FUNDAMENTAL OF PYTHON & SHELL SCRIPT

CLASS-VI

Instructor: Palash Gupta

Re-Cap

- Design a Spark Cluster
- Key Procedure Setting up a cluster
- Configuring a Cluster
- Basic Administration
- Under practically how spark works

What we are going to Cover today?

- Introduction to Python
- Practical Session with Python
- Installing & utilizing New Python Package
- Introduction to UNIX and Shell
- Practical Session with Shell Scripting
- Running a python script in UNIX Cron Job with Shell Script Support

Python Programming

Introduction to Python

❑What is Python?

-Python is a high level language

-Python is Interpreted and Interactive

-Python supports both object oriented & functional

❑History of Python?

-It was created by Guido van Rossum (BDFL) during 1985- 1990. Like Perl
-Python source code is also available under the GNU General Public License (GPL).
-Python is named after a TV Show called 'Monty Python's Flying Circus' and not after Python-the snake.
-Python 1.0 is released on January 1994
-Python 2.0 released (2.7) on October 16, 2000 [Garbage collector, Unicode]
-Python 3.0 released (3.5.2) on December, 2008

Features of Python

- Features of Python?

-Easy to learn
-Easy to maintain
-Easy to read
-Interactive Mode
-Portable
-Extendable
-Database Interfaces
-GUI Programming
-Scalable
-Supports automatic garbage collection
-Can be integrated with C, C++, COM, Java etc.

Difference between Python 2 and 3?

Python-2	Python-3
print function brackets optional	print() function brackets mandatory
Prefix string with u to make Unicode string	String is Unicode by default
Division of integer always returns integer -> 7 / 2 = 3	Division of integer may return float -> 7 / 2 = 3.5
raw_input() reads string	raw_input() is not available
input() evaluates data read	input() always read string
	py2to3 utility
print x, # Trailing comma suppresses newline in Python 2	print(x, end=" ") # Appends a space instead of a newline in Python 3

Environment Setup

- Download Python from the link: https://www.python.org/downloads/windows/ OR use given Windows EXE.

- You can take Windows x86 executable installer

Note: In order to install Python 3.5.2, minimum OS requirements are Windows 7 with SP1. For versions 3.0 to 3.4.x Windows XP is acceptable.

- Setting Path at Windows

-To add the Python directory to the path for a particular session in Windows
- Right Click on My PC/Computer -> Properties -> Advance System Settings -> Environment Variable -> Edit Path and Add following python directory
- Note:
- C:\Users\User\AppData\Local\Programs\Python\Python35-32 is our python directory
- C:\Users\User\AppData\Local\Programs\Python\Python35-32\Scripts
- We can also setup **PYTHONPATH**

Running Python Script

There are three different ways to start Python −

1. Interactive Interpreter

    You can start Python from Unix, DOS, or any other system that provides you a command-line interpreter or shell window.

    Enter **python** the command line.

    Start coding right away in the interactive interpreter.

    # python [In Linux/UNIX]
    C:> python

2. Using a python script

    # python test.py
    C:> python test.py

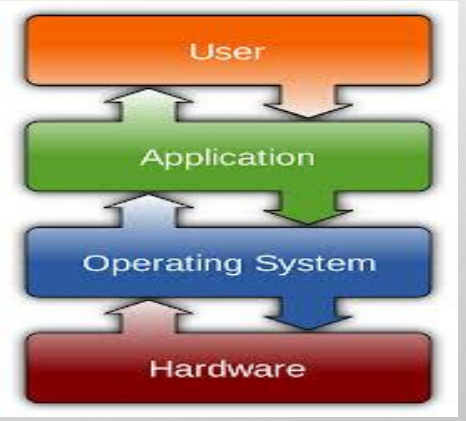
3. Using a IDE – We will use Pycharm in our lab

UNIX Shell Scripting

What is an Operating System?

