Unix Shell Scripting

Course Objectives

- Understand Unix Shells
- Learn Shell Scripting from basics to advance
- Write Shell Scripts
- Learn Best Practices

Overview

UNIX

Unix is a multitasking, multi-user computer operating system originally developed in 1969 by a group of AT&T employees at Bell Labs, including Ken Thompson, Dennis Ritchie.

Unix trademark is now owned by The Open Group, an industry standards consortium.

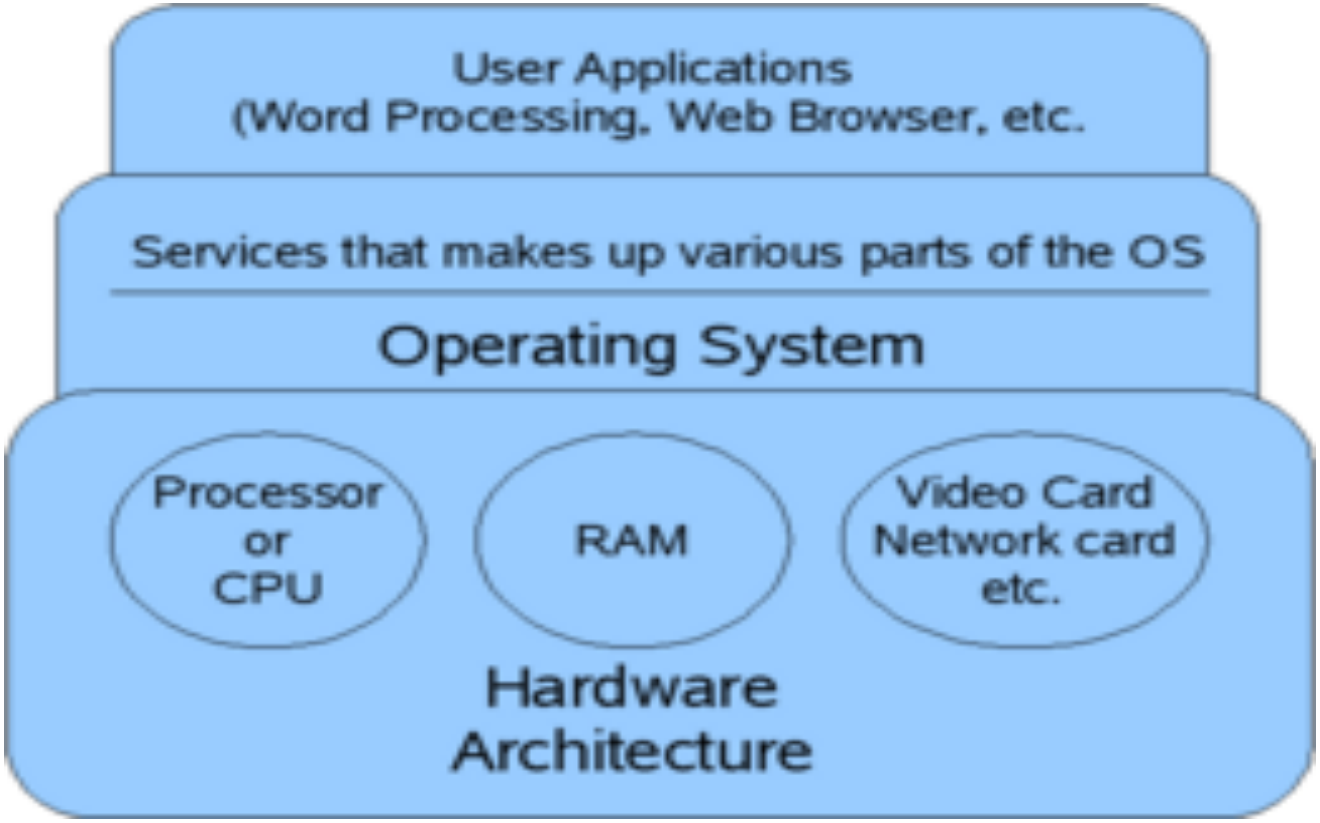
Unix-like

A Unix-like (sometimes referred to as UN*X or *nix) operating system is one that behaves in a manner similar to a Unix system, while not necessarily conforming to or being certified to any version of the Single UNIX Specification.

