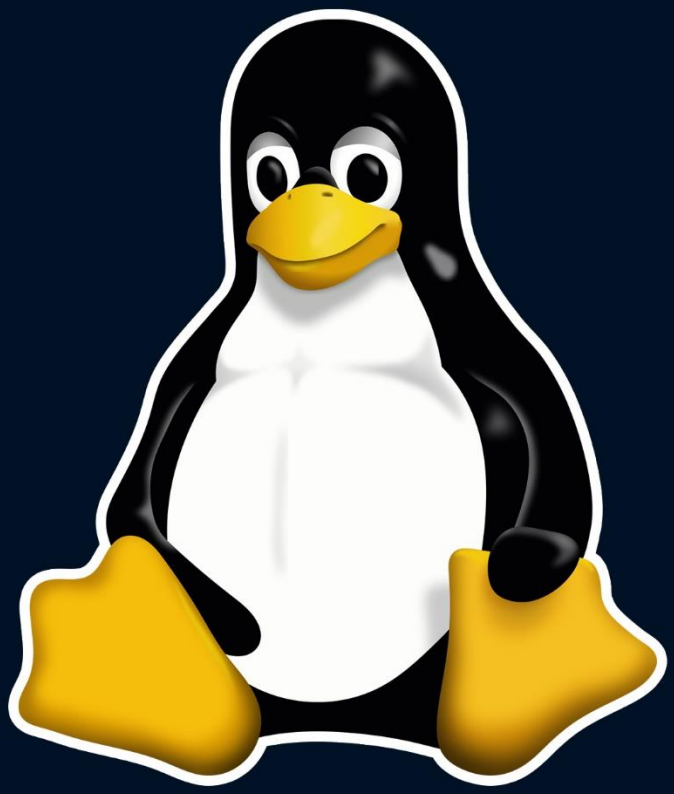
LINUX SIMPLIFIED

Part 01

Linux File System

Let's go →



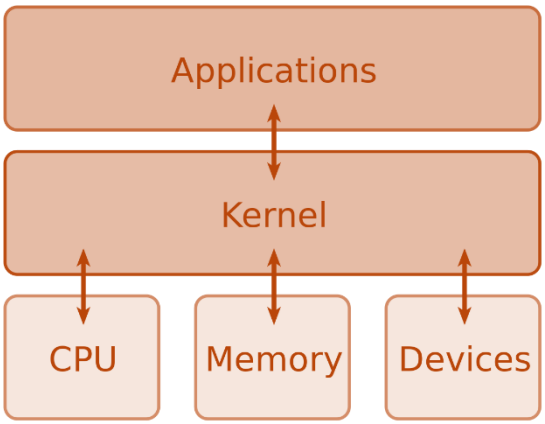
linkedin.com/in/shehan-arampola
github.com/shehan404

Linux

Linux is a family of open-source, Unix-like operating systems based on the Linux kernel, first released on September 17, 1991, by Linus Torvalds. Linux is typically packaged as a distribution, which includes the kernel, supporting system software, and libraries, most of which are provided by third parties to create a complete operating system.

Kernel

The kernel is a core component of an operating system and serves as the main interface between the computer's physical hardware and the processes running on it. The kernel enables multiple applications to share hardware resources by providing access to CPU, memory, disk I/O, and networking.



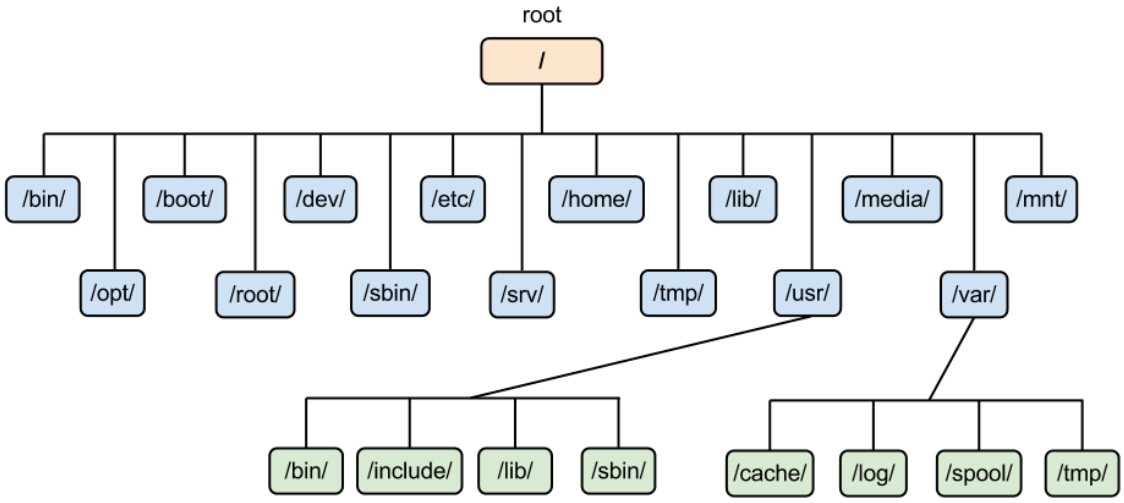
Shell

A shell in an operating system is a user interface that allows users to access the operating system's services. It acts as a bridge between the user and the kernel, enabling users to execute scripts, run commands, and monitor running processes. Shells can use either a command-line interface (CLI) or a graphical user interface (GUI). The most commonly available shells are,

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

File System in Linux

The default Linux file system is ext4. You can navigate to root by `cd /`



`/boot` - contains files that are used by the boot loader.

`/root` - root user home directory. It is not the same as /

`/dev` - System devices(eg. Disk, speakers, flash drive, keyboard, etc.)

`/etc` - configuration files

`/bin >> /usr/bin` - every user commands

`/sbin >> /usr/sbin` - System/filesystem commands

`/opt` - Optional add-on applications

`/proc` - Running processes(Only exist in memory)

`/lib >> /usr/lib` - C programming library files needed by commands and apps

`/temp` - Directory for temporary files

`/home` - Directory for user

`/var` - system logs

`/run` - System daemons that start very early to store temporary runtime files

`/mnt` - To mount external filesystem

`/media` - For cdrom mounts

File System Navigation

- `ls` – List file in a Directory

```
shehan@DESKTOP-DPJDV96:/$ ls
bin   dev  home  lib    lib64   lost+found  mnt  proc  run   snap  sys  usr  wslDhaPif  wslHcdfJB  wsldFEjjk  wslljBmAe
boot  etc  init  lib32  libx32   media       opt  root  sbin  srv   tmp  var  wslHEHcjA  wslHcgcLH  wsldGHLdH  wslpDAfPi
```

- `ls -l` or `ls -ltr` – List with file properties
- `ls -la` – List with hidden files (Hidden files are started with .)

```
shehan@DESKTOP-DPJDV96:/$ ls -ltr
total 2156
drwxr-xr-x   2 root root    4096 Apr 18  2022 boot
lrwxrwxrwx   1 root root       7 Nov 23  2023 bin -> usr/bin
lrwxrwxrwx   1 root root       8 Nov 23  2023 sbin -> usr/sbin
lrwxrwxrwx   1 root root      10 Nov 23  2023 libx32 -> usr/libx32
lrwxrwxrwx   1 root root       9 Nov 23  2023 lib64 -> usr/lib64
lrwxrwxrwx   1 root root       9 Nov 23  2023 lib32 -> usr/lib32
lrwxrwxrwx   1 root root       7 Nov 23  2023 lib -> usr/lib
drwxr-xr-x   2 root root    4096 Nov 23  2023 srv
drwxr-xr-x   2 root root    4096 Nov 23  2023 opt
drwxr-xr-x   2 root root    4096 Nov 23  2023 media
drwxr-xr-x  14 root root    4096 Nov 23  2023 usr
drwxr-xr-x  13 root root    4096 Nov 23  2023 var
drwxr-xr-x   8 root root    4096 Nov 23  2023 snap
-rwxrwxrwx   1 root root 2105816 Dec  1  2023 init
```

