Arrays

Method of creating an array variable

array_name[index]=value
i.e.

```
NAME[0]="stu1"
NAME[1]="stu2"
NAME[2]="stu3"
NAME[3]="stu4"
NAME[4]="stu5"
```

Syntax of array initialization

array_name=(value1 ... valuen)

Accessing Array Values

${array_name[index]}
i.e.

```
#!/bin/sh

NAME[0]="stu1"
NAME[1]="stu2"
NAME[2]="stu3"
NAME[3]="stu4"
NAME[4]="stu5"
echo "First Index: ${NAME[0]}"
echo "Second Index: ${NAME[1]}"
```

Access all the items in an array

${array_name[*]}
${array_name[@]}

i.e.

```
#!/bin/sh

NAME[0]="stu1"
NAME[1]="stu2"
NAME[2]="stu3"
NAME[3]="stu4"
```

```
NAME[4]="stu5"
echo "First Method: ${NAME[*]}"
echo "Second Method: ${NAME[@]}"
```

Operators

Assume Values of given variable

a=10
b=20

Arithmetic Operators

Operator	Description	Example
+ (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)	Assigns 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.

i.e.

```
#!/bin/sh

a=10
b=20
```

```
val=`expr $a + $b`
echo "a + b : $val"

val=`expr $a - $b`
echo "a - b : $val"
```

```
val=`expr $a \* $b`
echo "a * b : $val"

val=`expr $b / $a`
echo "b / a : $val"

val=`expr $b % $a`
echo "b % a : $val"

if [ $a == $b ]
then
   echo "a is equal to b"
fi

if [ $a != $b ]
then
   echo "a is not equal to b"
fi
```

Relational Operators

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

i.e.

```
#!/bin/sh

a=10
b=20

if [ $a -eq $b ]
then
   echo "$a -eq $b : a is equal to b"
else
   echo "$a -eq $b: a is not equal to b"
fi

if [ $a -ne $b ]
then
   echo "$a -ne $b: a is not equal to b"
else
   echo "$a -ne $b : a is equal to b"
fi
```

```
if [ $a -gt $b ]
then
   echo "$a -gt $b: a is greater than b"
else
   echo "$a -gt $b: a is not greater than b"
fi

if [ $a -lt $b ]
then
   echo "$a -lt $b: a is less than b"
else
   echo "$a -lt $b: a is not less than b"
fi

if [ $a -ge $b ]
then
   echo "$a -ge $b: a is greater or  equal to b"
else
   echo "$a -ge $b: a is not greater or equal to b"
fi

if [ $a -le $b ]
then
   echo "$a -le $b: a is less or  equal to b"
else
   echo "$a -le $b: a is not less or equal to b"
fi
```

String Operators

Operator	Description	Example
=	Checks if the value of two operands are equal or not; if yes, then the condition becomes true.	[$a = $b] is not true.
!=	Checks if the value of two operands are equal or not; if values are not equal then the condition becomes true.	[$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 nonzero length, then it returns true.	[-n $a] is not false.
str	Checks if **str** is not the empty string; if it is empty, then it returns false.	[$a] is not false.

i.e.

```
#!/bin/sh

a="abc"
b="efg"

if [ $a = $b ]
then
   echo "$a = $b : a is equal to b"
else
   echo "$a = $b: a is not equal to b"
fi

if [ $a != $b ]
then
   echo "$a != $b : a is not equal to b"
else
   echo "$a != $b: a is equal to b"
fi

if [ -z $a ]
then
   echo "-z $a : string length is zero"
else
   echo "-z $a : string length is not zero"
fi
```

```
if [ -n $a ]
then
   echo "-n $a : string length is not zero"
else
   echo "-n $a : string length is zero"
fi

if [ $a ]
then
   echo "$a : string is not empty"
else
   echo "$a : string is empty"
fi
```

Decision Making

- **if...fi statement**

  Syntax

  ```
  if [ expression ]
  then
     Statement
  fi
  ```

- **if...else...fi statement**

  Syntax

  ```
  if [ expression ]
  then
     Statement
  else
     Statement
  fi
  ```

- **if...elif...else...fi statement**

  Syntax

  ```
  if [ expression 1 ]
  then
     Statement
  elif [ expression 2 ]
  then
     Statement
  elif [ expression 3 ]
  ```

```
then
  Statement
else
  Statement
fi
```

- **case...esac statement**
  Syntax

```
case word in
  pattern1)
    Statement
    ;;
  pattern2)
    Statement
    ;;
  pattern3)
    Statement
    ;;
  *)
    Default condition to be executed
    ;;
esac
```

i.e.

```
#!/bin/sh

BRANCH="cs"
#or
#echo “Enter Your Branch name(cba/bda/cs)”
#read BRANCH

case "$BRANCH" in
  "cba") echo "cba student."
  ;;
  "bda") echo "bda student."
  ;;
  "cs") echo "cs student."
  ;;
esac
```