



# FILE OPERATIONS AND PROCESSES

SOFTWARE ENGINEERING

# CONTENTS

- Working with files (touch, cp, mv, rm)
- File archiving and compression (tar, gzip)
- Text editors in Linux (nano, vim)
- Basic text manipulation (grep, cat, less)
- Processes and process management (ps, top, kill)
- Running and managing background processes
- Redirecting input and output (>, >>, <, |)

# WORKING WITH FILES

- `cp`: to make a copy of a file or copy file to another location
- `touch`: create an empty file
- `mv`: To move a file from one location to the other, or to rename it
- `rm`: delete a file permanently
- `find`: to find a file

# FILE ARCHIVING AND COMPRESSION

- To compress a file in Linux: `tar -zcvf file.tar.gz [PATH]`
- ex: `tar -zcvf file.tar.gz file.cpp`
- -c: Create a new archive.
- -z: Compress the archive using gzip.
- -v: Verbose mode (optional, shows the progress).
- -f: Specifies the archive file name.

# EXCLUDE DIRECTORIES AND FILES

- Compress files but excluding files: `tar -zcvf archive.tar.gz --exclude='[DIRECTORY]'`
- Compress files but excluding directories: `tar -czvf /nfs/backup.tar.gz --exclude=" DIRECTORY " /home/vivek/`

# EXTRACT A FILE

- `tar -xzvf file.tar.gz`
- `tar -xjvf file.tar.bz2`
- extract the contents of the archive into a specific directory: `tar -xzvf my.tar.gz -C [DIRECTORY]`
- `tar -xjvf archive.tar.bz2 -C [DIRECTORY]`

# EXTRACT A FILE

- -x : Extract files from an archive
- -t : List the contents of an archive
- -v : Verbose output
- -f file.tar.gz : Use archive file
- -C DIR : Change to DIR before performing any operations
- --exclude : Exclude files matching PATTERN/DIR/FILENAME



# TEXT EDITORS IN LINUX

- Text Editor: nano
- If you already have a txt file: nano \_text\_file\_name.txt
- CTRL + O → to save
- CTRL + X → to exit
  
- Undo: ALT + U
- Redo: ALT + E



# NANO

Flag	Description	Example
-B	Makes a backup of the current file before saving changes.	<code>nano -B myfile.txt</code>
-I	Enables automatic indentation.	<code>nano -I myfile.txt</code>
-N	No conversion from DOS/Mac format.	<code>nano -N myfile.txt</code>
-T	Sets the size of a tab to the given number of spaces.	<code>nano -T 4 myfile.txt</code>
-U	Enables undo functionality.	<code>nano -U myfile.txt</code>
-Y	Syntax highlighting.	<code>nano -Y sh myfile.sh</code>
-c	Constantly show the cursor position.	<code>nano -c myfile.txt</code>
-i	Automatically indents new lines.	<code>nano -i myfile.txt</code>
-k	Toggle cut so it cuts from cursor position.	<code>nano -k myfile.txt</code>
-m	Enable mouse support.	<code>nano -m myfile.txt</code>

# OTHER TEXT EDITORS

- While the nano command is a powerful and user-friendly text editor, Linux provides other text editors that might better suit your needs, particularly as you become more experienced. Two of the most popular alternatives are Vi and Emacs.

# VI

- `vi myfile.txt`
- a. press 'i' to enter insert mode
- ex: This is some text.
- c. press 'Esc' to exit insert mode
- d. type `:wq` to save and quit
- e. Output: 'myfile.txt' 1L, 18C written

## Nano vs. Vi vs. Emacs

Feature	Nano	Vi	Emacs
Learning Curve	Easy	Moderate	Hard
User Interface	Simple and intuitive	Modal, requires learning commands	Extensible, requires learning commands
Extensibility	Limited	Yes, with Vimscript	Yes, with Emacs Lisp
Accessibility	Universally available	Universally available	Not always installed by default

# BASIC TEXT MANIPULATION

- 1. grep
- 2. cat
- 3. less
- 4. head
- 5. tail

# GREP

- The grep filter searches a file for a particular pattern of characters and displays all lines that contain that pattern. The pattern that is searched in the file is referred to as the regular expression (grep stands for global search for regular expression and printout).
- `grep [options] pattern [files]`
- [options]: These are command-line flags that modify the behavior of grep.
- [pattern]: This is the regular expression you want to search for.

# GREP

Options	Description		
-c	This prints only a count of the lines that match a pattern	-n	Display the matched lines and their line numbers.
-h	Display the matched lines, but do not display the filenames.	-v	This prints out all the lines that do not matches the pattern
-i	Ignores, case for matching	-w	Match whole word
-l	Displays list of a filenames only.		