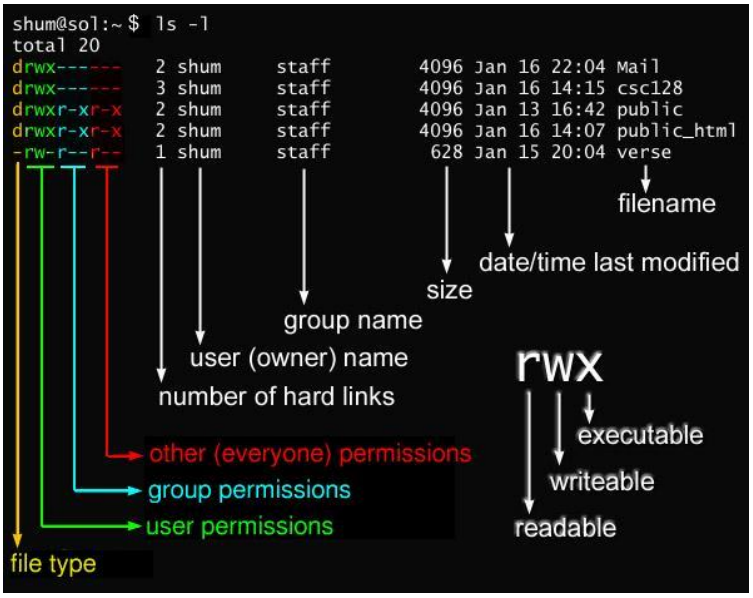
- `cd <directory>` - Change directory
- `cd ..` - Go to previous directory
- `pwd` – Get present working directory

```
shehan@DESKTOP-DPJDV96:~/demo$ pwd
/home/shehan/demo
```

Linux File or Directory Properties

- `ls -l` - For display the properties

We can see there are multiple columns. File sizes are shown in bytes.



The first letter indicates the type of file

File Types	Character in File Listing	Description
Regular file	-	text, data, or executable files
Directory	d	Folder
Link	l	A shortcut that points to the location of the actual file
Special File	c	Mechanism used for input and output, such as files in /dev
Block Disk	b	Any kinds of dis, it will be read as block disk.
Socket	s	A special file that provides inter-process networking protected by the file system's access control
Pipe	p	A special file that allows processes to communicate with each other without using network socket semantics.

What is Root

There are three types of root on Linux

1. Root account - Most powerful account
2. Root as / -The very first directory in Linux is also referred as root directory
3. Root home directory - The root filesystem is the primary filesystem on a Linux system, containing the root directory and all essential system files needed for the operating system to function. Located in /root

File System Paths

Absolute Path

Absolute Path can be used to go directly to a specific path

- `cd /var/log/my`

Relative Path

Relative Path can be used for navigate to a path relative to your current path

- `cd my`

Creating Files and Directories

We can use the following commands to create files

- `touch <filename>` or for multiple files `touch <file1> <file2> <file 3>`
- `Vi <filename>` or `vim <filename>`

- `mkdir <nameofdirectory>` - to create a directory

Copy Files

- `cp <source> <destination>`

Copy Directories

For copy directories with all files and sub directories we use -R

- `cp -R <source> <destination>`

Move Files and Directories

mv command can be used for moving and renaming files and directories.

- `mv <source> <destination>`

Delete Files and Directories

`Rm <filename>` - Delete a file

`rm -R <directory>` - Delete a directory

Find Files and Directories

find

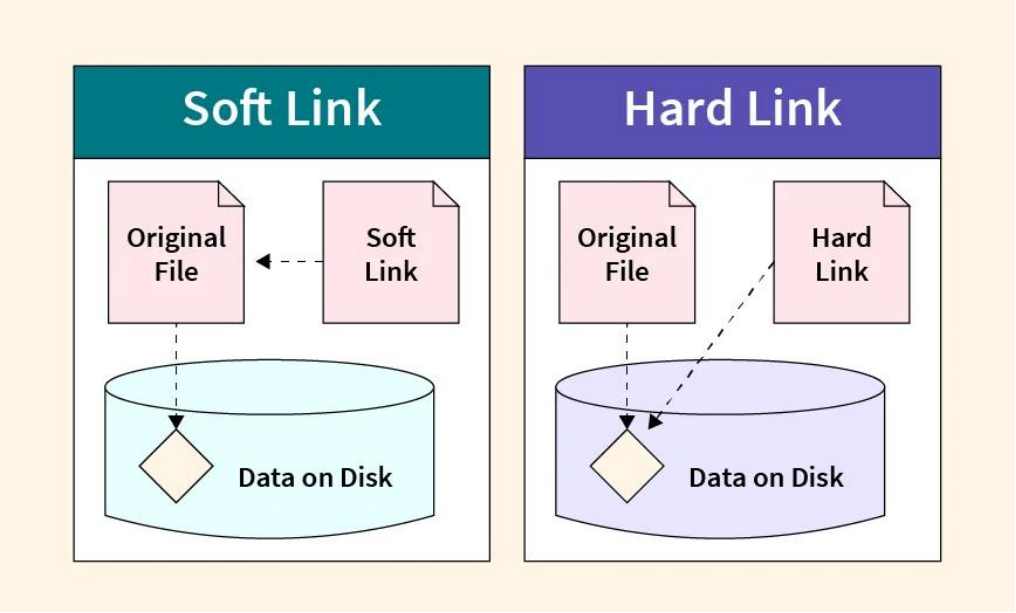
`find <directory> -name “<filename>”`

locate

Locate uses a prebuilt database, which should be regularly updated. Find iterates over a filesystem to locate files. Locate is much faster.

To update locate database run `updated`

Soft Link and Hard Link



A soft link is a pointer to the original file, similar to a shortcut in Windows

A hard link is an additional name for the same data on disk.

`ln -s /home/user/original.txt /home/user/link.txt` – Create a Soft Link

`ln /home/user/original.txt /home/user/link.txt –` Create a Hard Link

```
shehan@DESKTOP-DPJDV96:~/demo/new$ ls -l
total 0
-rw-r--r-- 2 shehan shehan  0 Jan  9 21:53 hardlink
-rw-r--r-- 2 shehan shehan  0 Jan  9 21:53 originalfile
lrwxrwxrwx 1 shehan shehan 12 Jan  9 21:53 softlink -> originalfile
```

When we run `ls -l` command it shows soft link and hard link as above. Note that soft link file type shown as `l`

Wildcards

Wildcards are special characters used in Linux to represent other characters in filenames and paths. They are commonly used in command-line operations to simplify tasks that involve multiple files. Here are some common wildcards

Asterisk (*)

Matches zero or more characters.

- Example: `*.txt` matches all files with a `.txt` extension.

Question Mark (?)

Matches exactly one character.

- Example: `file?.txt` matches `file1.txt`, `file2.txt`, but not `file10.txt`.

Square Brackets ([])

Matches any one of the enclosed characters.

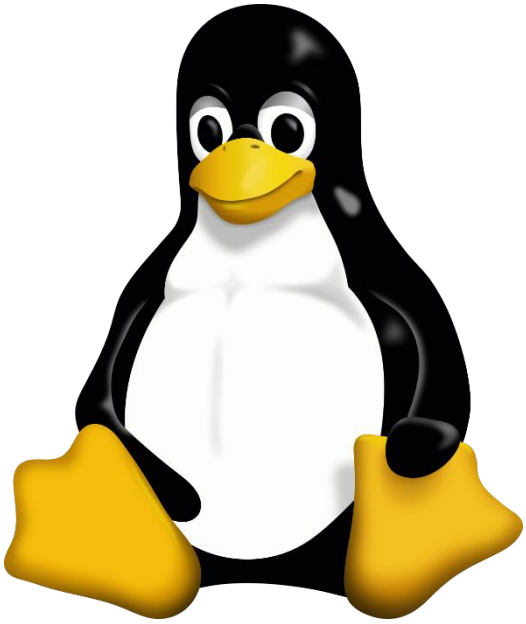
- Example: `file[123].txt` matches `file1.txt`, `file2.txt`, `file3.txt`.

Caret (^)

(Used in regular expressions, not as a wildcard in standard shell operations):

In regular expressions, the caret `^` is used to match the beginning of a line.

- Example: `^file` matches any line that starts with "`file`".



