Course Title: Introduction to Linux and Commands

Faculty: Bharath.G, Srinivas.K & Rajesh Sola

L&T Technology Services

LTTS GLOBAL ENGINEERING ACADEMY

Agenda

- Introduction to Linux OS
- Basic Linux Commands
- Filter commands
- Shell Features
- Environment
- Files and utilities

Syllabus: Introduction to Linux & Commands

- Introduction to Linux & Open Source
- CLI Environment – Shell, Terminal
- General Purpose Commands
- Handling File s & Directories
- Customizing Environment
- Disk & File System Utilities
- Filter Commands
- Shell Features
- Shell Scripting

Introduction to Linux OS

Linux History

Important key words: **UNIX, LINUX, GNU, FSF, GPL**

UNIX: Multitasking, multi-user OS developed in 1969 by AT&T employees at Bell Labs.

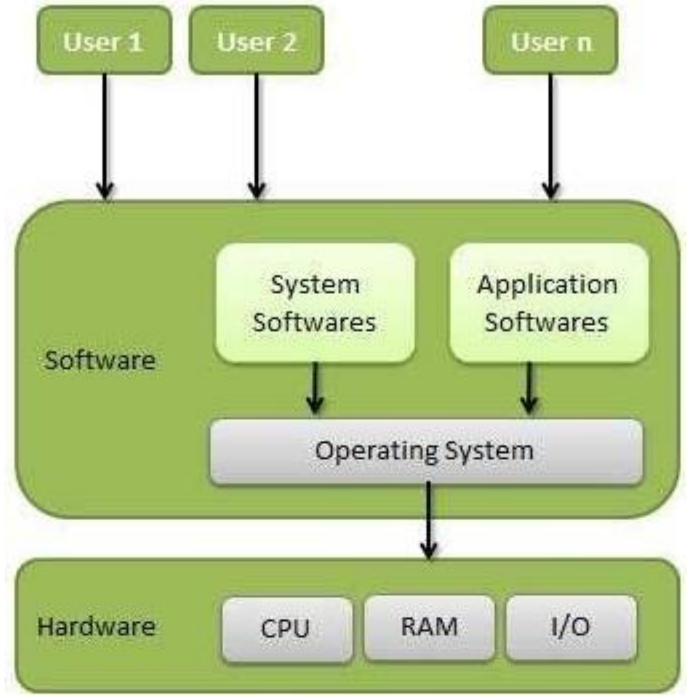
Linux: Kernel of Unix developed by Linus Torvalds in 1991, official name is 'GNU/Linux'

GNU: Free UNIX like OS developed by Richard Stallman in 1984

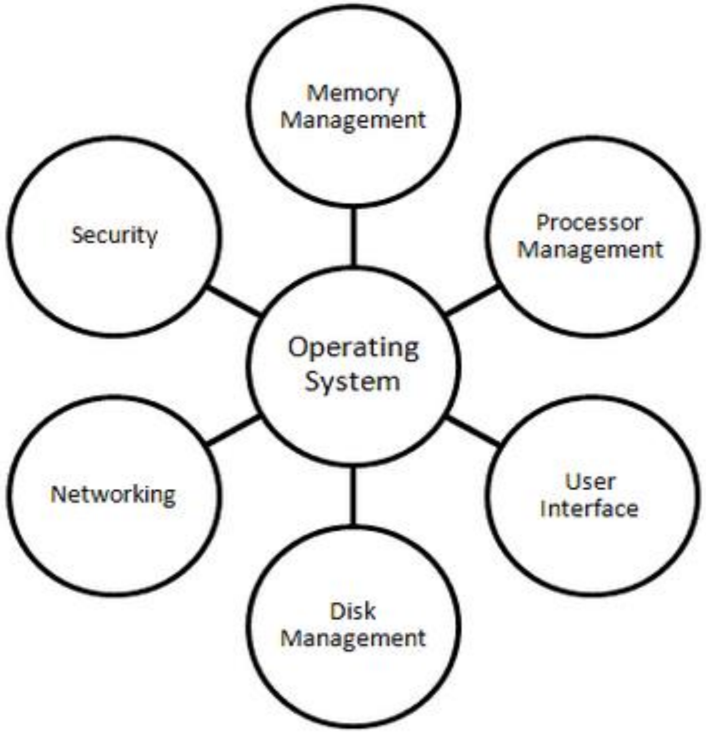
FSF and **GPL:** To promote GNU, Richard established FSF(the Free Software Foundation) and develop a public copyright agreement named GPL(General Public License).

