100 Unusual & Obscure Linux Commands

A guided tour through the historical nooks and crannies of Unix and Linux

Odd facts, history, and the origins behind commands most people never type.

100 commands · 10 categories · Compiled July 2026 · SplitZrate.com

---

Contents

1. The Forgotten Corners of Text Processing 2
2. ASCII Art & Terminal Jokes 3
3. Odd Calculators & Mathematical Curiosities 4
4. Communication From a Bygone Era 5
5. Calendars, Time & Waiting 5
6. Archives, Encoding & Binary Archaeology 6
7. System Archaeology — Tools Older Than You Think 7
8. Etymological Deep Dives 7
9. Games & Hidden Easter Eggs 8
10. Odd Networking & System Diagnostics 9

1. The Forgotten Corners of Text Processing

1.1 rev — Reverse every character in a line One of the most minimalist Unix tools around — its entire job is to mirror each line, character by character. Originally a BSD tool that's shipped with Unix since the early 1980s, often used in joke one-liners and for quickly checking palindrome strings. Example: `echo "Linux" | rev` gives you `xuniL`.

1.2 tac — cat spelled backward (literally) The name is simply "cat" spelled in reverse, and that's exactly what it does: print a file with its lines in reverse order. One of the clearest examples of Unix developers' fondness for wordplay in command names. Example: `tac logfile.txt` shows the most recent line first.

1.3 shuf — Shuffle the order of lines Added to GNU coreutils in version 6.0 (August 2006) as an answer to people otherwise cobbling together random line ordering with `sort -R` or Perl one-liners. Widely used in scripts to, say, draw a random winner from a list. Example: `shuf -n 1 participants.txt` — pick one random winner.

1.4 nl — Number lines with style Unlike `cat -n`, `nl` offers fine-grained control over numbering format, starting value, and which lines get counted — a legacy of the era when printouts on line printers needed precisely formatted page numbers.

1.5 pr — Format text for printing on paper A tool from a time when "printing" a file to a physical line printer was an everyday task. `pr` adds page headers, page breaks, and columns — almost nobody uses it for its original purpose anymore, but it still turns up in scripts that format reports.

1.6 paste — Merge lines side by side The opposite of cutting out columns: `paste` glues together corresponding lines from multiple files horizontally. The name is a direct metaphor for "pasting" columns together, in keeping with the Unix tradition of office-supply metaphors (cut, paste, and similar tools).

1.7 join — SQL joins before SQL was common `join` merges lines from two sorted files based on a shared field — in practice, a relational-database join performed on the command line decades before most developers had heard of SQL. Both files must be sorted on the key field for it to work.

1.8 comm — Compare two sorted files column by column Shows three columns: lines unique to file 1, lines unique to file 2, and lines common to both. A cousin of `diff`, but optimized for set comparisons rather than showing text changes.

1.9 expand / unexpand — Convert between tabs and spaces A throwback to a time when different terminals and printers handled tab stops completely differently, which made text files look different depending on where they were opened. Still relevant today for code formatting and whitespace fights in pull requests.

1.10 column — Neat columns out of raw text Takes unstructured, delimited text and lines it up into even columns — of BSD origin, often forgotten in favor of `printf` or `awk`, even though it does exactly one thing very well with no extra code.

1.11 tsort — Topological sorting, invented for the linker Originally created to solve a very specific problem: figuring out the order in which a set of `.o` object files needed to be handed to the Unix linker (`ld`) so that all symbol dependencies would resolve correctly. Nowadays it's occasionally used to sort dependency graphs in build systems or package tools.

1.12 ptx — Permuted index (KWIC) Generates a "Keyword In Context" index, where every word in a text is listed alongside its surrounding context, sorted alphabetically by keyword. A tool that once mattered for building searchable indexes for printed documents, long before full-text search existed.

1.13 look — Binary-search a sorted word list by prefix Quickly looks up words in a sorted word list (`/usr/share/dict/words` on many systems) that start with a given prefix — a precursor to autocomplete, originally meant for quickly checking spelling. Because the list is already sorted, `look` can binary-search it directly instead of reading through the whole file the way `grep` would

have to — a good example of how an assumption-based, hyper-specialized tool can be significantly faster than a general-purpose one.

1.14 fold — Wrap text at a given width A simple tool with roots in an era when terminals and line printers had a fixed width, often 80 characters, and text needed to be wrapped manually so it wasn't cut off mid-line.

