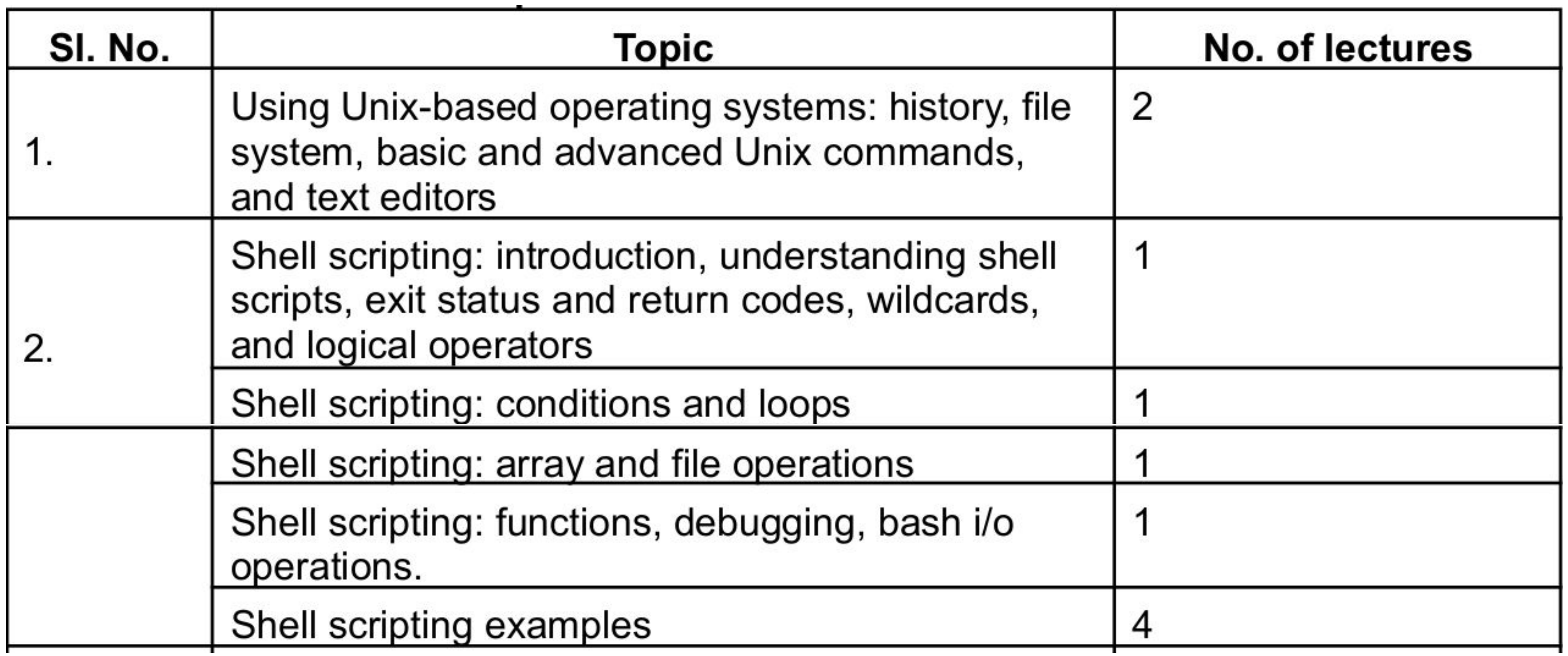
CD61203: Essential Tools for Scientific Computing



Sl. No.	Topic	No. of lectures
1.	Using Unix-based operating systems: history, file system, basic and advanced Unix commands, and text editors	2
2.	Shell scripting: introduction, understanding shell scripts, exit status and return codes, wildcards, and logical operators	1
	Shell scripting: conditions and loops	1
	Shell scripting: array and file operations	1
	Shell scripting: functions, debugging, bash i/o operations.	1
	Shell scripting examples	4

Topic: Basic Linux/Unix Commands

- Linux operating system
- Linux commands
- Vi/vim text editor
- Introduction to bash scripting
- Basic usage