An operating system (OS), in its most general sense, is software that allows a user to run other applications on a computing device. While it is possible for a software application to interface directly with hardware, the vast majority of applications are written for an OS, which allows them to take advantage of common libraries and not worry about specific hardware details. The operating system manages a computer's hardware resources, including:

- Input devices such as a keyboard and mouse
- Output devices such as display monitors, printers and scanners
- Network devices such as modems, routers and network connections
- Storage devices such as internal and external drives
- The OS also provides services to facilitate the efficient execution and management of, and memory allocations for, any additional installed software application programs



Example of Operating Systems are **Windows, Linux, Android, MAC etc.**

UNIX History & Overview

History:

- UNIX was developed by Bell Labs of AT&T and AT&T versions including version 7, System III, System V
- The most widespread version is BSD 4.1 and BSD 4.2 versions developed in University of California
- Now-a-days the free open source UNIX kernel is LINUX kernel

Time Table:

- After AT&T had dropped out of the Multics project, the Unix operating system was conceived and implemented by Ken Thompson and Dennis Ritchie (both of AT&T Bell Laboratories) in 1969 and first released in 1970
- The **history of Linux** began in 1991 with the commencement of a personal project by Finnish student Linus Torvalds to create a new free operating system kernel. Since then, the resulting Linux kernel has been marked by constant growth throughout its history. Since the initial release of its source code in 1991, it has grown from a small number of C files under a license prohibiting commercial distribution to the 4.15 version in 2018 with more than 23.3 million lines of source code without comments[1] under the GNU General Public License

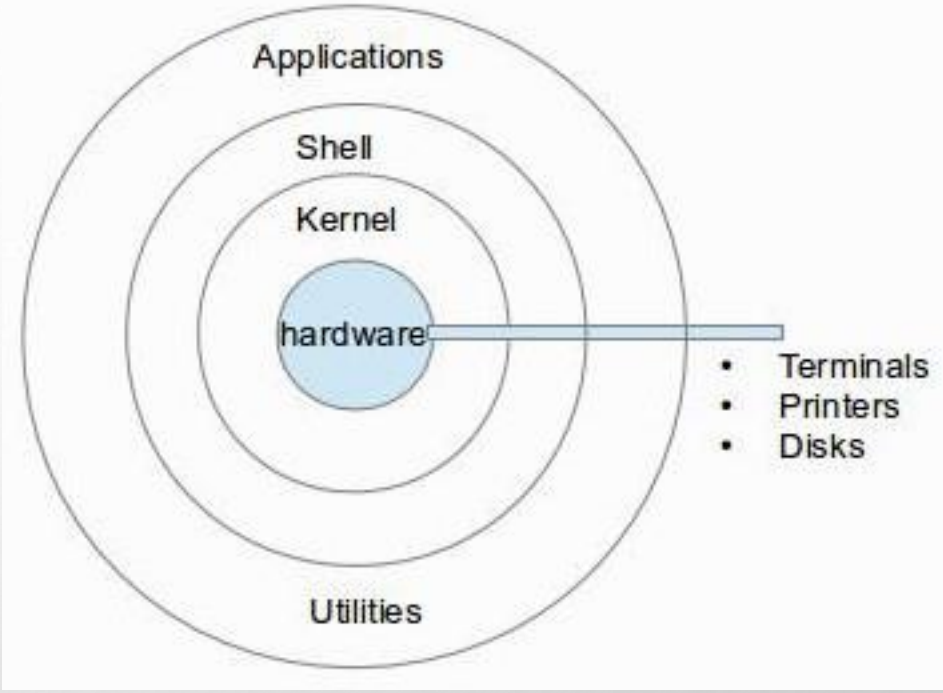
UNIX History & Overview

UNIX Overview questions in mind?

- What is UNIX really like?
- What are the shell and the kernel?
- How does the system keep track of your files?
- What actually happens when you type a command name?

What does Kernel do?

This provides basic-level control over all of the computer hardware devices. Main roles include reading data from memory and writing data to memory, processing execution orders, allocating resources to users, determining how data is received and sent by devices such as the monitor, keyboard and mouse, and determining how to interpret data received from networks.



