Examples of small bash Scripts

**Here is a set of small scripts, which demonstrate some features of
bash programming.**

```
//=============================================================================
// set new prompt
//=============================================================================

PS1=">"
PS1="[${LOGNAME}@$(hostname)] # "
PS1="[${LOGNAME}] # "
PS1='$PWD $ '
```

```
//=============================================================================
// set (and automatically create) shell variable
//=============================================================================

$ test="test"

$ homedir='pwd'

string="The man said \" hello \"."
```

```
//=============================================================================
// To use the variable within the shell, it is preceded by a $
//=============================================================================
homedir=$HOME
cd $homedir
```

```
//=============================================================================
// how to print shell variable
//=============================================================================

echo PS1

echo $PS1

echo $USERNAME
```

```
//=============================================================================
// predefined shell variables
//=============================================================================
HOME    name of users login directory
IFS     internal field separators
PATH    search path used for finding commands
PS1     shell prompt
OSTYPE
USERNAME
SHELL
```

```
//=============================================================================
// The shell supports pattern matching
//=============================================================================
* Match all characters in a string
? Match a single character

ls *.dat
```

```
//=============================================================================
// Command Substitution
//=============================================================================
today=`date`
```

```
//=============================================================================
#-------------------------------------------------
# empty sh script program
#-------------------------------------------------
#!/bin/sh
```

```
#-------------------------------------------------
# comments
#-------------------------------------------------
#!/bin/sh
# this is comment
```

```
#-------------------------------------------------
# printing of string constant
#-------------------------------------------------
#!/bin/sh
echo 'hello'
echo "hello"
echo hello
```

```
#-------------------------------------------------
# declaration and printing of string variable
#-------------------------------------------------
#!/bin/sh
```

```
x='Wonderful new World'
echo $x
echo x # just string 'x'
```

```
#-------------------------------------------------
# call of other programs
#-------------------------------------------------
#!/bin/sh
ls
```

```
#-------------------------------------------------
# indirect call of other programs
#-------------------------------------------------
#!/bin/sh

x='pwd'
$x

x='ls -l'
$x
```

```
#-------------------------------------------------
# indirect call with indirect parameters
#-------------------------------------------------
#!/bin/sh
u='-l'

x='ls '

$x $u
```

```
#-------------------------------------------------
# print current shell name ???
#-------------------------------------------------
#!/bin/sh

echo $SHELL
```

```
#-------------------------------------------------
# Anything enclosed in double quotes is passed on exactly
# as presented with the exception that the values of
# shell variables are substituted
#-------------------------------------------------
#!/bin/sh

v1="abc "
v2="$v1 d"
```

```
echo $v1 $v2
```

```
#-------------------------------------------------
# Any matter enclosed in single quotes is passed on
# exactly as presented. The values of shell variables
# are not substituted.
#-------------------------------------------------
#!/bin/sh

v1="abc"
v2='$v1 d'
echo $v1 $v2
```

```
#-------------------------------------------------
# Back quotes are used to enclose commands. An item
# enclosed in back quotes is replaced by the standard
# output of the command. Shell variable values
# are substituted within back quotes.
#-------------------------------------------------
#!/bin/sh

date=`date`
echo the date is $date
```

```
#-------------------------------------------------
# escaping
#-------------------------------------------------
#!/bin/sh

msg=`echo Your Current Directory is \`pwd\``

echo $msg
```

```
#-------------------------------------------------
# reading of text line from keyboard
#-------------------------------------------------
#!/bin/sh

read x

echo $x
echo $x
```

```
#-------------------------------------------------
# reading of text line from keyboard with help comment
# echo without new line at the end
#-------------------------------------------------
#!/bin/sh
```

```
echo -n "Input text line=? "
read x

echo $x
echo $x
```

```
#-------------------------------------------------
# syntax:  many commands in one line !!!!
#-------------------------------------------------
#!/bin/sh

echo "a"; echo "b"; echo "c"

var=5; echo `expr $var + $var`
```

```
#-------------------------------------------------
# integer variable and its increment
# does not works in sh !!!
#-------------------------------------------------
#!/bin/bash

var=12345
let var=$var+1 # let is important
echo $var

v=12345
v=$v+1 # result "12345+1"
```

```