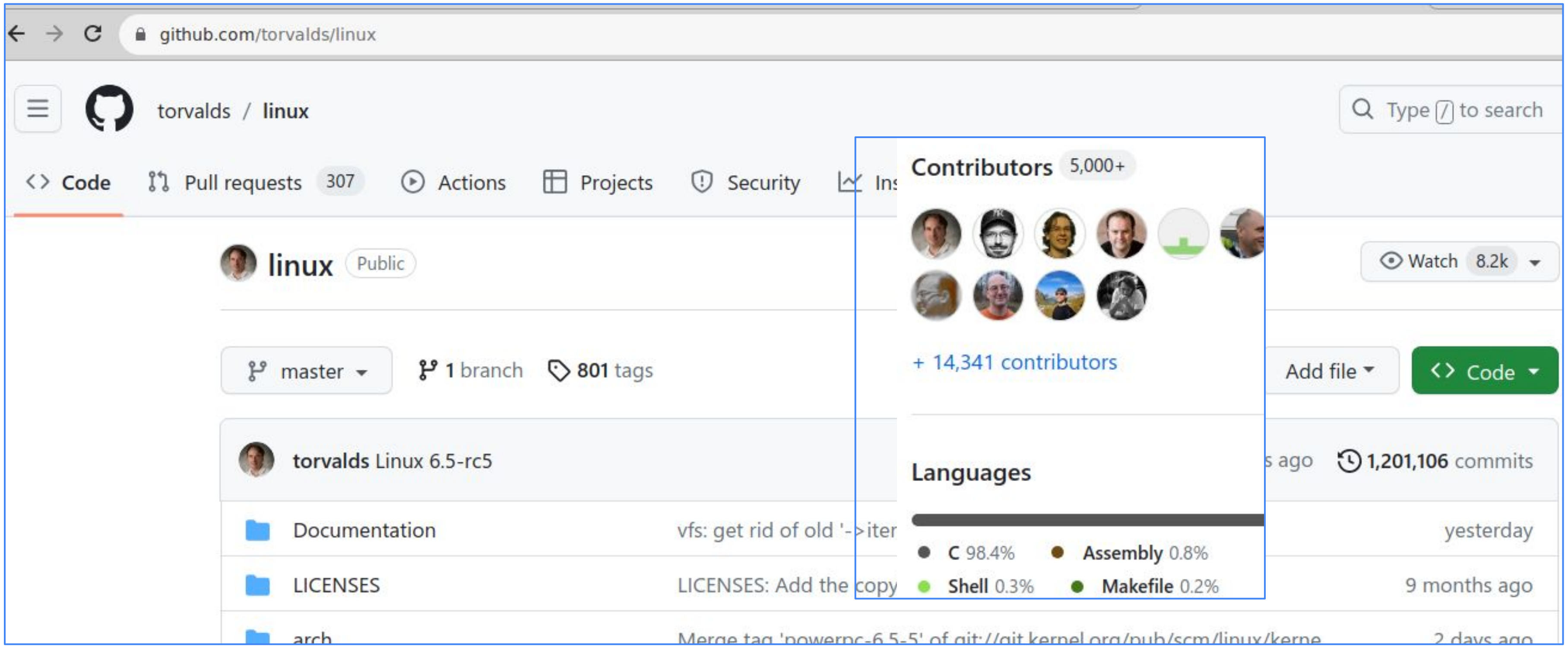
Why learning Linux is helpful

- Command line proficiency
- Server management
- Security and stability
- Development platform
- Flexibility and customization

Unix vs Linux

- Linux is derived from Unix; developed by Linus Torvalds in 1991
- Original Unix is commercial

