System Administration

Session 7 CONTENT

- Case
- Select
- Loops
- Shift
- Break
- Continue
- Arrays

The case command

```
case variable in
value1)
            Command(s)
            ;;
value2)
            Command(s)
            ;;
*)
            Command(s)
            ;;
esac
```

Sub Patterns

- ?(pattern(s))
  - Match one or zero occurrence of any of the patterns
- *(pattern(s))
  - Match zero or more occurrence of any of the patterns
- @(pattern(s))
  - Match exactly one occurrence of any of the patterns
- +(pattern(s))
  - Match one or more occurrence of any of the patterns
- !(patten(s))
  - Match all strings except any of the patterns

Example

```
case $var  in
@([a-z]) )    echo     " lower case "
      ;;
@([A-Z]) )   echo    " upper case "
                  ;;
@([0-9]) )    echo     " integer "
      ;;
esac
```

The `while` command

```
while command
do
   … command …
done
```

- Examples

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

Guess Game Activity

Write a bash script game that asks the executer to guess a secret name and exit only when he gets the secret correctly..

The until command

```
until command
do
    command(s)
done
```

Example

```
hour=1
until [ $hour -gt 24 ]
do
  case $hour in
    [0-9] | 1[0-1]) echo good morning ;;
    12) echo lunch time ;;
    1[3-7]) echo work time ;;
    *) echo Good Night ;;
  esac
  let hour=$hour+1
done
```

The `for` command

- It is used to execute commands a finite number of times on a list of items (files/usernames)

```
for variable in word list
do
  … commands …
done
```

The `for` command

Example:

```
for pal in mona ahmed maha
do
   echo hi $pal
done
```

Example:

```
for person in `cat mylist`
do
   mailx $person < letter
   echo mail to $person was sent
done
```

The `select` command and Menus

- The `select` loop is an easy way for creating menus.
- The input should be one of the numbers in the menu list.
- The input is stored in the special bash shell `RELPY` variable.
- The `case` command is used with the `select` command to make it possible for the user to make a select from the menu.

Examples

```
select choice in Ahmed Adel Tamer
do
   case $choice in
      Ahmed) print Ahmed is good boy
        ;;
      Adel) print Adel is the best
        ;;
      Tamer) print Tamer is a bad boy
        ;;
      *) print $REPLY is not one of the choices.
        ;;
   esac
done
```

Examples

```
select choice in Ahmed Adel Tamer
do
   case $REPLY in
      1) print Ahmed is good boy
         break;;
      2) print Adel is the best
         break;;
      3) print Tamer is a bad boy
         break;;
      *) print $REPLY is not one of the choices.
         print Try again
              ;;
   esac
done
```

The break command

- The break command is used to force immediate exit from the loop, but not from the program.

- Example

```
while true
do
    echo "Are you ready to move on?"
    read answer
    if [[ $answer = [Yy]* ]]  (the new test command to evaluate wild cards)
        then
          break
        else
            echo type Y - y or yes when you are!
    fi
done
print "Here are you?"
```

The `continue` command

- The `continue` command is used to starts back at the top of the loop

- Example

```
#! /bin/bash
for name in `cat names`
do
    if [ $name = naggar ]
    then
      continue
    else
      echo $name
    fi
done
```

Example for Nested Loops

```
#!/bin/ksh
while true
do
   for user in Ahmed Tamer Samy
   do
      if [[ $user = [Tt]* ]]
      then
                print A Hi from Tamer
                continue
      fi
      while true
      do
                 if [[ $user = [S]* ]]
                then
                            print A Hi from Samy
                            break 3
                fi
                print A Hi from Ahmed
                continue 2
                done
   done
done
print Out of the Loop
```

Arrays

- Bash supports one-dimensional numerically indexed and associative arrays types. Numerical arrays are referenced using integers, and associative are referenced using strings.
- Index starts with zero.
- Each element can be set and unset individual.
- Values do not have to be set in any particular order.
- Bash does not support multidimensional arrays, and you can't have array elements that are also arrays.

Examples

- To set the value of array element

```
array[0]=ahmed
array[1]=ali
array_name=( element_1 element_2 element_N )
```

- To add new elements you can assign by new index or

```
myArray+=( "newElement1" "newElement2" )
```

- To delete a single element

```
unset my_array[index]
```

- To print the values of the array elements

```
echo ${array[index]}
```

Examples

- To display all the elements in the array

```
echo ${ele[*]}
echo ${ele[@]}
```

- To loop over an array

```
for i in "${my_array[@]}"
do
  echo "$i"
done
```

- To display the number of elements in the array

```
echo ${#ele[@]}
```

Questions?!

Thank YOU!