**#Some Important Command:**

* **cd :** change directory
* **ls**: Show list of all files in that directory
* **mkdir**: make folder
* **rm -R <dir>**: delete all files in directory
* **touch**: create a file
* **chmod <permission> <filename>**: Add permissions

| **Permission** |
| --- |
| * + - **r:**read     - **w:** write     - **x:** execute     - **X:** execute only if the file is a directory or already has execute permission for some users.     - **t:** restricted deletion flag or sticky bit     - **s**: Set user / group id on execution |

* **ls -al:** Show list of all files with **permission** in that directory
* The first character represents
  + "**-**" for a regular file,
  + "**d**" for a directory,
  + "**l**" for a symbolic link.
* The next combination of character (adjacent three character) represents the file permission for owner
* The later three combination represents the file permission for the file group.
* The last three combination represents the file permission for the others.

**#To use vim as editor:**

-vim <filename>

- i + enter : to enter inset mode

- esc to exit insert mode

- **:wq** to save and quit

- **:q** to quit without saving

#**To use nano as editor:**

* nano <filename>
* Ctrl + o to save
* Ctrl + x to exit (You will be prompted if file is not saved)

**#Shells**: Shells are command-line interpreter.

# Find out the shell your environment has :

**Command**: cat /etc/shells

* /bin/bash : (Bourne Again Shell)is a interpreter which is improved version of sh shell.
* /bin/csh : Bourne Shell
* /bin/ksh:
* /bin/sh
* /bin/tcsh
* /bin/zsh

#Find Location of your bash:

**Command**: which bash

**1)First Shell Code:**

hello.sh

| #! /bin/bash  echo “hellow world” |
| --- |

**command:** ./hello.sh

**2)Variables:**

$CAPITAL\_CASE\_VARIABLE\_NAME = System Variable

$lower\_case\_variable\_name = User Variable

Into\_variable.sh

| #! /bin/bash  a=10  b=20  echo a = $a and b = $b |
| --- |

**3)Arithmetic Operation:**

| #! /bin/bash  a=10  b=20  echo $((a+b)) |
| --- |

| #! /bin/bash  a=10  b=20  c=$((a+b))  echo $c |
| --- |

**Try it yourself: (a+b)^2**

| #! /bin/bash  a=10.11  b=10.11  c=$a+$b  echo $c|bc |
| --- |

**Precision of number:**

| #! /bin/bash  echo "scale=5;11.211/3" | bc |
| --- |

**Power:**

| #! /bin/bash  echo "2^8" | bc -l  #-l is used to invoke math library |
| --- |

**Square root:**

| #! /bin/bash  echo "scale=4;sqrt(13)" | bc -l |
| --- |

**4)Input from User**

| #! /bin/bash  echo "Enter a:"  read a  echo "Enter b:"  read b  echo a = $a and b= $b |
| --- |

| #! /bin/bash  echo "Enter a & b:"  read a b  echo a = $a and b= $b |
| --- |

| #! /bin/bash  read -p "Enter a:" a  read -p "Enter b:" b  echo a = $a and b= $b |
| --- |

| #! /bin/bash  read -p "Enter id:" id  read -sp "Enter password:" pass  echo id = $id and pass= $pass |
| --- |

**5) Pass Argument during execution**

| #! /bin/bash  echo $0 $1 $2 $3 |
| --- |

| #! /bin/bash  args=("$@")  echo $@  echo $# |
| --- |

| #! /bin/bash  args=("$@")  echo ${args[0]} ${args[1]} ${args[2]} |
| --- |

**6) Conditional Statement- If:**

**Syntax:**

| if [ condition ]  then  #code to be executed if the condition is satisfied  else  #code to be executed if the condition is not satisfied  fi |
| --- |
| if [ condition ] && [condition]  then  #code to be executed if the condition is satisfied  else  #code to be executed if the condition is not satisfied  fi |
| if [ condition ] **||** [condition]  then  #code to be executed if the condition is satisfied  else  #code to be executed if the condition is not satisfie  fi |

**Condition:**

* **-eq :** equals to

example: if [ $var -eq 0 ]

* **-ne :** not equals to

example: if [ $var -q ne 0 ]

* **-gt Or > :** Greater than

example: if [ $var -gt 0 ]

if [ $var > 0 ]

* **-lt Or <** : Less than

example: if [ $var -gt 0 ]

if [ $var > 0 ]

* **-ge Or >= :** Greater than equals to

example: if [ $var -ge 10 ]

if [ $var >= 10 ]

* **-le Or <= :** Greater than equals to

example: if [ $var -le 10 ]

if [ $var <= 10 ]

| #! /bin/bash  a=10  if [ $a -eq 10 ]  then  echo $a is equal to 10  else  echo $a is not equal to 10  fi |
| --- |

| #! /bin/bash  a=13  if [ $a -ge 10 ]  then  echo $a is greater than or equal to 10  fi |
| --- |

**Conditional for String:**

* **== :** equals to

example: if [ $str == “value” ]

* **!= :** not equals to

example: if [ $str != “value” ]

* **< :** is less than in ASCII value

example: if [ $var -q ne 0 ]

* **> :** is greater than in ASCII value

example: if [ $var -q ne 0 ]

| #! /bin/bash  pass=abc123  read -sp "Enter your password:" inp  echo  if [ $pass == $inp ]  then  echo welcome  else  echo incorrect password  fi |
| --- |

**7) Loop Statement:**

* **While:**

**Syntax:**

| while [ condition ]  do  #code to be executed as long as the condition is satisfied  done |
| --- |

| #! /bin/bash  i=1  while [ $i -lt 10 ]  do  echo $i  ((i++))  done |
| --- |

| #! /bin/bash  i=1  while (($i <= 10 )) #we can use relational sign inside (( ))  do  echo $i  ((i++))  done |
| --- |

* **For :**

**Syntax:**

| for variable in {range\_start..range\_end}  #code to be executed as long as the condition is satisfied  done |
| --- |
| for ((start; condition; stepsize))  do  #code to be executed as long as the condition is satisfied  done |

| #! /bin/bash  for i in {1..10}  do  echo $i  done |
| --- |

| #! /bin/bash  for ((i=1;i<=10;i++))  do  echo $i  done |
| --- |

**8) Array**

1. **Indirect Declaration**

**ARRAYNAME[INDEXNR]=value**

1. **Explicit Declaration**

**declare -a ARRAYNAME**

1. **Compound Assignment**

**ARRAYNAME=(value1 value2 .... valueN)**

**Or**

**ARRAYNAME=([1]=10 [2]=20 [3]=30)**

* To print all the value of an array:

echo ${ARRAYNAME[\*]}

[@] & [\*] means All elements of Array.

Run this program:

#! /bin/bash

# To declare static Array

arr=(Soykot Shuvra Barik Dibbo)

# To print all elements of array

echo ${arr[@]}

echo ${arr[\*]}

echo ${arr[@]:0}

echo ${arr[\*]:0}

* To print elements from a particular index

echo ${ARRAYNAME[WHICH\_ELEMENT]:STARTING\_INDEX}

Run the following code:

# To print elements from a particular index

echo ${arr[@]:0}

echo ${arr[@]:1}

echo ${arr[@]:2}

echo ${arr[0]:1}