Linus Torvalds



github.com/torvalds/linux

torvalds / linux

Code | Pull requests 307 | Actions | Projects | Security | Ins

linux Public

master | 1 branch | 801 tags

Contributors 5,000+

+ 14,341 contributors

Languages

● C 98.4% ● Assembly 0.8% ● Shell 0.3% ● Makefile 0.2%

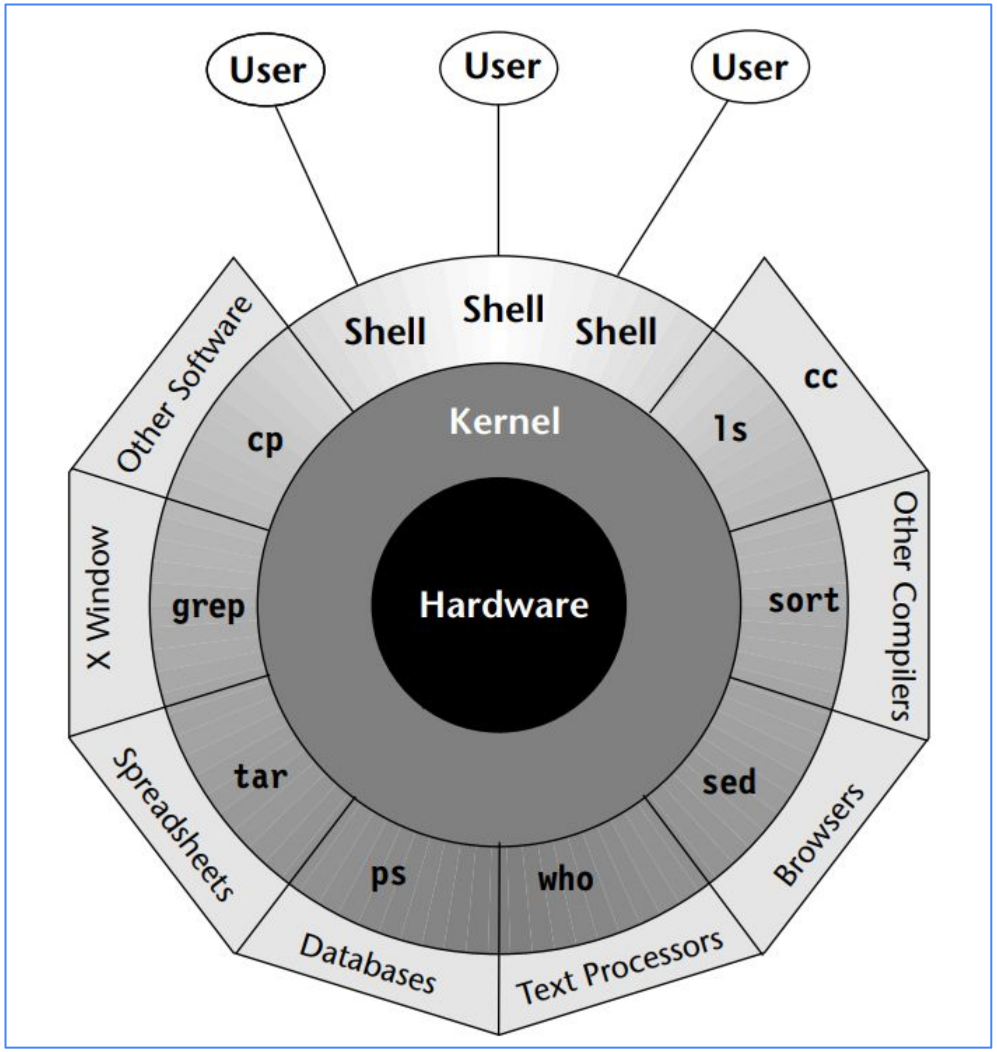
Watch 8.2k

Add file | Code

torvalds Linux 6.5-rc5 | s ago | 1,201,106 commits

Documentation	vfs: get rid of old '->iter	yesterday
LICENSES	LICENSES: Add the copy	9 months ago
arch	Merge tag 'powerpc-6.5-5' of git://git.kernel.org/pub/scm/linux/kerne	2 days ago

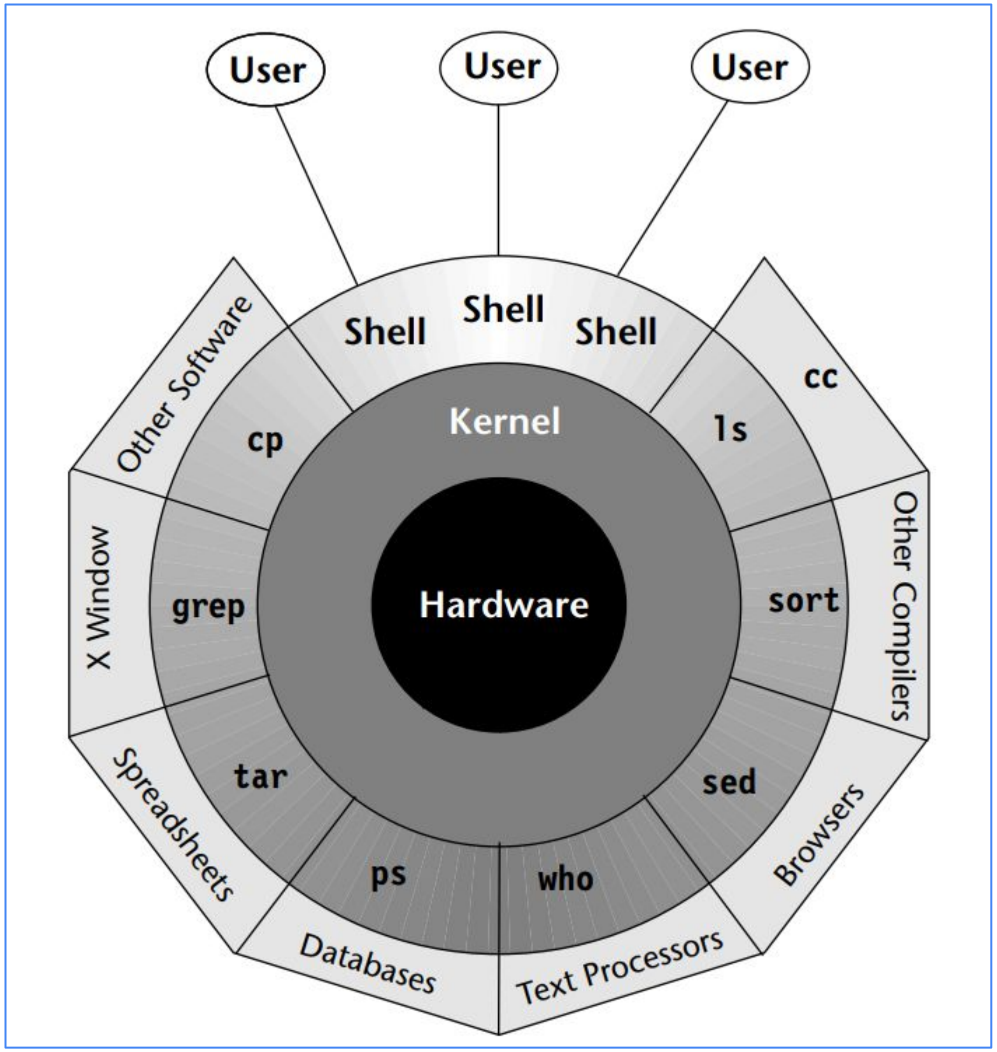
Operating system = Kernel + system applications