UNIX History & Overview

Let's take an example:

- The Kernel controls computer hardware
- When you log in, it is the kernel that runs init and getty to check to see if you are an authorized user and have the correct password.
- The kernel keeps track of all the various program being run, allocating time to each, which one stops and which ones to starts etc. (Time sharing)
- The kernel assign storage and runs the shell programs

In Reality Kernel is made with a group of System Calls and a command is basically one or more system calls

What does Shell do?

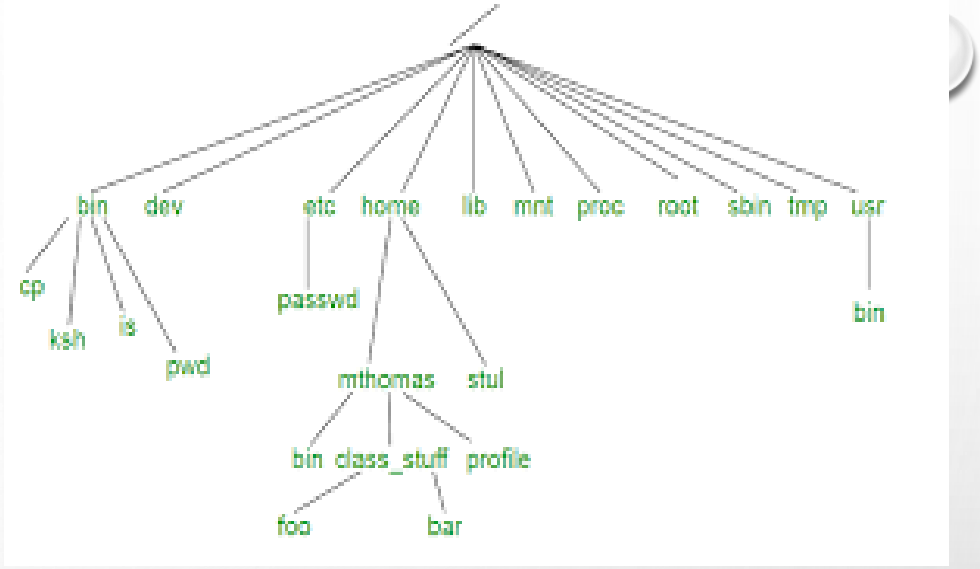
Users generally don't interact with Kernel, hence it is duty for shell to take inputs from user and transfer it to kernel for proper execution. There are different types of shell:

- Bourne (/bin/sh)
- Bash (/bin/bash)
- C (/bin/csh)
- Korn (bin/ksh)

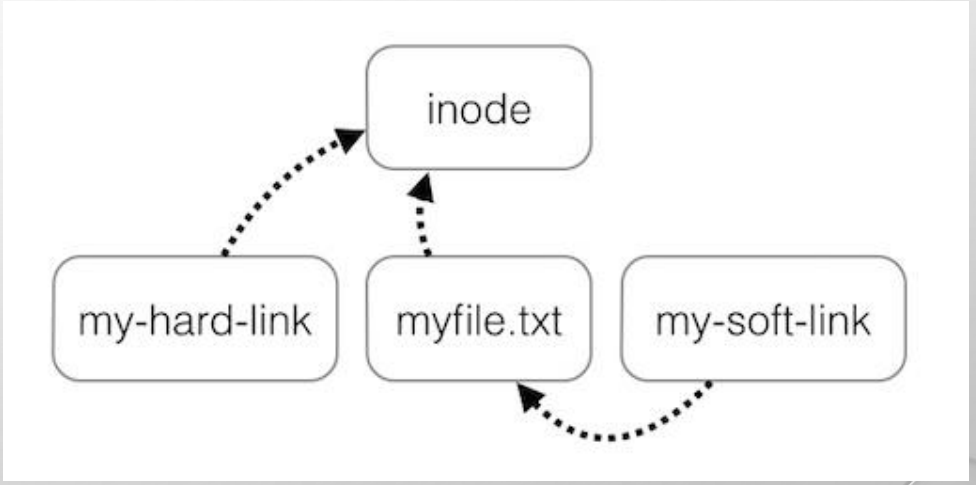
UNIX History & Overview

Let's understand File & File System:

