Unix Detialed Notes Accroding Syllabus

Unit-1

1. Overview of UNIX

UNIX is a multi-user, multitasking operating system created at AT&T Bell Labs in the early 1970s to provide a portable, powerful environment for programmers. It follows the idea of many small command-line tools that can be combined, instead of one big program, which makes it very flexible.[1][2][3]

Hindi video (general UNIX / Linux intro):

- "Introduction to Unix Operating System in Hindi" – good short theory overview. https://www.youtube.com/watch?v=qVMZNFSeBT8[4]

2. History and Features of UNIX

Originally, UNIX was written in assembly for a PDP-7, but it was soon rewritten in C language, which made it easy to move to many different machines and led to many versions like System V and BSD. Over time, its design influenced almost all modern operating systems, including Linux and macOS.[3][5][1]

Main features you should remember:

- Multi-user and multitasking with time-sharing of CPU.
- Portability due to C implementation.
- Strong security with user accounts and file permissions.
- Hierarchical file system starting at root /.
- Rich set of small utilities that can be piped together.[2][6]

Hindi video (history + features):

- "Unix History – Hindi tutorial" – explains why UNIX was created and its main advantages. https://www.youtube.com/watch?v=tmjxQlZ3ThA[7]

3. UNIX Architecture

UNIX is usually shown as four layers: **hardware → kernel → shell → utilities/applications**. The kernel runs in privileged mode and manages hardware, while shells and utilities run in user mode and request services from the kernel.[5][8][9]

Hindi video (architecture explanation):

- "UNIX / Linux Architecture in Hindi" section inside this general intro gives a diagram-based explanation.
  https://www.youtube.com/watch?v=Vz0wmUWzgys[10]

4. Kernel – Core of the System

The **kernel** is the heart of UNIX that always resides in memory and controls everything. It handles:[8][1]

- **Process management:** creating, scheduling, and terminating processes so many programs can run together safely.[9][8]
- **Memory management:** giving each process its own protected memory and using techniques like paging or swapping if RAM is low.[11][8]
- **File system management:** creating, reading, writing, and deleting files and directories using a unified interface.[12][2]
- **Device management:** accessing hardware (disks, terminals, printers, network cards) via device drivers.[13][8]
- **System call interface:** offering functions like `open`, `read`, `write`, `fork`, `exec` that user programs call to get kernel services.[14][2]

Hindi video (kernel concept in OS / UNIX):

- "Operating System Full Course in Hindi – Process & Kernel basics" (relevant lecture in playlist).
  https://www.youtube.com/playlist?list=PLspGXahxbrNz0Cz2_X9UZKNIPmz-pCelz[15]

5. Shell – Command Interpreter

The **shell** is a user-level program that reads commands from keyboard or script, interprets them, and asks the kernel to carry them out. It also performs expansions (variables, wildcards), I/O redirection and piping before commands run.[6][2][13]

Common shells are Bourne shell (`sh`), Bash, C shell (`csh`) and Korn shell (`ksh`), each with slightly different features but same basic role. Many users can run different shells at the same time, all talking to the same kernel.[16][5][9][14]

Hindi video (shell and basic commands):

- "Linux Tutorial For Beginners in Hindi" – long video but early parts clearly show shell prompt, entering commands and getting output.
  https://www.youtube.com/watch?v=_tCY-c-sPZc[17]

6. Basic UNIX File System View

Even though detailed file system is Unit-2, Unit-1 expects a simple overview. UNIX keeps all files in a **single tree** starting from root directory `/` instead of drive letters. Under root there are standard directories such as `/bin`, `/etc`, `/home`, `/tmp`, and `/dev`, each with a specific purpose.[18][19][12]

This design makes it easy to mount extra disks or partitions anywhere under the tree, and the user just sees a continuous structure of directories and files.[12][18]

Hindi video (file system basics):

- In the same CodeWithHarry Linux tutorial, there is a section on Linux/UNIX file system and directory structure with clear Hindi explanation.
  https://www.youtube.com/watch?v=_tCY-c-sPZc[17]

7. Login and Logout Process

Because UNIX is multi-user, every person must log in before using system resources.[20][6]

7.1 Login steps

1. A program (like `getty` or graphical greeter) shows a **login prompt** on the screen or terminal.[21][6]
2. The user types their **username**; system checks it in `/etc/passwd` and related files.[11][21]
3. System asks for **password**, compares it with stored encrypted password, and if correct, authenticates the user.[20][21]

4. Kernel starts the user's default **shell**, sets environment variables, and places user in their home directory.[22][21]

7.2 Logout steps

- When finished, user types `exit` or `logout` or closes the terminal; the shell process ends and any remaining jobs are handled according to system rules.[23][22]
- System returns to login prompt, ready for the next session.[21][20]

Hindi video (logging in, terminal use):

- "Unix Tutorial for Beginners (Hindi)" – shows terminal opening, login/terminal session and simple logout.
  https://www.youtube.com/watch?v=IrDUcdpPmdI[24]

8. How to Study Unit-1 with These Videos

- First read these notes once to understand the **big picture** of UNIX: what it is, its architecture, kernel, shell and file-system idea.[2][8]
- Then watch the Hindi videos in the order given (intro → history → architecture → shell/commands → file system → login-logout) and pause to write your own short bullet points for each heading.[10][17].