Your UNIX/Linux: The Ultimate Guide

- Kernel is core of the OS
- Kernel interacts with machine's hardware while shell interacts with the user
- When programs are not running, the kernel has to do the work. Controls the execution of the program
- Manages system memory, schedule processes, decides their priorities, etc

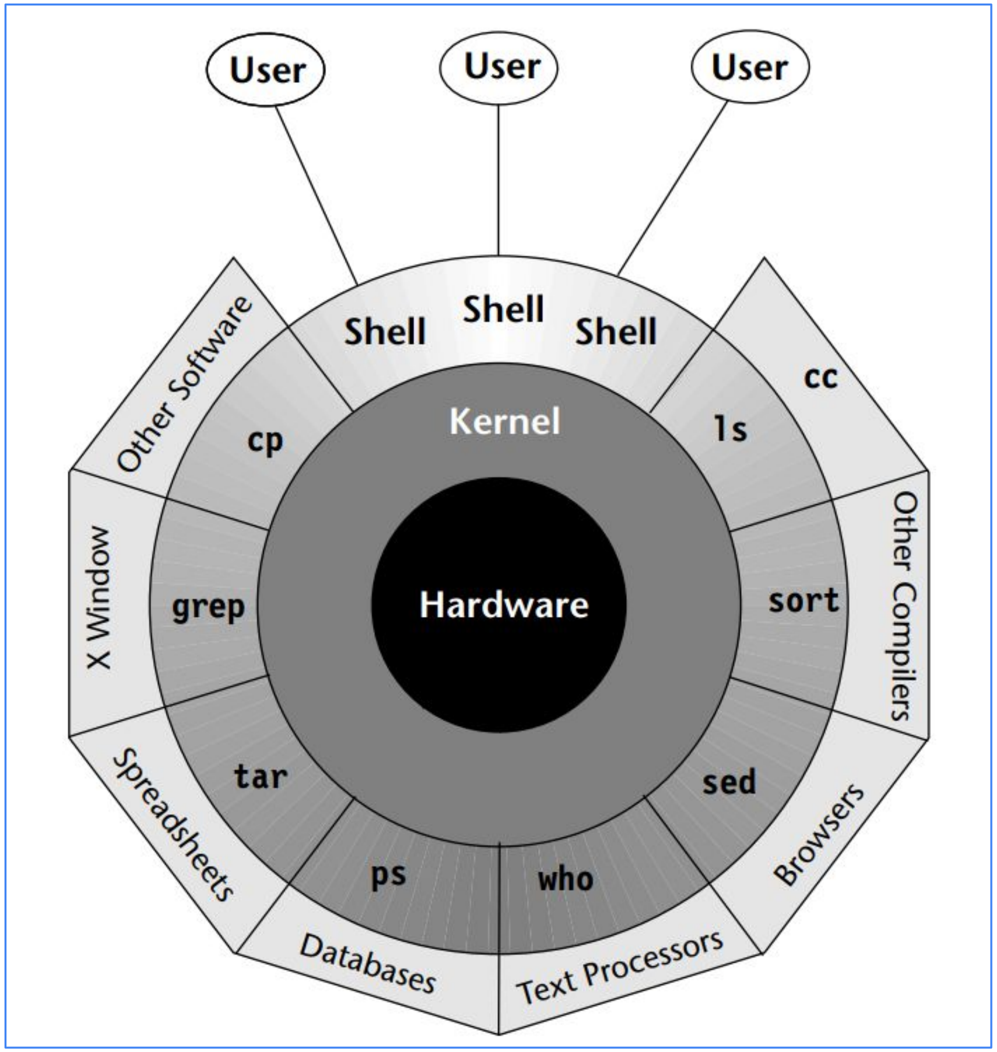
Operating system = Kernel + system applications



- Computers require an interpreter to understand user commands - shell
- Interface between the user and the kernel
- Files & process
- Files - containers for storing static information
- Process - program in execution

Your UNIX/Linux: The Ultimate Guide

Operating system = Kernel + system applications

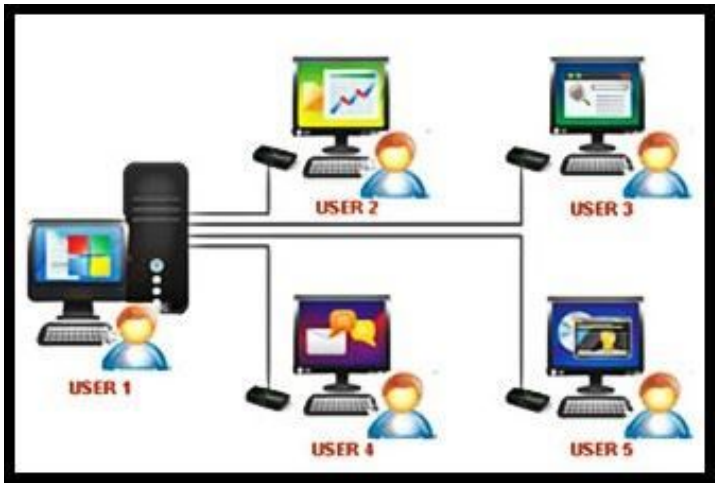


$$\begin{tikzcd}
\boxed{\text{User inputs command via keyboard}} \arrow[d] \\
\boxed{\text{Shell analyzes command, constructs simplified command line}} \arrow[d] \\
\boxed{\text{Shell communicates with Kernel}} \arrow[d] \\
\boxed{\text{Kernel executes the streamlined command}}
\end{tikzcd}$$