There is no standard for defining the term, and some difference of opinion is possible as to the degree to which a given operating system is "Unix-like".

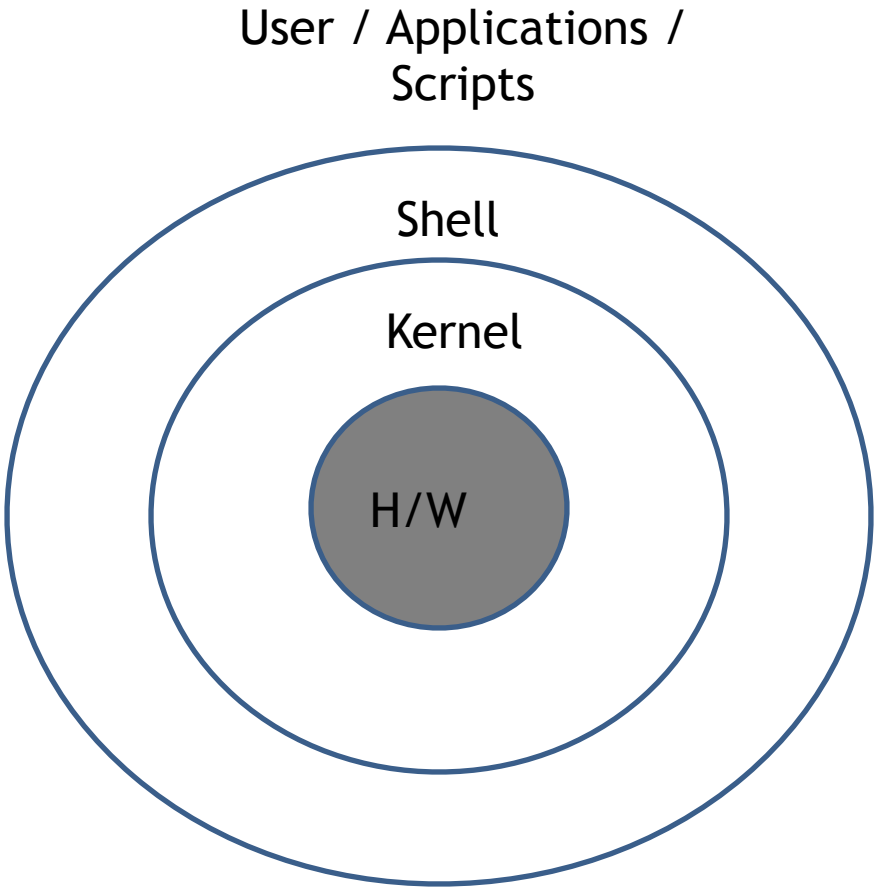
Architecture

OS Architecture



Architecture

Unix Architecture

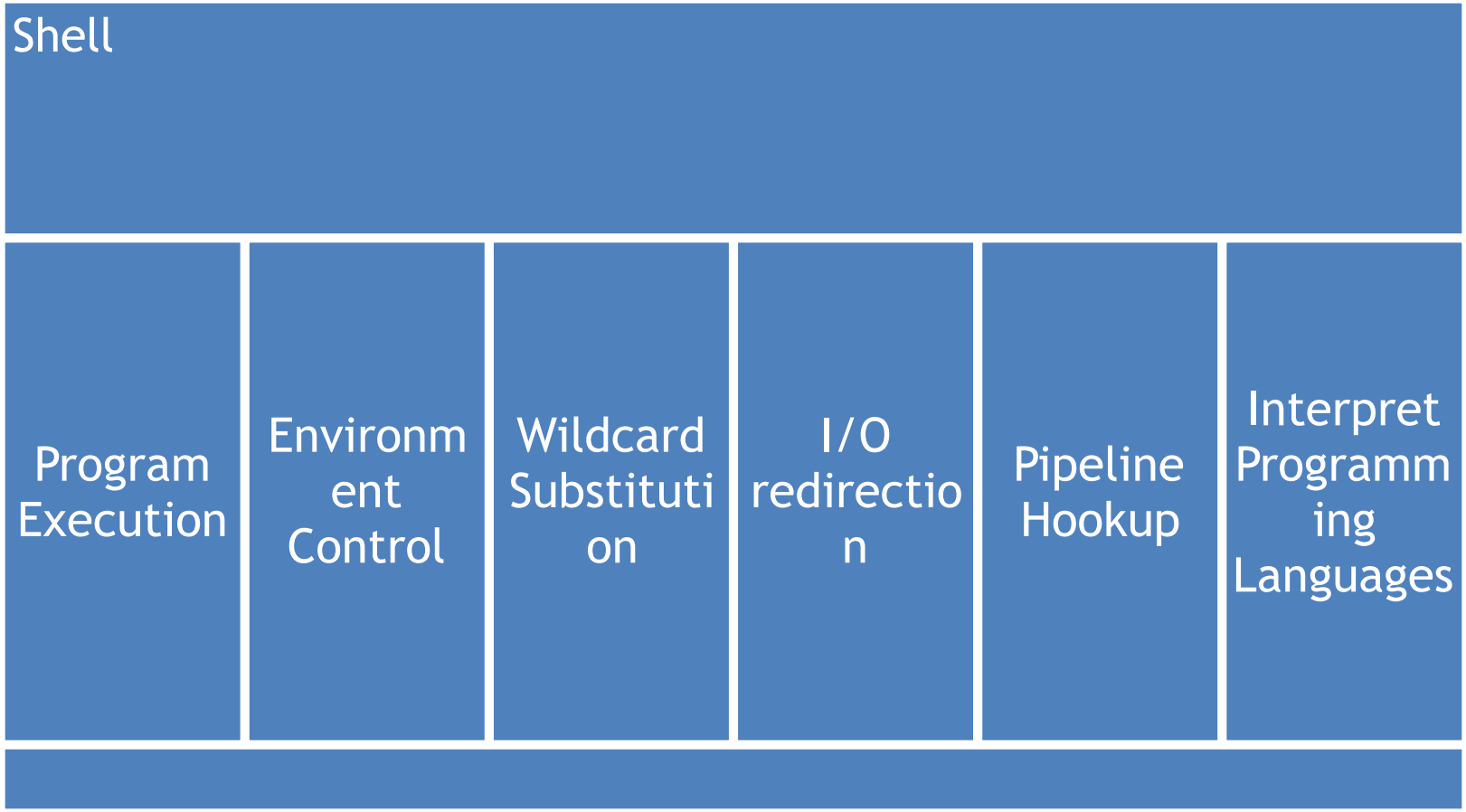


What is Shell?

Shell is a UNIX term for the interactive user interface with an operating system. The shell is the layer of programming that understands and executes the commands a user enters.

Architecture

Functions & Role of SHELL



Types of Shells

Unix / Linux OS offers a variety of shell environments to select from. Below are the popular Unix / Linux shells available.

- Borne shell (sh)
- Bourne again shell (bash)
- ‘C’ Shell (csh / tcsh)
- Korn shell (ksh)

Bourne Shell (sh)
The Bourne shell, called "sh," is one of the original shells, developed for Unix computers by Stephen Bourne at AT&T's Bell Labs in 1977.

Bourne Again Shell (bash)
The "Bourne-again Shell," based on sh -- has become the new default standard. One attractive feature of bash is its ability to run sh shell scripts unchanged.

C – Shell (csh / tcsh)
Using C syntax as a model, Bill Joy at Berkeley University developed the "C-shell," csh, in 1978. Ken Greer, working at Carnegie-Mellon University, took csh concepts a step forward with a new shell, tcsh, which Linux systems now offer.

Korn Shell (ksh)
David Korn developed the Korn shell, or ksh, about the time tcsh was introduced. Ksh is compatible with sh and bash. Ksh improves on the Bourne shell by adding floating-point arithmetic, job control, command aliasing and command completion. AT&T held proprietary rights to ksh until 2000, when it became open source.