Unit-2

1. Idea of the UNIX File System

UNIX treats almost everything as a file, including normal data, directories and many devices, and stores them in **one big tree** that starts at root `/`. There are no drive letters like C: or D:; extra disks are "mounted" somewhere under this tree so users just see more directories and files.geeksforgeeks+1

Important system directories include `/bin` (basic commands), `/etc` (configuration), `/home` (users' home folders), `/tmp` (temporary files) and `/dev` (device files mapping to hardware).geeksforgeeks+1

Hindi video (file system intro):

- Linux Tutorial For Beginners in Hindi – section on Linux file system and important directories.
  https://www.youtube.com/watch?v=_tCY-c-sPZc[youtube]

2. File Types and Directory Structure

Every entry in a directory has a name and points to an **inode**, which stores type, size, owner, permissions and data block addresses. From the user point of view the main file types are:[geeksforgeeks]

- **Regular file:** Normal data or program file, can be text, image, binary, etc.
- **Directory:** Special file that holds a list of names and their inodes; this creates the tree structure.
- **Character/block special file:** Represents devices (e.g. terminals, disks) under `/dev`.
- **Symbolic link:** "Shortcut" file that points to another pathname.
- **FIFO/pipe and socket:** Used for inter-process communication.[homepages.uc+1]

The directory structure is hierarchical: root `/` at top, then subdirectories forming paths such as `/home/rahul/docs/report.txt`.[geeksforgeeks]

Hindi video (file & directory basics):

- Basic Linux Commands FULL DEMO in Hindi – demonstrates files, folders and navigation.
  https://www.youtube.com/watch?v=4FPiPcEhjYw[youtube]

3. Absolute and Relative Paths

A **path** tells UNIX where a file or directory is in the tree.

- **Absolute path:**
  - Starts from root `/` and shows full location, e.g. `/home/rahul/notes/unix.txt`.
  - Always works, independent of current directory.[tutorialspoint+1]
- **Relative path:**
  - Written from current working directory, does not start with `/`.
  - Uses `.` (current) and `..` (parent), e.g. `../bin`, `./script.sh`.[hpc.iastate+1]

Absolute paths are safer in scripts; relative paths are faster to type when working interactively.

Hindi video (absolute vs relative path):

- In CodeWithHarry's Linux tutorial, there is a clear explanation with examples. https://www.youtube.com/watch?v=_tCY-c-sPZcyoutube

4. Basic File and Directory Commands

These commands are used daily and are important for viva and lab exams.geeksforgeeks+1

- `pwd` – show present working directory.
- `ls` – list files; `ls -l` gives long listing, `ls -a` shows hidden files.tutorialspoint
- `cd directory` – change current directory (`cd ..` goes to parent, `cd` alone goes to home).
- `mkdir name` – create a new directory.
- `rmdir name` – remove an empty directory.
- `cp source dest` – copy file or directory; options like `-r` copy recursively.
- `mv old new` – move or rename files/directories.
- `rm file` – remove file; `rm -r` removes directory tree, so it must be used carefully.
- `cat`, `more`, `less` – view file contents.tutorialspoint+1

Hindi video (core file commands):

- Basic Linux Commands FULL DEMO in Hindi – covers `pwd`, `cd`, `ls`, `cp`, `mv`, `rm`, `mkdir`, `rmdir`, `cat`, `more` etc. https://www.youtube.com/watch?v=4FPiPcEhjYwyoutube

5. File Permissions and Ownership

UNIX protects files using **permissions** and **ownership**.geeksforgeeks+1

5.1 Ownership

Every file has:

- **Owner (user ID):** usually the creator of the file.
- **Group (group ID):** a set of users who may share some rights.startertutorials+1

The system stores these IDs in the inode; user information comes from `/etc/passwd` and group info from `/etc/group`.geeksforgeeks+1

5.2 Permission bits

For each file or directory there are three classes: **owner, group, others**, and for each class three permissions: **read (r), write (w), execute (x)**.sscasc+1

- For **files**:
  - read = view contents,
  - write = modify contents,
  - execute = run as a program or script.
- For **directories**:
  - read = list names,
  - write = create/delete entries,
  - execute = enter and search the directory.sscasc+1

Permissions are shown as a 10-character string like `-rwxr-x--x` or as octal numbers like `751`, where each digit is a sum of r=4, w=2, x=1.sscasc

5.3 Permission commands

- `chmod` – change permission bits.
  - Symbolic: `chmod u+x file` (add execute for user).
  - Numeric: `chmod 755 file` (owner rwx, group r-x, others r-x).computerhindinotes+1
- `chown` – change file owner, usually by root user.
- `chgrp` – change group ownership.sscasc+1

Hindi videos (users & permissions):

- Linux Tutorial For Beginners in Hindi – part on "Users in Linux" and "Permissions in Linux" explains ownership and `chmod`.
  https://www.youtube.com/watch?v=_tCY-c-sPZcyoutube
- Linux for Beginners in One Video – 100 Commands – has a dedicated segment for `chmod`, `chown`, `chgrp` with practice examples.
  https://www.youtube.com/watch?v=Byx4sgLR88Eyoutube