Your UNIX/Linux: The Ultimate Guide

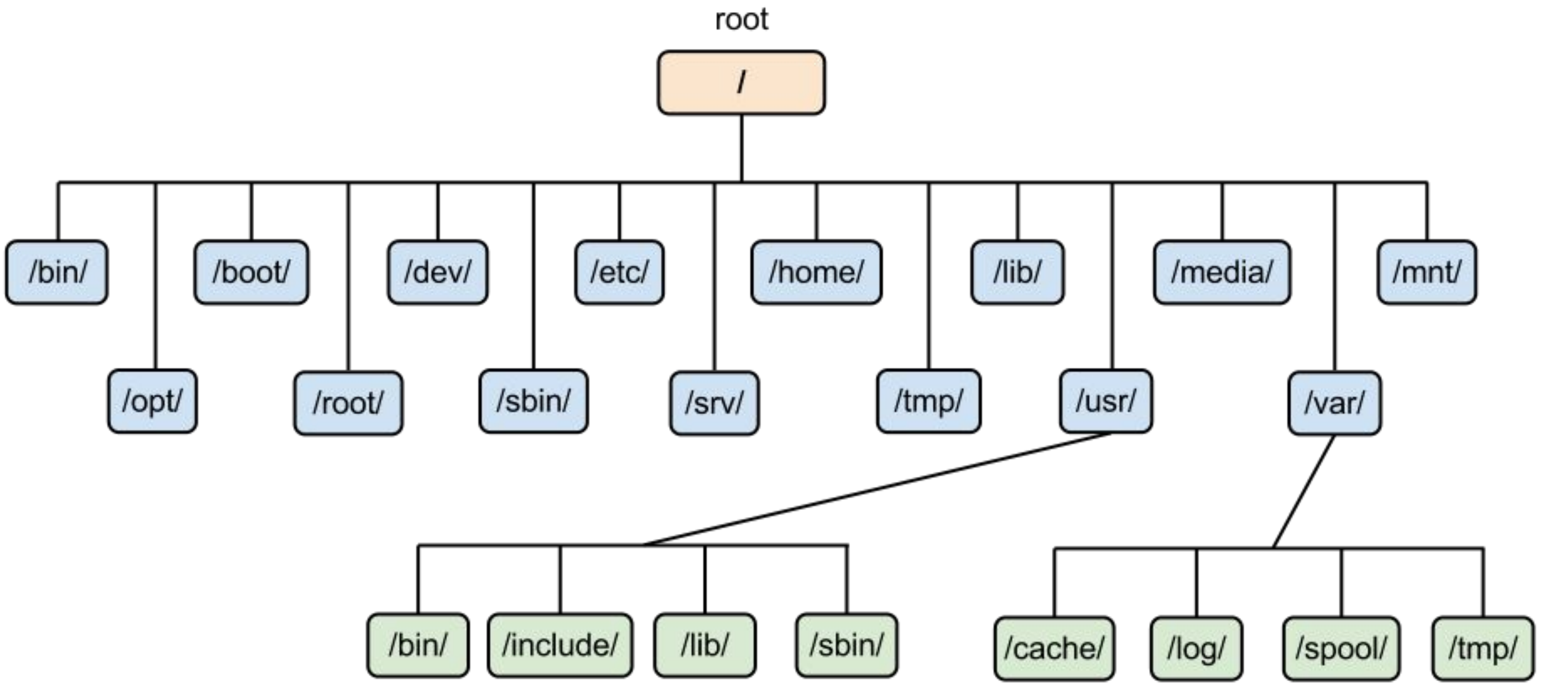
Why Linux

- A Multi-user system
- A multitasking system
- A Repository of applications
- Pattern matching
- Programming facility
- Documentation



https://www.itrelease.com/2014/08/advantages-disadvantages-multi-user-operating-system/

Linux file system



- Directory in Linux is similar to a “Folder” in Windows OS
- Files are organized into directories and sub-directories
- Directories are separated by forward slash (/)

Source: https://freedompenguin.com/articles/how-to/learning-the-linux-file-system/

