

CD61203: Essential Tools for Scientific Computing

[syllabus_CD61203_autumn2025-26.pdf](#)

Grading

WEIGHTAGE given for each component,

- Mid-term exam: 30% (Syllabus: SKR + LR + SM)
- End-term exam: 30% (Syllabus: All topics)
- Classroom participation, class tests, etc: 20%
- Term project: 20%

IMPORTANT

Mid-term and end-term semester exams are conducted in the PC lab without network connectivity

Open notebook exams: TWO A4 sheets are allowed during the exams

CD61203: Essential Tools for Scientific Computing

First two weeks

Teacher	Topic	No. of lectures (in hr)	Date
SR	Using Unix-based operating systems: history, file system, basic and advanced Unix commands, and text editors	4	W1-W1 28 Jul – 01 Aug
	Python programming: installation, environment setup and modules		

Topic: Basic Linux/Unix Commands

- Linux operating system
- Linux commands
- Vi/vim text editor
- 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



torvalds / linux
Public

Notifications
 Fork 57.1k
 Star 198k

[Code](#)
[Pull requests](#) 481
 [Actions](#)
[Projects](#)
[Security](#)
[Insights](#)

master
1 Branch
893 Tags

Code

torvalds Merge tag 'drm-fixes-2025-07-24' of ht... 25fae0b · 4 hours ago 1,369,295 Commits

Documentation	Merge tag 'char-misc-6.16-rc7' of git://g...	4 days ago
LICENSES	LICENSES: add CC0-1.0 license text	2 months ago
arch	Merge tag 'for-linus' of git://git.kernel....	8 hours ago
block	Merge tag 'block-6.16-20250718' of git:...	5 days ago
certs	sign-file,extract-cert: use pkcs11 provid...	10 months ago

Contributors 5,000+
 [+ 16,347 contributors](#)