Computer Architecture and OS

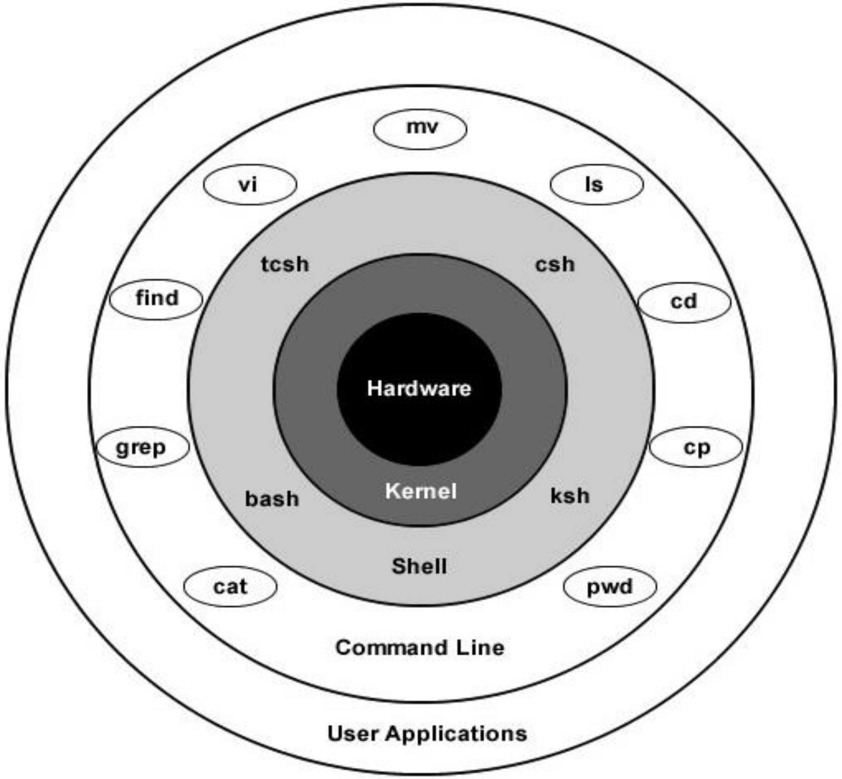
Computer Architecture



Functions of OS



Linux OS Architecture



Features of Linux OS:

1. Open source
2. Portable
3. Multi-user
4. Multi programming
5. Hierarchical File system
6. Shell
7. Security

Linux Commands

Command Line Interface

- **Terminal**
  - gnome-terminal
  - Konsole
  - xterm
  - mate-terminal
- **Shell**
  - command interpreter
  - Shell prompt
- Types of shells – bash, ksh, csh, tcsh, zsh etc.
- Current Shell - echo $SHELL, ps, cat /etc/passwd
- Available shells - /etc/shells

Command Syntax

Syntax:

- Command [options] [arguments]

Symbols used in commands:

-	#Single letter options start with
--	#Multi lettered options start with
\	#To continue command is next line
/	#To escape spaces in the arguments

- **Note:** <u>Shell is case sensitive</u>

Software installation

Command to Install software:

- sudo apt-get install <software>

Usage:

```
sudo apt-get install build-essential
```

Command Help and Documentation

Help command:

- -h or --help

User manual of command:

- Syntax:

  man [options]… [command]…

- Man page sections: 0 to 9

• 1	# User commands
• 2	# System calls
• 3	#Library calls
• 0	#Library header files

- Usage
  - man sleep, man 1 sleep, man 3 sleep,
- Shell variable **MANPATH**

Command Help and Documentation

• man -k keyword	#Search for keyword as a regular expression in man pages
• apropos keyword	#Search for keyword as a regular expression in man pages
• whereis keyword	#Prints location of executable, source code, man page
• whatis command	#Prints one line manual page description of command
• Info command pages	# Documentation in info format, can navigate and have link
• /usr/share/doc	#More documentation

Basic Linux Commands

Simple Commands

- date
- cal
- echo
- bc
- uname
- clear
- seq
- rev
- wc

date Command

Display or set System date

- Syntax:

  date [options]... [+Format]

- Usage:
  - date

Format options:

• %Y	#Display four-digit year
• %H	#Display the hour
• %M	#Display the minute
• %S	#Display the seconds
• %D	#Display date as mm/dd/yy
• %d	#Display the day of the month (01 to 31)
• %h	#Displays abbreviated month name (Jan to Dec)
• %m	#Displays the month of year (01 to 12)
• %y	#Displays last two digits of the year(00 to 99)
• -s	#For changing date & time by super user

Example:

```
date "+%D“
date "+%D %T“
date "+%Y-%m-%d"
date "+%Y/%m/%d"
date "+%A %B %d %T %y“
date --date="tomorrow“
date --date="2 year ago"
```

cal Command

Display Calendar for specific month or year

- Syntax:

```
cal [ [month] year]
```

- Usage:
  - cal
  - cal 5 2020
  - cal 2020
  - cal -3

echo Command

Display message or output the result of other commands

Syntax:

```
echo [-neE] [arguments]
```

Usage:

- echo “Welcome to Linux”
- echo -e “Hello\tWorld”
- echo -n “Hello World”
- var=100
  echo var=$var
- echo “Welcome to \"Linux\"“
- echo `date`   # Not single quotes, its back quote

bc Command

Basic command line calculator and interactive interpreter

Syntax:

```
bc [ -hlwsqv ] [long-options] [ file ... ]
```

The bc command supports the following features:

- Arithmetic operators
- Increment or Decrement operators
- Assignment operators
- Comparison or Relational operators
- Logical or Boolean operators
- Math functions
- Conditional statements
- Iterative statements
- Functions

Example:

```
echo “12+5” | bc
echo “10^2” | bc
echo “12>5” | bc
echo “12<5” | bc
bc
   12+5
    10^2
    12>5
    quit
```

uname Command

Display system information

Syntax:

```
uname [options]
```

Options:

-v	#kernel version
-r	#kernel build time
-n	#host name
-m	#machine name
-i	#hardware platform
-s	#kernel name
-o	#operating system
-a	#all of above info

seq and clear Commands

Print Sequence of Numbers

Syntax:

- seq [Option] LAST
- seq [Option] FIRST LAST
- seq [Option] FIRST INCREMENT LAST

Usage:

- seq 1 10
- seq 1 3 20
- seq 10 -1 1
- seq 20 -2 0

Clear the terminal Screen

- clear

rev and wc Commands

Print contents of arguments

Syntax:

- rev [Option] [file]

Example:
```
rev
    "Any string"
     CTRL + D

rev filename
```

Word Count

Syntax:

- wc [Option]..[File]...

- Output Description:
  - Column1 -Number of lines present in file
  - Column2 -Number of words present in the file
  - Column3 -Number of characters present in file
  - Column4 -File name which was given as argument

Example:
```
wc
    "Any string"
    CTRL + D

wc filename
```

Shortcuts in Command line

Editing in command line

• CTRL + a or **home**	# Go to beginning of line
• CTRL + e or **end**	# Go to end of line
• CTRL + u	# Discard the current command

Process control in command line

• CTRL + c	# Interrupt the running process (by **SIGINT** signal)
• CTRL + d	# End of input
• CTLR + \	# Quit process (by **SIGQUIT** signal)
• CTRL + z	# Suspend process (by **SIGTSTP** signal)

