Shell Programming

- *String Test*

```
if test “$s1” = “$s2”
  then
    echo “identical string”
fi
```

#must have a blank space right before and after =
# equivalent to if [“$s1” = “$s2”];
a blank space is needed after [and before].

#operators: ***=, !=, >, <***
< and > can be used if [[ ]] is included, i.e., if [[“$s1” > “$s2”]].

- *Numerical Test*

```
if test $# -eq 0
  then
    echo “must provide an argument”
    exit 1
fi
```

operators: *-gt, –lt, -ne, -eq, -ge, -le*

#must have a blank space right before and after -eq
-eq: equal to =
-gt: greater >
-lt: less <
-ne: not equal to !=
-ge: greater or equal to >=
-le: less or equal to <=

>, >=, <, <= can be used if (( )) is included, i.e., if (($v1 > $v2))

- *Special Test*

```
if test -f “$1”        # if [ -f “$1” ]
  then
    echo “$1 is a file”
  else
    echo “$1 is not a file”
fi
```

others: *-d, -e, -r, -s, -w, -x*

-f: is a file?
-d: is a directory?
-e: the file exists?
-r: the file has read permission?
-w: the file has write permission?
-x: the file has execute permission?

Also: if [f1 -nt f2]   -ot and -ef
f1 -nt f2: f1 is newer than f2?
f1 -ot f2: f1 is older than f2?
f1 -ef f2: f1 and f2 are hard linked to the same file?

- *Nested if*

```
if test-command
  then
    commands
elif test-command
  then
    commands
else
    commands
fi
```

- *Logical operators*

```
if [ “$s1” = “$s2” -a “$s2” = “$s3” ]
  then
    echo “match”
  else
    echo “s1, s2, & s3 are not all the same”
fi
```

operators: *-a, -o, !*
-a: logical and, equivalent to &&
-o: logical or, equivalent to ||
!: logical negation #if [! $a = 1]

[[ ]] should be included if &&, || are used, i.e., *if [[“$s1” = “$s2” && “$s2” = “$s3”]]*

- *For-in loop*

```
for letter in a b c d
do
      echo $letter
done
#equivalent to for letter in {a..d}

for i in * #files and directories in current path
do
      if [ -d “$i” ]
      then
            echo “$i”
      fi
done
# also a path can be included after “for i in”,
e.g., for i in ../*   or   for i in dir1/*

for i in L*    #files and directories in current
path starting with ‘L’; also supports usage
of ‘?’, e.g, for i in L?.txt

for i            #from command line arguments
do
      echo “$i”
done

for i in {1..10..2}
# step is 2; 1 3 5 7 9 are printed

for i in {a..g..2}          # a c e g are printed
```

New version of Linux Shell also supports for loop in C-Style:

```
for (( i=0; i<5; i=i+1 ))
do
  echo $i
done
```

- *While loop*

```
num=0
while [ $num –lt 5 ]
do
      echo –n “$num ”
      let num=$num+1
done

ls *.c > list
while read f
do
      echo $f
done < list
# read each line in the file list

ls *.c | while read f; do echo $f; done < list
```

- Arithmetic Expansion

```
x=1                          ((x=x+1))
echo $((x+1))                echo $x

echo $[ $(ls –l | wc –l) – 1 ]

let x=x+1
((x++))

operators: +, -, ++, --, *, /, %
```

- Until loop

```
count=10
until test $count –eq 0
do
      echo $count
      let count=$count-1
done
#execute the loop if the condition is false
```

- Break and Continue

```
for index in 1 2 3 4 5 6 7 8 9 10
   do
      if [ $index –le 3 ] ; then
              echo “continue”
              continue
      fi
      echo $index
      if [ $index –ge 8 ] ; then
              echo “break”
              break
      fi
done
```

- The Case Structure

```
set $(date)
case $1 in
     Fri )            echo “It’s Friday”;;
     Sat | Sun )      echo “it’s weekend!”;;
     *)               echo “a weekday”;;
esac

echo –n “Enter A or B: “
read letter
case “$letter” in
     a|A)
             echo “You entered A”
             ;;    #similar to “break” in C
     b|B)
             echo “You entered B”
             ;;
     *)      #similar to “default” in C
             “Wrong input”
             ;;
esac


echo “0: Exit”
echo “1: Show Date and Time”
echo “2: List my HOME directory
echo “3: Display Calendar”
echo “Enter your choice: “
read option
case $option in
     0)  echo good bye ;;
     1)  date ;;
     2)  ls $HOME ;;        #same as “ls ~”
     3)  cal ;;  #display the calendar
     *)  echo “Invalid input. Good bye.” ;;
esac


hour=$(date +%H) #get hour
case $hour in
     0? | 1[0-1] ) echo “Good Morning!” ;;
     1[2-7]  ) echo “Good Afternoon!” ;;
      *  ) echo “Good Evening!” ;;
esac
```

- Function

```
func1 ()
{
    echo “hello”
}
func1         $call func1()

func2 ()
{
        echo $1 $2      #display arguments
        return $(($1+$2))
}
func2 10 20       #passing arguments
total=$?  #return value of func2()
echo $total
```

- Array

```
list[0]="abc"
list[1]="def"
list[2]="xyz"
echo ${list[0]}    #abc
echo ${list[*]}   #abc def xyz
echo ${#list[*]} #3

scores=(75 92 87 95)
echo ${scores[*]} #75 92 87 95
echo ${#scores[*]} #4, # of elements
scores[0]=80
echo ${scores[*]} #80 92 87 95

for i in {0..50..10}
do
    array[i]=$i
    echo ${array[i]}
done
```