6. Putting It Together for Exams

- Draw a small **file system tree** starting at `/` with `/bin`, `/etc`, `/home`, `/tmp`, `/dev` and at least one user directory like `/home/student`.geeksforgeeks+1
- Memorise definitions: **file system**, **inode**, **absolute path**, **relative path**, **file types**, and the meaning of each permission bit for files and directories.geeksforgeeks+1
- Practise each listed command on a Linux machine or online terminal so that you remember real outputs; this makes it easy to write short examples in answers.homepages.uc+1

Unit-3

1. Redirection

Every command normally reads from **standard input** (keyboard) and writes to **standard output** (terminal). Redirection operators change these defaults so that commands read/write from files instead of the screen or keyboard.[swcarpentry.github]

- `command > file` – send output to file, overwrite existing content.
- `command >> file` – append output at end of file.
- `command < file` – take input from file, not keyboard.
- `command 2> file` – send error messages (standard error) to file.[ebookbou+1]

Redirection is very useful for saving results, creating logs and feeding data to other commands.

Hindi video (redirection basics):

- In CodeWithHarry's Linux tutorial, there is a part that shows > and < with examples. https://www.youtube.com/watch?v=_tCY-c-sPZc[youtube]

2. Pipes

A **pipe** connects the output of one command to the input of another using the symbol |.[geeksforgeeks+1] This allows you to build a chain (pipeline) of simple commands that together do complex processing.

Example:

- `ls -l | sort -k 5` – list files in long format, then sort them by size (column 5).[geeksforgeeks]
- `cat file.txt | grep "error" | sort | uniq` – show only unique lines containing "error", sorted alphabetically.[swcarpentry.github+1]

Key idea: each program does one small job; pipes connect them so the data flows smoothly from left to right.

Hindi video (pipes + basic filters):

- Many Hindi Linux tutorials show practical use of | with `grep`, `sort` etc.; you can follow along in the terminal for practice. https://www.youtube.com/watch?v=Byx4sgLR88E[youtube]

3. Concept of Filters

A **filter** is a small command-line program that reads text from standard input, transforms it and writes the result to standard output. Because they read from stdin and write to stdout, they fit naturally into pipelines.geeksforgeeks+1

Common filters in your syllabus are: `pr`, `head`, `tail`, `cut`, `paste`, `sort`, `uniq`, `tr`, `grep`, `egrep`. All of them can be combined with pipes for powerful text processing tasks on log files, data files, or program outputs.slideshare+1

Hindi video (filter overview – head, tail, cut, sort, uniq):

- "Unix: Filters commands (head/tail/cut)" – explains several filter commands with demos.
  https://www.youtube.com/watch?v=Vf9RcdUF0-Iyoutube

4. Important Filter Commands

Below is a short explanation of each filter you need to know.

4.1 `pr` – prepare file for printing

`pr` formats text files into pages with headers, page numbers and columns, useful when printing long outputs. It can set page length, number of columns, and margins so that printed reports are neat.vskub+1

4.2 `head` and `tail` – beginning and end of file

- `head file` shows first 10 lines by default; `head -n 5 file` shows first 5 lines.normalesup+1
- `tail file` shows last 10 lines; `tail -n 20 file` shows last 20 lines, useful for checking recent log entries.normalesup+1

These commands are often used with pipes to quickly inspect part of long outputs, e.g. `dmesg | tail`.

4.3 `cut` and `paste` – work with columns

- `cut` selects specific columns or fields from each input line, using a delimiter like comma or tab. For example, `cut -d":" -f1 /etc/passwd` prints only usernames.slideshare+1
- `paste` joins lines from two or more files side by side, creating combined records.vskub+1

Both are very useful for working with CSV or table-like data.

4.4 `sort` and `uniq` – ordering and duplicates

- `sort` arranges lines alphabetically or numerically; options like `-n` (numeric), `-r` (reverse), `-k` (key column) control the order.geeksforgeeks+1
- `uniq` removes or counts adjacent duplicate lines; normally used after `sort` because it needs identical lines to be together.normalesup+1

Example: `sort names.txt | uniq -c` prints each name once with a count of how many times it appeared.normalesup+1

4.5 `tr` – translate or delete characters

`tr` reads characters and replaces or removes them; for example, `tr 'a-z' 'A-Z'` converts lowercase letters to uppercase, and `tr -d '0-9'` deletes digits from the input stream. It is handy for simple text cleanup tasks.geeksforgeeks+1

4.6 `grep` and `egrep` – search with patterns

- `grep pattern file` prints lines that match a **regular expression** pattern, such as a word or string.ebookbou+1
- `egrep` (or `grep -E`) supports extended regular expressions, allowing more complex patterns using `+`, `?`, `|` and parentheses.geeksforgeeks+1

These commands are among the most powerful tools in UNIX because they can quickly find useful information inside large text files.

