Unix (officially trademarked as UNIX, sometimes also written as Unix with small caps) is a computer operating system originally developed in 1969 by a group of AT&T employees at Bell Labs, including Ken Thompson, Dennis Ritchie, Brian Kernighan, Douglas McIlroy, and Joe Ossanna. Today the term Unix is used to describe any operating system that conforms to Unix standards, meaning the core operating system operates the same as the original Unix operating system. Today's Unix systems are split into various branches, developed over time by AT&T as well as various commercial vendors and non-profit organizations.

As of 2007, the owner of the trademark is The Open Group, an industry standards consortium.Only systems fully compliant with and certified according to the Single UNIX Specification are qualified to use the trademark; others are called "Unix system-like" or "Unix-like".

**What is Shell and its Function**

The shell provides you with an interface to the UNIX system. It gathers input from you and executes programs based on that input. When a program finishes executing, it displays that program's output.

A shell is an environment in which we can run our commands, programs, and shell scripts. There are different flavors of shells, just as there are different flavors of operating systems. Each flavor of shell has its own set of recognized commands and functions.

**Shell Script**

The basic concept of a shell script is a list of commands, which are listed in the order of execution. A good shell script will have comments, preceded by a pound sign, #, describing the steps.

In Linux (Shell), there are two types of variable:

(1) **System variables** - Created and maintained by Linux itself. This type of variable defined in

CAPITAL LETTERS.

(2) **User Defined Variables(UDV)** - Created and maintained by user. This type of variable defined in lower letters.

You can see system variables by giving command like **set or env.**

some of the important System variables are:

System Variable

BASH=/bin/bash

BASH\_VERSION=1.14.7(1) Our shell version name

COLUMNS=80 No. of columns for our screen

HOME=/home/abhay Our home directory

LINES=25 No. of columns for our screen

LOGNAME=students students Our logging name

OSTYPE=Linux Our Os type

PATH=/usr/bin:/sbin:/bin:/usr/sbin Our path settings

PS1=[\u@\h \W]\$ Our prompt settings

PWD=/home/students/Common Our current working directory

SHELL=/bin/bash Our shell name

USERNAME=abhay User name who is currently login to this PC

**User defined variables (UDV)**

To define UDV use following syntax

Syntax:

variable name=value

'value' is assigned to given 'variable name' and Value must be on right side = sign.

**Defining variables using declare keyword**

declare -i : integer

declare -a : array

declare -f : function

declare -r : readonly

declare -x: export

**Constant variables**

readonly is the keyword to make variables constants.

**Local and Global Shell Variables**

**Unsetting Variable**

unset a

**Special variable:**

($0 $n $# $\* $@ $$ $! $? )

Command line arguments and Exit status

echo "Number of arguments=$#"

echo "Name of shell =$0"

echo "First argument=$1"

echo "second argument=$2"

echo "ALL arguments=$\*"

echo "ALL arguments=$@"

exit 5;

**command substitution $ (command ) or with backticks, `command `**

d=$(date)

sum =$((2+3))

d =`date`

sum=`expr 2 + 3`

Mathematical Expressions:

1. POSSIX: echo $((2+3))
2. expr 2 + 3
3. echo “2 + 3” | bc OR bc <<< 2 + 3

cat add.sh

echo "First num is = $1"

echo "Second num is = $2"

echo "process id = $$"

sum=`echo "$1 + $2" | bc`

echo "Sum = $sum"

**test command or [ expr ]**

test expression or use [expression]

**Operators in Shell Script:**

There are following operators which we are going to discuss −

* Arithmetic Operators.
* Relational Operators.
* Boolean Operators.
* String Operators.
* File Test Operators.