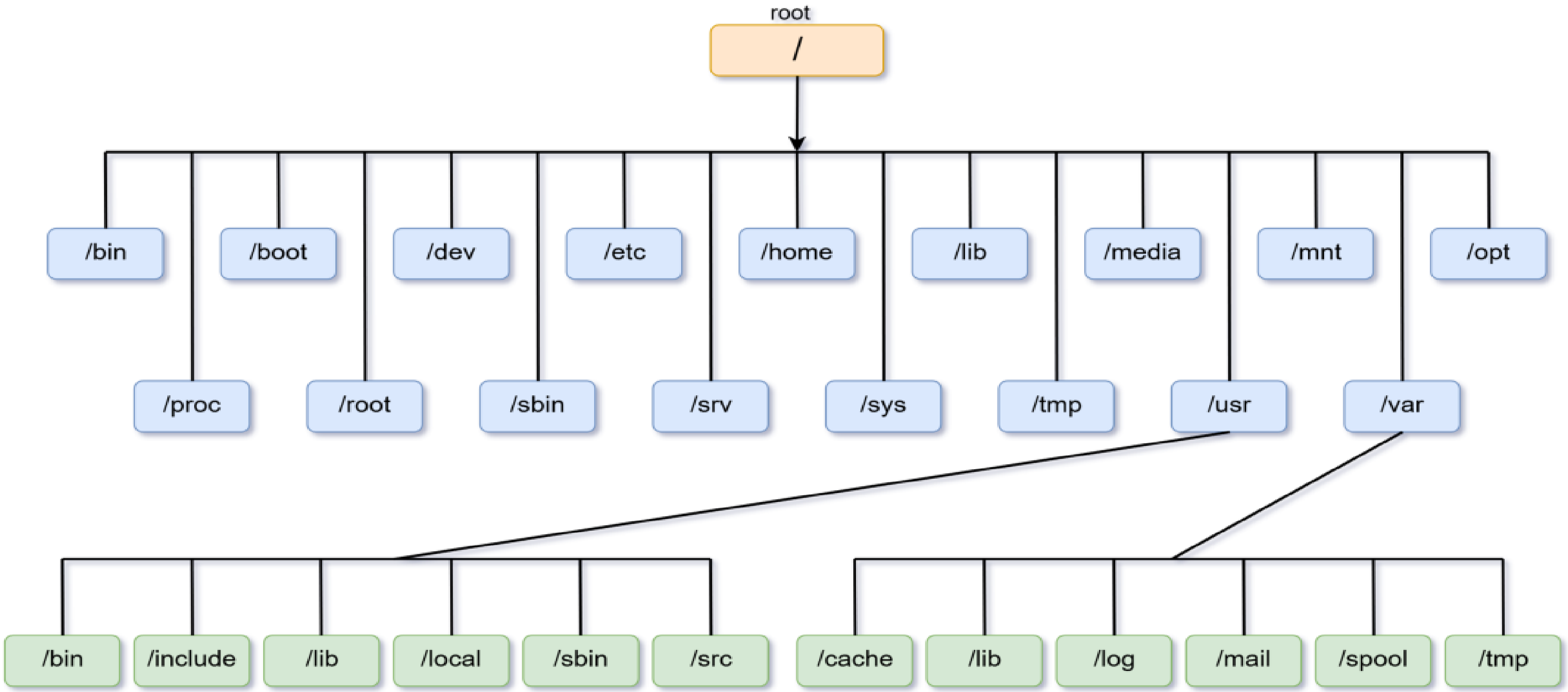
Command Navigation

• CTRL + r	# Reverse search a already used command
• CTRL + l	# Clear the screen
• ↑ and ↓	#Navigate between previous commands
• **TAB**	#Auto complete command

File System and Commands to Handle Files and Directories

Linux File System Hierarchy

Filesystem Hierarchy Standard (FHS)



File Paths

• /	# Root directory of Linux
• Absolute path	# Path w.r.t / directory
• Relative path	# Path w.r.t current directory
• Special Directories	
• .	# Current directory
• ..	# Parent Directory
• Home directory (~)	# Represents home directory, feature of shell

Directory navigation Commands

Print Working Directory

• pwd	# Current directory
• pwd -L	# Logical directory name, default
• pwd -P	# Physical directory name
• Shell variable **PWD**	

Change Directory

• cd [options] directory	
• Options:	
-L	# Follow symbolic links, default
-P	# Don't follow symbolic links
• cd	# home dir
• cd /	# root dir
• cd ..	# parent dir
• cd ~	# home directory
• cd -	# recent dir
• cd .	# Change to current directory, does nothing

Create and Delete directory Commands

Create new directory

- mkdir <folder name>
- mkdir abc
- mkdir abc/pqr
- mkdir abc/pqr/xyz
- mkdir dir1 dir2

Delete empty directory

- rmdir <folder name>
- rmdir dir1 dir2
- rmdir -p abc/pqr/xyz

File handling Commands

File handling:

- ls
- touch
- rm
- cp
- mv
- file
- cat, tac, ln
- chown, chmod

ls Command

List files and directories

- Syntax:

```
ls [options]... [files]...
```

- Usage:
  - ls # List all files and folders in the current directory
  - ls -l # List files and folders in long list format
  - ls -a # List hidden files and folders
  - ls -R # List content of subdirectory recursively
  - ls -l output fields
    - The file type
    - The file permissions
    - Number of hard links to the file
    - File owner
    - File group
    - File size
    - Date and Time
    - File name

