10 Comandos Linux que talvez nunca ouviu falar

Linux, a powerhouse of flexibility and control, often reveals its true magic through the command line. While `ls`, `grep`, and `cd` are universally known, the operating system houses a constellation of lesser-known utilities—each with unique capabilities. These obscure commands, once discovered, can enhance your workflows, increase productivity, and turn you into a command-line artisan.

1. `look` — Fast Dictionary Lookup

The `look` command performs a binary search on a sorted file, typically a dictionary, and prints all lines that begin with a given string. It's perfect for autocomplete tools, word games, or when verifying the existence of a term.

```
look pro
```

This will return all dictionary entries beginning with “pro.” Fast, lean, and astonishingly handy.

2. `rev` — Reverse the Characters of a Line

A surprisingly effective tool, `rev` reverses each line of input character by character. It might sound like a novelty, but it’s invaluable in scenarios involving cryptic text transformations or palindromic algorithms.

```
echo "Linux" | rev
```

This returns “xuniL”. Simple, elegant, and precise.

3. `tac` — The Reverse of `cat`

While `cat` displays file content from top to bottom, `tac` (cat spelled backward) prints lines in reverse order. For tail-heavy logs, or when parsing data from the bottom up, `tac` can be a lifesaver.

```
tac access.log
```

This allows you to read logs in a reverse chronological sequence without needing `tail -r`.

4. `yes` — Repetitive Stream Generator

The `yes` command outputs a string repeatedly until interrupted. When automating scripts or testing buffer behavior, this tool shines.

```
yes | sudo apt install mypackage
```

This command auto-confirms every prompt, useful in scripted installs.

5. `nl` — Number Lines of Files

A more sophisticated cousin of `cat -n`, the `nl` command adds line numbers with robust formatting control.

```
nl file.txt
```

With support for logical page delimiters and line numbering styles, `nl` is ideal for structured file documentation.

6. `column` — Format Output into Columns

`column` transforms textual data into well-aligned columns, making output dramatically more readable—especially when viewing CSVs or tabular data.

```
cat data.txt | column -t -s,
```

This aligns comma-separated data neatly into tabular format.

7. `shuf` — Shuffle Lines Randomly

Need to randomize a playlist or test against unpredictable data? `shuf` randomizes input line order effortlessly.

```
shuf list.txt
```

It’s also useful in shell-based gaming, simulation, and statistical sampling.

8. `comm` — Compare Two Sorted Files Line by Line

`comm` is an unsung hero in file comparison. It contrasts two sorted files line-by-line and categorizes them: lines unique to file1, file2, and lines common to both.

```
comm file1.txt file2.txt
```

Ideal for syncing datasets or identifying deltas.

9. `chrt` — Manipulate Real-Time Scheduling Policies

For those tinkering with performance tuning, `chrt` adjusts a process’s real-time scheduling policy. Combined with `ps` or `top`, it’s a potent performance tool.

```
sudo chrt -f 99 ./my_program
```

This elevates your process to the highest fixed-priority level.

10. `watch` — Execute a Program Periodically

Observe command output in near real-time with `watch`. It’s perfect for monitoring resource usage, service health, or file changes.

```
watch -n 2 df -h
```

This runs `df -h` every 2 seconds, refreshing the terminal view dynamically.

Mastering Linux means going beyond the commonplace. These ten underutilized commands unlock new layers of potential, allowing developers, admins, and enthusiasts to operate with greater fluency and finesse. With just a bit of curiosity, even the most obscure tool can become an indispensable ally in your command-line journey.

Revision #2
Created 2025-08-31 12:56:14 UTC by Administrador
Updated 2025-08-31 12:58:18 UTC by Administrador