Vi Editor - brushup

Cursor movement

- h - move left
- j - move down
- k - move up
- l - move right
- w - jump by start of words (punctuation considered words)
- W - jump by words (spaces separate words)
- e - jump to end of words (punctuation considered words)
- E - jump to end of words (no punctuation)
- b - jump backward by words (punctuation considered words)
- B - jump backward by words (no punctuation)
- 0 - (zero) start of line
- ^ - first non-blank character of line
- $ - end of line
- G - Last Line (prefix with number - 5G goes to line 5)

Note: Prefix a cursor movement command with a number to repeat it. For example, 4j moves down 4 lines.

Inserting / Appending

- i - start insert mode at cursor
- I - insert at the beginning of the line
- a - append after the cursor
- A - append at the end of the line
- o - open (append) blank line below current line (no need to press return)
- O - open blank line above current line
- ea - append at end of word
- Esc - exit insert mode
- r - replace a single character (does not use insert mode)
- J - join line below to the current one
- cc - change (replace) an entire line
- cw - change (replace) to the end of word
- dd - delete (cut) a line
- dw - delete (cut) the current word
- x - delete (cut) current character

Cut and Paste

- yy - yank (copy) a line
- 2yy - yank 2 lines
- yw - yank word
- y$ - yank to end of line
- p - put (paste) the clipboard after cursor
- P - put (paste) before cursor

Search/Replace

- /*pattern* - search for pattern
- ?*pattern* - search backward for pattern
- n - repeat search in same direction
- N - repeat search in opposite direction
- :%s/*old*/*new*/g - replace all *old* with *new* throughout file
- :%s/*old*/*new*/gc - replace all *old* with *new* throughout file with confirmations

Exiting

- :w - write (save) the file, but don't exit
- :wq - write (save) and quit
- :q - quit (fails if anything has changed)
- :q! - quit and throw away changes

What is Shell Script?

A Shell Script is computer program which consists of Unix / Linux shell commands in the order of their execution and which is designed to run under the Unix / Linux OS

Why to Write Shell Script ?

- System boot scripts
- Automate repetitive day to day task such as backups
- Customized commands
- Saves time and efforts in performing lengthy tasks
- Simplifies operations and minimizes human errors
- Application startup scripts

Why not to use Shell Scripts?

- Developing business applications such as billing, inventory management etc.
- Writing complex engineering / mathematical programs
- Large complex applications
- Web Development

Writing Shell Scripts

A simple shell scripts is nothing but a list of commands in the order of their execution. For example the following shell script clean up log files. As a best practice always have .sh as extension of shell script file name.

```
# My first  Shell Script
 clear
 echo  "From shell script"
 echo "Hello All,  I am Sami"
 echo  "Bye  for Now"
```

To write and execute above code do the following

Step 1: # vi myfirst.sh
Step 2: write and save the script file
Step 3: # chmod +x myfirst.sh (give execute permission)
Step 4: # ./myfirst.sh

Note: To execute your any program available in current directory you would execute using **./**program_name and if it is in a different directory the use full path **/home/sami/myscripts/program_name.sh**

Using SHEBANG we can tell OS to use a particular Shell to intrepret the script. Shebang must be the first line of the shell script. Shebang consist of # (hash SHA) & ! (exclamation or bang).

#!/bin/sh -- tell OS that the following script should be executed in Bourne Shell

Consider the following script:

```
#!/bin/bash
# This script runs in bash shell
pwd
ls
```

Writing Some Shell Scripts

1) Writing a shell script to find biggest file in the directory.

```
#!/bin/bash
 ls -lSr|tail -n1
```

2) List out only user names with 'nologin' as default shell from /etc/passwd

```
#!/bin/bash
 grep  nologin /etc/passwd|cut  -d':'  -f1
```

3) Write a script to do messages & wtmp log file cleanup from /var/log

```
# Log Cleanup  Tool
#  Works only as root user
Echo  "Initiating Log cleanup……"
cd /var/log
cat /dev/null > messages
cat /dev/null > wtmp
echo "Log files cleaned up."
```

Try these

1) Write a shell script to find 5 big file in directory
2) Write a shell script to find system uptime and user names who are all users logged in
3) Write a shell script to find out number of user logged in (like 2 users etc)