Commands to Create and Display file

- **touch**
  - touch file_name # Create file_name without content if not exist, change timestamp if exists
- **cat**
  - cat file_name # Display content of file_name
  - cat file1 file2 # Display content of file1 and file2
  - cat > file_name # Create file file_name and store standard input to file
  - cat -n file_name # Display content of file with line numbers
  - cat file1 > file2 # Copy file1 contents to file2
  - cat file1 >> file2 # Append file1 contents to file2 contents
- **tac**
  - tac file_name # Display contents from last line
- **nl**
  - nl file_name # Display line numbers, same as cat -n

cp Command

Copy files

- Syntax:
  - cp [Option] Source Destination
  - cp [Option] Source Directory
  - cp [Option] Source-1 Source-2 Source-3 Source-n Directory

Common options

- -r recursive, including subdirs
- -v verbose output
- -i interactive
- -f force

Usage:

- cp src_file dest_file # Copy source file to destination file
- cp -r src_dir dest_dir # Copy all contents of src_dir to dest_dir recursively
- cp file1 file2 file3 dir # Copy file1, file2, file3 to dir

mv Command

Move and rename files

- Syntax:
  - mv[Option] Source Destination

Common options

- -r recursive, including subdirs
- -v verbose output
- -i interactive
- -f force
- -b backup

Usage:

```
mv src_file dest_file        # Rename src_file to dest_file, overwrite if exists without prompt
mv file1 file2 file3 dir     # Move file1, file2, file3 to dir
```

rm Command

Remove files

- Syntax:
  - rm[Option]… File…

Common options

- -r recursive, including subdirs
- -v verbose output
- -i interactive
- -f force

Usage:

• rm file	# Remove file
• rm file1 file2 file3	# Remove file1, file2, file3
• rm file*	# Remove files with matching name file.

Filter Commands

Filter Commands

- head
- tail
- more
- less
- cut
- paste
- sort
- tr
- grep
- uniq
- sed

head & tail Commands

- **head**
  - head file # Display first 10 lines from file
  - head -n 5 file # Display first 5 lines from file

- **tail**
  - tail file # Display last 10 lines from file
  - tail -n 5 file # Display last 5 lines from file
  - tail -n +11 file #Displays 11th line onwards
  - tail -f file # Continuously monitor last 10 lines

- Activity:

  Display Lines from 2 to 5 in a emp.lst file from handbook.

  Hint : |

more & less Commands

- **more**
  - more file # Display file/stdin page by page according to display height
  - cat file.txt | more # Display stdout page by page according to display height

- **less**
  - less file # Display file/stdin page by page according to display height
  - cat file.txt | less # Display stdout page by page according to display height
  - less has additional support for arrow keys

“More is Less and Less is More”

cut Command

Split a file vertically character wise or field wise separated by given delimiter

Syntax:

```
cut option.. [File]...
```

Options

- -c character range
- -d delimiter
- -f field as per delimiter separated

Usage:

- cut -c 1-3 sample.txt
- cut -d':' -f2,3 sample.txt
- echo ‘01/02/2020’ | cut -c 4-5
- echo ‘01/02/2020’ | cut -c 1,5
- echo ‘10/12/2020’ | cut -d'/' -f3

sample.txt:-

```
    101 : Pune : India : 5200
    100 : Chennai : India : 4800
    103 : Delhi : India : 7000
    104 : Kolkata : India : 1200
    102 : Mumbai : India : 5200
```

paste Command

Merge two or more files Horizontally (parallel merging) with tab as separator

Syntax:

```
paste [Option]... [Files]...
```

Options

- -d custom separator
- -s sequential merging

Usage:

- paste f1 f2
- paste -d':' f1 f2
- paste -d' ' f1 f2
- paste -d'\n' f1 f2
- paste -s f1 f2

```
f1:-
        red
        green
        blue
f2:-
        one
        two
        three
```

sort and uniq Command

- **sort**
  - sort sample.txt   # Sort file contents line wise
  - sort -t':' -k2 sample.txt   #Sort file based on second column after :
  - sort -t':' -k2 sample.txt -r   #Same as above, but in reverse order

```
sample.txt:-
    101 : Pune : India : 5200
    100 : Chennai : India : 4800
    103 : Delhi : India : 7000
    104 : Kolkata : India : 1200
    102 : Mumbai : India : 5200
```

- **uniq**
  - uniq [Option] [input[output]]
  - uniq file   # Eliminates consecutively repeated lines

```
Example.txt:-
    Welcome to Linux.
    Welcome to Linux.
    Welcome to Programming.
    Welcome to Linux.
```

tr Command

Translate or delete characters

Syntax:

- tr [Option] set1 [set2]

Options:

- -d delete each char from given set
- -s truncate/squeeze multiple chars into single char