Hindi video (grep, sort, uniq, cut, etc.):

- Use this one-video Linux commands tutorial in Hindi which covers `grep`, `sort`, `uniq`, `cut`, `head`, `tail` and more with practice.
  https://www.youtube.com/watch?v=Byx4sgLR88Eyoutube

5. Compression and Archiving: `tar` and `gzip`

5.1 Why archiving and compression?

On multi-user systems, many files must be backed up or transferred over network, so it is convenient to **collect them into one archive** and **compress** it to save space. UNIX traditionally uses `tar` as the archiver and `gzip` as the compressor.linuxjournal+1

5.2 `tar` – tape archive

`tar` groups many files and directories into one archive file, often called a "tarball". Important options:cherryservers+1

- `-c` – create new archive.
- `-x` – extract files from archive.
- `-v` – verbose (show file names).
- `-f` – specify archive file name.
- `-z` – filter through gzip for compression (creates `.tar.gz`).geeksforgeeks+1

Examples:

- Create archive: `tar -cvf backup.tar /home/student` – archive home directory without compression.freecodecamp
- Create compressed archive: `tar -czvf backup.tar.gz /home/student` – archive and compress using gzip.linuxjournal+1
- Extract: `tar -xzvf backup.tar.gz` – unpack tar.gz into current directory.freecodecamp+1

5.3 `gzip`

`gzip file` compresses a single file and replaces it with `file.gz`, while `gunzip file.gz` restores original content. When combined with `tar`, it gives compact `.tar.gz` files commonly used for backups and software distribution.xtom+1

Hindi video (tar + gzip):

- "Linux Backup and Restore Using TAR AND GZIP Command | Hindi" – shows how to create and extract archives practically.
  https://www.youtube.com/watch?v=rEkIrSP1aCoyoutube

6. How to Study Unit–3

- Understand the **idea**: redirection and pipes are ways of connecting commands; filters are small tools that transform text.linfo+1
- Practise each filter with small text files so you can write simple examples in exams, like "Example of `cut` command" or "Example using `grep` and `sort` together".geeksforgeeks+1
- Remember that `tar` is mainly an **archiver** (combines files), and `gzip` is the **compressor**; together they produce `.tar.gz` archives for backup and transfer.linuxjournal+1

UNIT-4

1. Vi Editor: Basic Idea and Modes

`vi` is the standard text editor available on almost every UNIX or Linux system, mainly used to edit configuration files and code inside a terminal. It is a **modal editor**, which means the same keys do different things depending on the current mode.[cse.iitm+3]

Main modes:

- **Command mode:** Default mode after opening `vi`; keys move the cursor, delete or copy text, search, etc.
- **Insert mode:** Used to type actual text into the file.
- **Last-line (ex) mode:** Entered with `:` to give commands like `:w`, `:q`.[sscasc+1]

Hindi video (intro to vi editor):

- "Linux vi Editor Tutorial in Hindi" – shows modes and basic editing. https://www.youtube.com/watch?v=4kW9x8Xy0iM[youtube]

2. Basic Vi Commands

2.1 Starting and exiting vi

- Open file: `vi filename` – creates file if it does not exist.[sscasc]
- Quit without saving: from command mode type `:q!` and press Enter.
- Save and quit: `:wq` or `:x`.[sscasc+1]

2.2 Switching modes

- From command mode to insert mode: `i` (insert before cursor), `a` (after cursor), `o` (open new line below), `O` (new line above).[cse.iitm+1]
- From insert mode back to command mode: press `Esc` key.[sscasc]

2.3 Navigation and editing

In command mode:

- Move cursor: `h` left, `j` down, `k` up, `l` right; or use arrow keys.[cse.iitm]
- Word and line movement: `w` next word, `b` previous word, `0` start of line, `$` end of line, `G` end of file, `1G` first line.[sscasc]
- Delete: `x` delete character, `dw` delete word, `dd` delete entire line.
- Copy (yank) and paste: `yy` copy line, `p` paste after cursor.[sscasc+1]
- Undo: `u` undo last change.

Hindi video (practical vi usage):

- Many Linux command playlists in Hindi include a specific “vi Editor” part showing these commands in action.
  https://www.youtube.com/watch?v=Byx4sgLR88E youtube

3. Shell Types and Environment Variables

3.1 Shell types

A **shell** is a command interpreter program that provides the user interface to UNIX. Popular shells are: hpc.iastate+1

- Bourne shell (`sh`)
- Bourne Again Shell (`bash`)
- C shell (`csh`)
- Korn shell (`ksh`)

Each shell supports slightly different scripting syntax and extra features (history, completion, etc.), but all read commands, perform expansions and call the kernel. interviewbit+1

3.2 Environment variables

Environment variables are name–value pairs that store information about the user’s session and configuration. Common examples: geeksforgeeks+1

- `HOME` – path to user’s home directory.
- `USER` or `LOGNAME` – current username.
- `PATH` – list of directories to search for commands.
- `PS1` – shell prompt format. geeksforgeeks+1

You can:

- View all variables with `env` or `printenv`.
- Show one variable: `echo $HOME`.
- Set variable: `VAR=value`.
- Export to child processes: `export VAR` in Bourne/Bash. startertutorials+1

Hindi video (shell and environment variables):