Keep Learning 😃

linkedin.com/in/shehan-arampola

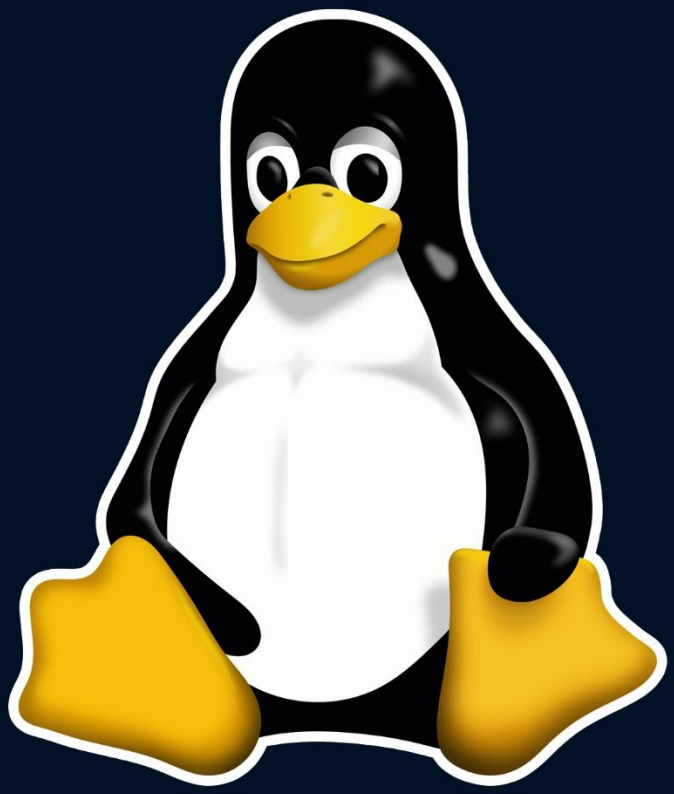
github.com/shehan404

LINUX SIMPLIFIED

Part 02

File Permissions

Let's go →



linkedin.com/in/shehan-arampola
github.com/shehan404

Previous Lessons

- Linux Simplified Part 01 - Linux File System

File Permissions

File permissions in Linux systems are important for ensuring security, privacy, and proper access control. They define who can read, write, or execute a file, securing data and preventing unauthorized actions. In organizational settings, they help to reduce vulnerabilities to data breaches. Properly configured permissions not only prevent accidental or malicious modifications but also allow efficient resource management and role-based access.

To see the permissions go to a directory which contains some files and run `ls -ltr` or `ls -l`

```
shehan@DESKTOP-DPJDV96:~/perm$ ls -ltr
total 4
-rw-r--r-- 1 shehan shehan    0 Jan 10 13:19 run.sh
-rw-r--r-- 1 shehan shehan    0 Jan 10 13:19 my.txt
-rw-r--r-- 1 shehan shehan    0 Jan 10 13:19 file1
-rw-r--r-- 1 shehan shehan    0 Jan 10 13:19 doc.txt
drwxr-xr-x 2 shehan shehan 4096 Jan 10 13:20 myfolder
```

As an example, consider the first column of the my.txt file. It is as follow

-rw-r--r--

Those characters represent following permissions

-	File type
rw-	User Permissions
r--	Group Permissions
r--	Other Permission

Each permission has three letters

- First letter - `read` permission (`r`)
- Second letter – `write` permission (`w`)
- Third letter – `execute` permission (`x`)

If there are `no` permissions, it indicates with a dash (`-`)

Change permissions

`chmod` command is used for changing permissions files and directories.

Adding permissions

To add permissions, we can use

```
chmod <user/group/other/all>+<type of permissions> <filename>
```

As an example, if you need to add execute permission for the user to my.txt fil, you can run `chmod u+x my.txt`

Before adding execute permissions

```
-rw-r--r-- 1 shehan shehan    0 Jan 10 13:19 my.txt
```

After adding execute permissions

```
-rwxr--r-- 1 shehan shehan    0 Jan 10 13:19 my.txt
```

We can add multiple permissions at once

```
chmod u+rw my.txt
chmod g+rwx my.txt
chmod o+rx my.txt
chmod a+wx my.txt
```

Removing permissions

To remove permissions, we can use

```
chmod <user/group/other/all>-<type of permissions> <filename>
```

Examples

```
chmod u-rw my.txt
chmod g-rwx my.txt
chmod o-rx my.txt
chmod a-wx my.txt
```

Chmod with numeric mode

There is a numerical method add and remove permissions in chmod

Symbolic	Numeric	Permission
---	0	None
--x	1	Execute
-w-	2	Write
-wx	3	Write + Execute
r--	4	Read
r-x	5	Read + Execute
rw-	6	Read + Write
rwx	7	Read + Write + Execute

drwxrwxrwx

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

chmod 777

rwx | rwx | rwx
Owner | Group | Others

Example

```
chmod 345 my.txt
```

write + execute for user (3) **|** read for group (4) **|** read + execute for others (5)

When change permissions in a directory if you need to change the permission of subdirectories and files as well use `-R`

```
chmod -R 345 myfolder
```

```
chmod -R u-rw myfolder
```

File Ownership

In Linux/Unix systems, file ownership determines who has control over a file or directory. Each file or directory has three types of owners

1. User (Owner): The person who created the file, typically the default owner.
2. Group: A collection of users who share access to the file.
3. Others: Everyone else on the system.

The chown Command

The `chown` command is used to change the ownership of a file or directory

```
chown <options> <user> <file>
```

For the option we can use -R for recursive changes for directories.

Run `ls -ltr` to see the current user and group

```
-rw-r--r-- 1 shehan shehan    0 Jan 10 13:19 doc.txt
```

It shows that in second column user(owner) is shehan and in third column group is shehan

Run `sudo chown root doc.txt` for change ownership to the root

```
-rw-r--r-- 1 root   shehan    0 Jan 10 13:19 doc.txt
```

It changes owner as the root.

The chgrp Command

The `chgrp` command is used to change the ownership of a file or directory

`chgrp <options> <group> <file>`

For the option we can use -R for recursive changes for directories.

Run `ls -ltr` to see the current user and group

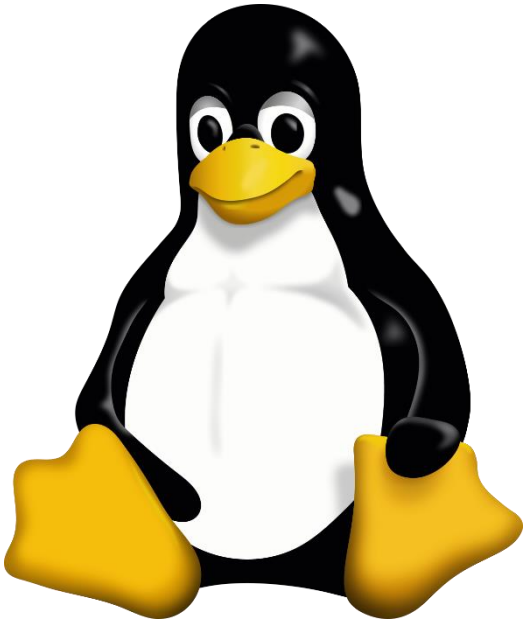
```
-rw-r--r-- 1 root   shehan    0 Jan 10 13:19 doc.txt
```

It shows that in second column user(owner) is root and in third column group is shehan

Run `sudo chgrp root doc.txt` for change ownership to the root

```
-rw-r--r-- 1 root   root      0 Jan 10 13:19 doc.txt
```

It changes group as the root.



Keep Learning 🚀

linkedin.com/in/shehan-arampola

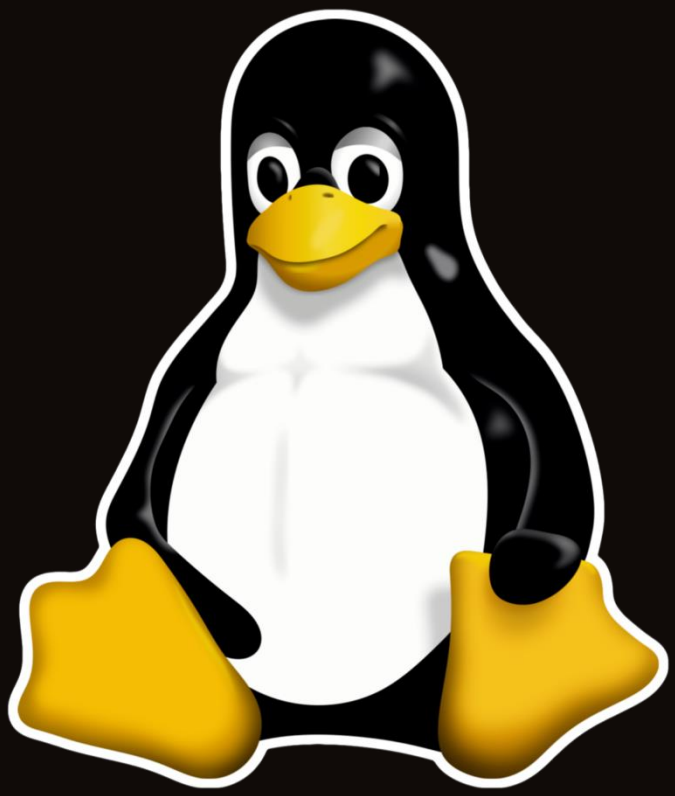
github.com/shehan404

LINUX SIMPLIFIED

Part 03

Text Processing

Let's go →



linkedin.com/in/shehan-arampola
github.com/shehan404

Previous Lessons

- Linux Simplified Part 01 - Linux File System
- Linux Simplified Part 02 – File Permissions

