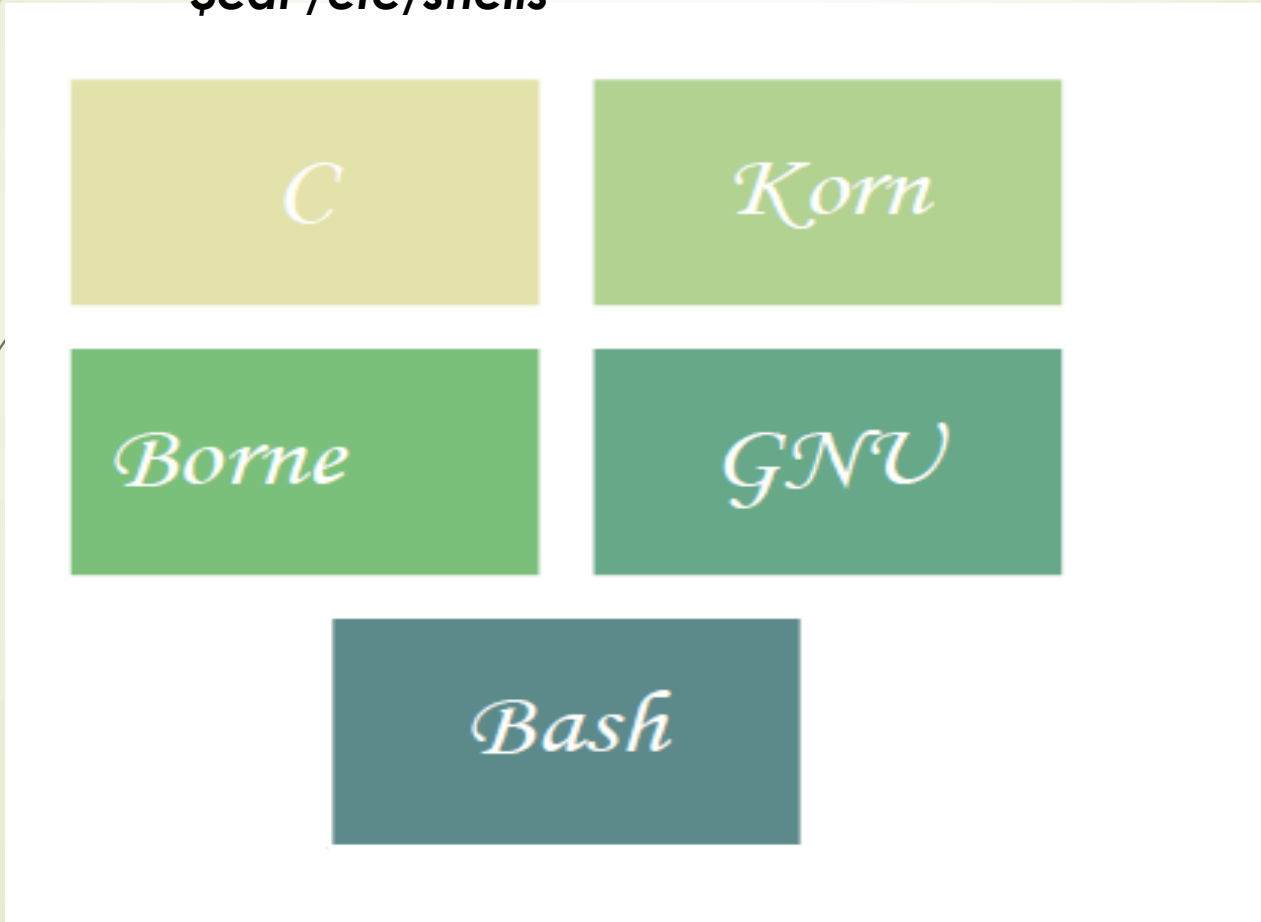
Shell Programming

What is Shell Scripting?

- *Script written for a shell or command line interpreter of an OS*
- *Different Command Line Interfaces - Unix Shell, Windows PowerShell , MS-DOS command.com*
- *Whatever that can be entered at the shell can be grouped together and written in a shell script like batch jobs*
- *Different Unix shells – Bourne (known as sh; Stefen Bourne AT&T labs), bash (Bourne again), csh (Bill Joy @ Berkley), tcsh, ksh (David Korn)*

Shells

Know your shell
$ echo $SHELL
$cat /etc/shells



Shell Script

- vi hello.sh

```
#!/bin/bash
#Above line is called sheline
echo "Let's say hello from shell script"
echo "Some variables in shell"
course="DAC Aug 2015"
module="Linux and OS"
echo "Wlecome to $module module of $course course"
echo "Exiting ..."
exit 0
```

- Execution

```
chmod +x hello.sh
./hello.sh
```

OR

```
bash hello.sh
```

Comments and Variables

- *Comments in Shell start with # and goes until end of line*
- *Assign a variable*
  - *variable=value*
- *Accessing a variable*
  - *$variable*
- *To separate variable from attached text use {}*
  - *Num=13*
  - *echo "It's ${Num}th April today"*

Variables

- *Must begin with an alphabet or underscore*
  - *Can contain (a to z, A to Z, 0 to 9 and _);*
- *Case sensitive*
  - *no and No are different*
- *Do not use ? , * or some punctuation marks in variable names*

Types of variables

- *Local Variables - variables that are present in the current instance of a shell. These will not be available to the programs started by the shell, if they are declared on the command prompt.*

- *Environement Variables - Environement variables are available to its child processes as well.*

- *Shell Variables - Special variables that are set by the shell and are required for proper functioning of the shell.*

Variables

Variable	Meaning
$0	File name of the current script
$n	Here n is an integer which corresponds to the position of an argument at command line
$#	No of arguments supplied to the script
$?	Exit status of the last command executed
$$	Process number of the current shell
$!	