# CAT, LESS

- The cat command is the simplest way to view the contents of a file. It displays the contents of the file(s) specified on to the output terminal. Used for simple and short files: `cat FILENAME`
- The less command is similar to the more command but provides extensive features. One important one is that it allows backward as well as forward movement in the file: `less FILENAME`
- Use `g` and `G` to go up or down, `q` to exit

# HEAD, TAIL

- As their names imply, the head command will output the first part of the file, while the tail command will print the last part of the file. Both commands write the result to standard output: head [OPTIONS] FILES
- Both allow the display of a certain number of lines
- tail [OPTIONS] FILES

# PROCESSES AND PROCESS MANAGEMENT

- Processes and process management are fundamental concepts in Linux and other operating systems. Understanding and effectively managing processes is crucial for the stability, performance, and security of a Linux system.
- A process is an instance of a program currently running on a computer system. In Linux, processes are managed by the operating system's kernel, which allocates system resources and schedules processes to run on the CPU. Understanding and managing processes is a critical skill for Linux administrators and developers.

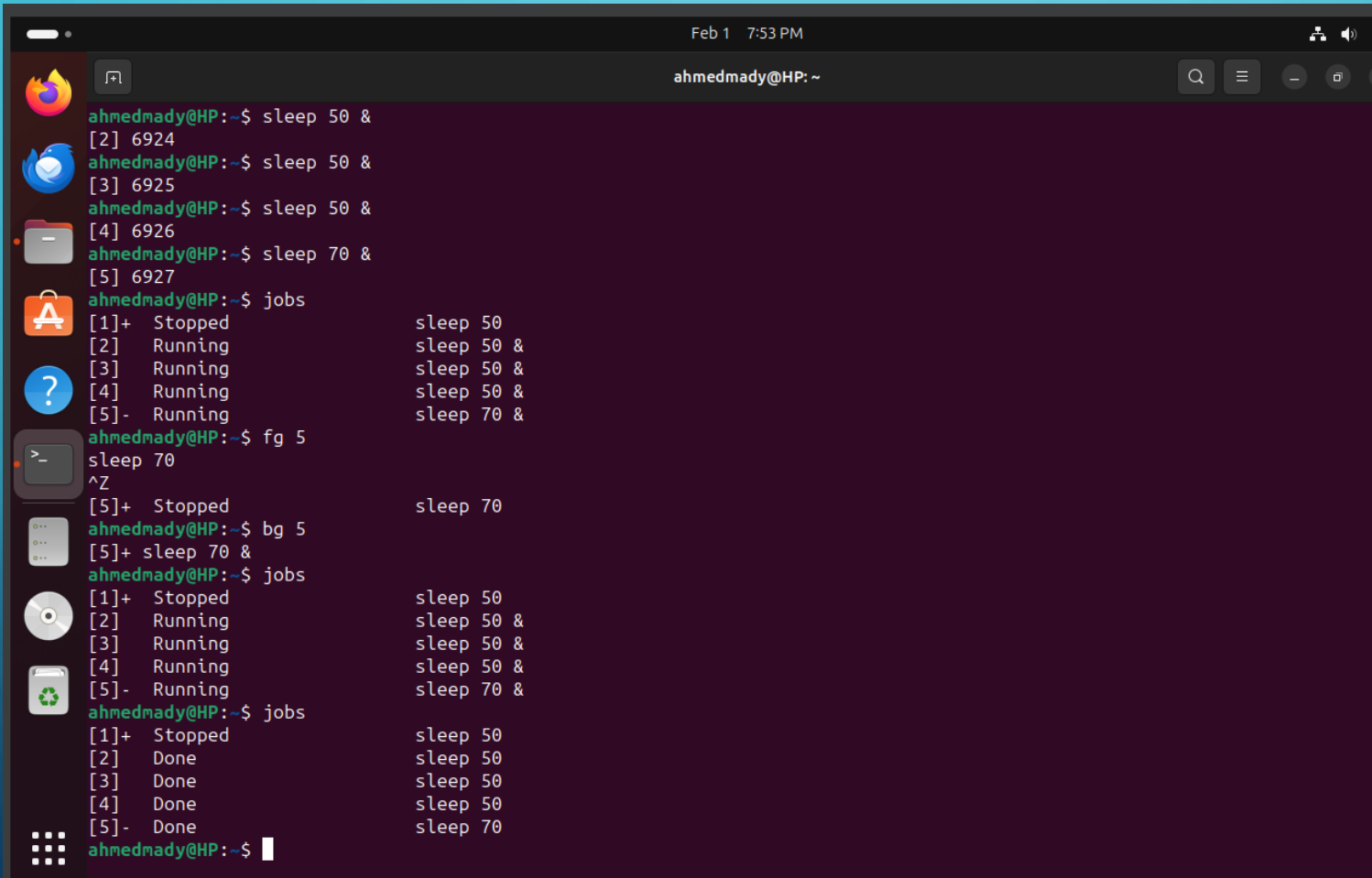
# PS

- Sometimes, we want to close unresponsive programs or check if a background process has started in our Linux system.
- The `ps` command in Linux is used for displaying information about processes. It provides a snapshot of the currently running processes on the system. The `ps` command has various options that allow you to customize the output according to your requirements: `ps [options]`
- PID – the process id
- TTY – terminal associated with the process
- TIME – elapsed CPU utilization time for the process
- CMD – the executable command