#-------------------------------------------------
# integer arithmetics - bash only !
#-------------------------------------------------
#!/bin/bash

echo 'number=?' ; read x

let y=$x+$x ; echo 'x+x=' $y

let y=$x*$x ; echo 'square=' $y

let y=$x/3 ; echo 'x/3=' $y

let y=$x%7 ; echo 'x%7=' $y
```

```
#-------------------------------------------------
# integer arithmetics in sh !!! using expr - slow
#-------------------------------------------------
#!/bin/sh

a=123
b=12
```

```
c=`expr $a + $b` # addition
echo $c

c=`expr $a \* $b` # multiplication
echo $c

c=`expr $a / $b` # division
echo $c

c=`expr $a % $b` # residual
echo $c
```

```
#-------------------------------------------------
# very simple 'if'
#-------------------------------------------------
#!/bin/sh

echo 'number=?'
read x

if [ $x -eq 5 ]
then
        echo "five"
fi
```

```
#-------------------------------------------------
# if ... else
#-------------------------------------------------
#!/bin/sh

echo 'number=?'
read x

if [ $x -eq 5 ]
then
        echo "five"
else
        echo "not 5"
fi
```

```
#-------------------------------------------------
# if ... elif ... else
#-------------------------------------------------
#!/bin/sh

echo 'number=?'
read x

if [ $x -eq 5 ]
then
        echo "five"
elif [ $x -eq 7 ]
then
```

```
        echo "seven"
else
        echo "not 5 and not 7"
fi
```

```
#-------------------------------------------------
# comparison -lt  and  -gt, nested if
#-------------------------------------------------
#!/bin/sh

echo -n 'number=?'
read x

if [ $x -gt 0 ]
then
        if [ $x -lt 10 ]
        then
                echo "0 < x < 10"
        fi
fi
```

```
#-------------------------------------------------
# while loop - print first 10 integers from 0
#-------------------------------------------------
#!/bin/bash

x=0

while  [ $x -lt 10 ]
do
        echo $x
        let x=$x+1
done
```

```
#-------------------------------------------------
# 10 random numbers generation
#-------------------------------------------------
#!/bin/bash

i=0
while [ $i -lt 10 ]
do
        x=$RANDOM
        echo $x
        let i=$i+1
done
```

```
#-------------------------------------------------
# endless loop: interrupting by ctrl-c
#-------------------------------------------------
#!/bin/sh
```

```
while [ 1 ]
do
        read x
        echo $x$x
done
```

```
#-------------------------------------------------
# divisors of integer number
#-------------------------------------------------
#!/bin/bash

echo -n 'number=?'
read x

i=2 # possible divisor
k=1
let n=$x/2 # top limit for divisor

while [ $i -le $n ]
do
        let k=$x%$i # residual
        if [ $k -eq 0 ]
        then
                echo -n "Divisor= "
                echo $i
        if
        let i=$i+1
done
```

```
#-------------------------------------------------
# simple use of  for ... in ...
#-------------------------------------------------
#!/bin/sh

for i in "abc" "xyz" 1 2 99
do
        echo $i
done
```

```
#-------------------------------------------------
# use for as in C-programming
# sum of the first n integer numbers
#-------------------------------------------------
#!/bin/bash

echo -n "number=?"
read n

s=0 # here sum

for((i=1; i <=n ; i++))
do
```

```
        let s=$s+$i
done

echo "sum= "$s
```

```
#-------------------------------------------------
# operator case for selection of logical branches
# end marker ;; of branch
#-------------------------------------------------
#!/bin/sh

echo "input string=?"
read str

case "$str" in
        abc) echo "string = abc"
                ;;
        xyz) echo "string = xyz"
                ;;
        *)   echo "not abc, not zyz" ;;
esac
```

```
#-------------------------------------------------
# exit operator
#-------------------------------------------------
#!/bin/sh

while [ 1 ]
do
        read x
        echo $x
        if [ $x -eq 0 ] # in $x must be number!
        then
                echo "script done ..."
                exit 0
        fi
done
```

```
#-------------------------------------------------
# string comparing
#-------------------------------------------------
#!/bin/sh

echo "Input string=?"
read str

if [ $str = "abc" ]
then
        echo "You got it!"
else
        echo "Its not 'abc'"
fi
```

```
#-------------------------------------------------
# simple strings concatenation
#-------------------------------------------------
#!/bin/sh

echo "Input string=?"
read str

s2=$str"AAAA"
echo $s2

s3="XXX"$s2
echo $s3
```

```
#-------------------------------------------------
# strings concatenation
#-------------------------------------------------
#!/bin/sh

echo "Input string=?"
read str1

echo "Input second string=?"
read str2

s3=$str1$str2 # it works!
echo $s3

s4=${str1}${str2} # it works too!
echo $s4
```

```
#-------------------------------------------------
# testing whether a string is null
#-------------------------------------------------
#!/bin/sh

echo "Input string=?"
read str

if [ $str ]
then
        echo "Not empty"
else
        echo "Empty"
fi
```

```
#-------------------------------------------------
# length of string
#-------------------------------------------------
#!/bin/sh

echo "Input string=?"
```

```
read str

leng=`expr length $str`
echo "length= "$leng
```

```
#-------------------------------------------------
# how to insert string to constant string
#-------------------------------------------------
#!/bin/sh

var="good"

echo "This is $var test"
```

```
#-------------------------------------------------
# simplest function example
#-------------------------------------------------