Usage:

- tr ':' '|' < sample.txt
- tr ':0' '|$' < sample.txt
- tr -d ':' < sample.txt
- tr -s'0' < sample.txt

```
sample.txt:-
        101 : Pune : India : 5200
        100 : Chennai : India : 4800
        103 : Delhi : India : 7000
        104 : Kolkata : India : 1200
        102 : Mumbai : India : 5200
```

grep Command

Searching for a pattern of characters inside files

Syntax:

- grep [options] pattern [files]

Usage:

grep ^pattern file	#Starting with pattern
grep pattern$ file	#Ending with pattern
grep ^pattern$ file	#Starting and ending with pattern(whole line match)
grep -c ^$ file	#Count blank lines, -vc for non empty lines

Example:

grep ^# test.c	#lines starting with #
grep \/$ test.c	#lines ending with /
grep \;$ test.c	#lines ending with ;, \ to nullify special meaning of ;

test.c:-

```
 #include <stdio.h>
 int main()
  {

     printf("welcome /
               to Linux");
     return 0;
  }
```

sed Command

Stream editor

sed actions

• p print	#option -n is useful to suppress all lines by default
• d delete	
• i insert	
• s substitute	

Line based actions

• sed -n '5,8p' file	#display line no.s 5 to 8, try without -n option
• sed '5,8d' file	#delete line no.s 5 to 8

Pattern based actions

• sed -n '/int/p' test.c	# Print line containing supplied pattern
• sed -n '/\bint/p' test.c	# Exclude partial words
• sed '/int/d' test.c	#Delete line containing supplied pattern

Insert new lines

• sed 5i’string’ file	#Insert new line after 5th line with the supplied pattern

sed Command

Substitution

Syntax:

```
sed -i 's/SEARCH_REGEX/REPLACEMENT/g'  INPUTFILE
```

Usage:

sed 's/printf/puts/’ for.c	#Substitute puts in place of printf
sed 's/var/count/' for.c	# Substitute count in place of var
sed 's/var/count/g' for.c	# Substitute all the global occurrences
sed 's/var/count/2' for.c	# Substitute on 2nd occurrence on every line
sed ‘s/\bint\b/long/’ for.c	# Ignore partial words
sed '/int/s/var/count/' for.c	#Substitute if followed by int

```
//for.c
#include <stdio.h>
int main()
{
 int var ;
 for( var =1; var <=10; var++)
 printf(“%d\n”, var );
 return 0;
}
```

Shell Features

Internal vs. External Commands

- Inbuilt commands vs. External Commands
- **PATH** variable
- Add custom dir to PATH
  - export PATH=<custom-dir-location>:$PATH
- Adding to startup script files
  - ~/.bashrc # Every local login
  - ~/.bash_profile # Every remote login
  - /etc/environment # System wide configuration for all users on all logins
- **Identification**
  - **type** # check for internal or external command
  - **which** # Path to external command

Shell Features

- **Shell Variables**
  - X=100
    echo $X
- **Continuously generate string at stdout**
  - yes [String] # Generates supplied string at stdout until aborted.
- **Combining commands**
  - cmd1 && cmd2 # Execute cmd2 only if cmd1 succeeds
    - ls dir1 && ls dir2
  - cmd1 || cmd2 # Execute cmd2 only if cmd1 fails
    - ls dir1 || dir2
  - cmd1 ; cmd2 # Execute cmd2 unconditionally
    - ls dir1;ls dir2

Meta Characters and Quoting

- **Meta characters**
  - $, <, >, `, \, ‘, “, ;, |, &, (,), new-line char, tab char, space char

Quoting to suppress special meaning of all Meta Characters

- Escape character
  - echo \#
  - echo *
  - echo \(
- Single Quotes #Except single quotes itself, even when followed by \
  - echo ‘\’
- Double Quotes #Except **$**, `, **** (only if followed by **$**, `, ****, **“**)
  - echo “\” vs echo “\\”
  - echo “ls” vs echo “`ls`”
  - echo “\””
- ANSI-C Quoting
  - echo $’Hi\thello’ #Supports all ANSI C escape chars

Pattern Matching

- **Wildcard substitution in file & directory names (pattern matching)**
  - ***** # Matches any string, including the null string
    - cat file*
    - nl file.*
  - **?** #Matches any one character
    - cat file.tx?
    - nl ?ile.txt
  - **[. . .]** #Matches any one of the characters supplied
    - cat f[aeiou]le
    - cd dir[aeiou]ct[!xyz]ry
  - **[! . . .]** #Matches any character *other than* one of the characters supplied
    - cat f[!xyz]le