File Display Commands

There are multiple ways to display the content in a file

- `cat` - Display entire file
- `more` - Displays the content of a file one screen (or page) at a time.
- `less` - Displays the content of a file with more advanced features.
- `head` - Display first 10 lines of a file
- `tail` - Display last 10 lines of a file

`less` command is a powerful tool which has following features

- Spacebar: Move to the next page.
- b: Move to the previous page.
- Up/Down Arrows: Move line by line.
- g: Go to the beginning of the file.
- G: Go to the end of the file.
- /pattern: Search for a pattern (moves forward).
- ?pattern: Search for a pattern (moves backward).
- n: Repeat the search in the same direction.
- N: Repeat the search in the opposite direction.
- q: Quit.

Adding text to a file

There are multiple text editors in Linux. Vi, Vim and Nano are widely used editors.

Redirects

Apart from the text editors we can write output of a command using redirects (> and >>). > use for write output to a file and >> use for write output in new line.

```
shehan@DESKTOP-DPJDV96:~$ ls -ltr > filelist
shehan@DESKTOP-DPJDV96:~$ cat filelist
total 100
-rw-r--r-- 1 shehan shehan     0 Jul 12  2024 newnew
-rw-r--r-- 2 shehan shehan     0 Jul 12  2024 heyhard
-rw-r--r-- 2 shehan shehan     0 Jul 12  2024 hey
-rw-r--r-- 1 shehan shehan     0 Jul 12  2024 wow
-rwxrwxrwx 1 shehan shehan     8 Jul 12  2024 this
lrwxrwxrwx 1 shehan shehan     5 Jul 12  2024 heysoft -> ./hey
-rw-r--r-- 1 shehan shehan   587 Jul 24 22:20 listing
-rw-r--r-- 1 shehan shehan    53 Jul 24 22:27 errorfile
-rw-r--r-- 1 shehan shehan   629 Jul 24 22:38 lsnew
-rw-r--r-- 1 shehan shehan     0 Jul 25 13:18 wowrenamed
-rw-r----- 1 shehan shehan 32914 Jul 25 13:37 dmesg
-rw-r--r-- 1 shehan shehan    47 Jul 25 13:52 mee
-rw-r--r-- 1 shehan shehan    56 Jul 25 13:55 newfit
-rw-r--r-- 1 shehan shehan   182 Jul 25 16:06 errorfile.tar.gz
-rw-r--r-- 1 shehan shehan    55 Jul 25 21:44 myfirstvi
-rw-r--r-- 1 shehan shehan   140 Jul 25 21:55 newvi
-rw-rw-r-- 1 shehan shehan    26 Jul 29 18:09 crontab-entry
-rw-r--r-- 1 shehan shehan     0 Jul 30 18:21 error.txt
drwxr-xr-x 2 shehan shehan  4096 Jul 30 19:40 shelscripting
-rw-r--r-- 1 shehan shehan    10 Nov 12 12:28 my.sh
drwxr-xr-x 4 shehan shehan  4096 Jan  1 14:07 monitoring
drwxr-xr-x 4 shehan shehan  4096 Jan  9 21:52 demo
drwxr-xr-x 3 shehan shehan  4096 Jan 10 13:19 perm
drwxr-xr-x 2 shehan shehan  4096 Jan 13 04:18 text
-rw-r--r-- 1 shehan shehan     0 Jan 13 04:40 filelist
```

```
shehan@DESKTOP-DPJDV96:~$ pwd > currentpath.txt
shehan@DESKTOP-DPJDV96:~$ cat currentpath.txt
/home/shehan
shehan@DESKTOP-DPJDV96:~$
```

```
shehan@DESKTOP-DPJDV96:~$ ls >> currentpath.txt
shehan@DESKTOP-DPJDV96:~$ cat currentpath.txt
/home/shehan
crontab-entry
currentpath.txt
demo
dmesg
error.txt
errorfile
errorfile.tar.gz
filelist
hey
heyhard
heysoft
listing
lsnew
mee
monitoring
my.sh
myfirstvi
newfit
newnew
newvi
perm
shelscripting
text
this
wow
wowrenamed
```

Tee command

Tee command can display the result while writing it into a file.

```
shehan@DESKTOP-DPJDV96:~$ pwd | tee presentdir
/home/shehan
shehan@DESKTOP-DPJDV96:~$ cat presentdir
/home/shehan
shehan@DESKTOP-DPJDV96:~$
```

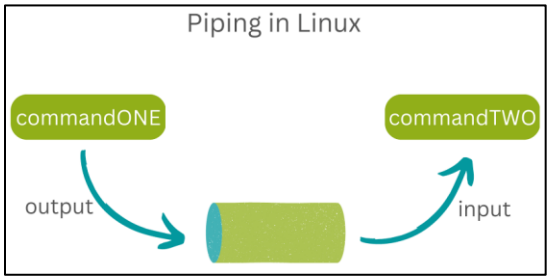
To avoid the override the file use -a option

```
<command> | tee -a <filename>
```

Pipe "|"

A pipe used to direct the output of one command as the input to another command. It enables chaining multiple commands.

`commandONE | commandTWO`




```
shehan@DESKTOP-DPJDV96:~$ ls -ltr | head
total 112
-rw-r--r-- 1 shehan shehan     0 Jul 12  2024 newnew
-rw-r--r-- 2 shehan shehan     0 Jul 12  2024 heyhard
-rw-r--r-- 2 shehan shehan     0 Jul 12  2024 hey
-rw-r--r-- 1 shehan shehan     0 Jul 12  2024 wow
-rwxrwxrwx 1 shehan shehan     8 Jul 12  2024 this
lrwxrwxrwx 1 shehan shehan     5 Jul 12  2024 heysoft -> ./hey
-rw-r--r-- 1 shehan shehan   587 Jul 24 22:20 listing
-rw-r--r-- 1 shehan shehan    53 Jul 24 22:27 errorfile
-rw-r--r-- 1 shehan shehan   629 Jul 24 22:38 lsnew
```

Filters

When working with files occasionally we have filter out some outputs. In those scenarios filters are important. The following are the widely used filters in Linux.

`cut, awk, grep and egrep, sort, uniq, wc`

Here is a file with list of names we use for the demonstration.

```
shehan@DESKTOP-DPJDV96:~/text$ cat names
John Smith
Emma Johnson
Michael Brown
Olivia Davis
William Wilson
Sophia Moore
James Taylor
Isabella Anderson
Benjamin Thomas
Mia Jackson
```

cut Command

Extracts parts of each line based on delimiter or byte/character position.

- **Extract specific characters (`cut -c1,2,7 file`)**

```
shehan@DESKTOP-DPJDV96:~/text$ cut -c1,2,7 names
Jom
Emo
Mil
Ol
Wim
So
JaT
Isl
Bei
Mic
```

- **Extract range of characters (`cut -c1-3,5-7 file`)**

```
shehan@DESKTOP-DPJDV96:~/text$ cut -c1-3,5-7 names
Joh Sm
Emm Jo
Micael
Oliia
Wiliam
Sopia
Jams T
Isaell
Benami
MiaJac
```

- **Extract by bite size (`cut -b1-3,5-7 file`)**

```
shehan@DESKTOP-DPJDV96:~/text$ cut -b1-3,5-7 names
Joh Sm
Emm Jo
Micael
Oliia
Wiliam
Sopia
Jams T
Isaell
Benami
MiaJac
```

- **Extract Specific Fields**

`cut -d':' -f1 /etc/passwd`

Above command extract `field 1` from fields separated by ':' of the `passwd` file.

```
shehan@DESKTOP-DPJDV96:~/text$ cat /etc/passwd | head
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
```

```
shehan@DESKTOP-DPJDV96:~/text$ cut -d':' -f1 /etc/passwd
root
daemon
bin
sys
sync
games
man
lp
mail
news
```

- **It is not only limited to files**

```
shehan@DESKTOP-DPJDV96:~/text$ echo "Hello world" | cut -c1-7
Hello w
```

awk Command

The awk command is a powerful text-processing tool in Linux. It is mainly used for pattern scanning, data extraction, and text manipulation.