**Arithmetic Operators :**

+ - \* / %, =, ==, !=, +=, ++, --,

|  |  |  |
| --- | --- | --- |
| + | Addition - Adds values on either side of the operator | `expr $a + $b` will give 30 |
| - | Subtraction - Subtracts right hand operand from left hand operand | `expr $a - $b` will give -10 |
| \* | Multiplication - Multiplies values on either side of the operator | `expr $a \\* $b` will give 200 |
| / | Division - Divides left hand operand by right hand operand | `expr $b / $a` will give 2 |
| % | Modulus - Divides left hand operand by right hand operand and returns remainder | `expr $b % $a` will give 0 |
| = | Assignment - Assign right operand in left operand | a=$b would assign value of b into a |
| == | Equality - Compares two numbers, if both are same then returns true. | [ $a == $b ] would return false. |
| != | Not Equality - Compares two numbers, if both are different then returns true. | [ $a != $b ] would return true. |

**Relational Operators:**

-eq

-ne

-gt

-lt

-ge

-le

|  |  |  |
| --- | --- | --- |
| -eq | Checks if the value of two operands are equal or not, if yes then condition becomes true. | [ $a -eq $b ] is not true. |
| -ne | Checks if the value of two operands are equal or not, if values are not equal then condition becomes true. | [ $a -ne $b ] is true. |
| -gt | Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true. | [ $a -gt $b ] is not true. |
| -lt | Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true. | [ $a -lt $b ] is true. |
| -ge | Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true. | [ $a -ge $b ] is not true. |
| -le | Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true. | [ $a -le $b ] is true. |

## Logical (Boolean) Operators :

!

-o (||)

-a (&&)

if [ $condition1 -a $condition2 ]

if [ $condition1 ] && [ $condition2 ]

if [[ $condition1 && $condition2 ]]

|  |  |  |
| --- | --- | --- |
| ! | This is logical negation. This inverts a true condition into false and vice versa. | [ ! false ] is true. |
| -o | This is logical OR. If one of the operands is true then condition would be true. | [ $a -lt 20 -o $b -gt 100 ] is true. |
| -a | This is logical AND. If both the operands are true then condition would be true otherwise it would be false. | [ $a -lt 20 -a $b -gt 100 ] is false. |

## **String Operators :**

|  |  |  |
| --- | --- | --- |
| = | Checks if the value of two operands are equal or not, if yes then condition becomes true. | [ $a = $b ] is not true. |
| != | Checks if the value of two operands are equal or not, if values are not equal then condition becomes true. | [ $a != $b ] is true. |
| -z | Checks if the given string operand size is zero. If it is zero length then it returns true. | [ -z $a ] is not true. |
| -n | Checks if the given string operand size is non-zero. If it is non-zero length then it returns true. | [ -z $a ] is not false. |
| Str | Check if str is not the empty string. If it is empty then it returns false. | [ $a ] is not false. |

## **File Test Operators :**

|  |  |  |
| --- | --- | --- |
| -b file | Checks if file is a block special file if yes then condition becomes true. | [ -b $file ] is false. |
| -c file | Checks if file is a character special file if yes then condition becomes true. | [ -c $file ] is false. |
| -d file | Check if file is a directory if yes then condition becomes true. | [ -d $file ] is not true. |
| -f file | Check if file is an ordinary file as opposed to a directory or special file if yes then condition becomes true. | [ -f $file ] is true. |
| -g file | Checks if file has its set group ID (SGID) bit set if yes then condition becomes true. | [ -g $file ] is false. |
| -k file | Checks if file has its sticky bit set if yes then condition becomes true. | [ -k $file ] is false. |
| -p file | Checks if file is a named pipe if yes then condition becomes true. | [ -p $file ] is false. |
| -t file | Checks if file descriptor is open and associated with a terminal if yes then condition becomes true. | [ -t $file ] is false. |
| -u file | Checks if file has its set user id (SUID) bit set if yes then condition becomes true. | [ -u $file ] is false. |
| -r file | Checks if file is readable if yes then condition becomes true. | [ -r $file ] is true. |
| -w file | Check if file is writable if yes then condition becomes true. | [ -w $file ] is true. |
| -x file | Check if file is execute if yes then condition becomes true. | [ -x $file ] is true. |
| -s file | Check if file has size greater than 0 if yes then condition becomes true. | [ -s $file ] is true. |
| -e file | Check if file exists. Is true even if file is a directory but exists. | [ -e $file ] is true. |