- “Everything about Linux from Scratch – Part 1 (Hindi/Urdu)” – covers shell basics and environment variables.
  https://www.youtube.com/watch?v=CTZfWmMAdoI youtube

4. Shell Features

Shells offer many useful features **before** a command is executed.geeksforgeeks+1

Key features:

- **Command history:** Recall previous commands using arrow keys or `history` command.
- **Filename expansion (globbing):** Patterns like *, ?, `[a-z]` expand to matching filenames (e.g. `*.txt`).geeksforgeeks
- **I/O redirection and pipes:** Using >, <, >>, | as covered in Unit–3.geeksforgeeks+1
- **Aliases:** Short names for longer commands, set with `alias l='ls -l'`.geeksforgeeks
- **Shell scripts:** Files containing a list of commands that the shell can execute as programs, covered deeply in Unit–5.freecodecamp

These features make the shell a powerful programming environment as well as an interface.

Hindi video (shell features + history, wildcards, alias):

- Linux For DevOps In One Shot (Hindi) explains shell features such as history, wildcards and aliases.
  https://www.youtube.com/watch?v=e01GGTKmtpcyoutube

5. Quoting Mechanisms in Shell

Quoting controls how the shell treats special characters like spaces, $, *, ?, ! etc.geeksforgeeks+1

- **No quotes:** Shell treats spaces as separators and expands wildcards and variables. Example: `echo $HOME` prints your home directory.geeksforgeeks
- **Single quotes `'...'` :** Everything inside is taken literally; no variable or wildcard expansion happens. Example: `echo '$HOME'` prints `$HOME` text, not the directory.makautexam+1
- **Double quotes `"..."` :** Protects most characters but still allows variable and command substitution. Example: `echo "Home is $HOME"` will substitute `$HOME` value.geeksforgeeks+1
- **Backslash `\` :** Escapes only the next character so it is treated literally (for example `\*` to avoid wildcard expansion).makautexam+1

Quoting is very important for safe shell scripts, especially when filenames have spaces or special characters.

Hindi video (quoting & variables):

- Shell Scripting Tutorial #1 (Hindi) covers variables and explains the effect of quotes while printing and using them.
  https://www.youtube.com/watch?v=M79HMJqRaVUyoutube

6. How to Study Unit–4

- Practise `vi` on a real Linux system so that switching modes and using `dd`, `yy`, `p`, `:wq` become automatic.cse.iitm+1
- Memorise key environment variables (`PATH`, `HOME`, `USER`, `PS1`) and practise changing them temporarily in a terminal session.geeksforgeeks
- Try small experiments with quoting, for example `echo *`, `echo "*"`, `echo "$HOME"` and `echo '$HOME'`, to see exactly how shell behaviour changes.swcarpentry.github+1

Unit-5

1. What is Shell Programming?

Shell programming (shell scripting) means writing a text file that contains a sequence of shell commands, which the shell runs one by one like a small program. It is mainly used to automate routine tasks such as backups, monitoring, file operations or system administration, using all the normal UNIX commands plus control structures.pressbooks.senecapolytechnic+2

A typical shell script:

- starts with a **shebang line** (for example `#!/bin/bash`) telling which shell should run it,
- contains commands, variables, and logic,
- is given execute permission so it can run like a program.training-course-material+1

Hindi video (basic idea + first script):

- Shell Scripting Tutorial for Beginners | In HINDI | Crash Course https://www.youtube.com/watch?v=LMWx1WpLXGIyoutube

2. Script Structure and Variables

2.1 Basic script structure

A minimal Bash script often looks like: shebang line, optional comments, variable definitions, commands and logic. Scripts are saved with `.sh` (not compulsory) and executed by `bash script.sh` or by `chmod +x script.sh` then `./script.sh`.shellscript+1

Important points:

- Use comments starting with # to explain steps.

- Organise script in logical sections (input, processing, output) for readability.[pressbooks.senecapolytechnic+1]

2.2 Shell variables

Variables in shell are created simply by assignment, with no data type declaration.[geeksforgeeks+1]

- Assign: `name="Rahul"` (no spaces around =).
- Use: `echo "Hello $name"` – `$name` expands to its value.
- Shell also has many built-in variables: `$HOME`, `$PWD`, `$PATH`, `$USER` etc.[pubs.opengroup+1]

There are also **positional parameters** used for command-line arguments: `$0` (script name), `$1`, `$2`, … for arguments, `$#` for count of arguments, `$@` for all arguments.[redhat+1]

Hindi video (variables & first scripts):

- Bash scripting tutorial for beginners (Hindi/English mix) – covers shebang, running scripts, variables and user input.
  https://www.youtube.com/watch?v=9T2nEXlLy9o[youtube]

3. Operators and Expressions

Shell scripts use operators mostly inside `[ ]` or `[[ ]]` test expressions.[geeksforgeeks+1]

Main categories:

- **Arithmetic:** `+ - * / %` plus comparison operators like `-eq` (equal), `-ne` (not equal), `-gt` (greater than), `-lt` (less than), `-ge`, `-le`.[pluralsight+1]
- **String tests:** = or == (equal), `!=` (not equal), `-z` (empty string), `-n` (non-empty string).[geeksforgeeks]
- **File tests:** `-e` exists, `-f` regular file, `-d` directory, `-r` readable, `-w` writable, `-x` executable, `-s` non-empty.[geeksforgeeks]
- **Logical:** ! (NOT), `-a` or && (AND), `-o` or || (OR) used to combine conditions.[training-course-material]

Example of arithmetic test:
`if [ $a -gt 10 ]; then echo "a is big"; fi` – executes echo only if value of `a` is greater than 10.[digitalocean]

Hindi video (operators with conditions):

- Master SHELL SCRIPTING in ONE VIDEO (Hindi) – has dedicated sections for variables, operators and conditions.
  https://www.youtube.com/watch?v=119TtGM9GfBuok[youtube]

4. Conditional Statements

Conditions control which block of commands runs, based on test results.pressbooks.senecapolytechnic+1

4.1 if, if…else, if…elif…else

General form:

```
if condition; then commands; elif other_condition; then other commands; else default commands; fi
```
digitalocean+1

Uses:

- Check if files or directories exist before using them.
- Validate user input.
- Choose different actions for different argument values.digitalocean+1

4.2 case statement

`case` is used when there are many possible values of one variable (like a menu choice).phoenixnap+1

Basic pattern:

```
case $choice in 1) commands ;; 2) commands ;; *) default ;; esac
```

This makes scripts easier to read than long chains of `if…elif`.

Hindi video (if/else, case, file tests):

- Bash scripting tutorial (Hindi, 4 hours) – parts around "Conditional Statement" and "Case statement".
  https://www.youtube.com/watch?v=9T2nEXlLy9oyoutube

5. Loops

Loops repeat commands while a condition holds or for each item in a list.geeksforgeeks+1

- **for loop:** iterates over a list of words or numbers.
  Example: `for f in *.txt; do echo $f; done` – prints all `.txt` filenames.geeksforgeeks
- **while loop:** runs as long as condition is true.
  Example: `while [ $i -le 10 ]; do echo $i; i=$((i+1)); done`.geeksforgeeks
- **until loop:** opposite of while; runs until condition becomes true.geeksforgeeks

Loops are used for tasks like scanning many files, reading lines from a file or performing repeated checks.

Hindi video (for, while, until loops):

- Master SHELL SCRIPTING in ONE VIDEO (Hindi) – covers for, while, until, and loop examples with files.
  https://www.youtube.com/watch?v=9T2nEXlLy9oyoutube

6. Functions

Functions let you group related commands into reusable blocks, improving structure and avoiding repetition.training-course-material+1

Basic definition in Bash:

```
Text:
myfunc() {
      commands
}
```

- Call using `myfunc`.
- Parameters inside function appear as `$1`, `$2`, etc., similar to script arguments.redhat+1
- You can return an exit status using `return` or by echoing values to be captured by caller.shellscript

Functions are very useful for large scripts, where you might have one function to check input, another to log messages and another to perform a task.

Hindi video (functions & modular scripts):

- Master SHELL SCRIPTING in ONE VIDEO (Hindi) – section "Functions in shell script" and "Arguments passing in shell script".
  https://www.youtube.com/watch?v=9T2nEXlLy9oyoutube

7. Command-Line Arguments and Pattern Matching

7.1 Command-line arguments

When you run `./script.sh file1 file2`, inside the script:

- `$1` is `file1`, `$2` is `file2`, `$#` is 2, `$@` is the list `file1 file2`.training-course-material+1

Scripts typically:

- check argument count (`if [ $# -lt 1 ] ...`),
- use arguments as filenames, options, or parameter values.

7.2 Pattern matching (globbing and regex in tools)

At shell level, **globbing** expands patterns like `*.txt`, `data??.csv`, `[a-c]*.sh` to matching filenames. This is not full regular expressions but is often enough for selecting files.pubs.opengroup

Inside tools like `grep` or `egrep`, **regular expressions** are used to match patterns in text data, e.g. `grep "^error"` to show lines starting with "error". Shell scripts combine these with loops and conditions to perform powerful searching and reporting.geeksforgeeks+1

Hindi video (arguments, patterns, practical scripts):

- Shell Scripting in One Shot | DevOps (Hindi) – covers `$1`, `$#`, case patterns and real scripts.
  https://www.youtube.com/watch?v=9Xl1ZTk3BQwyoutube

8. How to Practise Unit–5

- Write very small scripts: one that prints your name, one that adds two numbers, one that checks if a file exists, one that loops over all `.txt` files.freecodecamp+1
- Gradually add `if`, `case`, `for`, `while`, and functions to build slightly bigger scripts, like a simple menu-driven calculator or backup script.pressbooks.senecapolytechnic+1
- Run the Hindi videos in parallel with practice so you both **see** and **type** each concept; this will also help you remember syntax for theory questions.youtube+1

Unit-6

1. Basic idea of a process

A **process** is a program in execution; when you run any command or application, the OS creates a process with its own memory, registers, open files and a unique **PID (Process ID)**. On a multi-user UNIX system, hundreds of processes (system + user) can exist together, and the kernel manages them so that they share CPU, memory and other resources safely.guru99

Hindi video (concept of process & process management):

- Process Management in Operating System [Hindi] – theory of what a process is, process states and PCB.
  https://www.youtube.com/watch?v=RL17f5KdO8Iyoutube

2. Process states and lifecycle

During its life a process moves through standard **states**:

- **New:** Process is being created and PCB (Process Control Block) is allocated.
- **Ready (Runnable):** Process is in memory, ready to run, waiting in ready queue for CPU time.
- **Running:** Process is currently executing on CPU.
- **Waiting / Blocked:** Process is waiting for some event (I/O completion, signal, child exit) so it cannot run right now.
- **Terminated:** Process has finished; kernel keeps minimal info (zombie) until parent collects exit status.