Basic syntax: `awk 'pattern {action}' file`

- **Print the Entire File (`awk '{print}' file`)**

```
shehan@DESKTOP-DPJDV96:~/text$ awk '{print}' names.txt
John Smith
Emma Johnson
Michael Brown
Olivia Davis
William Wilson
Sophia Moore
James Taylor
Isabella Anderson
Benjamin Thomas
Mia Jackson
Daniel Brown
Will taylor
```

- **Print Specific Columns, Fields (`awk '{print $1}' file`)**

```
shehan@DESKTOP-DPJDV96:~$ awk '{print $1,$3}' filelist
total
-rw-r--r-- shehan
-rw-r--r-- shehan
-rw-r--r-- shehan
-rw-r--r-- shehan
-rwxrwxrwx shehan
lrwxrwxrwx shehan
```

- **Specify a Field Separator (`awk -F ':' '{print $2}' file`)**

```
shehan@DESKTOP-DPJDV96:~$ awk -F ':' '{print $6}' /etc/passwd
/root
/usr/sbin
/bin
/dev
/bin
/usr/games
/var/cache/man
/var/spool/lpd
/var/mail
```

- **Match Patterns (`awk '/pattern/ {print}' file.txt`)**

```
shehan@DESKTOP-DPJDV96:~/text$ awk '$2 == "Brown" {print $1,$2}' names.txt
Michael Brown
Daniel Brown
```

- **Print Number of Fields (`awk '{print NF}' file`)**

```
shehan@DESKTOP-DPJDV96:~/text$ awk '{print NF}' names.txt
2
2
2
2
2
```

And there are much more….

grep/egrep Command

grep searches for a specific pattern in a file or input.

`grep "pattern" file.txt`

- `grep <pattern> file` – Search for the pattern

```
shehan@DESKTOP-DPJDV96:~/text$ grep John names
John Smith
Emma Johnson
```

- `grep -c <pattern> file` – Search for keyword and give count

```
shehan@DESKTOP-DPJDV96:~/text$ grep -c John names
2
```

- `grep -i <pattern> file` – Search for pattern ignoring case-sensitive

```
shehan@DESKTOP-DPJDV96:~/text$ grep -i taylor names
James Taylor
Will taylor
```

- `grep -n <pattern> file` – Search for pattern & display with line numbers

```
shehan@DESKTOP-DPJDV96:~/text$ grep -n Brown names
3:Michael Brown
11:Daniel Brown
```

- `grep -v <pattern> file` – Display lines don't have the pattern

```
shehan@DESKTOP-DPJDV96:~/text$ grep -v Brown names
John Smith
Emma Johnson
Olivia Davis
William Wilson
Sophia Moore
James Taylor
Isabella Anderson
Benjamin Thomas
```

- `egrep -i "<pattern1>|<pattern2>" file` – Search with two patterns

```
shehan@DESKTOP-DPJDV96:~/text$ egrep -i "john|brown" names
John Smith
Emma Johnson
Michael Brown
Daniel Brown
```

sort/uniq Command

The sort and uniq commands are powerful tools for sorting and manipulating text data.

Sort Alphabetically (`sort file`)

```
shehan@DESKTOP-DPJDV96:~/text$ sort names.txt
Benjamin Thomas
Daniel Brown
Emma Johnson
Isabella Anderson
James Taylor
John Smith
Mia Jackson
Michael Brown
```

Sort in Reverse Order (`sort -r file`)

```
shehan@DESKTOP-DPJDV96:~/text$ sort -r names.txt
William Wilson
Will taylor
Sophia Moore
Olivia Davis
Michael Brown
Mia Jackson
John Smith
```

Sot by field number (`sort -k2 file`)

```
shehan@DESKTOP-DPJDV96:~/text$ sort -k2 names.txt
Isabella Anderson
Daniel Brown
Michael Brown
Olivia Davis
Mia Jackson
Emma Johnson
Sophia Moore
John Smith
```

- **Filter Duplicate Lines (`uniq file.txt`)**
- **Count Duplicate Lines (`uniq -c file.txt`)**
- **Sort and Remove Duplicates (`sort file.txt | uniq`)**

wc Command

- **Get new line count , Word count and byte count (`wc file`)**

```
shehan@DESKTOP-DPJDV96:~/text$ wc names.txt
 12  24 163 names.txt
```

- **Get the number of lines (`wc -l file`)**
- **Get the number of words (`wc -w file`)**
- **Get the number of characters (`wc -c file`)**

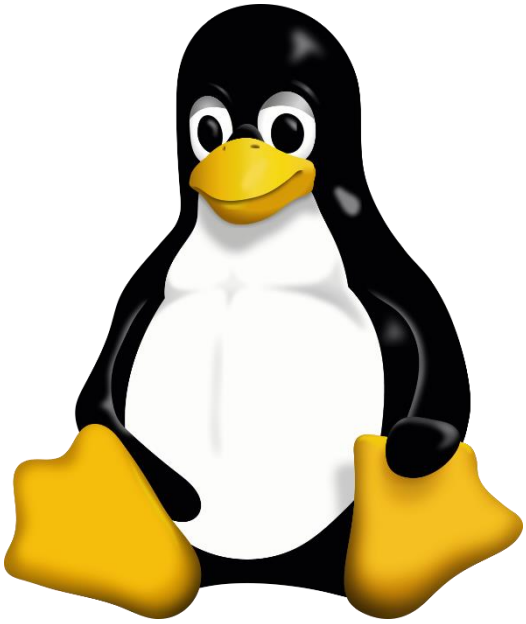
```
shehan@DESKTOP-DPJDV96:~/text$ wc -l names.txt
12 names.txt
shehan@DESKTOP-DPJDV96:~/text$ wc -w names.txt
24 names.txt
shehan@DESKTOP-DPJDV96:~/text$ wc -c names.txt
163 names.txt
```

Compare files

We can compare difference between two files

- `diff <file1> <file2>` - compare files line by line
- `cmp <file1> <file2>` - compare two files byte by byte

```
shehan@DESKTOP-DPJDV96:~/text$ diff names1.txt names3.txt
12d11
< Will taylor
shehan@DESKTOP-DPJDV96:~/text$ cmp names1.txt names3.txt
cmp: EOF on names3.txt after byte 151, line 11
```



Keep Learning 🚀

linkedin.com/in/shehan-arampola

github.com/shehan404

LINUX SIMPLIFIED

Part 04

User Account Management

Let's go →

linkedin.com/in/shehan-arampola
github.com/shehan404

Previous Lessons

- Linux Simplified Part 01 - Linux File System
- Linux Simplified Part 02 – File Permissions
- Linux Simplified Part 03 – Text Processing

User Account Management

User account management is the process of creating, managing, and deleting user accounts. It includes managing user permissions, access, and activity.

The following commands are mainly used for creating and deleting users and groups. You have to switch to root account using `su -` command to use these commands

- `useradd`
- `groupadd`
- `usermod`
- `userdel`
- `grouped`

These user accounts are managed in the following files

- `/etc/passwd`

  Stores information about user accounts. It is a plain text file readable by all users, but it doesn't contain sensitive information like passwords. File Format - each line represents a user and has seven fields separated by colons. X – placeholder for password

  `username:x:UID:GID:comment:home_directory:shell`

- `/etc/group`

  Stores group information. Like /etc/passwd, it is a plain text file readable by all users. File Format - each line represents a group and has four fields separated by colons.

  `groupname:x:GID:members`

- `/etc/shadow` Stores encrypted user passwords and other password-related settings. This file is readable only by the root user or processes with elevated privileges for security. File Format: Each line represents a user and has nine fields separated by colons.

  `username:password:last_change:min_days:max_days:warn_days:inactive_days:expire_date:reserved`

useradd

The `useradd` command creates a new user account.

