Linux Navigator: PATH, Repos, Cron, Bash

Check GitHub for helpful DevOps tools:

Michael Robotics
Hi, I'm Michal. I'm a Robotics Engineer and DevOps enthusiast. My mission is to create skill-learning platform that combats information overload by adhering to the set of principles: simplify, prioritize, and execute.

https://github.com/MichaelRobotics

Ask Personal AI Document assistant to learn interactively (FASTER)!

1 Download PDF

2 Go to website

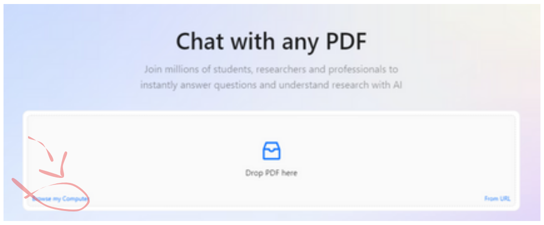
3 Browse file

4 Chat with Document

1 https://github.com/MichaelRobotics/DevOpsTools/blob/main/LinuxNavigator.pdf

Click there to go to ChatPdf website 2

3



Chat with any PDF

Join millions of students, researchers and professionals to instantly answer questions and understand research with AI

Ask questions about document! 4

Complety new to Linux and Networking?

Essential for this PDF is a thorough knowledge of networking. I highly recommend the HTB platform's networking module, which offers extensive information to help build a comprehensive understanding.

HTB - Your Cyber Performance Center

We provide a human-first platform creating and maintaining high performing cybersecurity individuals and organizations.

https://www.hackthebox.com/

What does Linux navigation means?

Linux navigation refers to the process of moving through the file system using the command line interface (CLI), allowing users to access, manage, and manipulate files and directories. It involves using commands to change directories, list files, and execute scripts or programs within the Linux environment.

How it is done?

Navigation in Linux is typically done using commands such as cd (change directory) to move between directories and ls (list) to display the contents of the current directory. Users can also use relative or absolute paths to specify locations and can combine commands with options to enhance functionality, such as ls -l for detailed file listings.

Linux Navigation: Why and When it is usefull to know

Knowing how to navigate in Linux is essential for system administration, troubleshooting, and development tasks, as many operations require command-line interaction.

For example, when managing server files or scripts, efficient navigation allows for quick access to the necessary resources, making tasks like file transfers or configurations more manageable.

System Requirements

- 1 GB RAM (minimum, 2 GB recommended for better performance)
- 10 GB free storage (more may be required depending on the size of the DNS zone files and logs)
- Ubuntu 22.04

Linux Navigation: Main components & packages

- **bash** – The Bourne Again Shell, a popular command-line interface for executing commands and writing shell scripts.
- **coreutils** – A package of essential command-line utilities for file manipulation and system management, including ls, cp, and rm.
- **findutils** – Tools for searching and locating files, primarily find, locate, and updatedb, useful for managing large directories.
- **man-db** – The manual system for Linux, providing access to documentation for commands and programs via the man command.

Linux Navigation: Bash

1) Standard commands

grep: Search for patterns in files or output.

```
grep "pattern" file.txt      # Search for "pattern" in file.txt
grep -r "pattern" dir/       # Recursively search for "pattern" in dir/
```

awk: Extract and manipulate fields from a file.

```
awk '{print $1, $3}' file.txt  # Print 1st and 3rd column from file.txt
awk -F',' '{print $2}' file.csv  # Use a comma as field separator and print 2nd column
```

sed: Stream editor to search, find, and replace text.

```
sed 's/old/new/g' file.txt   # Replace 'old' with 'new' globally in file.txt
sed -n '2,4p' file.txt       # Print lines 2 to 4 in file.txt
```

cut: Extract specific columns from input.

```
cut -d':' -f1 /etc/passwd    # Cut first field (username) from /etc/passwd
```

tr: Translate or delete characters.

```
echo "abc" | tr 'a-z' 'A-Z'  # Convert lowercase to uppercase
```

find: Search for files based on name, size, or other attributes.

```
find /path/to/search -name "*.txt"          # Find all .txt files in directory
find . -type f -size +100M                  # Find files larger than 100 MB
find /dir -name "filename" -exec rm {} \;     # Find and remove files with specific
name
```

locate: Quickly find files using a pre-built index (faster but needs database updates).

```
locate file.txt # Find file.txt anywhere in the system
sudo updatedb # Update the locate database
```

touch: Create an empty file or update its timestamp.

```
touch newfile.txt # Create a new empty file or update timestamp
```

ln: Create symbolic or hard links.

```
ln -s /path/to/file linkname # Create a symbolic link
ln /path/to/file linkname # Create a hard link
```

Pipes - redirect output

```
ls -l | grep ".txt" # List all .txt files in long format
cat file.txt | tr '[:lower:]' '[:upper:]' # Convert content of file.txt to uppercase
```

>: Redirect output to a file (overwrite).

```
echo "Hello" > file.txt # Write "Hello" to file.txt, overwriting it
```

>>: Redirect output to a file (append).

```
echo "World" >> file.txt # Append "World" to file.txt
```

2>: Redirect error output.

```
command 2> error.log # Redirect errors to error.log
```

You can use xargs to pass each line of the file as arguments to another command:

```
cat items.txt | xargs echo
```

&1: This means "redirect to wherever file descriptor 1 (stdout) is currently pointing."

```
ls existingfile.txt > output.txt 2>&1
```

View the first/last N lines

```
head -n 10 filename.txt # First 10 lines
tail -n 10 filename.txt # Last 10 lines
```

view file with pagination (viewing a file with pagination means displaying its content one screen at a time)

```
less filename.txt
```

2) Binaries

Common Bin Paths

/bin ->Contains essential user binaries required for basic system functionality (ls, vp, mv)

/sbin → Houses system binaries that are primarily used for system administration.