1.15 fmt — Simple text wrapping and paragraph justification Wraps running text to a given line width, a cousin of how classic email clients formatted quoted text. Still handy for quickly normalizing line lengths in a text file from the command line.

1.16 colcrt — Format nroff output for terminals without overstrike A tool built for a time when many terminals lacked the ability to "overstrike" a character to produce underlining or bold — `colcrt` converted that kind of formatting into something readable on simpler screens instead of just showing garbled control characters.

1.17 ul — Simulate underlining on terminals that don't support it A small sibling tool to `colcrt`, specialized in interpreting underline sequences from nroff output and displaying them as best it can, even on terminals with no formatting support at all.

---

2. ASCII Art & Terminal Jokes

2.1 cowsay — A cow that says whatever you want Generates ASCII-art cow art with a speech bubble around any text you give it. It quickly became part of hacker culture after its release and has lived on for decades — named Debian's "package of the day" in 2007. The classic combo: `fortune | cowsay`.

2.2 cowthink — The cow thinks instead of speaking The same ASCII cow as `cowsay`, but with a thought bubble instead of a speech bubble — a small but typical example of the Unix philosophy of turning a minor variation into its own separate tool instead of a flag.

2.3 sl — A train that punishes you for a typo Created by Toyoda Masashi in 1993 as a joke aimed at anyone who accidentally types "sl" instead of "ls". Instead of a directory listing, you get a full-scale ASCII steam locomotive chugging across the terminal. The project's own description puts it plainly: "It's just a joke command, and not useful at all."

2.4 figlet — Big ASCII-art letters Turns text into stylized letters built from ASCII characters, with support for dozens of different "fonts" stored as text files. The name stands for "Frank, Ian and Glenn's LETters," after its three creators.

2.5 toilet — figlet's colorful cousin Similar to `figlet` but with support for colors, gradients, and more output formats (including HTML and ANSI color codes) — often used in terminal banners for install scripts and CTF challenges.

2.6 banner — Enormous letters for line printers One of the oldest ASCII text tools, with roots going back to early Unix versions where printouts on wide continuous-feed paper could produce enormous letters spanning several pages — a physical precursor to today's digital banners.

2.7 lolcat — Rainbow-colored terminal output Colors text in a scrolling rainbow pattern, named after and inspired by the "lolcat" internet meme of the 2000s. Classically combined as `fortune | cowsay | lolcat` for maximum terminal joy.

2.8 asciiquarium — An aquarium in your terminal Animates fish, sharks, and a sinking Titanic in ASCII art right inside your terminal window. It's a Perl script built on the `Term::Animation` CPAN module — not, as sometimes assumed, on the aalib library. One of the more elaborate examples of just how far Unix joke culture can be taken for its own sake.

2.9 cmatrix — "The Matrix" rain in your terminal Simulates the falling green character rain from the 1999 film The Matrix directly inside a terminal window. One of the most widely shared screenshot tools among Linux users showing off their terminal setup.

2.10 hollywood — Turn your terminal into a hacker movie Fills the screen with multiple panes of rapidly scrolling code, hex dumps, and status fields at once — designed specifically to look like a Hollywood movie's (unrealistic) idea of "hacking."

2.11 nyancat — The rainbow cat, terminal edition A terminal port of 2011's viral Nyan Cat animation, with the same pixel cat and rainbow tail, fully recreated in ANSI colors and ASCII characters.

2.12 oneko — A cat chases your mouse cursor A small windowed program (requires X11) where an animated cat, or another animal, chases your mouse cursor around the screen — originally created in Japan in the 1990s and since ported to countless platforms.

2.13 xeyes — Eyes that follow your cursor A classic X11 demo program where a pair of cartoon eyes visually track your mouse cursor's movement across the screen. It has shipped with X Window System distributions since the 1980s as a simple test that X11 actually works.

2.14 aafire — Fire made of ASCII characters A genuine aalib demo program, rendering an animated campfire entirely in text-based ASCII art — originally built to show off what the aalib library could do. (Unlike `aafire`, `asciiquarium` above isn't actually part of the aalib family, despite the resemblance.)

2.15 genius — A tongue-in-cheek terminal "intelligence test" One of several small BSD-heritage joke programs still living on in the `bsdgames` package, often unknown even to experienced Linux users since it's rarely installed by default anymore.

---

3. Odd Calculators & Mathematical Curiosities

3.1 dc — The oldest surviving Unix application Written by Lorinda Cherry and Robert Morris at Bell Labs, and according to Ken Thompson, possibly the very first program run on Bell Labs' new PDP-11 — before even an assembler was available. `dc` uses reverse Polish notation (RPN) and was written in the language B, before C even existed. Example: `echo "2 3 + p" | dc` prints `5`.