**IF-THEN-ELSE**

If [ expr ]

then

commands;

else

commands;

fi

**elif :**

if [ $# -ne 3 ]

then

echo "$0: number1 number2 number3 are not given" >&2

exit 1

fi

n1=$1

n2=$2

n3=$3

if [ $n1 -gt $n2 ] && [ $n1 -gt $n3 ]

then

echo "$n1 is Biggest number"

elif [ $n2 -gt $n1 ] && [ $n2 -gt $n3 ]

then

echo "$n2 is Biggest number"

elif [ $n3 -gt $n1 ] && [ $n3 -gt $n2 ]

then

echo "$n3 is Biggest number"

elif [ $1 -eq $2 ] && [ $1 -eq $3 ] && [ $2 -eq $3 ]

then

echo "All the three numbers are equal"

else

echo "I can not figure out which number is bigger"

fi

**CASE :**

**case $variable-name in**

**pattern1)**

**command**

**...**

**..**

**command;;**

**pattern2)**

**command**

**...**

**..**

**command;;**

**patternN)**

**command**

**...**

**..**

**command;;**

**\*)**

**command**

**...**

**..**

**command;;**

**esac**

if test $# = 3

then

case $2 in

+) let z=$1+$3;;

-) let z=$1-$3;;

/) let z=$1/$3;;

x|X) let z=$1\*$3;;

\*) echo Warning - $2 invalied operator, only +,-,x,/ operator allowed

exit;;

esac

echo Answer is $z

else

echo "Usage - $0 value1 operator value2"

echo " Where, value1 and value2 are numeric values"

echo " operator can be +,-,/,x (For Multiplication)"

fi

**Loops**

**For loop :**

for variable in {List}

do

commands;

done

for (( exp; exp; exp ))

do

commands;

done

Examples:

1. for i in 1 2 3 4 5

do

echo $i

done

1. for i in {1..100..2}

do

echo $i > >file

done

1. for i in `cat file`

do

echo $line

done

1. for i in `ls`

do

echo “file = [$i]”

done

1. no=1

for i in $\*

do

echo “argument $no = $i”

no=echo $(( $no + 1 ))

done

6. i=1

for username in `awk -F: '{print $1}' /etc/passwd`

do

echo "Username $((i++)) : $username"

done

7.

|  |
| --- |
| **for (( i=1; i <= 5; i++ ))**  do  echo "Random number $i: $RANDOM"  done |

8. for server in 192.168.1.{1..10}

Do

Echo $server

Ping –c $server

Done

**Multiplication table:**

if [ $# -lt 1 ]; then

echo "Usage: $0 Table\_Number Table\_limit"

exit

fi

if [[ $1 =~ ^[0-9]+$ ]]; then

table\_no=$1

else

echo "Error: Table Number should be integer"

exit

fi

if [ $2 ]; then

table\_limit=$2

else

table\_limit=10

fi

for (( i=1; i<=table\_limit; i++))

do

echo "$table\_no \* $i = $(($table\_no \* i))"

done

**While loop :**

i=1;

while [ $i -le 10 ]

do

echo $i

i=`expr $i + 1`

done

**Until Loop :**

#!/bin/bash

option=1

until [ $option -eq 5 ]; do

echo ""

echo "MAIN-MENU"

echo "\*\*\*\*\*\*\*\*\*"

echo "1 - System Info"

echo "2 - Uptime"

echo "3 - Disk Info"

echo "4 - Memory Info"

echo "5 - Exit"

echo ""

echo -n "Enter option: "

read option

echo ""

case $option in

1 ) echo "System Info :"

uname -a

;;

2 ) echo "Uptime :"

uptime

;;

3 ) echo "Disk Info :"

df -h

;;

4 ) echo "Memory Info :"

free

;;

5 ) exit

;;

\* ) echo "Please Enter 1-5"

esac

done

**Break and continue:**

**File Handling:**

* 1. using for loop

for line in `cat /etc/users`

do

user=`echo $line | cut –d : -f1`

echo $user

done

* 1. using while loop

while read line

do

echo $line

done < file

OR

cat file | while read line

do

echo $line

done

* 1. Using head and tail

Print lines 6 to 10

Head -10 | tail -5

**Internal Field Separator (IFS):**

Str= “a:b:c:d:e:f:g”

IFS=:

For i in $str

Do

Echo $i

Done

**Awk:**

awk - pattern scanning and text processing language

Awk breaks each line of input passed to it into [field](#FIELDREF)