Switch Case

Syntax

case $var in pattern) commands to execute;;

pattern1) commands to execute;;

pattern2) commands to execute;;

pattern3) commands to execute;;

\*)

Default condition and commands to execute;;

esac

Code

fruit = “kiwi”

case $fruit” in “apple”) echo “apple is tasty”;;

“banana”) echo “I like banana”;;

“kiwi”) echo ”Newzeland is famous for kiwi”;;

\*)

Echo “default case”;;

esac

Output

New Zealand is famous for kiwi

Syntax

case $var in pattern|pattern1|pattern2) list of commands need to execute;;

pattern3| pattern4| pattern5) list of commands need to execute;;

pattern6) commands need to execute;;

\*)

Default condition and statements need to execute

esac

Code

mode = “jeep”;

case $mode in “lorry") echo "For $mode, rent is Rs.40 per k/m.";;

"jeep") echo "For $mode, rent is Rs.30 per k/m.";;

\*) echo "Sorry, I cannot get a $mode rent for you!";;

esac

Output

For jeep rental is Rs.5 per k/m

Code:

mode = “bike”;

case $mode in "sportscar") echo "For $mode, rent is Rs.20 per k/m.";;

"lorry") echo "For $mode, rent is Rs.50 per k/m.";;

"sumo") echo "For $mode, rent is Rs.30 per k/m.";;

"bicycle") echo "For $mode, rent is Rs. 5 per k/m.";;

\*) echo "Sorry, I can not get a $mode rent for you!";;

esac

Output

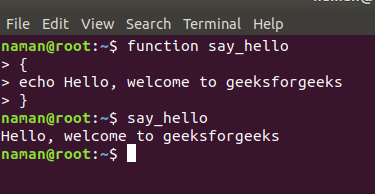
Sorry, I can not get a bike rental for you!

Function Commands

using function keyword : A function in linux can be declared by using keyword function before the name of the function. Different statements can be separated by a semicolon or a new line.

SYNTAX

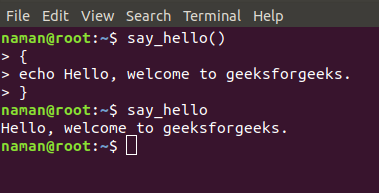
function name { COMMANDS ; }



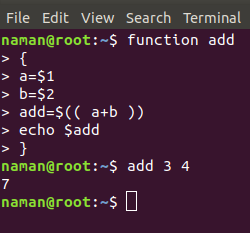
2. Using parenthesis: A function can also be declared by using parenthesis after the name of the function. Different statements can be separated by a semicolon or a new line.

SYNTAX

name () { COMMANDS ;



3. Parameterized function



$1 will displays the first argument that will be sent and $2 will display the second ans so on…

**Arrays**

Code 1:

#!/bin/bash

array1[0]=one

array1[1]=1

echo ${array1[0]}

echo ${array1[1]}

array2=( one two three )

echo ${array2[0]}

echo ${array2[2]}

array3=( [9]=nine [11]=11 )

echo ${array3[9]}

echo ${array3[11]}

read -a array4

for i in "${array4[@]}"

do

echo $i

done

Code2:

#!/bin/bash

array=( apple bat cat dog elephant frog )

#print first element

echo ${array[0]}

echo ${array:0}

#display all elements

echo ${array[@]}

echo ${array[@]:0}

#display all elements except first one

echo ${array[@]:1}

#display elements in a range

echo ${array[@]:1:4}

#length of first element

echo ${#array[0]}

echo ${#array}

#number of elements

echo ${#array[\*]}

echo ${#array[@]}

#replacing substring

echo ${array[@]//a/A}

exit 0

Output

apple

apple

apple bat cat dog elephant frog

apple bat cat dog elephant frog

bat cat dog elephant frog

bat cat dog elephant

5

5

6

6

Apple bAt cAt dog elephAnt frog

Code 3:

#!/bin/bash

read -p "Enter the matrix order [mxn] : " t

m=${t:0:1}

n=${t:2:1}

echo "Enter the elements for first matrix"

for i in `seq 0 $(($m-1))`

do

for j in `seq 0 $(($n-1))`

do

read x[$(($n\*$i+$j))]

done

done

echo "Enter the elements for second matrix"

for i in `seq 0 $(($m-1))`

do

for j in `seq 0 $(($n-1))`

do

read y[$(($n\*$i+$j))]

z[$(($n\*$i+$j))]=$((${x[$(($n\*$i+$j))]}+${y[$(($n\*$i+$j))]}))

done

done

echo "Matrix after addition is"

for i in `seq 0 $(($m-1))`

do

for j in `seq 0 $(($n-1))`

do

echo -ne "${z[$(($n\*$i+$j))]}\t"

done

echo -e "\n"

done

exit 0