Languages

C 98.3%

Assembly 0.7%

Shell 0.4%

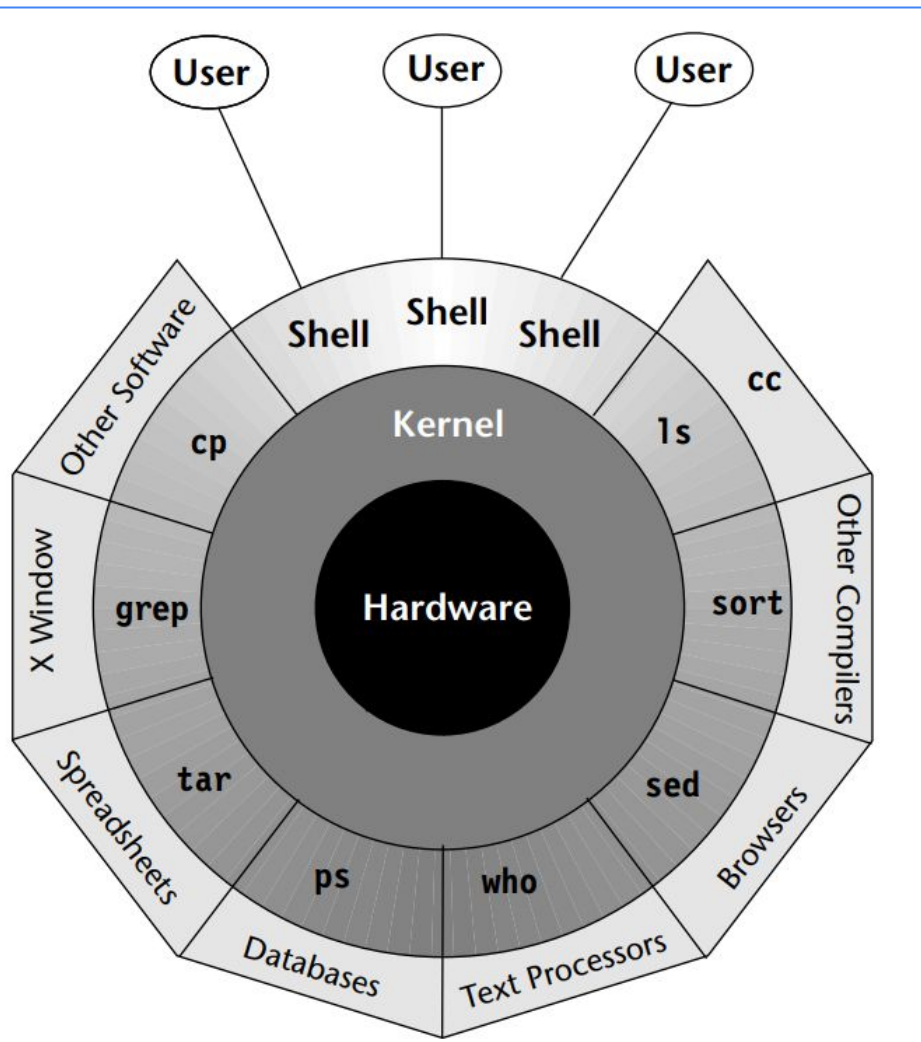
Python 0.2%

Makefile 0.2%

Rust 0.1%

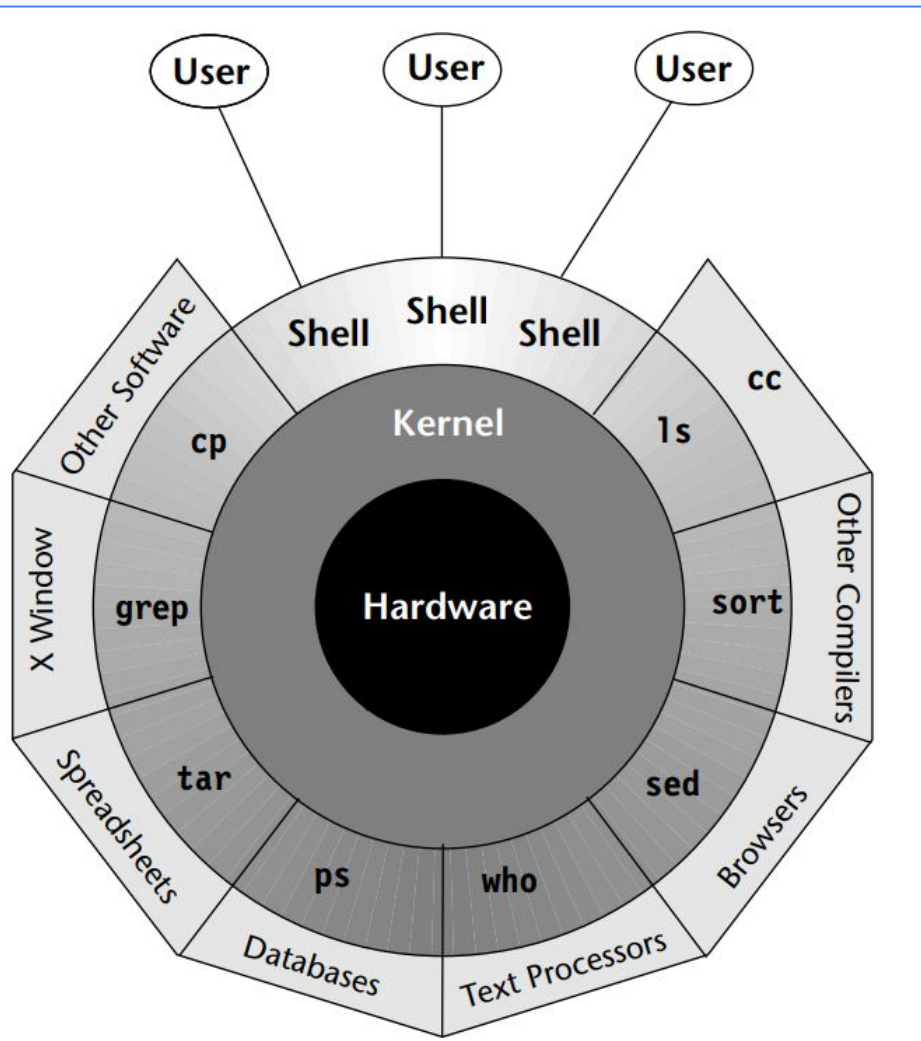
Other 0.1%

Operating system = Kernel + system applications



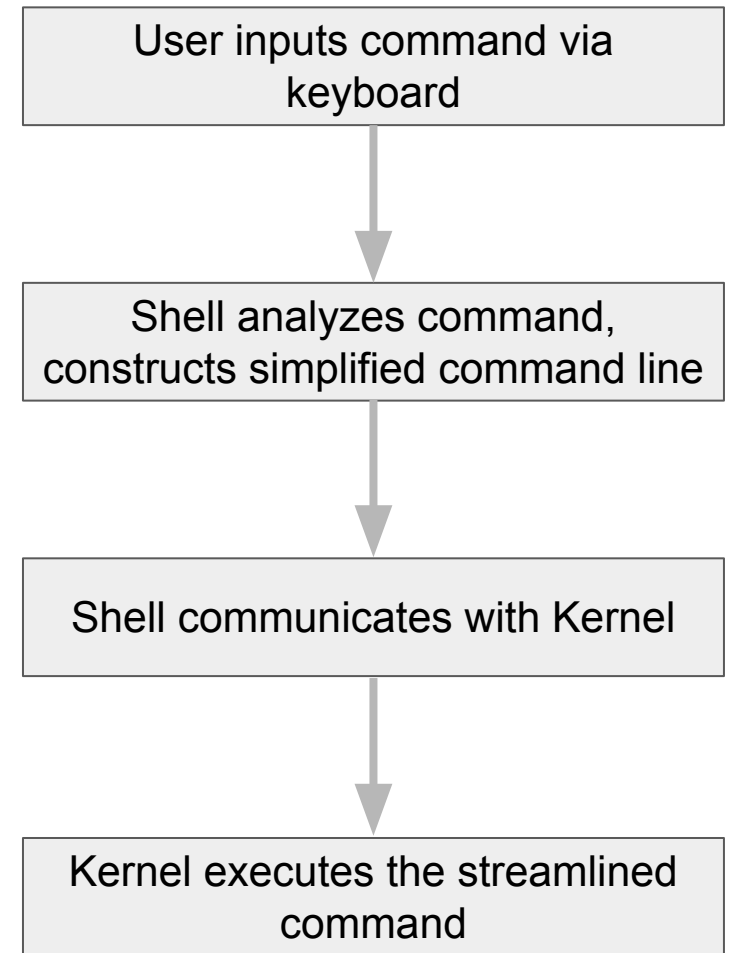
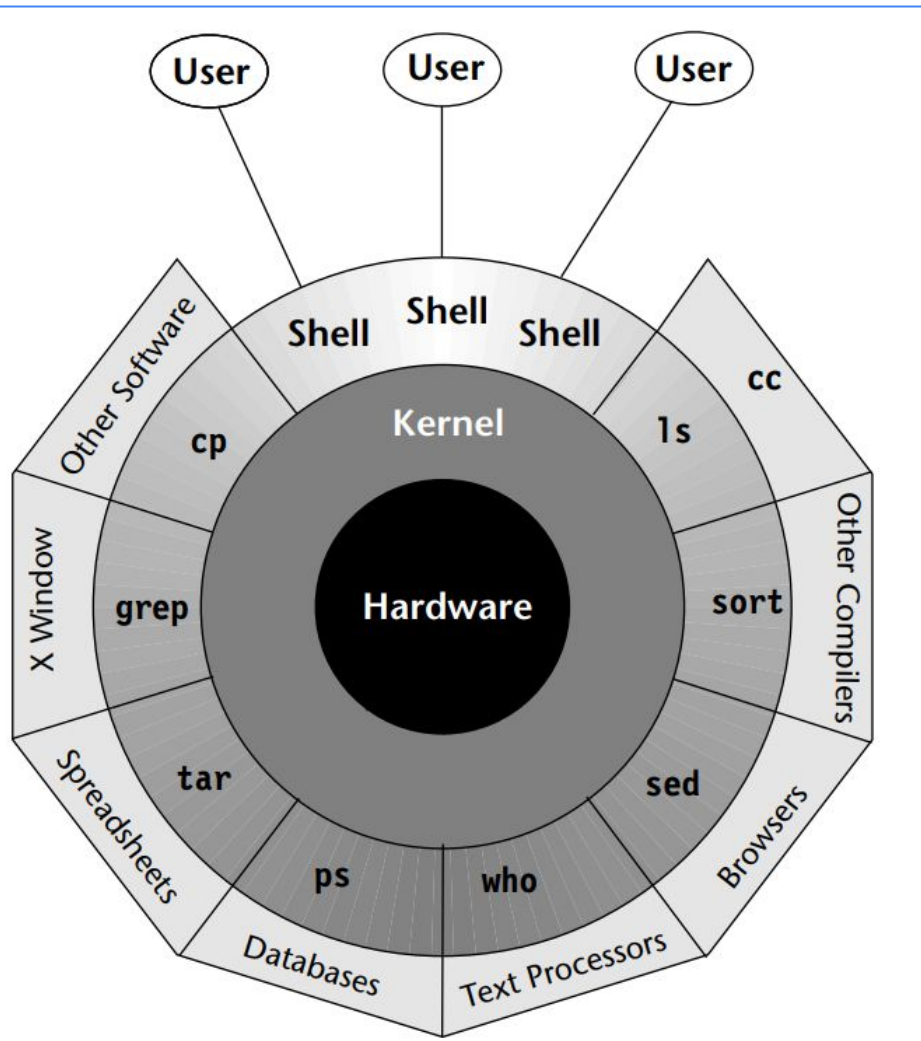
- Kernel is core of the OS
- Kernel interacts with machine's hardware while shell interacts with the user
- When programs are not running, the kernel has to do the work. Controls the execution of the program
- Manages system memory, schedule processes, decides their priorities, etc