3.2 bc — The calculator once built on top of dc `bc` ("basic calculator") provided a more readable infix language for math and traditionally compiled its expressions down to `dc` instructions internally. Modern GNU implementations have flipped that relationship, but the names and the lineage live on side by side.

3.3 units — Convert units with a Bell Labs pedigree A tool for converting between essentially any unit of measurement you can think of — from light-years to hectares, or US gallons to liters. It traces back to Bell Labs and contains a database of thousands of units, including quite a few joke units for the curious.

3.4 factor — Prime-factorize any number A surprisingly deep tool hiding inside GNU coreutils: feed it an integer and get back its prime factors, with support for arbitrarily large numbers thanks to the GMP library under the hood. Example: `factor 91` gives `91: 7 13`.

3.5 expr — Arithmetic and string operations in plain POSIX shell Before `$(( ))` became standard in shell languages, `expr` was the only way to do basic arithmetic in a script. It still lives on in older scripts and POSIX sh-compatible environments.

3.6 seq — Generate sequences of numbers A small but heavily used tool for printing a sequence of numbers with any start value, end value, and step size — surprisingly often reinvented by hand by developers who didn't know it already existed built in.

3.7 numfmt — Human-readable numbers, back and forth Converts between raw digits and readable formats like "1.5K" or "2.3G" — a relatively modern addition to GNU coreutils that lets you write scripts that present numbers just as neatly as `df -h` does.

3.8 primes — Crunch out prime numbers at lightning speed with old algorithms A small tool from the BSD games collection that prints every prime number in a given range, historically optimized to be as fast as possible despite decades-old code — a curious mathematical companion to `factor` that does the opposite job: finding primes instead of factoring them.

---

4. Communication From a Bygone Era

4.1 wall — Write a message to every logged-in user Short for "write all." Sends a message to the terminal of every user logged into the same system — a precursor to group chat, from a time when several people often shared a single Unix machine over terminals.

4.2 write — Chat directly with another logged-in user Opens a one-way text channel to a specific logged-in user's terminal, line by line — in practice, one of the earliest forms of instant messaging on Unix systems, long before ICQ or IRC.

4.3 talk / ytalk — Real-time chat with a split screen An improvement on `write` that split the terminal screen in two so both parties could see each other's text appear in real time, character by character. `ytalk` later added support for more than two participants at once.

4.4 mesg — Control whether others can write to your terminal A small but important companion to `write` and `talk`: `mesg n` blocks unsolicited messages from popping up in the middle of whatever you were doing, while `mesg y` allows them.

4.5 finger — Look up information about a user Displayed information about logged-in users: full name, last login, and sometimes a free-text "plan" file they'd written themselves. It became infamous in 1988 when a vulnerability in the `fingerd` daemon was one of the ways the Morris worm spread — one of the first major internet security incidents ever.

---

5. Calendars, Time & Waiting

5.1 cal — The terminal's simplest calendar Prints a text-based calendar for a given month or year right in the terminal — one of the most undramatic yet indispensable tools for a quick date check without leaving the command line.

5.2 ncal — The calendar lying on its side A variant of `cal` that displays the month sideways instead of upright, and can also show Julian dates and moon phases depending on the flags — a BSD alternative many people never discover exists alongside `cal`.

5.3 ddate — What day is it according to Discordianism? Converts today's date to the satirical Discordian calendar, invented by Kerry Wendell Thornley and Greg Hill in the 1960s as part of the parody religion Discordianism. One of the clearest examples of Unix developers happily weaving their own absurdist humor into serious toolkits. Example: `ddate` prints today's date in the Discordian calendar, e.g. "Pungenday, Discord 15, YOLD 3192."

5.4 pom — What moon phase is it right now? Short for "phase of moon." Calculates and describes the moon's current phase directly in the terminal — an odd but fully serious little astronomy tool that shipped with the BSD games collection, meant more as trivia than practical navigation.

5.5 leave — Remind yourself to go home A nearly forgotten BSD tool that keeps track of the time and beeps a warning shortly before (and exactly when) a specified end time arrives — a digital precursor to today's calendar reminders, just confined to the terminal.

5.6 calendar — Personal reminders before Google Calendar existed Reads a simple text file of dates and events and shows the ones relevant today or in the near future — an entire reminder system built on nothing more than a text file and a small script, decades before cloud calendars existed.

