SYIMCA SEM III

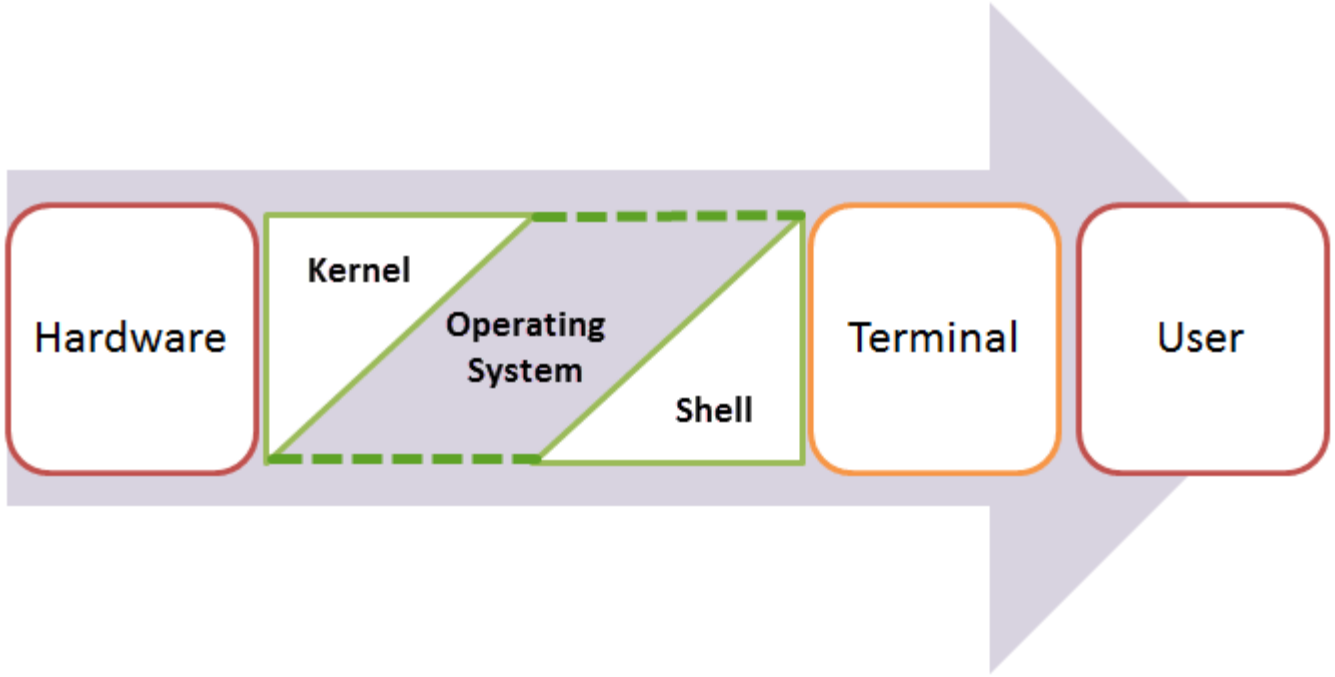
222301307 Practicals on OS & Networks

222301307 Practicals on OS & Networks

UNIT	MODULES	WEIGHTAGE
1	INTRODUCTION TO SHELL PROGRAMMING	20 %
2	ADVANCED SHELL PROGRAMMING	20 %
3	FILTERS	20 %
4	FILE MANAGEMENT & COMPRESSION USING SHELL SCRIPTING	20 %
5	COMMUNICATION COMMANDS	20 %

Linux shell script

- An Operating system is made of many components, but its two prime components are -
  - Kernel
  - Shell



Linux shell script

A Kernel is at the nucleus of a computer. It makes the communication between the hardware and software possible. While the Kernel is the innermost part of an operating system, a shell is the outermost one.

A shell in a Linux operating system takes input from you in the form of commands, processes it, and then gives an output. It is the interface through which a user works on the programs, commands, and scripts. A shell is accessed by a terminal which runs it.

Shell

A Shell provides you with an interface to the Unix system. It gathers input from you and executes programs based on that input. When a program finishes executing, it displays that program's output.

Shell is an environment in which we can run our commands, programs, and shell scripts. There are different flavors of a shell, just as there are different flavors of operating systems. Each flavor of shell has its own set of recognized commands and functions.

Shell prompt

- The prompt, $, which is called the command prompt, is issued by the shell. While the prompt is displayed, you can type a command.