Standard Input, Standard Output Redirection

- **Process File Descriptor:**
  - 0 #stdin
  - 1 #stdout
  - 2 #stderr
- **Standard Input Redirection**
  - command < File
    - cat < filename
    - bc < filename
    - rev < filename
- **Standard Output Redirection**
  - command > File #Overwrite if exists, otherwise creates new.
  - command >> File #Appends if exists, otherwise creates new.
  - command > dev/null #Discard the output
    - cat file1 > file2
    - cat file1 >> file2
    - cat music.mp3 > /dev/audio #Play audio file
    - echo “Some string” >>file1
    - cat file1 file2 file3 >> file1 #Observe the Result
    - cat file1 >> /dev/null # Discard the output

Standard Error and Inline input Redirection

• **Standard Error Redirection**	
• command 2> File	#Uses File to store standard error. Overwrite if exists.
• command 2>>File	#Uses File to store standard error. Append if exists.
• command &> File	# Store both stdout & stderr to File.
• gcc dumbfile 2> file	
• ls ext_dir non_ext_ dir 2> file	#Example that provides output and error
• ls ext_dir non_ext_ dir &> file	#Example that provides output and error
• **Feeding Inline input**	
• cmd <<< expression	# Expression or string is supplied as stdin to cmd.
• rev <<< “Number of characters ”	
• **Piping:**	
• cmd1 | cmd2	# stdout of cmd1 is supplied as stdin for cmd2
• ls -l | wc -l	

- **Display stdin to stdout and store to file**
  - command | tee -a file
    - cat file1| tee -a file2

Additional Features

- **Arithmetic in shell**
  - **expr** $a + $b
  - **bc** <<< “$a + $b”
  - **let** c=a+b
- **Command Substitution**
  - cal \`date +%m\` 2020 #Its backquote
  - list=\`ls\`

    echo $list
- **Background processing**
  - command & # Run the process in the background
    - cat &

• Command line Editors

- **Text File editor**
  - nano #Simple one to start with
- **Vi Editor**
  - vim #vi improved
  - gvim #GTK based vi
- **vi modes**
  - command mode
  - input mode
  - ex mode

Power of History

- **Command history**
  - history # History of commands executed
  - **HISTSIZE** # Variable to define the size of history

- **Repeat recent commands from history**
  - !num #Repeat the command at that line in history file
    - !10
  - !! #Most recent command
  - !key #Recall last executed matching command
    - !any
  - history | grep pattern #Searching in history file
    - History | grep his
  - ^key1^key2 #Replace key1 by key2 in recent command only
    - cat f1
    - ^f1^f2

Repeat Arguments from history

- !* #Repeat all arguments from last command
  - **!$** or **$_** # Repeat last argument. **Alt + .**
- !^ #Repeat first argument
- !!:n #Repeat the argument number

```
echo “one” “two” “three” “four”
echo !*

echo “one” “two” “three” “four”
echo !$

echo “one” “two” “three” “four”
echo !^

echo “one” “two” “three” “four”
echo !!:2
```

- **Remove from History**
  - history -d line #Clear line from history file
  - history -c #Clear whole history
  - cat /dev/null > ~/.bash_history #Clear history

Environment

Environment

Environment Variables:

- printenv, env, set #Display all Environment variables
- $VARIABLE #Access value of Environment variable
- export $VARIABLE=new_val # Change value
- export $VARIABLE=new:$VARIABLE #Append a path

- **Set and delete new Environment Variable**
  - VARIABLE=value
  - unset VARIABLE
- **Shell Prompt:**
  - PS1
  - PS2
- **Friendly Shell:**
  - alias list='ls'
  - unalias list

PATH
HOME
SHELL
PWD
RANDOM
LOGNAME
USER
HOSTNAME
HISTSIZE
TERM
MANPATH
LD_LIBRARY_PATH

Environment

Startup script files containing configuration

- User specific configuration

~/.bash_profile	#User environment & startup programs, every login
~/.bashrc	# Aliases and Functions, every launch of bash
~/.profile	# Contains configuration accessible by other shells, login console

- System wide Configuration - Applicable on every boot

/etc/profile	# System Environment & startup programs, every login
/etc/bashrc	# Aliases and Functions, every launch of bash
/etc/environment	# Paths for configuration
/etc/profile.d	# Directory containing custom changes for environment