#!/bin/sh

#--------------------------
func()
{
        echo "Inside function"
}

#--------------------------
echo "Now function call..."
func
echo "end of main"
```

```
#-------------------------------------------------
# function can see variables of main program
#-------------------------------------------------
#!/bin/sh

#--------------------------
func()
{
        echo $var
}
#--------------------------

var="test of global "
func
```

```
#-------------------------------------------------
# pass of parameters to function
#-------------------------------------------------
#!/bin/sh

#--------------------------------
```

```
func()
{
        echo "We are in function now"
        echo $0  # shell script name
        echo $1  # first parameter
        echo $2  # second parameter
        echo "We leave function..."
        exit 0
}

#---------------------------------

func  123  "abc"
```

```
#-------------------------------------------------
# passing variable parameters
#-------------------------------------------------
#!/bin/bash

#------------------------------
func2()
{
        let r=$1*$1
        echo $r
}
#------------------------------
var=123
func2 $var
```

```
#-------------------------------------------------
# recursive function example
# calculation of factorial
#-------------------------------------------------
#!/bin/sh
#------------------------------
factorial()
{
        if [ "$1" -gt "1" ]
        then
                i=`expr $1 - 1`
                j=`factorial $i`
                k=`expr $1 \* $j`
                echo $k
        else
                echo 1
        fi
}
#------------------------------
read x
factorial $x
```

```
#-------------------------------------------------
# using of function library ????????
#-------------------------------------------------

file with name my.lb

func2()
{
        echo $1$1
}
func3()
{
        echo $1$1$1
}

shell program:

#!/bin/sh

./my.lb

var=123

func2 123

func3 123
```

```
#-------------------------------------------------
# floating point numbers
#-------------------------------------------------
#!/bin/sh

# does not support !!!!
```

```
#-------------------------------------------------
# simpiest array : declaration, element access and assignment
#-------------------------------------------------
#!/bin/bash

arr=(aa bb cc dd)

echo ${arr[0]} # curly bracket notation
echo ${arr[1]}
echo ${arr[2]}
echo ${arr[3]}

arr[2]="CCCCCCC"
echo ${arr[2]}
```

```
#-------------------------------------------------
# number of elements in array
#-------------------------------------------------
```

```
#!/bin/bash

arr=(aa bb cc dd)

n=${#arr[@]}
echo $n
```

```
#-------------------------------------------------
# array with filenames of current directory
#-------------------------------------------------
#!/bin/sh

arr=(*)  # * is list of all file and dir names

n=${#arr[@]}
echo "number of files and dirs "$n

echo ${arr[0]}
echo ${arr[1]}
```

```
#-------------------------------------------------
# print all array elements -not good
# works then no holes in indexes
#-------------------------------------------------
#!/bin/bash

arr=(aa bb cc dd ee ff gg)
n=${#arr[@]}
i=0
while test $i -lt $n
do
        echo ${arr[$i]}
        let i=$i+1
done
```

```
#-------------------------------------------------
# dynamic expansion of array
# one array element in reality is couple (index, value)
#-------------------------------------------------
#!/bin/bash

arr=()

n=${#arr[@]}
echo "number of array elements "$n

arr[0]=a
n=${#arr[@]}
echo "number of array elements "$n

arr[1]=b
n=${#arr[@]}
echo "number of array elements "$n
```

```
arr[2]=c
n=${#arr[@]}
echo "number of array elements "$n

arr[10]=h
n=${#arr[@]}
echo "number of array elements "$n


echo ${arr[10]}

echo ${arr[4]} # empty string
echo ${arr[6]} # empty string
```

```
#-------------------------------------------------
# get all array and print it
#-------------------------------------------------
#!/bin/bash

arr=(aa bb cc dd ee ff gg)

echo ${arr[*]} # all array

echo ${arr[@]:0}   # aa bb cc dd ee ff gg

echo ${arr[@]:1}   # bb cc dd ee ff gg

echo ${arr[@]:2:3} # cc dd ee

for i in ${arr[*]}
do
        echo $i
done
```

```
#-------------------------------------------------
# adding element to array
#-------------------------------------------------
#!/bin/bash

arr=(aa bb cc dd ee ff gg)

echo ${arr[*]}

arr=( "${arr[@]}" "newElem" ) # from right

echo ${arr[*]}

arr=( "newElem" "${arr[@]}" ) # from left

echo ${arr[*]}
```

```
#-------------------------------------------------
```

```
# move last element from array
#-------------------------------------------------
#!/bin/bash


arr=(aa bb cc dd ee ff gg)

echo ${arr[*]}

unset arr[${#arr[@]}-1] #  move last element
echo ${arr[*]}
```

```
#-------------------------------------------------
# copying of array
#-------------------------------------------------
#!/bin/bash

arr=(aa bb cc dd ee ff gg)

echo ${arr[*]}

arr2=( "${arr[@]}" )

echo ${arr2[*]}
```

```
#-------------------------------------------------
# get substring from string
#-------------------------------------------------
#!/bin/bash

echo "long string input=?"
read st

st2=${st:2:4}

echo $st2
```

```
#-------------------------------------------------
# substring replacement "abc" to "xyz"
#-------------------------------------------------
#!/bin/bash

echo "string input=?"
read str

st2=${str/abc/xyz} # only ones

echo $st2
```

```
#-------------------------------------------------
# search of character 'a' in a string
```