Operating system = Kernel + system applications



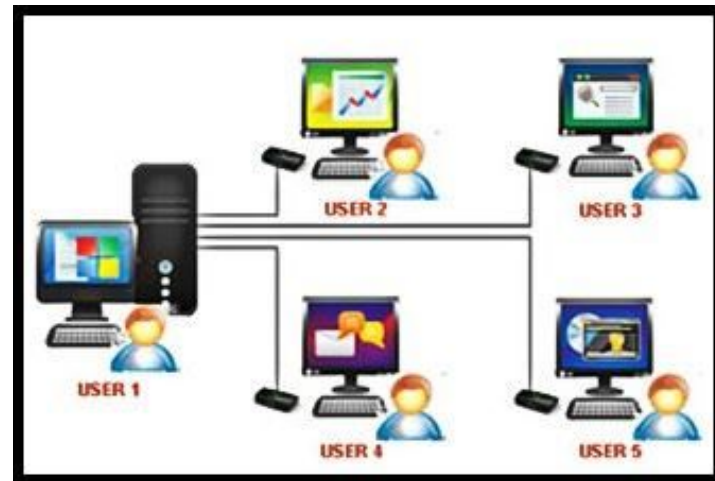
- Shell is an interpreter: computers require an interpreter to understand user commands
- Interface between the user and the kernel
- Files & process
- Files - containers for storing static information
- Process - program in execution