/usr/bin → administrative commands that are not required for basic system operation (useradd, cron)

/usr/local/bin → A place for locally installed binaries, typically used for software compiled from source or installed manually

/usr/local/sbin → Similar to /usr/local/bin, but specifically for system administration binaries that are locally installed.

PATH variable → The PATH variable is a colon-separated list of directories that the shell searches when a command is entered.

view path:

```
echo $PATH
>
/home/laptopdev/.local/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin:/opt/java/jdk-17.0.10/bin:/snap/bin
```

Add a directory to PATH (temporary):

```
export PATH=$PATH:/new/directory
```

to add permanently modify ~/.bashrc or ~/.bash_profile.

bash exporting variables

The source command is used to run a script in the current shell, allowing variables and settings from the script to persist in the shell after the script completes.

```
source script.sh
```

The export command is used to make a variable available to subprocesses of the current shell. It does not affect existing child processes or other unrelated processes.

```
export my_var
```

Create variable:

```
my_var="Hello"
```

print variable:

```
echo $my_script_var
```

Linux Navigation: Cron, nohup, services

throug cron you can define when specific script need to be run

```
* * * * * command-to-run
| | | | |
| | | | |
| | | | ----- Day of the week (0 - 7) (Sunday=0 or 7)
| | | ------- Month (1 - 12)
| | --------- Day of the month (1 - 31)
| ----------- Hour (0 - 23)
------------- Minute (0 - 59)
```

and this how it looks like:

```
0 3 * * * /path/to/your/script.sh
```

View cron jobs for the current user & edit cron jobs

```
crontab -l
crontab -e
```

Nohup (No Hang Up) is used to run a command that should continue running even after the user logs out of the terminal session

```
nohup /path/to/your/script.sh &
```

Create a service and manage it to run after startup:

create service in /etc/systemd/system/my-script.service

```
[Unit]
Description=Run my custom script at startup
After=network.target

[Service]
ExecStart=/path/to/your/script.sh
Restart=always

[Install]
WantedBy=multi-user.target
```

then enable:

```
sudo systemctl enable my-script.service
```

Another method to run scripts at boot is using cron's special @reboot directive. This allows a script to execute once, right after the system starts.

```
@reboot /path/to/your/script.sh
```

/etc/rc.local → /etc/rc.local file to run commands at system startup

Edit /etc/rc.local and add the path to the script before the exit 0 line.

```
#!/bin/bash
/path/to/your/script.sh
exit 0
```

Ensure the script has execute permissions

```
chmod +x /etc/rc.local
```

.bashrc is a configuration file for the Bash shell (or any compatible shell). It is specific to user sessions and is executed every time a new interactive shell is started.

# Set the PATH in .bashrc

```
export PATH=$PATH:/custom/path
```

Linux Navigation: Repositories managment

In Linux, software packages are managed using package managers, with tools varying by distribution, such as APT for Debian-based systems and YUM or DNF for Red Hat-based systems

Software packages in Linux are stored in repositories (repos), which are servers hosting packages and metadata, and package managers fetch these from repos during installation or updates; on Debian-based systems like Ubuntu, repositories are configured in specific files.

The main repository sources for Debian-based systems are listed in **/etc/apt/sources.list**, while additional repositories can be managed via separate .list files in **/etc/apt/sources.list.d/**, allowing easy organization without altering the main file.

On Debian-based systems, packages use the .deb format, managed mainly by **APT** (for installing, upgrading, and dependency handling) and **dpkg** (for direct package file operations)

Key APT commands include:

```
apt update: Update the package list from repositories
apt upgrade: Upgrade all installed packages to their latest versions
apt install <package_name>: Install a package
apt remove <package_name>: Remove a package
apt search <keyword>: Search for a package
```

Key dpkg commands:

```
dpkg -i <package.deb>: Install a .deb package
dpkg -r <package_name>: Remove a package
dpkg -l: List all installed packages
```

To add a new repository on Ubuntu, you can edit **/etc/apt/sources.list** directly or add a **.list** file in **/etc/apt/sources.list.d/.** For example, to add a PPA (Personal Package Archive):

```
sudo add-apt-repository ppa:some/ppa
sudo apt update
```

common troubleshooting

1) Command Not Found
If a command isn't recognized, it usually means its executable path isn't included in the PATH variable.

2) Repository Not Found
This usually means the repository URL is incorrect or no longer available.

3) Cron Job Issues

If cron jobs are not running, it could be due to syntax errors, permissions issues, or environmental variable problems.

4) **Just learn more about bash**

5) **Output File Issues (nohup)**

By default, nohup redirects output to a file called nohup.out. If this file is not being created or is empty, it could indicate a problem with how the command is executed.

Learn more about Bash & Linux

Check TecMint - great docs!

Understand Linux Shell and Basic Shell Scripting

The Linux shell, or command-line interface, is a powerful program

https://www.tecmint.com/understand-linux-shell-and-basic-shell-scripting-language-tips/

Linux File System Explained

The concept of boot loading

https://www.tecmint.com/linux-file-system-explained/

Share, comment, DM and check GitHub for scripts & playbooks created to automate process.

Check my GitHub

Michael Robotics

Hi, I'm Michal. I'm a Robotics Engineer and DevOps enthusiast. My mission is to create skill-learning platform that combats skill information overload by adhering to the set of principles: simplify, prioritize, and execute.

https://github.com/MichaelRobotics

PS.

If you need a playbook or bash script to manage KVM on a specific Linux distribution, feel free to ask me in the comments or send a direct message!