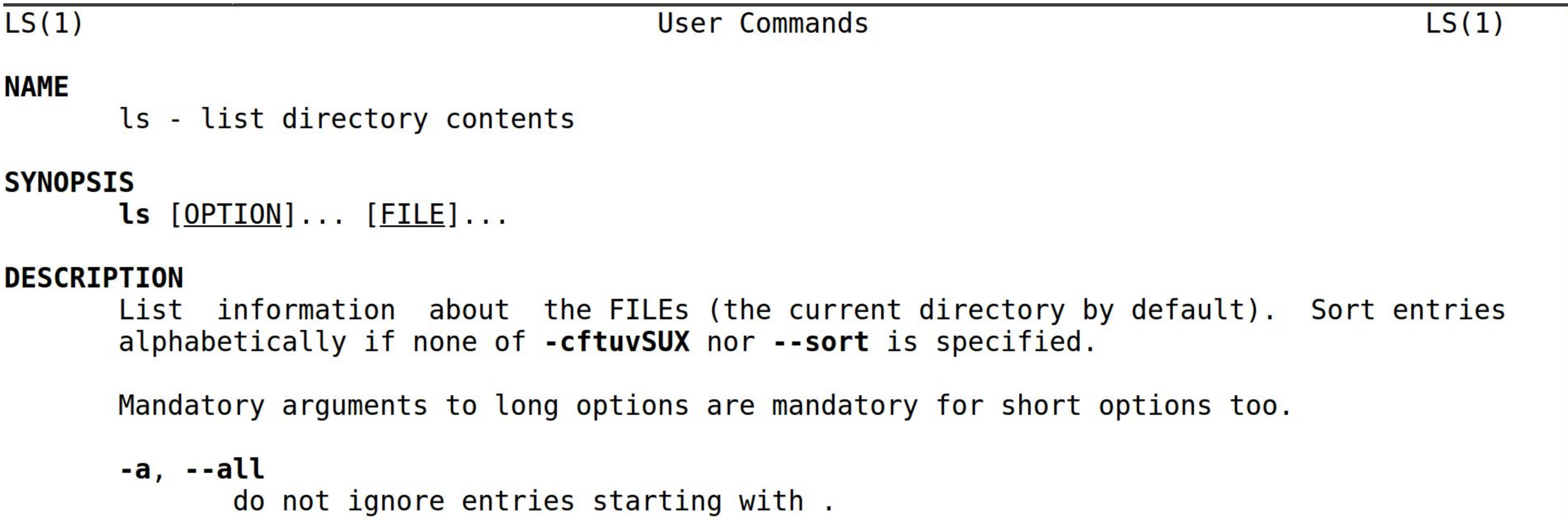
Getting started

- Log in with your username and passwd
- Normal user vs Super user
- Locate terminal
- Applications → System Tools → Terminal
- Program & Process – Process: Executes the program/command or performs the work
- All Linux commands are CASE SENSITIVE
- Type one or two characters of a command and HIT TAB (twice) for knowing the list of all commands starting with these characters

Man page

man

Manual - Provides help information for the command



```
LS(1)                          User Commands                          LS(1)

NAME
       ls - list directory contents

SYNOPSIS
       ls [OPTION]... [FILE]...

DESCRIPTION
       List  information  about  the FILEs (the current directory by default).  Sort entries
       alphabetically if none of -cftuvSUX nor --sort is specified.

       Mandatory arguments to long options are mandatory for short options too.

       -a, --all
              do not ignore entries starting with .
```

Getting started

- “*” character means match everything
- “?” character means match any one character
- [0-4] or [m-s] matches a range of characters
- **Piping**
  - ls -l | grep "file"
  - cat text.txt | wc -w
  - sort file.txt | uniq
  - ps aux | grep "chrome" | awk '{print $2}'
  - echo "Hello, World!" | tr '[:lower:]' '[:upper:]'

Getting started

- ctrl+c halts the current command
- ctrl+z stops the current command and resume it with fg in the foreground or bg in the background
- Login to remote system: ssh
  - ssh -X remote_username@remote_hostname OR

    Eg: ssh -X sandeep@10.3.55.120
- Copying files: scp
  - scp -r remote_username@remote_hostname:~/file1 . OR
  - scp -r file1 remote_username@remote_hostname:~

Getting started

Sl. No.	Command	Description
1	mkdir dir	create directory
2	cd dir	change directory
3	pwd	present working directory
4	ls	list files and directories
5	ls -ltr	sorting the listing by time modification
6	touch file	creates an empty file
7	cp file1 file2	copy files
8	cp -r dir1 dir2	copy directory dir1 to directory dir2
9	mv file1 file2	rename file1 to file2
10	rm -i file	remove file1 (interactive mode)

