr $num % 10`
   PrintGreaterThan20 $Quo
   printDigit $num1
   exit 0
fi

if [ $num -gt 9 ]; then
      PrintTenToNineTeen $num
else
      printDigit $num
fi
```

SAMPLE INPUT OUTPUT:

- Test case 1:

  Enter any number between 1-9999 : 9873
  nine thousand
  eight hundred
  and
  seventy
  three

- Test Case 2:
  Enter any number between 1-9999 : 112
  one hundred
  and
  twelve

RESULT:

Thus the above shell script was executed and verified.

Ex. No: 12

DATE:

PALINDROME CHECKING

AIM:

To write a Shell script for Palindrome Checking.

ALGORITHMS:

1. Start.
2. Read the string.
3. Check the string is palindrome or not using wc, cut and ne operations.
4. Stop.

SOURCE CODE:

```
#!/bin/sh
read -p "Enter the string : " s1

len_of_s1=`echo "$s1" | wc -c`

until [ $len_of_s1 -lt 1 ]
do
char=`echo $s1 | cut -c $len_of_s1`
rev=${rev}${char}
len_of_s1=`expr $len_of_s1 - 1 `
done
echo "reversed String is $rev"
if [ $rev = $s1 ]; then
   echo "String is palindrome"
else
   echo "String is not a palindrome"
fi
```

SAMPLE INPUT OUTPUT:

- Test case 1:
  Enter the string : jenly
  reversed String is ylnej
  String is not a palindrome

- Test Case 2:
  Enter the string : deified
  reversed String is deified
  String is palindrome

RESULT:

Thus the above shell script was executed and verified.