The **scheduler** switches CPU from one ready process to another, using context switch, so many processes appear to run in parallel.youtube+1

Hindi video (process states, PCB, scheduling intro):

- Lecture on Process Management in OS (Code Hacker / GATE-style Hindi explanation). https://www.youtube.com/watch?v=Koqx_wiYz84youtube

3. Viewing processes: `ps` and `top`

3.1 `ps` – process status

`ps` shows a snapshot of processes:

- `ps` – processes for current terminal.
- `ps -ef` or `ps aux` – almost all processes with PID, user, CPU%, MEM%, TTY and command.
- `ps -u username` – processes for a particular user.

You can combine it with `grep`, for example `ps -ef | grep sshd`, to find specific programs.guru99+1

3.2 `top` – live process view

`top` displays running processes in real time, updating CPU and memory usage every few seconds.guru99
From inside `top` you can:

- sort by CPU or memory,
- press `k` to send kill signal to a process,
- press `q` to exit.teaching.healthtech.dtu

Hindi video (ps, top basics):

- Process Management Commands in Linux | ps, top, kill, jobs, nice, renice (Hindi). https://www.youtube.com/watch?v=vh7jwXo_tLcyoutube

- Linux top Command | Hindi – focused video on using `top`.
  https://www.youtube.com/watch?v=XDeWSfHrXQw youtube

4. Killing and prioritising processes: `kill`, `nice`, `renice`

4.1 `kill` and signals

Processes are controlled by **signals**; `kill` sends a signal to a PID:

- `kill PID` – sends default `SIGTERM` (15), a polite request to stop.
- `kill -9 PID` – sends `SIGKILL`, which the kernel enforces immediately if process does not respond to TERM. digitalocean+1

Before using `kill`, you usually find PID via `ps` or `pidof`.

4.2 `nice` and `renice` – priority (niceness)

Each process has a **nice value** between −20 and +19; lower value = higher priority. youtube guru99

- Start a command with lower priority: `nice -n 10 long_job.sh`.
- Change nice of running process: `renice +5 PID` (increase niceness, give it less CPU). guru99

Only privileged users can reduce nice (give more CPU) to prevent misuse.

Hindi videos (ps + kill + nice/renice):

- Linux Process Management | ps, fg, bg, jobs (Urdu/Hindi) – includes basic `ps` and `kill`.
  https://www.youtube.com/watch?v=HfAs1ReenUM youtube
- Linux Kill Command Tutorial in Hindi – detailed examples of killing processes using PID.
  https://www.youtube.com/watch?v=gXByMLmzOpo youtube
- How to Manage Processes on Linux with nohup, nice, bg, fg, jobs Commands (Hindi).
  https://www.youtube.com/watch?v=kmk3_kEiJvk youtube

5. Foreground/background and job control: &, `jobs`, bg, `fg`

The shell lets you manage **jobs** – commands started from that shell:

- `command &` – run command in background; shell prints a job number like `[1]` and returns prompt.
- `jobs` – list current jobs with their status (Running, Stopped).

- `Ctrl+Z` – stop (suspend) current foreground job and put it in "Stopped" state for that terminal.
- `bg %1` – continue job 1 in background so you can keep using terminal.
- `fg %1` – bring job 1 to foreground, attaching it back to keyboard.guru99+1

Foreground jobs read from keyboard and write to terminal; background jobs continue without blocking your shell.

Hindi video (bg, fg, jobs, nohup, nice):

- How to Manage Processes on Linux with nohup, nice, bg, fg, jobs Commands (Hindi). https://www.youtube.com/watch?v=kmk3_kEiJvkyoutube

6. Concept of scheduling and key system calls

6.1 CPU scheduling overview

The **scheduler** chooses which ready process runs next, based on policies like round-robin, priority or multi-level queues.
Goals include:

- Fairness between users and processes.
- Good response time for interactive tasks.
- Efficient CPU usage by avoiding idle time.

The OS performs **context switches**, saving state of current process (registers, program counter) into its PCB and loading another process's state so it can resume from where it stopped.youtube+1

6.2 Main process system calls (conceptual level)

In UNIX-like systems, process lifecycle uses a few classic system calls:

- `fork()` – parent process creates a copy (child process) with a new PID.
- `exec()` family – child replaces its code with a new program (for example, shell forks, then child execs `/bin/ls`).
- `wait()` – parent waits for child to finish and collects its exit status.
- `_exit()` / `exit()` – process terminates and returns status to parent.