5.7 at / batch — Schedule a one-off job Unlike `cron`, which runs recurring jobs, `at` runs a command exactly once at a specific time you specify in plain language (like `at 17:00` or `at now + 1 hour`). `batch` does the same but waits until system load is low enough.

5.8 sleep — The simplest possible delay, with an odd history `sleep` itself is hardly mysterious, but its syntax for time units (`sleep 1h`, `sleep 30m`) is a relatively late addition to GNU coreutils — older scripts had to convert everything to seconds by hand.

---

6. Archives, Encoding & Binary Archaeology

6.1 uuencode / uudecode — Send binary files over plain-text email Invented for Unix-to-Unix Copy (UUCP) in the 1980s to send binary files through email systems that could only handle seven-bit text. A precursor to today's base64 encoding, and one of the reasons old Usenet posts sometimes contain cryptic text blocks for attached images.

6.2 cpio — The archive format still living inside your initramfs The name stands for "copy in/out." Even though `tar` is better known today, it's actually the `cpio` format that's still used to pack the Linux initramfs image loaded at every system boot.

6.3 ar — The archiver behind every static library Used to create and manipulate `.a` archive files (static libraries) that the linker merges with your code at compile time — a tool most developers never call directly, but which runs in the background every time a C or C++ project is built.

6.4 od — Octal (or hexadecimal) dump of raw data One of the very oldest ways to inspect binary data in Unix, long before `xxd` or `hexdump` existed. The name stands for "octal dump," even though these days it can just as easily show hexadecimal, decimal, or ASCII.

6.5 xxd — A hex dump that can run in reverse Not only can it show a file's binary contents as hexadecimal numbers, it can also do the opposite: rebuild a binary file from a text-based hex dump — originally written as a companion to the Vim editor.

6.6 strings — Find readable text hidden inside binary files Scans a binary file for sequences of printable characters — a classic way to quickly find error messages, URLs, or version strings hidden inside compiled programs without needing a full disassembler.

6.7 cksum / sum — Checksums before md5sum was standard Computes a simple checksum (CRC) to detect accidental data corruption. Considerably older and weaker than modern hash functions like SHA-256, but still useful for quickly checking whether two files differ.

6.8 bcd — Format text as punch cards or Morse code A purely nostalgic tool that turns text into a visual representation of old IBM punch cards (or, alternatively, Morse code and paper tape) right in the terminal — with zero practical use today, but a fascinating window into a time when punch cards really were the primary way to feed programs into a computer.

6.9 morse — Translate text to Morse code and back Converts plain text into Morse code's dots and dashes (or the reverse) right on the command line — part of the same BSD tool family as `bcd`, with roots in an era when amateur radio and telegraphy were still everyday technology for many computer users.

6.10 split — Split a large file into smaller pieces Developed for a time when storage media (floppy disks, tapes) had limited capacity and large files needed to be broken up to fit — used today mostly for handling files that are too large for email attachments or certain filesystems.

---

7. System Archaeology — Tools Older Than You Think

7.1 ed — Unix's original text editor Written by Ken Thompson in the very first version of Unix (1969–1971), long before screen-based editors like vi or emacs existed — `ed` edits text line by line without ever showing a full screen. Its man page has long carried the half-ironic, half-serious motto that "ed is the standard editor."

7.2 m4 — The macro processor behind sendmail and autoconf A general-purpose text macro processor that's been the hidden engine behind generating `sendmail.cf` configuration files and GNU Autoconf's `configure` scripts for decades — most developers never encounter `m4` directly, but have very likely run code it generated.

7.3 troff / nroff — Typesetting before HTML and LaTeX Developed at Bell Labs to format text for printing, and used to typeset the entire Unix manual page collection for decades — man pages are technically still written in a troff-like macro language under the hood, even if you rarely see it directly.

7.4 yes — One of the simplest programs to ever become legendary Written by Ken Thompson for Version 7 Unix in 1979 to automatically answer "y" to programs asking for confirmation. Even though the original was a single line of code, modern implementations (GNU's especially) have become famous for extremely aggressive buffering optimization — GNU's `yes` can output text at over 800 MB/s, pure I/O buffering in its rawest form.

7.5 true / false — The two smallest programs you'll ever run They literally do nothing except return exit code 0 or 1 respectively — but since the entire shell-scripting ecosystem is built on exit codes, these "empty" programs are actually critical building blocks in countless conditionals and infinite loops (`while true; do ...; done`).