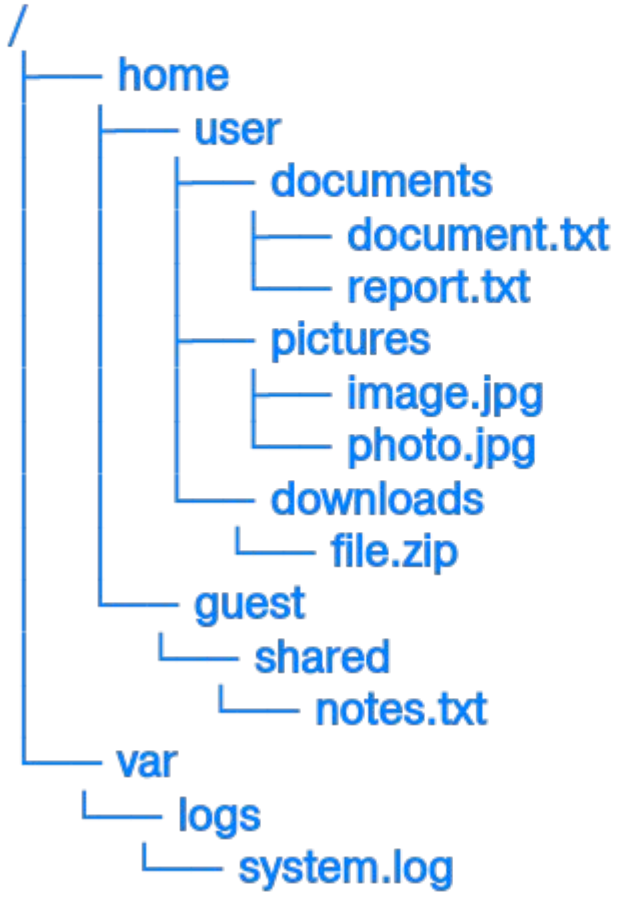
Getting started

Sl. No.	Command	Description
11	rmdir dir1	remove directory
12	history	prints all recent commands to stdout
13	cat file	prints contents of the file
14	head file	outputs first 10 lines
15	tail file	outputs last 10 lines
16	top	display all running processes
17	kill -9 PID	kill the process with process id
18	bg	list all stopped jobs or resume the stopped job in the background
19	fg	Brings the most recent job to the foreground
20	date	displays today date and time

Getting started

Sl. No.	Command	Description
21	grep -i [pattern] file	search for a pattern in file and print it to stdout
22	find -iname “file”	search for a file in current directory recursively
23	./file	executes a file in current directory
24	../../file	executes a file two levels above the current directory
25	zip file.zip file1 file2 or unzip file.zip	compress/uncompress the files

Absolute and relative path



```
/
├── home
│   ├── user
│   │   ├── documents
│   │   │   ├── document.txt
│   │   │   └── report.txt
│   │   ├── pictures
│   │   │   ├── image.jpg
│   │   │   └── photo.jpg
│   │   └── downloads
│   │       └── file.zip
│   └── guest
│       └── shared
│           └── notes.txt
└── var
    └── logs
        └── system.log
```

Absolute path

- /home/guest/shared/notes.txt
- /var/logs/system.log

Relative path (assuming you are in /home/user)

- documents/document.txt
- ../guest/shared/notes.txt

Relative path (assuming you are in /var/logs)

- ../../home/guest/shared/notes.txt

Echo

- echo [options] [string(s)]
- Examples:
  - echo "Hello, World!"
  - name="CD61203" ; echo "Hello, $name!"
  - echo "New content" >> file.txt
  - echo -e "Column 1\tColumn 2\tColumn 3"
  - text="Hello World" ; echo "$text" | tr '[:lower:]' '[:upper:]'
  - text=”Hello World”; echo "Length of $text is ${#text}"
  - result=$(echo "5 + 7" | bc) ; echo "$result"
  - result=$(echo "sqrt(25)" | bc -l) ; echo "√25 = $result"

File permissions

- File Permissions are represented using characters:
  - r: Read permission
  - w: Write permission
  - x: Execute permission
  - Type ls -l to see file permissions

File permissions

- Examples:
    - rw-r--r--: Owner has read and write permissions, group and others have read-only permissions.
    - rwxr-x---: Owner has read, write, and execute permissions, group has read and execute permissions, others have no permissions.
    - rwxrwxrwx: All users have full read, write, and execute permissions. This is not recommended for security reasons, as it gives unrestricted access.

File permissions

- Examples:
  - Read-only file
    - `-r--r--r-- 1 owner group 100 Aug 1  file.txt`
  - Executable File:
    - `-r-xr-xr-x 1 owner group 100 Aug 1  script.sh`
  - Full Access for Owner:
    - `-rwx------ 1 owner group 100 Aug 1  file.txt`

Changing file permissions

- Use the 'chmod' command to modify permissions
- chmod [options] permissions file(s)
- Examples:
  - Add execute permission: chmod u+x file.txt
  - Remove read for group/others: chmod go-r file.txt
  - Set rwx for owner, rx for group&other: chmod 755 script.sh

    here r=4, w=2 and x=1

File ownership

- Files have an owner and a group associated with them.
- Owners create files, groups provide shared access.
- Use chown to change ownership
  - chown [options] new_owner:new_group files
- Examples:
  - chown hpc:ccds file.txt
  - chown :ccds file.txt
  - chown root:root dir/
  - chown -R user:group dir/

grep

- grep [options] pattern [file...]
- Examples:
  - Basic text search: grep "keyword" file.txt
  - Case-Insensitive search: grep -i "word" file.txt
  - Counting matches: grep -c "pattern" file.txt
  - Display line numbers: grep -n "pattern" file.txt
  - Whole word match: grep -w "word" file.txt
  - Invert match: grep -v "pattern" file.txt
  - display the names of files with a match: grep -l "pattern" file

grep

- grep [options] pattern [file...]
- Examples:
  - Recursive search: grep -r "pattern" directory/
  - Regular expressions: grep "^[0-9]*$" numbers.txt
  - Matching multiple patterns: grep "pattern1\|pattern2" file.txt
  - Show context around match: grep -C 2 "pattern" file.txt
  - Search file types: grep "pattern" --include "*.txt" dir/
  - Exclude file types: grep "pattern" --exclude "*.txt" dir/

Arithmetic Computation

- expr: Evaluate arithmetic expressions and perform string operations.
  - Example: expr 5 + 3 outputs 8 (note spaces between 5,+,3)
- bc: An arbitrary precision calculator language.
  - Example: echo "5 + 3" | bc outputs 8.
- awk, sed: NEXT LECTURE
- (( )): Shell arithmetic expansion for integer computations.
  - Example: ((x = 5 + 3)) followed by echo $x outputs 8. OR
  - Example x=$((5+3)) followed by echo $x outputs 8.
- let: Perform arithmetic operations inside a shell script.
  - Example: let "x = 5 + 3" followed by echo $x outputs 8.

