OSSA – Lec 6 – UNIX Commands & Shell Scripting.

Learning Topics

1. Introduction to the Shell
2. Important Shell Features
3. Getting Started with Shell Scripting.

1. Introduction to the Shell

What is Shell?

- Shell is a **Program.**
- Shell is **created to interact with the system for users.**
- When user **create a Terminal** it **connects with a Shell** in the System.
- In the **Terminal user can write commands** related to the Shell and **Shell will executes those commands.**
- **User Commands are taken to the Kernel** using Shell.

How to change Shells?

- 3 categories of Shell,
    - **C shell**
    - **Bourne shell**
    - **Bash shell**
- *To find all available shells* **cat / etc/shells**
- *To see working shell* **echo $SHELL**
- *change back to previous shell* **Ctrl +D**

What is a Shell Script?

- Shell Script is a **Sequence of Commands (Programs).**
- A Command,
    - **Utilities coming with OS.**
    - **Used to do a particular tasks.**

Use of Shells

- Shells are using by system administrators.
- Executes complex operations efficiently.
- Perform repititive tasks.
- Maximize the power of the Command Line.
- System can be control during startup and shutdown.
- Execute portable shell scripts across UNIX systems. (Linux, ubuntu etc.)
- Modify the system environment.

2. Important Shell Features

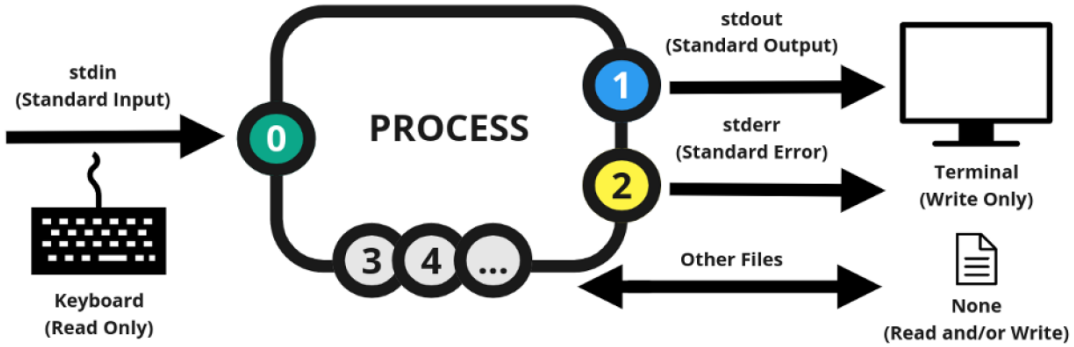
Command Execution

- UNIX commands can be executes in the Shell (ls , cd, echo etc.).
- Also built-in commands can executes.
- Commands can **separate** using this **;** .
- Commands can be **grouping** between **parenthesis ( )** .
- To print a value of command, we can use **grave operator `ex`** .

Filename subsitution

- ' ***** ' Substitute **Zero or More Characters in a Filename.**
  - Ex : *ls -l *ab**
- ' **?** ' Substitutes any **Single Character in a Filename.**
  - Ex : *ls -l fi?el*
- ' **[...]** ' Substitute a **Letter from a restricted range of single characters enclosed within the brackets.**
  - Ex : *ls -l fil[aeg]l*

I/O & Error Redirection



Input Redirection

- **" < "** symbol is used to **redirect input.**
- **Syntax :** " *command* < *input_file* "
- Example : ***" sort < names.txt "***

Output Redirection

- **" > "** symbol is used to **redirect output.**
- **Syntax :** " *command* > *output_file* "
- Example : ***" ls > file_list.txt "***

- **" >> "** symbol is used to **append output.**
- **Syntax :** *"command >> output_file"*
- Example : ***"ls >> file_list.txt"***

Error Redirection

- **“ 2> ”** symbol is used to **redirect error messages.**
- **Syntax :** *“ command **2>** error_file ”*
- Example : ***“ls non_existent_file 2> error.log”***

Pipe Mechanism

- Pipe Operator **“ | ”**
- **Makes** the **output of one program** the **input of another program.**

Background Processes

- Executes **commands in the Background** of the system. When **Foreground is use for other activity.**
- Syntax : **& (ampersand)**
- **Example : “ *$ date &* ”**

Subshell

- To perform some tasks current (parent) shell creates a new (child) shell
- ***Child shell is called “subshell”***

Variables

Environment Variables	**Shell Variabels**
Global variables	User Defined variables
Normally created and available in the system	Can create and use in programming
Written in Uppercase Letters	Define in Lowercase Letters

3. Getting Started with Shell Scripting

- To Start Shell Script → ***" #!/bin/bash "***
- Save Shell Script → ***" first "***
- Change Permission → ***" $chmod 755 first "***
- Run Script → ***" $./first "***

Ingredients for Shell Programming

- Variables and Comments
  - ***#comment***
  - ***name = value*** → **declare variable**
  - ***$ echo $name*** → **value of the variable print**

- Expressions
  - **Arithmetic**
  - **Conditional**

- Flow Controlling Structures
  - **Branching Structures**
  - **Looping Structures**

Special Shell Variables

- **$?** –The exit status of the last executed command
- **$$** – The process number (PID) of the current shell
- **$!** – The process number (PID) of the last background command