- Shell reads your input after you press Enter. It determines the command that you want to be executed by looking at the first word of your input. A word is an unbroken set of characters. Spaces and tabs separate words.

Shell types

- In Unix, there are two major types of shells −
  - **Bourne shell** − If you are using a Bourne-type shell, the **$** character is the default prompt.
  - **C shell** − If you are using a C-type shell, the % character is the default prompt.

The Bourne Shell have following subcategories−

- Bourne shell (sh)
- Korn shell (ksh)
- Bourne Again shell (bash)
- Z shell (zsh)
- Public domain korn shell (pdksh)
- **The C-type have following subcategories −**
- C shell (csh)
- TENEX/TOPS C shell (tcsh)

Bourne Shell (sh)

- The original Unix shell was written in the mid-1970s by Stephen R. Bourne while he was at the AT&T Bell Labs in New Jersey.
- It was the first shell used for the Unix operating system.
- It provides a basic set of features and is known for its simplicity and efficiency.
- However, all Unix and many Linux versions allow users to switch to the original Bourne Shell, known simply as "sh," if they choose to forgo features such as file name completion and command histories that later shells have added.

C Shell (csh)

- The C shell was developed by Bill Joy as an alternative to the Bourne shell.
- The C shell, as its name might imply, was designed to allow users to write shell script programs using a syntax very similar to that of the C programming language. It is known as “csh”.

TC Shell (tcsh)

- It is an enhanced version of the C shell, incorporating additional features and improvements.
- The name "tcsh" stands for "Tenex C shell," referring to the Tenex operating system for which it was initially developed.
- tcsh provides advanced command-line editing capabilities, allowing users to easily edit and modify previously entered commands using keyboard shortcuts.
- It includes an extensive command history mechanism, enabling users to recall and reuse previously executed commands easily. The history can be searched, edited, and saved to a file for future reference.

Korn Shell (ksh)

- Korn Shell was also written by a developer at Bell Labs, David Korn.
- It attempts to merge the features of the C shell, TC shell and Bourne shell under one package.
- It also includes the ability for developers to create new shell commands as the need arises.
- It is known as "ksh".

Bourne Again Shell (bash)

- The Bash shell is one of the most widely used Bourne shell variants today.
- It was developed as part of the GNU Project and is the default shell for many Linux distributions.
- Its syntax is similar to that used by the Bourne shell, however it incorporates some of the more advanced features found in the C, TC and Korn shells.
- Bash retains compatibility with the Bourne shell while introducing new features like command-line editing, improved scripting capabilities, and advanced job control.
- Among the added features that Bourne lacked are the ability to complete file names by pressing the TAB key, the ability to remember a history of recent commands and the ability to run multiple programs in the background at once.

Z Shell (zsh)

- The Zsh shell is another powerful and feature-rich shell that evolved from the Bourne shell.
- It offers extensive customization options, advanced tab completion, spelling correction, and a more modern scripting environment.
- Zsh is known for its usability and configurability, and it has gained popularity among power users and developers.

Public Domain Korn Shell (pdksh)

- It is an open-source implementation of the Korn shell (ksh) programming language.
- It was developed as a free alternative to the proprietary AT&T Korn shell (ksh88).
- One notable aspect of Pdksh is its portability. This allows users to utilize the Korn shell language and scripts across different environments.
- Pdksh has been widely adopted in open-source Unix-like systems and distributions, serving as the default or alternative shell for scripting and interactive use.

Shell scripts

- The basic concept of a shell script is a list of commands, which are listed in the order of execution. A good shell script will have comments, preceded by # sign, describing the steps.
- all the scripts would have the .sh extension.
- Steps to create sh files -
  - Create empty files using touch command
    - $ touch hello.sh
  - Create empty file and start adding contents using cat > filename
    - $ cat > hello.sh

Shell scripts

- **Save the content and make the script executable:**

  $ cat > test.sh

- **To execute:**

  $ test.sh

List of commands

1.	Creating files: Cat
2.	Creating Directories- mkdir
3.	Changing Directories: cd
4.	Creating blank files: touch
5.	Cat >, >> (redirection)
6.	Copying Files and Directories- cp
7.	Removing Files and Nonempty Directories- rm
8.	Renaming Files and Directories- mv
9.	Comparing Two Files- cmp
10.	What is Common- comm
11.	Converting One File to Other- diff
12.	wc- word count in a file
13.	File permission
14.	Editors: vi, emacs, pico
15.	Printing message on the screen- Echo, Printf