7.6 tee — Named after a pipe fitting, not a T-shirt The name comes from the T-fitting used in plumbing, even though `tee` actually duplicates the data stream to multiple destinations rather than splitting it apart — a small naming quirk that's confused new Unix users for decades. It was already described in the classic book "The Unix Programming Environment."

7.7 nohup — Literally "no hang up" The name refers directly to the phone line's "hangup" signal (SIGHUP), which old dial-up terminal connections sent when the line dropped — `nohup` protects a process from being killed when that exact signal occurs, a legacy of the modem era that's still used daily to keep processes alive after logout.

7.8 nice — A priority scale from -20 to 19, purely out of history The odd bounds on process priority (instead of, say, 0–100) are a direct legacy of how the scheduler on early PDP-11 Unix systems counted "niceness" internally — the numbers have carried through unchanged across fifty years of Unix development.

---

8. Etymological Deep Dives

8.1 grep — The command named after an ed editor command The name comes from the command sequence `g/re/p` in the `ed` editor — "global, regular expression, print." Ken Thompson wrote the first version in a single hour in 1973 after colleague Doug McIlroy asked for a standalone search tool, built on a private utility Thompson already used called "s." The word "grep" is now officially in the Oxford English Dictionary, as both a noun and a verb.

8.2 tar — The legacy of magnetic tape Stands for "tape archive." The format was originally designed specifically to stream data sequentially to and from magnetic tape — which is why classic tar archives still lack a real index and must be read through sequentially, a direct consequence of 1970s tape technology's limitations.

8.3 su — Superuser, or substitute user? Early Unix documentation described `su` as a way to "become the superuser," while later manuals define it as running a command with a "substitute"

user ID. Both interpretations are historically correct depending on which era of Unix documentation you're reading — the command itself already existed in Unix version 1.

8.4 sudo — Born at a university, not at Bell Labs Unlike many core tools, `sudo` wasn't created by AT&T but by Bob Coggeshall and Cliff Spencer at SUNY Buffalo around 1980, on a VAX-11/750. The name combines "su" with "do," and the tool spread via Usenet in the mid-1980s before Todd C. Miller took over maintenance and turned it into the standard it is today.

8.5 awk — Three surnames in one command name The name is simply the initials of its three creators: Alfred Aho, Peter Weinberger, and Brian Kernighan, all of Bell Labs. Unusually transparent naming in a world of cryptic two-letter commands.

8.6 sed — The editor born from ed's batch mode The name stands for "stream editor." `sed` grew out of wanting to run `ed`-like editing commands on streaming text without interactivity — in practice, `ed`'s search-and-replace power detached from its line-by-line interactive workflow.

8.7 vi — The visual part of an older editor `vi` (pronounced "vee-eye") stands for "visual" and was originally just a visual command mode added on top of the `ex` editor, which in turn built on `ed`. What feels today like a completely standalone editor started out as just one flag away from `ed`.

8.8 emacs — The editor named after its own macros The name stands for "Editor MACroS," after the original TECO macros ("Editing MACroS") that Richard Stallman and others built to create a more extensible editing system in the 1970s — the foundation of the entire extensible philosophy Emacs still runs on today.

---

9. Games & Hidden Easter Eggs

9.1 bsdgames — An entire text-based game collection hiding in the package repository A package of classic text-based games (including `adventure`, `rogue`, `worms`, and `robots`) that once shipped by default with many Unix and BSD installations, but these days usually has to be installed manually — a fossil from an era when the terminal doubled as a game console.

9.2 wtf — The terminal's own acronym dictionary Short for exactly what you're thinking: type `wtf is NASA` and the command looks up and explains the acronym for you. A genuinely useful little reference tool hiding behind a joke of a name, pulled from the same BSD games collection as `rogue` and `worms`.

9.3 rain — Terminal rain that predates screensavers Animates concentric "raindrops" spreading across the entire terminal screen in an endless loop — designed as an early form of screensaver for terminals, long before graphical screensavers with flying toasters became commonplace.

9.4 caesar — Automatically crack Caesar ciphers with frequency analysis A tool that guesses the correct shift in a Caesar cipher by analyzing letter frequencies in the encrypted text and comparing them to typical English text — a small, surprisingly serious cryptography tool hiding in a collection of terminal jokes.

9.5 rogue — The game that spawned an entire genre One of the earliest "dungeon crawler" games with randomly generated levels, developed on early Unix systems in the early 1980s. Its influence was so large that an entire subgenre of games is now called "roguelikes" after it.