Operating system = Kernel + system applications

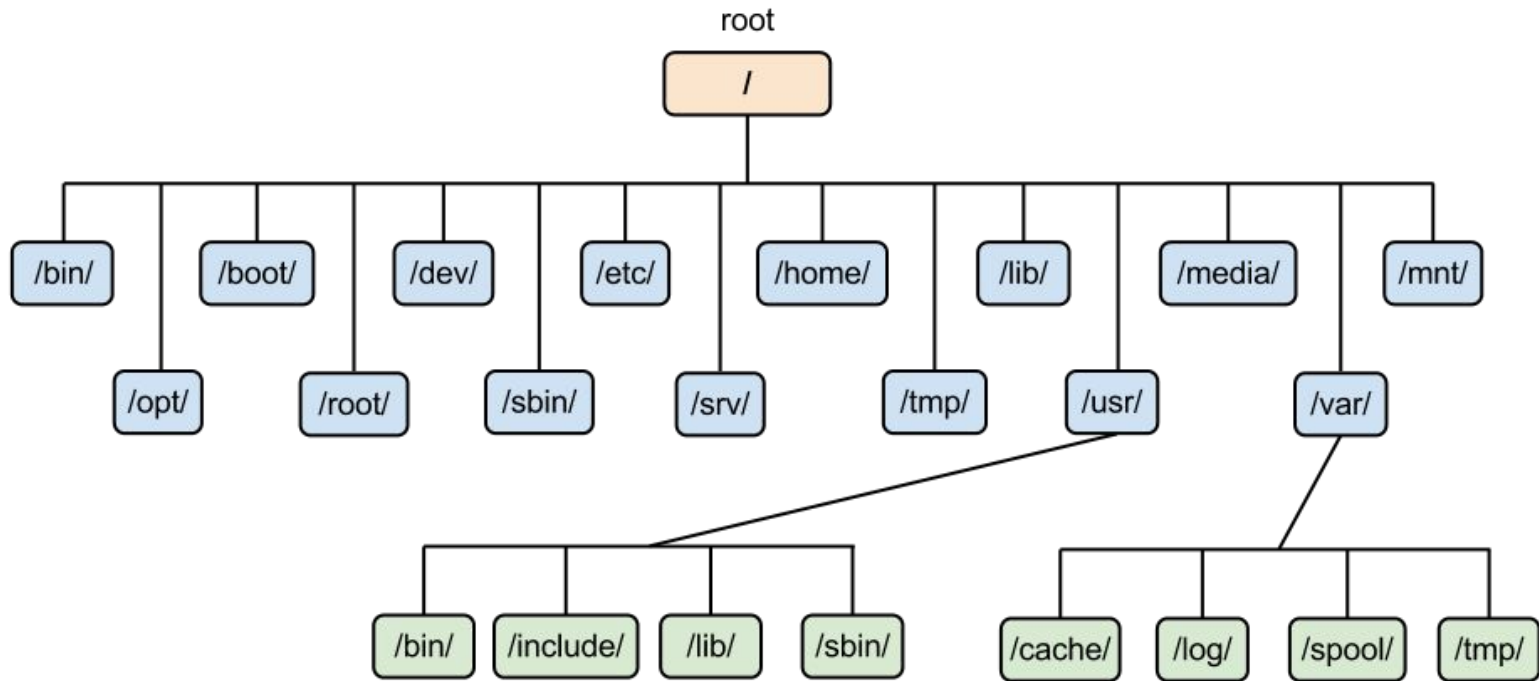


Why Linux

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



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 [<u>OPTION</u>]... [<u>FILE</u>]...		
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

Sl. No.	Command	Description
1	<code>mkdir dir</code>	create directory
2	<code>cd dir</code>	change directory
3	<code>pwd</code>	present working directory
4	<code>ls</code>	list files and directories
5	<code>ls -ltr</code>	sorting the listing by time modification
6	<code>touch file</code>	creates an empty file
7	<code>cp file1 file2</code>	copy files
8	<code>cp -r dir1 dir2</code>	copy directory dir1 to directory dir2
9	<code>mv file1 file2</code>	rename file1 to file2
10	<code>rm -i file</code>	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	<code>egrep -i [pattern] file</code>	search for a pattern in file and print it to stdout
22	<code>find -iname "file"</code>	search for a file in current directory recursively
23	<code>./file</code>	executes a file in current directory
24	<code>../../file</code>	executes a file two levels above the current directory
25	<code>zip file.zip file1 file2</code> or <code>unzip file.zip</code>	compress/uncompress the files

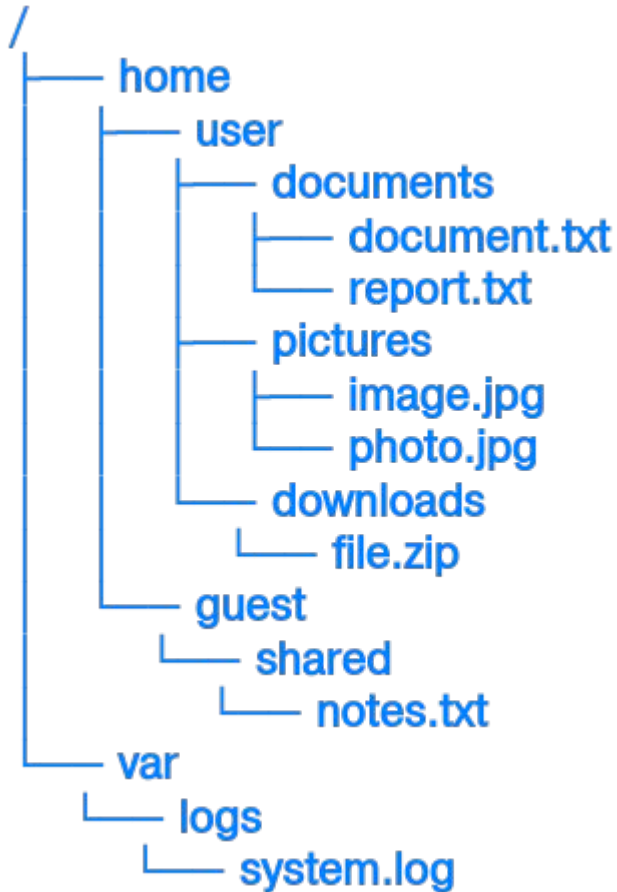
Getting started

- “*” character means match everything
- “?” character means match any one character
- [0-4] or [m-s] matches any single character in the range
- **Piping**
 - `ls -l | egrep "file"`
 - `cat text.txt | wc -w`
 - `sort file.txt | uniq`
 - `ps aux | egrep "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:~`

Absolute and relative path



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 " $\sqrt{25} =$ $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/`

egrep

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

egrep

- `egrep [options] pattern [file...]`
- Examples:
 - Recursive search: `egrep -r "pattern" directory/`
 - Regular expressions: `egrep "^[0-9]*$" numbers.txt`
 - Matching multiple patterns: `egrep "pattern1|pattern2" file.txt`
 - Show context around match: `egrep -C 2 "pattern" file.txt`
 - Search file types: `egrep "pattern" --include "*.txt" dir/`
 - Exclude file types: `egrep "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.
- `sort -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 | egrep "^d"**
 - **ls -p . | egrep /**
- **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** or **du -h**
 - Show total disk Usage: **du -sh**
 - Sort and display largest directories: **du -h | 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`: 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