`useradd [options] username`

Options

- -m: Creates the user's home directory if it doesn't exist.
- -s: Specifies the user's login shell (e.g., /bin/bash).
- -c: Adds a comment (e.g., full name or description).
- -G: Adds the user to supplementary groups.
- -d: Specifies a custom home directory.

We can run `ls -ltr` at `home` directory to see existing users. Here we have only user account called “shehan”

```
root@DESKTOP-DPJDV96:/# ls -ltr home/
total 4
drwxr-x--- 9 shehan shehan 4096 Jan 13 16:58 shehan
root@DESKTOP-DPJDV96:/#
```

Let’s create a user called “john”

`useradd -m -s /bin/bash -c "John Brown" john`

```
root@DESKTOP-DPJDV96:/# useradd -m -s /bin/bash -c "John Brown" john
root@DESKTOP-DPJDV96:/# ls -l home/
total 8
drwxr-x--- 2 john   john   4096 Jan 17 19:50 john
drwxr-x--- 9 shehan shehan 4096 Jan 13 16:58 shehan
root@DESKTOP-DPJDV96:/#
```

groupadd

The `groupadd` command creates a new group.

`groupadd [options] groupname`

Options:

- -g: Specifies the Group ID (GID).
- -r: Creates a system group (used for system tasks).

Let’s create a group called developers.

`groupadd developers`

```
root@DESKTOP-DPJDV96:/# groupadd developers
```

We can confirm that whether the group is created by checking `/etc/group` file by running `cat /etc/group`

```
admin:x:115:
netdev:x:116:shehan
shehan:x:1000:
john:x:1001:
developers:x:1002:
```

usermod

The `usermod` command modifies a user account.

`usermod [options] username`

Options:

- -G: Adds the user to supplementary groups (replacing current group membership).
- -aG: Appends the user to supplementary groups.
- -s: Changes the user's login shell.
- -d: Changes the user's home directory.
- -l: Changes the username.
- -L: Locks the user's account.
- -U: Unlocks the user's account.

Lets add “john” to the “devolopers” group

`usermod -G developers john`

```
admin:x:115:
netdev:x:116:shehan
shehan:x:1000:
john:x:1001:
developers:x:1002:john
```

userdel

The `userdel` command removes a user account.

`userdel [options] username`

Options:

- -r: Removes the user's home directory and mail spool

Let's delete user "john"

`userdel -r john`

```
root@DESKTOP-DPJDV96:/home# userdel -r john
userdel: john mail spool (/var/mail/john) not found
root@DESKTOP-DPJDV96:/home# ls
shehan
```

groupdel

The `groupdel` command removes a group.

`groupdel groupname`

Let's delete developers group

`groupdel developers`

```
admin:x:115:
netdev:x:116:shehan
shehan:x:1000:
```

Switch Users and sudo

- `su - <username>` - switch user
- `sudo <command>` - "superuser do" run commands as root
- `visudo` - The visudo command is used to edit the sudoers (`/etc/sudoers`) file safely. The sudoers file defines who has sudo privileges and what commands they can run.

Monitor users

- `who` - Displays information about users currently logged into the system.

Username | terminal logged into | time they logged in | IP address (if remote)

```
shehan@DESKTOP-DPJDV96:~$ who
shehan   pts/1        2025-01-19 17:13
```

- `Last` - Displays the login history of users.

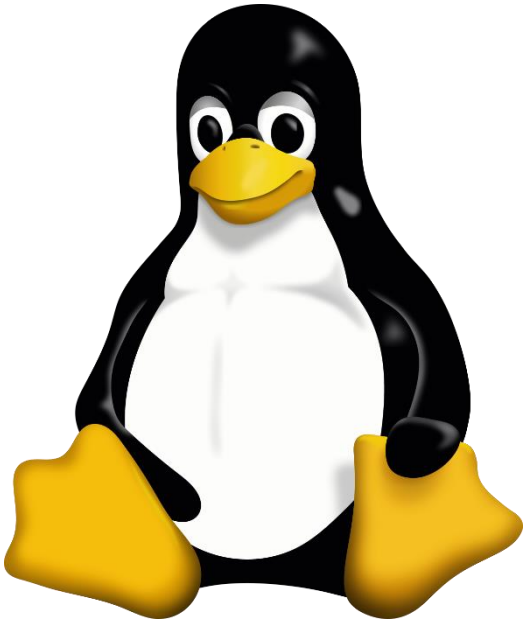
```
shehan@DESKTOP-DPJDV96:~$ last
shehan   pts/1                        Sun Jan 19 17:13   still logged in
reboot   system boot  5.15.133.1-micro Sun Jan 19 17:13   still running
shehan   pts/1                        Fri Jan 17 18:53 - crash (1+22:19)
reboot   system boot  5.15.133.1-micro Fri Jan 17 18:53   still running
shehan   pts/1                        Fri Jan 17 18:45 - crash  (00:08)
reboot   system boot  5.15.133.1-micro Fri Jan 17 18:45   still running
shehan   pts/1                        Wed Jan 15 02:49 - crash (2+15:55)
reboot   system boot  5.15.133.1-micro Wed Jan 15 02:49   still running
shehan   pts/1                        Mon Jan 13 16:17 - crash (1+10:32)
reboot   system boot  5.15.133.1-micro Mon Jan 13 16:17   still running
shehan   pts/1                        Mon Jan 13 04:15 - crash  (12:01)
reboot   system boot  5.15.133.1-micro Mon Jan 13 04:15   still running
shehan   pts/1                        Fri Jan 10 13:18 - crash (2+14:57)
```

- `w` - Shows information about logged-in users and what they are doing.

```
shehan@DESKTOP-DPJDV96:~$ w
 17:57:55 up 44 min,  1 user,  load average: 0.08, 0.05, 0.02
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
shehan   pts/1    -                17:13   44:24   0.07s  0.05s -bash
```

- `Id` - Displays user ID (UID), group ID (GID), and group memberships of a

```
shehan@DESKTOP-DPJDV96:~$ id shehan
uid=1000(shehan) gid=1000(shehan) groups=1000(shehan),4(adm),20(dialout),24(cdrom),25(floppy),27(sudo),29(audio),
30(dip),44(video),46(plugdev),116(netdev)
```



Keep Learning 🚀

linkedin.com/in/shehan-arampola

github.com/shehan404

LINUX SIMPLIFIED

Part 05

Utilities, Scheduling, and Process Management

Let's go →

Shehan Arampola

linkedin.com/in/shehan-arampola

github.com/shehan404

Previous Lessons

- Linux Simplified Part 01 - Linux File System
- Linux Simplified Part 02 – File Permissions
- Linux Simplified Part 03 – Text Processing
- Linux Simplified Part 04 - User Account Management

System Utility Commands

- `date` – Displays the system date and time.
- `hostname` – Displays the hostname of the system.
- `which <command>` - Locates the executable path of a command.
- `cal` - Displays a calendar for the current month or a specific date.
- `bc` – A command line calculator
- `uname -a` - Prints system information such as the kernel name, version, and architecture
- `uptime` - Shows how long the system has been running, current time, number of users, system load averages.

```
shehan@DESKTOP-DPJDV96:~$ date
Tue Jan 21 18:49:21 +0530 2025
shehan@DESKTOP-DPJDV96:~$ uptime
 18:49:25 up 9 min,  1 user,  load average: 0.00, 0.03, 0.01
shehan@DESKTOP-DPJDV96:~$ hostname
DESKTOP-DPJDV96
shehan@DESKTOP-DPJDV96:~$ uname
Linux
shehan@DESKTOP-DPJDV96:~$ which ls
/usr/bin/ls
shehan@DESKTOP-DPJDV96:~$ cal
    January 2025
Su Mo Tu We Th Fr Sa
          1  2  3  4
 5  6  7  8  9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28 29 30 31

shehan@DESKTOP-DPJDV96:~$ bc
bc 1.07.1
Copyright 1991-1994, 1997, 1998, 2000, 2004, 2006, 2008, 2012-2017 Free Software Foundation, Inc.
This is free software with ABSOLUTELY NO WARRANTY.
For details type `warranty'.
4+5
9
quit
```

Processes, Jobs and Scheduling

systemctl Command

systemctl is used for interacting with the systemd service manager in Linux systems. Systemd is a system and service manager for Linux, responsible for managing services and processes.