Together these allow shells and programs to start, manage and clean up other processes.

Hindi video (OS-level process management & scheduling):

- Process Management in Operating System [Hindi] – good for theory (process states, PCB, schedulers).
  https://www.youtube.com/watch?v=RL17f5KdO8Iyoutube

7. How to revise Unit–6

- On a Linux machine, practise: `ps -ef`, `top`, starting `sleep 60 &`, using `jobs`, `bg`, `fg`, `kill`, `nice` and `renice` so you remember syntax.linuxteck+1
- For theory questions, prepare 3–4 lines each on: definition of process, process states diagram, role of scheduler and PCB, and job-control commands (`bg`, `fg`, `jobs`, `&`).guru99 youtube

Unit-7

1. What is System Administration?

System administration in UNIX/Linux means taking care of the whole system so that it stays secure, fast and reliable for all users. A system administrator (sysadmin) manages user accounts, disk space, software, backups and networking, and also handles problems like hardware failure or crashes.geeksforgeeks+2

Key responsibilities:

- Create and manage users and groups.
- Control disk usage with partitions and quotas.
- Install, update and remove software.
- Plan and test backup/restore.
- Configure boot, shutdown and basic networking.linode+1

Hindi video (overall Linux admin basics):

- Linux Admin Tutorial for Beginners (Hindi sections on users, packages, networking). https://www.youtube.com/watch?v=F25W0Z-MNwsyoutube

2. User Management

Every person using the system has a **user account** with a UID, home directory and default shell stored in `/etc/passwd`, while encrypted passwords are stored in `/etc/shadow` and group information in `/etc/group`.purevoltage+1

Common commands:

- `useradd username` – create user; options can set home dir, shell, expiry, etc.
- `passwd username` – set or change password.
- `usermod` – modify existing user (change home, shell, groups).
- `userdel username` – delete user account, optionally remove home directory.
- `groupadd`, `groupdel`, `groupmod` – manage groups.geeksforgeeks+1

The admin also sets appropriate file permissions and group memberships so users can access only what they need.geeksforgeeks

Hindi videos (user & group management):

- User & Group Management Commands Tutorial | Master Linux in Hindi | Part 3 https://www.youtube.com/watch?v=Lky8ubmxrcoyoutube
- Linux User Account Management in HINDI with LAB | USERADD, USERMOD, USERDEL, PASSWD https://www.youtube.com/watch?v=F25W0Z-MNwsyoutube

3. Disk Management and Quotas

The sysadmin creates and mounts file systems, monitors free space and prevents any user from filling the disk completely. **Disk quotas** limit how much disk space (blocks) or how many files (inodes) a user or group can use on a file system.tuxcademy+3

Basic quota steps (concept level):

- Enable quotas for a file system in `/etc/fstab` (for example, user and group quotas on `/home`).
- Run tools to create quota database files and scan current usage.
- Set soft and hard limits for users/groups; soft limit can be exceeded for a grace period, hard limit cannot be crossed.
- Periodically check reports and adjust limits if needed.redhat+1

Hindi resource (quota concept inside admin notes/video):

- Many Linux admin Hindi courses mention quota when discussing disk management; pair these notes with any such segment for exams.vmou+1

4. Software Installation, Backup and Restore

4.1 Managing software

Modern UNIX/Linux systems use **package managers** to install and update software consistently, resolving dependencies automatically.linode+1

Examples:

- `apt`, `apt-get` on Debian/Ubuntu.
- `yum` / `dnf` on Red Hat based systems.
- `zypper` on SUSE.

Typical tasks: `apt install package`, `apt remove package`, `apt update` and `apt upgrade` to keep system patched and secure.linode

4.2 Backup and restore with `tar` and compression

Backups protect against accidental deletion, disk crash or malware. A simple method is to **archive** important directories with `tar` and optionally **compress** them using `gzip` or similar.translate.google+1

Typical commands:

- Create archive: `tar -cvf backup.tar /home/user` (archive only).
- Create compressed backup: `tar -czvf backup.tar.gz /home/user` (archive + gzip).
- Restore: `tar -xzvf backup.tar.gz` to extract files back to a directory.youtube translate.google

Good practice is to keep regular backups (daily/weekly), store them on a different disk or remote server, and test recovery periodically.translate.google+1

Hindi video (backup with tar & gzip):

- tar for backup and restore in Linux | Linux Tutorial – Hindi explanation of `tar` + compression.
  https://www.youtube.com/watch?v=fpx5lrvSdWMyoutube

5. Startup, Shutdown and Services

When a UNIX/Linux system boots, the kernel starts **init** (or systemd on modern systems), which then runs startup scripts and service units to bring the system into multi-user mode. Sysadmins configure which services (web server, database, SSH, etc.) should start automatically.tldp+1

With systemd, common commands are:

- `systemctl start service` – start service now.
- `systemctl stop service` – stop service.
- `systemctl enable service` – start at boot.
- `systemctl disable service` – do not start at boot.linode

For shutdown and reboot, admins use commands like `shutdown -h now`, `shutdown -r now`, `reboot` or `power