Select command

- Constructs simple menu from word list
- Allows user to enter a number instead of a word
- User enters sequence number corresponding to the word

Syntax:

```
select WORD in LIST
do
    RESPECTIVE-COMMANDS
done
```

- Loops until end of input, i.e. ^d (or ^c)

SELECT EXAMPLE

```
#! /bin/bash
select var in alpha beta gamma
do
     echo $var
done
```

- Prints:

```
1) alpha
2) beta
3) gamma
#? 2
beta
#? 4
#? 1
alpha
```

Select detail

- PS3 is select sub-prompt
- $REPLY is user input (the number)

```
#! /bin/bash
PS3="select entry or ^D: "
select var in alpha beta
do
     echo "$REPLY = $var"
done
```

<u>Output</u>:

```
select ...
1) alpha
2) beta
? 2
2 = beta
? 1
1 = alpha
```

Select example

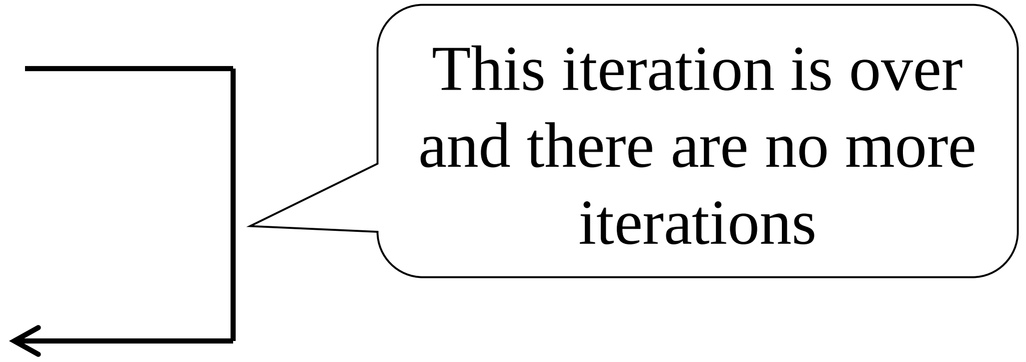
```
#!/bin/bash
echo "script to make files private"
echo "Select file to protect:"
select FILENAME in *
do
  echo "You picked $FILENAME ($REPLY)"
  chmod go-rwx "$FILENAME"
  echo "it is now private"
done
```

BREAK AND CONTINUE

- **Interrupt for, while or until loop**
- **The break statement**
  - **transfer control to the statement AFTER the done statement**
  - **terminate execution of the loop**
- **The continue statement**
  - **transfer control to the statement TO the done statement**
  - **skip the test statements for the current iteration**
  - **continues execution of the loop**

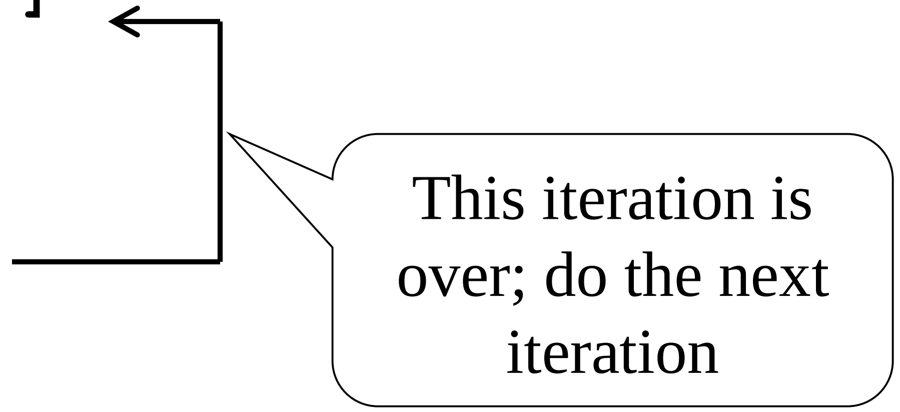
THE BREAK COMMAND

```
while [ condition ]
do
    cmd-1
    break
    cmd-n
done
echo "done"
```



The continue command

```
while [ condition ]
do
    cmd-1
    continue
    cmd-n
done
echo "done"
```



Example:

```
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
```

ARRAY VARIABLE

- This can hold multiple values at the same time.
- Arrays provide a method of grouping a set of variables.
- syntax of array initialization
- Array=(va1 va2 va3)
- echo “first value=${Array[0]}”

Syntax	Result
arr=()	Create an empty array
arr=(1 2 3)	Initialize array
${arr[2]}	Retrieve third element
${arr[@]}	Retrieve all elements
${!arr[@]}	Retrieve array indices
${#arr[@]}	Calculate array size
arr[0]=3	Overwrite 1st element
arr+=(4)	Append value(s)
str=$(ls)	Save ls output as a string
arr=($(ls))	Save ls output as an array of files

```
#!/bin/bash
 #Declare a string array

Array=("PHP"  "Java"  "C#"  "C++"  "VB.Net"  "Python"  "Perl")

# Print array values in  lines
echo "Print every element in new line"

for val1 in ${Array[*]}; do
     echo $val1
done

echo ""

# Print array values in one line
echo "Print all elements in a single line"
for val2 in "${Array[*]}"; do
    echo $val2
done
echo ""
```

SORT

- **sort – to sort contents of a text file**
- Usage:

  sort [options] *file name*

- Eg:-

```
# sort iacsd

# sort –r iacsd   for reverse sorting
```

-k - sort by column

-o - to redirect o/p to other file

-u –   sort & remove duplicates

locate

•**locate – used for file searching in linux**

Search in database.

•Usage:

locate [options] *file name*

•Eg:-

```
# locate passwd
# locate –r /passwd$        to find with exact filename
```

-e – print only file which are present

updatedb – to update database

Find

Find - is used to search and locate the list of files and directories based on conditions you specify for files that match the arguments.

Find can be used in a variety of conditions like you can find files by permissions, users, groups, file type, date, size, and other possible criteria.

Syntax :

$ find [where to start searching from] [expression determines what to find] [-options] [what to find]

1. Find Files Using Name in Current Directory

```
# find  . -name dbda.txt

./dbda.txt
```

2. Find Files Under Home Directory

```
# find  /home  -name dbda.txt

/home/dbda.txt
```

3. Find Files Using Name and Ignoring Case

```
# find  /home  -iname  dbda.txt

./dbda.txt
./Dbda.txt
```

4. Find Directories Using Name

```
# find  / -type  d    -name  dbda

/dbda
```

5. Find PHP Files Using Name

```
# find  .  -type  f  -name  dbda.php

./dbda.php
```

6. Find all PHP Files in Directory

```
# find  .  -type  f  -name  "*.php"

./tech.php
./login.php
./index.php
```

7. Find Files With 777 Permissions

```
# find  .  -type  f  -perm  777
```

8. Find Files Without 777 Permissions

```
# find / -type f ! -perm 777
```