- **Start, stop or get the status of a service**

  ```
  systemctl start|stop|status <servicename.service>
  ```

- **Enable or disable a service to start on boot**

  ```
  systemctl enable|disable <servicename.service>
  ```

- **Restart a service or reload service's configuration without stopping it**

  ```
  systemctl restart|reload <servicename.service>
  ```

- **List all units**

  ```
  systemctl list-units –all
  ```

- **Control system with systemctl**

  ```
  systemctl poweroff
  systemctl halt
  systemctl reboot
  ```

ps Command

ps command stands for process status and it displays all the currently running processes in the Linux system.

`ps` – shows the processes of the current shell

```
shehan@DESKTOP-DPJDV96:~$ ps
    PID TTY          TIME CMD
    418 pts/0    00:00:00 bash
   6238 pts/0    00:00:00 ps
```

PID = the unique process ID
TTY = terminal type that the user logged-in to
TIME = amount of CPU time (in min and sec) that the process has been running
CMD = name of the command

`ps -e` – Shows all running processes.

```
shehan@DESKTOP-DPJDV96:~$ ps -e
    PID TTY          TIME CMD
      1 ?        00:00:32 systemd
      2 ?        00:00:00 init-systemd(Ub
      9 ?        00:00:00 init
     47 ?        00:00:00 systemd-journal
     66 ?        00:00:00 systemd-udevd
     80 ?        00:00:00 snapfuse
     89 ?        00:00:00 snapfuse
     92 ?        00:00:00 snapfuse
     93 ?        00:00:01 snapfuse
    103 ?        00:00:07 snapfuse
    107 ?        00:00:00 snapfuse
    112 ?        00:00:00 snapfuse
    117 ?        00:00:03 snapfuse
    126 ?        00:00:00 systemd-resolve
```

`ps aux` - Shows all running processes in BSD format.

```
shehan@DESKTOP-DPJDV96:~$ ps aux
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  1.8  0.1 165872 11124 ?        Ss   15:15   0:35 /sbin/init
root           2  0.0  0.0   2460  1340 ?        Sl   15:15   0:00 /init
root           9  0.0  0.0   2460     4 ?        Sl   15:15   0:00 plan9 --control-socket 6 --log-level 4 --server-fd 7
root          47  0.0  0.1  47740 14916 ?        S<s  15:15   0:00 /lib/systemd/systemd-journald
root          66  0.0  0.0  21836  5668 ?        Ss   15:15   0:00 /lib/systemd/systemd-udevd
root          80  0.0  0.0 152992   156 ?        Ssl  15:15   0:00 snapfuse /var/lib/snapd/snaps/bare_5.snap /snap/bare/
root          89  0.0  0.0 152992  2236 ?        Ssl  15:15   0:00 snapfuse /var/lib/snapd/snaps/core22_1439.snap /snap/
root          92  0.0  0.0 153124   164 ?        Ssl  15:15   0:00 snapfuse /var/lib/snapd/snaps/gtk-common-themes_1535.
root          93  0.1  0.1 452048 15996 ?        Ssl  15:15   0:01 snapfuse /var/lib/snapd/snaps/core22_1722.snap /snap/
root         103  0.3  0.1 526812 15612 ?        Ssl  15:15   0:07 snapfuse /var/lib/snapd/snaps/snapd_23545.snap /snap/
root         107  0.0  0.0 152992  2196 ?        Ssl  15:15   0:00 snapfuse /var/lib/snapd/snaps/snapd_23258.snap /snap/
root         112  0.0  0.0 152992  2200 ?        Ssl  15:15   0:00 snapfuse /var/lib/snapd/snaps/ubuntu-desktop-installe
root         117  0.1  0.1 377284 14104 ?        Ssl  15:15   0:03 snapfuse /var/lib/snapd/snaps/ubuntu-desktop-installe
```

`ps -ef` - Shows all running processes in full format listing (Mostly used).

```
shehan@DESKTOP-DPJDV96:~$ ps -ef
UID          PID    PPID  C STIME TTY          TIME CMD
root           1       0  1 15:15 ?        00:03:18 /sbin/init
root           2       1  0 15:15 ?        00:00:00 /init
root           9       2  0 15:15 ?        00:00:00 plan9 --control-socket 6 --log-level 4 --server-fd 7 --pipe-fd 9 --
root          47       1  0 15:15 ?        00:00:00 /lib/systemd/systemd-journald
root          66       1  0 15:15 ?        00:00:00 /lib/systemd/systemd-udevd
root          80       1  0 15:15 ?        00:00:00 snapfuse /var/lib/snapd/snaps/bare_5.snap /snap/bare/5 -o ro,nodev,
root          89       1  0 15:15 ?        00:00:00 snapfuse /var/lib/snapd/snaps/core22_1439.snap /snap/core22/1439 -o
root          92       1  0 15:15 ?        00:00:00 snapfuse /var/lib/snapd/snaps/gtk-common-themes_1535.snap /snap/gtk
root          93       1  0 15:15 ?        00:00:01 snapfuse /var/lib/snapd/snaps/core22_1722.snap /snap/core22/1722 -o
root         103       1  0 15:15 ?        00:00:07 snapfuse /var/lib/snapd/snaps/snapd_23545.snap /snap/snapd/23545 -o
root         107       1  0 15:15 ?        00:00:00 snapfuse /var/lib/snapd/snaps/snapd_23258.snap /snap/snapd/23258 -o
root         112       1  0 15:15 ?        00:00:00 snapfuse /var/lib/snapd/snaps/ubuntu-desktop-installer_1276.snap /s
```

`ps -u <username>` - Shows all processes by username.

```
shehan@DESKTOP-DPJDV96:~$ ps -u shehan
    PID TTY          TIME CMD
    418 pts/0    00:00:00 bash
    479 ?        00:00:00 systemd
    480 ?        00:00:00 (sd-pam)
    485 pts/1    00:00:00 bash
  42704 pts/0    00:00:00 ps
```

top Command

Top command is used to show the Linux processes and it provides a real-time view of the running system

```
shehan@DESKTOP-DPJDV96:~$ top
top - 18:32:38 up  3:17,  1 user,  load average: 0.01, 0.02, 0.00
Tasks:  35 total,   1 running,  34 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.3 us,  0.2 sy,  0.0 ni, 99.5 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :   7865.1 total,   6570.2 free,    587.7 used,    707.3 buff/cache
MiB Swap:   2048.0 total,   2048.0 free,      0.0 used.   7043.6 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
      1 root      20   0  165872  11168   8152 S   1.7   0.1   3:36.68 systemd
    500 root      20   0   44220  37656  10176 S   1.0   0.5   1:23.72 python3
      2 root      20   0    2460   1340   1224 S   0.0   0.0   0:00.02 init-systemd(Ub
      9 root      20   0    2460      4      0 S   0.0   0.0   0:00.00 init
     47 root      19  -1   47740  14948  13936 S   0.0   0.2   0:00.50 systemd-journal
     66 root      20   0   21836   5668   4348 S   0.0   0.1   0:00.73 systemd-udevd
     80 root      20   0  152992    156      0 S   0.0   0.0   0:00.00 snapfuse
     89 root      20   0  152992   2236     36 S   0.0   0.0   0:00.00 snapfuse
     92 root      20   0  153124    164      0 S   0.0   0.0   0:00.00 snapfuse
     93 root      20   0  452048  15996    384 S   0.0   0.2   0:01.92 snapfuse
    103 root      20   0  526812  15612    264 S   0.0   0.2   0:07.25 snapfuse
    107 root      20   0  152992   2196      4 S   0.0   0.0   0:00.00 snapfuse
    112 root      20   0  152992   2200      4 S   0.0   0.0   0:00.00 snapfuse
    117 root      20   0  377284  14104    264 S   0.0   0.2   0:03.37 snapfuse
```

```
panlab@lab-virtual-machine:~$ top

top - 15:40:31 up 13 days,  2:07,  2 users,  load average: 0.09, 0.04, 0.01
Tasks: 343 total,   1 running, 342 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.7 us,  2.0 sy,  0.0 ni, 97.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :  32087.6 total,  27074.2 free,   1177.6 used,   3835.8 buff/cache
MiB Swap:   2048.0 total,   2048.0 free,      0.0 used.  30420.6 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
   1009 tet-sen+  20   0  496232  69196  59904 S  11.8   0.2  24:01.46 tet-sensor
1106615 panlab    20   0   13360   4480   3456 R   5.9   0.0   0:00.03 top
      1 root      20   0  168056  12996   8132 S   0.0   0.0   0:55.65 systemd
```

System Info

Process Info

- **PID** :Shows task's unique process id
- **USER** :Username of owner of task
- **PR** :The "PR" field shows the scheduling priority of the process from the perspective of the kernel
- **NI** :Represents a Nice Value of task. A Negative nice value implies higher priority, and positive Nice value means lower priority.
- **VIRT** :Total virtual memory used by the task
- **RES** :Memory consumed by the process in RAM
- **SHR** :Represents the amount of shared memory used by a task
- **S** :This field shows the process state in the single-letter form
- **%CPU** :Represents the CPU usage
- **%MEM** :Shows the Memory usage of task
- **TIME+** :CPU Time, the same as 'TIME', but reflecting more granularity through hundredths of a second.