File Operations

- tar: Back up files specified in the command line.
- cp -r: Copy directory tree.
- cp: Copy file.
- mv: Move files to another directory
- rm: Delete files and directories.
- touch: Create an empty file or update the modification timestamp of a file.
- mkdir and rmdir: Create and remove directories
- ln: Create hard or symbolic (soft) links to files or directories.
  - eg: ln -s /home/hpc/Downloads/file1.txt .

File Permissions and Ownership

- chgrp: Change file's group ownership.
- touch: Change file's last modification or access time.
- chown: Change file's ownership.
- chmod: Change file's permissions.

Text Processing

- head: Beginning of file.
- tr: Translate or delete characters
- cut: Cut columns or fields from a file.
- sort: Lines in ASCII collating sequence.
- sort -n: Lines in numeric sequence.
- tail -r: Lines in reverse order.
- sort -f: Lines sorted ignoring case.
- uniq -d: Lines that are repeated.
- uniq -u: Lines that occur only once.
- tr -s: Squeeze multiple spaces to a single space.

File Compression and Archiving

- bzip2: Compress file (to .bz2).
  - Example: bzip2 file.txt
- gzip: Compress file (to .gz).
  - Example: bzip2 file.txt
- zip: Compress multiple files to a single file (to .zip)
  - Example: zip files.zip file1.txt file2.txt file3.txt
- unzip: Uncompress .zip file.
  - Example: unzip files.zip

File Display and Manipulation

- cat: Concatenate files.
  - Display File Content: cat file.txt
  - Concatenate Multiple Files: cat file1.txt file2.txt > combined.txt
  - Append File Content: cat file1.txt >> file2.txt
  - cat file.txt | head -n 10
- Directory list
  - ls -l | grep "^d"
  - ls -p . | grep /
- du: Disk space utilization.
- tail -f: Monitor growth of a file.

File Display and Manipulation

- du: Disk space utilization.
  - Display directory usage: du -h directory/
  - Show total disk Usage: du -sh directory/
  - Sort and display largest directories: du -h directory/ | sort -rh | head -n 10
  - Display files with size > 100M: find . -type f -size +100M
- tail -f: Monitor growth of a file.

File Display and Manipulation

- tail: Display the last few lines (by default 10 lines)
  - Monitor growth of a file: tail -f file.txt
  - Display the last 10 lines of a file and follow changes:

    tail -n 10 -f file.txt
  - Monitor multiple files: tail -f file1.txt file2.txt file3.txt
- watch
  - Watch changes for every second: watch -n 1 "cat file.txt”

System and User Information

- hostname: Name of the local host.
- uname: Operating system name.
- uname -r: Operating system release.
- who -r, runlevel: System run level.
- who: Users and their activities.

User Account and Security

- passwd: Change own password.
- su: Superuser from a nonprivileged account.
- exit, logout: Terminate shell script.

SSH and Remote Access

- ssh: Log in to a remote machine.
  - `ssh username@remote_host`
  - `ssh -p port_number username@remote_host`
- scp, sftp: Copy file between machines.
  - Copy local to remote machine:

    ```
    scp local_file.txt username@remote_host:/path
    ```

  - Copy Remote machine to Local:

    ```
    scp username@remote_host:/remote_path/file.txt local_path/
    ```

System and Process Control

- uptime: Show system uptime
- top: Free space in memory and swap
- htop: Interactive process viewer
- kill: Terminate process by PID:
  - kill PID or kill -9 PID
- bg & fg: Move job to the background or foreground
- nohup: Run a command in the background
  - nohup command &
- reboot: Reboot the system

Environment and Variables

- set: Assign values to positional parameters
- export: Pass variable value to sub-shell
    - export VAR_NAME=value

Input and Output

- read: Input data to a shell script interactively

  -
    ```
    echo "Enter your name:"

    read name

    echo "Hello, $name!"
    ```
- paste: Join two files laterally
  - `paste file1.txt file2.txt`

System Date and Time

- date
  - Current date: `date +"%Y-%m-%d"`
  - Current time: `date +"%H:%M:%S"`
  - Future date: `date -d "+3 days" +"%Y-%m-%d"`
  - Display year: `date +%Y`
  -
    ```
    start_date="2023-01-01"
    end_date=$(date +"%Y-%m-%d")
    diff=$((($(date -d "$end_date" +%s) - $(date -d "$start_date" +%s)) / 86400))
    echo "Days between $start_date and $end_date: $diff"
    ```

References

1. Your Unix :The Ultimate Guide, Das, Sumitabha