9.6 worms — Worms eating up your terminal screen A small terminal-based animation/screensaver where digital "worms" wriggle across the screen and slowly fill it up — more demonstration than game, an early example of generative terminal animation.

9.7 apt-get moo — A cow-secret easter egg in Debian's package manager Type `apt-get moo` on a Debian- or Ubuntu-based system and you get an ASCII cow saying "Have you mooed

today?" — one of the best-known easter eggs in the entire Linux ecosystem, hidden inside a tool millions of administrators use daily without ever finding it.

9.8 aptitude moo — More levels of the same joke The older package management tool `aptitude` has its own `moo` command, and if you run it repeatedly with more `-v` flags (`aptitude moo -vvvvv`), the joke escalates through several increasingly absurd stages — a joke building on another joke.

9.9 sudo apt-get moo as "root privileges required for cows" An everyday classic among sysadmins: combining `sudo` with `apt-get moo` to confirm that even cow humor requires root privileges — completely pointless technically, but a fixture of Linux folklore.

9.10 emacs -f doctor — A terminal psychotherapist Emacs's built-in `doctor` mode simulates a Rogerian psychotherapist in the style of the classic 1966 ELIZA program — type out your problems and the "doctor" reflects the question back at you, one of the oldest examples of a chatbot joke embedded in serious software.

9.11 emacs -f tetris — A fully playable Tetris inside your text editor Emacs has long shipped with a handful of built-in minigames, including a working Tetris clone, part of its philosophy of being "an operating system that happens to lack a decent text editor" rather than just an editor.

9.12 yes | rm -i — A joke about bypassing safety prompts A classic (if somewhat dangerous) example of how `yes` gets abused to automatically answer "y" to every confirmation prompt `rm -i` throws up — often used as a cautionary example in tutorials about why you should be careful piping `yes` into destructive commands.

9.13 :(){ :|:& };: — The fork bomb that became a cultural phenomenon An infamous bash function that makes a process duplicate itself in an endless, exponentially growing chain until the system crashes from resource exhaustion. Rather than a useful command, it's become a kind of running joke and cautionary demonstration in security courses about why process limits (`ulimit`) exist — never try this on a system you care about.

---

10. Odd Networking & System Diagnostics

10.1 xargs — Invented to work around an old argument limit Created to solve a concrete historical problem: early Unix shells had a hard limit on how many arguments a command could receive at once. `xargs` solves that by automatically splitting input into appropriately sized command invocations — a limit that's long since been raised much higher, but the tool lives on as a powerful way to build commands dynamically.

10.2 watch — Repeat a command and see the difference Reruns a command at regular intervals and can even highlight what's changed between runs — a surprisingly simple tool that replaces a lot of homegrown `while true; do ...; sleep 1; done` loops.

10.3 script / scriptreplay — Record and replay an entire terminal session `script` records everything that happens in a terminal session to a log file, while `scriptreplay` can later play back that exact session in real time, including the original's exact timing between keystrokes — a kind of screen recording from an era long before video formats made it unnecessary for most purposes.

10.4 lsof — List literally every open file The name stands for "list open files," and in true Unix spirit, network sockets and devices also count as "files" — one of the most powerful troubleshooting tools for seeing exactly which process is holding a file, port, or device busy.

10.5 arping — Ping without needing IP to work yet Sends ARP requests instead of ICMP echo requests, so it can tell whether a device is present on the local network segment even before it has an IP address configured, or when ICMP is blocked entirely — a diagnostic tool operating one layer below where ordinary `ping` works.

10.6 yes > /dev/null — An innocent CPU stress test Piping `yes` to `/dev/null` (`yes > /dev/null`) is a surprisingly common, minimal way to load a single CPU core to 100% for quick, harmless performance tests — nothing needs to be installed, the command already exists on essentially every Unix system.

10.7 tput bel — Make the terminal beep via the terminfo database `tput` queries the system's terminfo database to send the correct control sequence for a given terminal type — `tput bel` is the most harmless example, simply making the terminal emit its classic beep sound, a digital remnant from when terminals had an actual physical bell built in.

10.8 factor $(date +%s) — Factorize the current Unix time for fun No practical use, but a popular way to demonstrate `factor`: feed in the current number of seconds since January 1, 1970 and see its prime factors — a small, harmless example of how Unix tools love being combined in unexpected ways.

© SplitZrate 2026 · 100 Unusual & Obscure Linux Commands