`top -u <username>` - shows tasks/processes by user owned

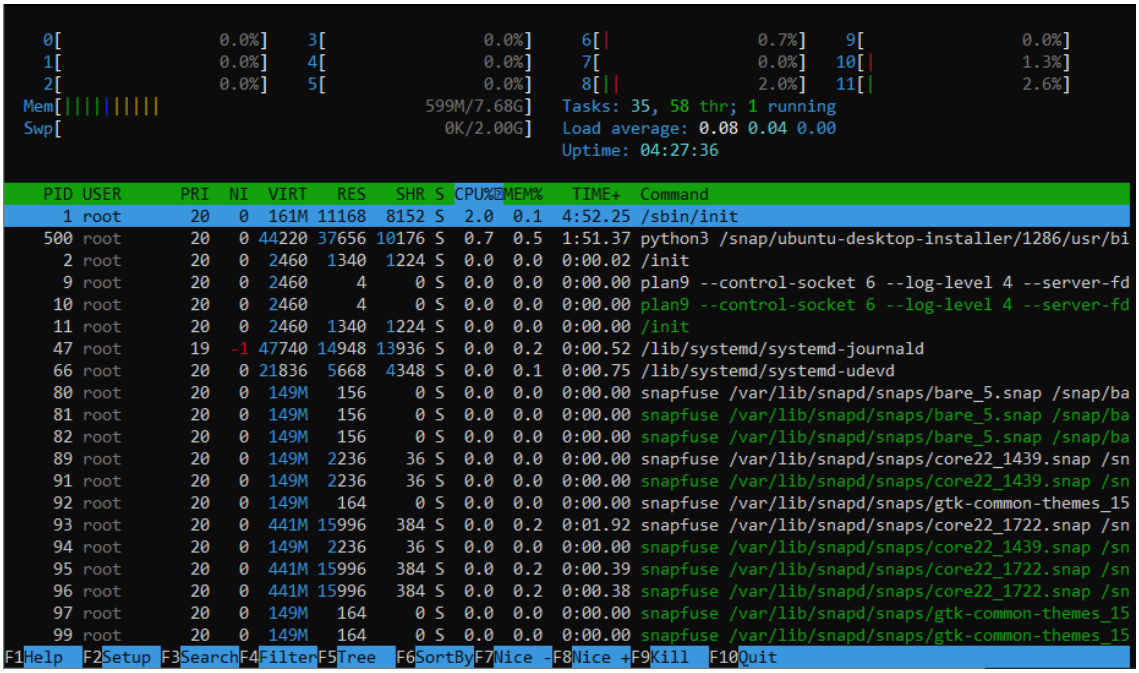
`top then press c` - shows commands absolute path

`top then press k` - kill a process by PID within top session

`top then M and P` - To sort all Linux running processes by Memory usage

htop command

htop is an interactive, user-friendly alternative to the top command in Linux. It allows you to monitor system performance, manage processes, and observe resource usage in real time.



kill Command

kill command use to terminate processes manually

`kill -l` - to get a list of all signal names or signal number

Most used signals are

`kill PID` - Kill a process with default signal

`kill -1 PID` - Restart

`kill -2 PID` - Interrupt from the keyboard just like Ctrl C

`kill -9 PID` - Forcefully kill the process

`kill -15 PID` - Kill a process gracefully

crontab Command

crontab is use for scheduling task. We can schedule repetitive tasks to execute automatically in specific times and intervals. These tasks can be scripts, commands, or programs

- `crontab -e` - Edit the crontab
- `crontab -l` - List the crontab entries
- `crontab -r` - Remove the crontab

```
🎯 A crontab file has five fields for specifying:

*   *   *   *   *   command to be executed
-   -   -   -   -
|   |   |   |   |
|   |   |   |   +--------------- **DAY OF WEEK** (0-6) (Sunday=0)
|   |   |   +------------ **MONTH** (1-12)
|   |   +--------- **DAY OF MONTH** (1-31)
|   +------ **HOUR** (0-23)
+--- **MINUTE** (0-59)
```

Let's schedule a task that writes "`This is my first crontab`" to a file called `mycron` in `every day` at `21:17`

- Run command `crontab -e` and add following entry to that file.
- You can list crontab entries by `crontab -l`

```
# m h  dom mon dow   command
17 21 * * * echo "This is my first crontab" > mycron
```

- After 21:17 you will see a file called `mycron` which contains text "`This is my first crontab`"

```
shehan@DESKTOP-DPJDV96:~$ cat mycron
This is my first crontab
```

at Command

at Command is used for scheduling task as crontab but it executes only once at a specified time.

- `at HH:MM PM` - Schedule a job
- `atq` - List the at entries
- `atrm <Number>` - Remove entries

- `at 2:45 AM 101621` - Schedule a job to run on Oct 16th, 2021 at 2:45am
- `at 4PM + 4 days` - Schedule a job at 4pm four days from now
- `at now +5 hours` - Schedule a job to run five hours from now
- `at 8:00 AM Sun` - Schedule a job to 8am on coming Sunday
- `at 10:00 AM next month` - Schedule a job to 10am next month

Additional cronjobs

In Linux there are four directories for cronjobs which are running hourly, daily, weekly and monthly. We can add our script in to that directories for run in those default time intervals. Those directories are in `/etc` directory

```
shehan@DESKTOP-DPJDV96:/etc$ ls -l /etc/ | grep cron
drwxr-xr-x 2 root root    4096 Nov 23  2023 cron.d
drwxr-xr-x 2 root root    4096 Nov 23  2023 cron.daily
drwxr-xr-x 2 root root    4096 Nov 23  2023 cron.hourly
drwxr-xr-x 2 root root    4096 Nov 23  2023 cron.monthly
drwxr-xr-x 2 root root    4096 Nov 23  2023 cron.weekly
-rw-r--r-- 1 root root    1136 Mar 23  2022 crontab
```

Process Management

bg

The `bg` command resumes a suspended process by running it in the background.

- When you run a command with `&` at the end, it runs in the background.
- If you suspend a running process (using `Ctrl+Z`), you can use `bg` to move it to the background.
- Use the `jobs` command to view background jobs

fg

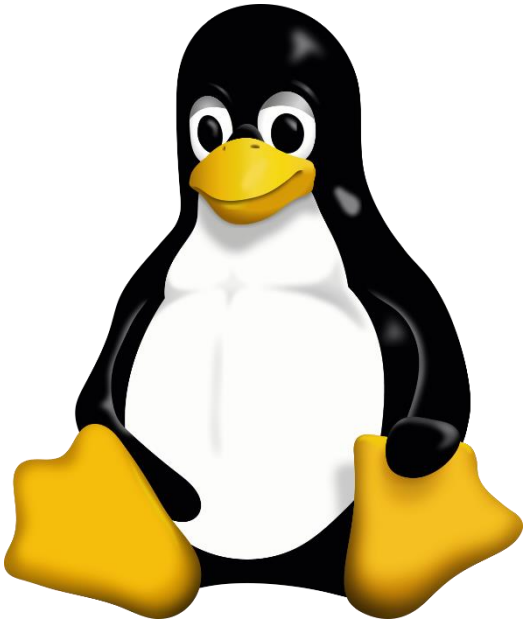
The `fg` command brings a background or suspended process to the foreground.

Use the job number to bring it to the foreground (`fg %1`).

nice

The `nice` command starts a process with a specified priority level. It determines how much CPU time a process gets. Priority ranges from -20 (highest priority) to 19 (lowest priority).

```
nice -n 10 <command>
sudo nice -n -5 <command>
```



Keep Learning 🚀

Shehan Arampola linkedin.com/in/shehan-arampola github.com/shehan404