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

echo "string input=?"
read str

pos=`expr index $str a`

echo "position of the first 'a' = "$pos
```

```
#-------------------------------------------------
# string list counting
#-------------------------------------------------
#!/bin/sh

for i in aa bb cc dd ee ff gg hh
do
        echo $i
done
```

```
#-------------------------------------------------
# command line arguments
# separated by spaces
#-------------------------------------------------
#!/bin/sh

echo $0  # script file name

echo $1  # first argument
echo $2  # second argument
echo $3  # third argument
```

```
#-------------------------------------------------
# command line arguments without script name number
# all command line without script name
#-------------------------------------------------
#!/bin/sh

echo $#  # argument number

echo $*  # command line

echo $@
```

```
#-------------------------------------------------
# get all files and dir names
#-------------------------------------------------
#!/bin/sh

echo *  # file and dir names of current dir
```

```
for i in *
do
        echo $i
done

echo ../* # file and dir names of parent dir

*/ just close comments
```

```
#-------------------------------------------------
# file search from root dir      ???????
# file name - parameter from command line
#-------------------------------------------------
#!/bin/sh

start=$HOME
date
find $start -name $1 -print
```

```
#-------------------------------------------------
# list of all files with extension .txt     !!!!!!
#-------------------------------------------------
#!/bin/sh

echo *.txt
```

```
#-------------------------------------------------
# combine a set of text files in one file use
# script >targetfile.lst , not txt-file !!!
#-------------------------------------------------
#!/bin/sh

lst=*.txt

for i in $lst
do
        echo
        echo "======================================"
        echo "File "$i
        echo "======================================"
        cat <$i
done
```

```
#-------------------------------------------------
# create new file and write string to it
# file name from command string - variable $1
#-------------------------------------------------
#!/bin/sh

echo "String=?"
```

```
read str

echo $str >$1
```

```
#-------------------------------------------------
# read textlines from console and add them to file
# file name from command string - variable $1
#-------------------------------------------------
#!/bin/sh

echo "Add strings=?"
str="1"

while [ $str ]
do
        read str
        echo $str >>$1
done
```

```
#-------------------------------------------------
# read first string from text file
#-------------------------------------------------
#!/bin/sh

read str <$1
echo $str
```

```
#-------------------------------------------------
# text file reading
# script res.txt
#-------------------------------------------------
#!/bin/sh

str="1"

while [ $str ]
do
        read str
        echo $str
        echo $str
done
```