- A file is a series of bytes – can be interpreted differently
- In UNIX system, everything is handled through a file
- A file can be a part of directory of sub-directory
- A file is known by his inode Number & it contains where the file is written in disk, start and end location, permissions, last modified data, creation data etc.
- Missing things within inode is: content of file and name of file



Soft and Hard Link of a File



$ ls –i file_name

7689 file_name

UNIX History & Overview

Let's understand File & File System:

- What happen if a disk or a part of disk is set aside to store files and the inodes entries. The entire functional unit is referred a file system.
- The chunk of memory is divided into number of blocks e.g. 512 byte, 4096 bytes or 8192 bytes etc. based on kernel settings.
- $ # blockdev --getbsz /dev/sda1
  512
  It means the file system block size is 512 KB
- Three types of blockings:

-The first group consists of block 1 and is called super block
-The second group consists of blocks devoted to inodes
-The third and final block is to store files

- A disk can have multiple file systems e.g. xfs, ext2, ext3, ext4 etc.
- **$ more /etc/fstab**

```
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
pi@raspberrypi:~$ df -Th
Filesystem     Type      Size  Used Avail Use% Mounted on
/dev/root      ext4       28G  3.7G   23G  14% /
devtmpfs       devtmpfs  214M     0  214M   0% /dev
tmpfs          tmpfs     218M     0  218M   0% /dev/shm
tmpfs          tmpfs     218M  4.5M  213M   3% /run
tmpfs          tmpfs     5.0M  4.0K  5.0M   1% /run/lock
tmpfs          tmpfs     218M     0  218M   0% /sys/fs/cgroup
/dev/mmcblk0p6 vfat       65M   21M   45M  32% /boot
tmpfs          tmpfs      44M     0   44M   0% /run/user/1000
/dev/mmcblk0p5 ext4       30M  398K   28M   2% /media/pi/SETTINGS
```

Why UNIX is better mostly in Server Side?

Why UNIX is better?

- Open source nature
- Secure
- Can revive older computers
- Perfect for programmers
- Software Updates
- Customizations
- Variety of distributions
- Free to Use
- Better Community Support
- Reliability
- Privacy

Better Community Support

Basic UNIX Commands

Listing files and directories
Making Directories
Changing to a different directory
The directories . and ..
Pathnames
~ (your home directory)
Copying Files
Moving files
Removing files and directories
Displaying the contents of a file on the screen (clear, cat, head, tail)
Searching the contents of a file (less and /, grep, wc)
Redirection
Redirecting the Output
Redirecting the Input
Pipes
Wildcards

Basic UNIX Commands

Getting Help

File system security (access rights)

Changing access rights

Processes and Jobs

Listing suspended and background processes

Killing a process

Other useful UNIX commands (df, du, compress, gzip, unzip, history, file)

Shell Scripting Overview

A shell script can contain one more UNIX commands & it can contact various logics to automate user jobs.

$ cat < simple.sh
echo “this is my first shell script”
date
Ctrl+D

$chmod u+x simple.sh

$./simple.sh

A detail can be done as part of our lab

Shell Scripting Overview Contd.

SN	Key Item	Process
1	Value of a variable	$variable
2	Input, output, error	Input – 0 $cat < a.txt Output – 1 echo “Hello…” > a.txt #Overwrite echo “Hello…” >> a.txt #Append Error – 2 <Command> > out 2>&1
3	Value of nth positional parameter	$n
4	Total number of positional parameter	$#
5	Command Substitute	`Command`
6	Common Shell Variable	$HOME, $LOGNAME, $HOSTNAME $PATH etc.
7	Showing all Shell Environment Variable	export
8	Set a Shell Environment Variable	export HADOOP_HOME=“/home/hadoop”

Shell Scripting Overview Contd.