# TOP, KILL

- Top is used to list all running Linux processes on your system: top
- To kill a process: kill [PID]

# RUNNING AND MANAGING BACKGROUND PROCESSES

A terminal window titled 'ahmedmady@HP: ~' with a dark purple background and white text. The window shows a series of commands and their outputs for managing background processes. The user runs 'sleep 50 &' five times, which are listed as jobs [2] through [6] with PIDs 6924 through 6928. Then, the user runs 'jobs', showing jobs [1] through [5] with their status (Stopped or Running) and the command they were run with. The user then runs 'fg 5', which brings job [5] to the foreground. The user presses Ctrl-Z (^Z), which suspends the foreground process. The user then runs 'bg 5', which resumes job [5] in the background. Finally, the user runs 'jobs' again, showing jobs [1] through [5] with their status (Stopped or Done) and the command they were run with. The terminal window has a standard Ubuntu-style top bar with the date and time (Feb 1 7:53 PM) and system icons on the right. On the left side of the terminal window, there is a vertical dock with icons for the Dash, Home, Applications, and various system utilities.

```
ahmedmady@HP:~$ sleep 50 &
[2] 6924
ahmedmady@HP:~$ sleep 50 &
[3] 6925
ahmedmady@HP:~$ sleep 50 &
[4] 6926
ahmedmady@HP:~$ sleep 70 &
[5] 6927
ahmedmady@HP:~$ jobs
[1]+  Stopped                  sleep 50
[2]   Running                  sleep 50 &
[3]   Running                  sleep 50 &
[4]   Running                  sleep 50 &
[5]-  Running                  sleep 70 &
ahmedmady@HP:~$ fg 5
sleep 70
^Z
[5]+  Stopped                  sleep 70
ahmedmady@HP:~$ bg 5
[5]+ sleep 70 &
ahmedmady@HP:~$ jobs
[1]+  Stopped                  sleep 50
[2]   Running                  sleep 50 &
[3]   Running                  sleep 50 &
[4]   Running                  sleep 50 &
[5]-  Running                  sleep 70 &
ahmedmady@HP:~$ jobs
[1]+  Stopped                  sleep 50
[2]   Done                     sleep 50
[3]   Done                     sleep 50
[4]   Done                     sleep 50
[5]-  Done                     sleep 70
ahmedmady@HP:~$
```





# ECHO

- The echo command is a way to communicate with your Linux terminal. It allows you to send text, variables, and special characters to the standard output, which is usually the terminal screen. The echo command is like a messenger that delivers your words to the terminal.



# REDIRECTING INPUT AND OUTPUT

- The ‘>’ symbol is used for output (STDOUT) redirection overriding the file: > :  
ls -al > record.txt
- If the contents of the file are to be added and not overwritten: >> : ls -n text  
>> record.txt
- The ‘<’ symbol is used for input(STDIN) redirection : Mail -s "Subject" to-  
address < Filename

- 
- 
- 
- 
- In Linux, the vertical bar (|) is known as a "pipe" and is used to connect the output of one command to the input of another. This allows you to create a pipeline of commands, where the output of one command becomes the input for the next command. The | symbol is commonly used in the command line to enhance the functionality and flexibility of Linux commands.

# WORD COUNT

- `wc [file]`: returns is the file's number of lines, words, and characters, followed by its path.

# DIFF COMMAND

- Its primary purpose is to compare the contents of two files and display the differences between them. `diff [FILE1] [FILE2]`
- Can compare 2 versions of the same file

- Create 2 text files, 1 file using nano, and the other using cp command
- Edit the second file any text u add
- Use diff command to display the added text

# REFERENCES

- <https://www.allhandsontech.com/linux/linux-commands-files/>
- <https://www.cyberciti.biz/faq/how-to-tar-a-file-in-linux-using-command-line/>
- <https://ioflood.com/blog/nano-linux-command/#:~:text=To%20use%20the%20nano%20command,will%20create%20it%20for%20you.>
- <https://www.baeldung.com/linux/head-tail-commands>