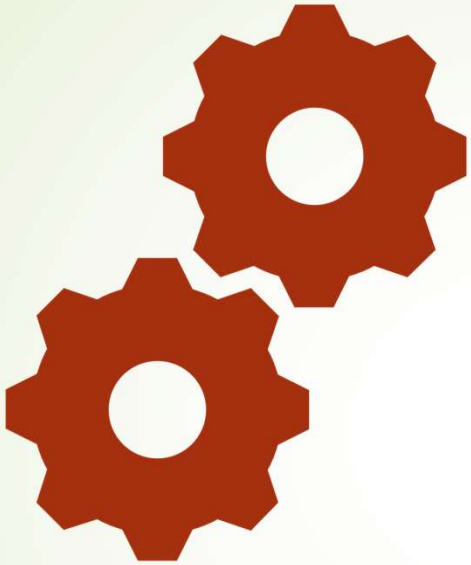

Shell Scripting

Shell Scripting Basics

- A shell is a command-line interpreter and typical operations performed by shell scripts include file manipulation, program execution, and printing text.
- Shell Script is a text file that contains list of commands placed in order which user wants the shell to execute to perform a task.
- Example
  - echo "What is your name?"

    read PERSON

    echo "Hello, $PERSON“

    What is your name?

    Ram Kumar

    Hello, Ram Kumar
  - echo "Hello World"

    date

    Hello World
    Wed Apr 23 15:23:40 IST 2025

Scripting Vs High level languages

- A scripting language is used to communicate directly with the shell environment in UNIX
- It doesn't require any compilation
- Shell interprets each command as it is reached according to the control structures used
- Example for difference
  - echo "Hello World"

    date

    Hello World
    Wed Apr 23 15:23:40 IST 2025
  - import java.util.Date;

    class Hello {

    public static void main (String [] args) {

    System.out.println("Hello World\n" + new Date());

    }

    }

Scripting - Variables

- Variable name can contain only letters (a to z or A to Z), numbers (0 to 9) or the underscore character (_)
  - _VAR
  - TOKEN_A
  - VAR_1
  - VAR_2
- Variable Definition
  - variable_name=variable_value
  - VAR1=“100”
  - VAR_2=“Ken”
- Read-Only Variable
  - readonly Variable_Name (readonly VAR1)
- unset VAR1 – directs the shell to delete the variable

Special Variables

Variable	Description
$0	The filename of the current script.
$n	These variables correspond to the arguments with which a script was invoked. Here **n** is a positive decimal number corresponding to the position of an argument (the first argument is $1, the second argument is $2, and so on).
$#	The number of arguments supplied to a script.
$*	All the arguments are double quoted. If a script receives two arguments, $* is equivalent to $1 $2.
$@	All the arguments are individually double quoted. If a script receives two arguments, $@ is equivalent to $1 $2.
$?	The exit status of the last command executed.
$$	The process number of the current shell. For shell scripts, this is the process ID under which they are executing.
$!	The process number of the last background command.

Scripting - Array

- Defining an array
  - array_name[index]=value
    - NAME[0]=“Shanjay"
    - NAME[1]=“Ayan"
    - NAME[2]=“Karthika"
- Accessing Array Values
  - ${array_name[index]}
  - NAME[0]=" Shanjay"

    NAME[1]=" Ayan"

    NAME[2]=“Karthika"

    echo "First Index: ${NAME[0]}"

    echo "Second Index: ${NAME[1]}“
  - Output:
    - First Index: Shanjay
    - Second Index: Ayan

Scripting – Arithmetic Operators

Operator	Description	VAR_A	VAR_B	Result
+ (Addition)	Adds values on either side of the operator	30	20	50
- (Subtraction)	Subtracts right hand operand from left hand operand	30	20	10
* (Multiplication)	Multiplies values on either side of the operator	30	20	600
/ (Division)	Divides left hand operand by right hand operand	30	20	1.5
% (Modulus)	Divides left hand operand by right hand operand and returns remainder	30	20	10
= (Assignment)	Assigns right operand in left operand	30	20	-
== (Equality)	Compares two numbers, if both are same then returns true.	30	20	FALSE
!= (Not Equality)	Compares two numbers, if both are different then returns true.	30	20	TRUE

Author – Sri Ram Kumar J (123159)

Scripting – Relational Operators

Operator	Description
-eq	Checks if the value of two operands are equal or not; if yes, then the condition becomes true.
-ne	Checks if the value of two operands are equal or not; if values are not equal, then the condition becomes true.
-gt	Checks if the value of left operand is greater than the value of right operand; if yes, then the condition becomes true.
-lt	Checks if the value of left operand is less than the value of right operand; if yes, then the condition becomes 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.
-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.

Scripting – Boolean & String Operators

Operator	Description
!	This is logical negation. This inverts a true condition into false and vice versa.
-o	This is logical **OR**. If one of the operands is true, then the condition becomes true.
-a	This is logical **AND**. If both the operands are true, then the condition becomes true otherwise false.

Operator	Description
=	Checks if the value of two operands are equal or not; if yes, then the condition becomes true.
!=	Checks if the value of two operands are equal or not; if values are not equal then the condition becomes true.
-z	Checks if the given string operand size is zero; if it is zero length, then it returns true.
=-N	Checks if the given string operand size is non-zero; if it is nonzero length, then it returns true.
str	Checks if **str** is not the empty string; if it is empty, then it returns false.

Author – Sri Ram Kumar J (123159)

Scripting – Decision Making & Loops

- **The if statement.**
  - if (cond) then
  - echo "reached the true part"
  - else
  - echo "reached the else part"
  - endif
- **The foreach statement.**
  - foreach name (`\ls -d $dest/*`)
  - echo "$name"
  - end
- **The while statement.**
  - while (cond)
  - echo "$name"
  - end

Scripting - Functions

Function Syntax & Invoke

- **function_name()**
  {
  ...
  <statements>
  ...
  }
- **Invoke - $ function_name**

Passing parameters

- **$ function_name $arg1 $arg2 $arg3**

  function_name()
  {
  ...
  c = $1 + $2
  ...
  }

Return parameter – Method 1

- function_name()
  {
  echo “hello $1”
  return 1
  }
- $ function_name ram
  hello ram
- $ echo $?
  1

Return parameter – Method 2

- $ var = `function_name ram`
  $ echo $var
- hello ram

Author – Sri Ram Kumar J (123159)