- Using source to export the values

Files System Utilities

Files types in Linux

Everything is File in Linux except CPU and Memory

• **-**	#Regular file
• **d**	#Directory
• **c**	#Character device file ex: /dev/tty, hardware peripherals
• **b**	#Block device file, Hard drive, memory
• **s**	#Local domain socket file, Communication between processes, /dev/log
• **p**	#Named pipe, Communication between two processes
• **l**	#Symbolic link, similar to shortcut or pointer
• soft links	#File name as reference
• ln -s src_file dest_file	# Create a soft link to file
• hard links	#Direct reference to file
• ln src_file dest_file	#Create a hard link to file
• unlink link	

Commands to check file type:

- ls -ld file
- stat file

Special Files Linux

Special Files represent Real Physical Devices

- Character special Files
    - ls -l marks c as the first character
    - Terminal devices
    - Data transfer is one char at a time
- Block special Files
    - ls -l marks b as the first character
    - Disk Devices
    - Data transfer is large fixed size blocks

Inode in Linux

Files are identified by Unique numbers in Linux called I-node

- ls -li
- The Inode doesn't contain file content, instead it has a pointer to that data.

Inode Table

- Created when the file system is created
- df -i   #Check used or free i-nodes

Mounting

Mount- Make a file system accessible at certain point in Linux

- Display all mounted file systems
  - mount
- Mount a CD-ROM under folder /mnt/cdrom
  - mount -t iso9660 /dev/cdrom /mnt/cdrom
- Mount a CD-ROM under folder /mnt/cdrom
  - mount -t vfat /dev/hda1 /win

- **Unmount a mounted device**
  - unmount /mnt/cdrom
- **File systems Table**
  - cat /etc/fstab #File containing info about file systems
  - man fstab

Disk Space

File system Disk Space Availability- Disk Free

• df	#Entire system
• df -h	#Human readable form
• df -m	#Display in MB
• df -i	#Inode information
• df -hT /dir	#Usage for folder
• man df	#More info

Disk Space Usage

- du -h dir
- man du

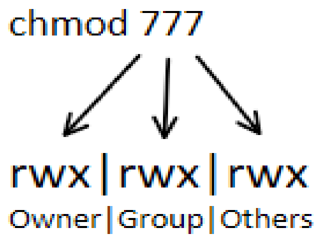
File Permissions

permission	on a file	on a directory
r (read)	read file content (cat)	read directory content (ls)
w (write)	change file content (vi)	create file in directory (touch)
x (execute)	execute the file	enter the directory (cd)

drwxrwxrwx

d = Directory
r = Read
w = Write
x = Execute

7	rwx	111
6	rw-	110
5	r-x	101
4	r--	100
3	-wx	011
2	-w-	010
1	--x	001
0	---	000



Check File Permissions

- ls -l

Default Permissions

- umask #New files only
- umask u=rwx, g=, o=

Changing File Permissions

Absolute (Numeric) mode

- chmod 764 sample

Symbolic mode

- chmod u=rwx, g=rw,o=rw sample
- chmod o-w sample
- chmod g+x sample

Operator	Description
+	Add
–	Remove
=	Override

User Denotion	Description
u	User/Owner
g	Group
o	Others
a	All

Changing Ownership and Group

Syntax:

• chown user sample	#Change Ownership for sample
• chown user:group sample	#Change User and group for sample
• chown group sample	#Change group owner for sample

Change group

- chgrp group file

Info about Groups

• groups	#Lists all groups the user is member of
• newgrp other_group	#Work as a member of other group

Looking for Files and Subdirectories

Find Command

Syntax:

- find [start_path] [Option] pattern

Usage:

- find start_dir -name file #Search for specific file name
- find start_dir -name *.txt #Search for file with pattern
    - find /usr -name 'std*.h‘ #Find all files matching the pattern
- find dest -empty #Find all empty files and dirs in dest
- find dest -perm 664 #Search for file with specified permissions
- find ./ -type d -name dir #Find Directory

- **Search for pattern within found files**
    - find /lib -name *.c | xargs grep -n printf
- **Find and delete file**
    - find ./ -name test.txt -exec rm -i {} \;

- locate command to find file #Searches in database, hence faster
    - locate file
    - locate file -n 10
    - locate -S, updatedb

Additional Commands

Self Exploration

- User Management
- Archives
- Networking
- Process related commands
- Others

Queries???

Thank You !

L&T Technology Services