Example

```
#!/bin/bash
echo "this is my script ****$0***"
echo "process number of shell is $$"
echo "My first name is $1"
echo "My surname is $2"
echo "my full name is $1 $2"
echo "total number of argument supplied $#"
```

Variables ..cont

- *All the global environment variables (ENV) can be viewed using*
- *$ printenv*
- *Displays global as well as local environment variables*
- *$ set*
- *Displays all global environment variables*
- *$ env*

Variable ..cont

- *User wide ENVs are set and configured in*
  - *~/.bashrc*
  - *~/.bash_profile*
  - *~/.bash_login*
  - *~/.profile*
- *System wide ENVs can be configured in*
  - */etc/environment*
  - */etc/profile*
  - */etc/profile.d/*
  - */etc/bash.bashrc*

Commonly used Environment variables

- *$ USER - Gives current User's name*
- *$ PATH –provides the list of search path*
- *$ PWD – current working directory*
- *$ HOME – path of home directory*
- *$ HOSTNAME – gives the name of the host*
- *$ LANG – language of the editor*
- *$ EDITOR – name of the default file editor*
- *$ UID – User ID*
- *$ SHELL – current shell*

Separator	Meaning
-lt	Less than
-gt	Greater than
-le	Less than or equal to
-ge	Greater than or equal to
-e or =	Equal to
-ne or !=	Not equal to

Comparator	Meaning
=	Equal to
!=	Not equal to
<	Sort string in ascending
>	Sort string in descending

Arithmetic Evaluation in Shell

```
#!/bin/bash
a=10;b=20;c=30;d=40;
addition=`expr $a + $b`   # $ is compulsory for variables, should have space around
                          # operator(+), no space around =, need to escape * operator
let sum=a+b   # No space around operator or =, $ is not mandatory before var name
total=$((a+b))      # $ before var name is not mandatory, no space around = and +
# Increment options (Make sure count is already initialized with some value)
count=`expr $count + 1`
let count=count+1
((count++))
```

Exit Status

<command invocation>

- echo $? ## $? contains exit status of last executed command

- Example

  grep “Sachin Tendulkar” dac-aug-2015-list.txt

  echo “$?”

  - 0 if pattern found else non zero

- Wrong Usage

  grep “Sachin Tendulkar” dac-aug-2015-list.txt

  echo “Performed pattern search using grep now checking exit status”

  echo “$?”

Conditional operators

- &&

  <command1> && <command2>

  - If execution of command1 is successful (0 exit status)

then command2 is executed else command2 will not be executed

- ||

  <command1> || <command2>

  - If execution of command1 is unsuccessful(non 0 exit status then

  command2 is executed else command2 will not be executed

- && and || can be combined, but be careful!

Example of conditional Operators

- conditional.sh

```
#!/bin/bash

grep "Sachin Tendulkar" e-dac2020-list.txt || echo "Pattern not found or file not present"

grep "Sunil Gavaskar" e-dac2020-list.txt && echo "pattern found"
```

- Execution

```
chmod +x conditional.sh

./conditional.sh
```

If statement

- *if …..; then*

  elseif….;then

  …..

  else

  …..

  fi
- *Conditions to be tested inside if are written in [ ]*

Example of If-else

```
if [ "$SHELL" = "/bin/bash" ]; then
echo "your login shell is the bash (bourne again
    shell)"
else
echo "your login shell is not bash but $SHELL"
fi
```

While Loop

while [test condition]

do

…

….

done

Example

```
flag=true
while [ "$flag" = true ]
do
    read choice
    echo $choice
    if [ "$choice" = 'Y' ]; then
    flag=true
    echo "continuing"
    else
    flag=false
    fi
Done
```

Case Statement

case … in

…) action on the match;;

esac

Case Statement

```
echo "Enter a number
between 1 and 10. "
read NUM
case $NUM in
1) echo "one" ;;
2) echo "two" ;;
3) echo "three" ;;
4) echo "four" ;;
5) echo "five" ;;
6) echo "six" ;;
7) echo "seven" ;;
8) echo "eight" ;;
9) echo "nine" ;;
10) echo "ten" ;;
*) echo "INVALID
NUMBER!" ;;
esac
```

For Loop

for i in {0..4} or for i in some range

Do

…

…

done

- *Conditional exit in for loop - break*

Example for loop

```
#method 1
for i in {0..10}
do
echo $i
done
```

```
#method 2
for i in {0..10..3}
do
echo $i
done
```

```
#Method 3
for (( i=1; $i<=5; i++ ))
do
echo $i
done
```

Function

function name( )

{…

}

- *Calling a function –*
  - *use only functionname*
- *Local variables – declare using "local" keyword*

Function Example

```
#Function Declaration
display()
{
local local_var=100
global_var=blessen
echo "local variable is
    $local_var"
echo "global variable is
$global_var"
}
```

```
#Function call
echo"====================="
display
echo"=======outside========"
echo "local variable outside
function is $local_var"
echo "global variable outside
function is $global_var"
```