SN	Key Item	Process
9	Running a command in background	<command> &
10	Piping	cat /proc/cpuinfo | grep 'core id' | wc –l
11	List of all positional parameter	$*
12	Exist Status of last command	$?
13	The process (PID) of current shell	$$
14	Quoting	‘’ or “”
15	Comment	#
16	Set a variable	a=30

Scheduling a Shell Script

Scheduling:

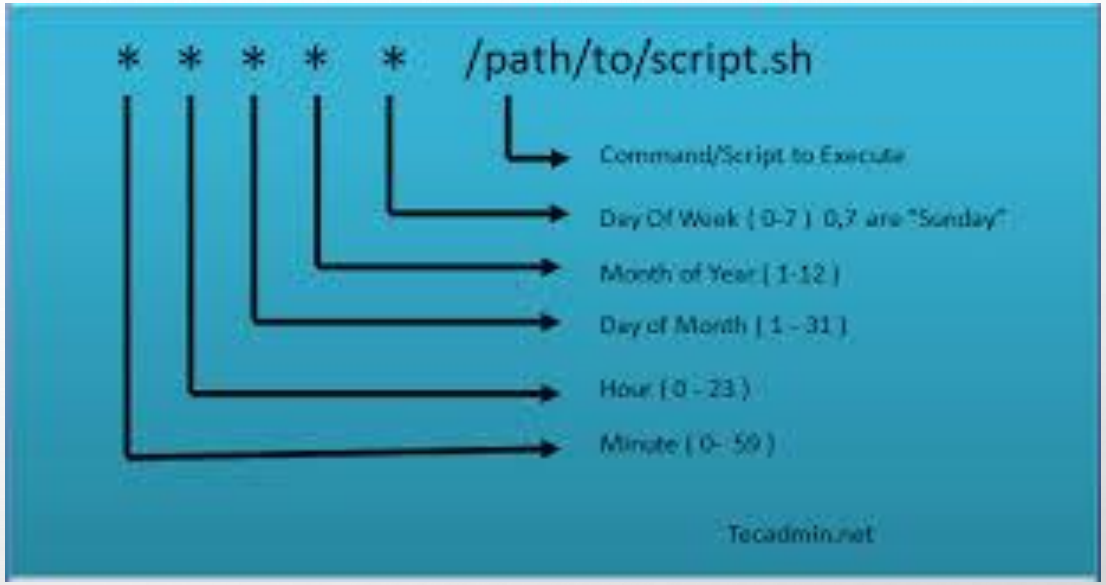
Crontab is a powerful utility to schedule any job in an UNIX machine.

$ crontab –l

$ crontab –e

If we want to run a job daily at 03:00PM then the entry shall be look like:

00 15 * * * /path/our_script > /tmp/log.txt



Running Python within UNIX Shell Scripting

Scheduling python program within shell script

KPIENGINEFORNSN3G.sh | Covid-19 | pythontesting.txt | Sample_Script_python_within.sh

```
1  #!/bin/bash
2  . /home/hadoop/.bashrc
3  changedname=`date '+%Y%m%d%H%M%S'`
4  cd /tmp/
5  export filename=`echo 'mypythonscript'$changedname'.log'`
6  ((
7  python mypythonscript.py
8  #Retention of log files generated by Application in /tmp directory
9  find /tmp/ -name 'mypythonscript*' -mtime +15 -exec rm -rf {} \;
10 )2>&1)| tee -a $filename
11
```

QUESTION & ANSWER

THANKS FOR ATTENDING THE CLASS & YOUR CO-OPERATION

References

- https://www.python.org/
- https://www.tutorialspoint.com/index.htm
- https://www.shellscript.sh/functions.html
- https://www.guru99.com/introduction-to-shell-scripting.html