jc CLI output to JSON

START HERE

`jc <command> [args]`
Run a command, get JSON (magic syntax)

`jc -h`
List all parsers

`<command> | jc --<parser>`
Convert piped output

`jc -h --<parser>`
Docs and schema of one parser

`jc -p <command>`
Pretty-print the JSON

`jc <command> | jq '.[]'`
Hand off to jq

`jc -y <command>`
Output YAML instead

`jc -r <command>`
Raw strings, no type conversion

CONCEPTS

jc only converts text to JSON; it does not filter. Pair it with jq. Most parsers return an array of objects, but some (like uptime) return one object, so check the schema with jc -h --<parser>.

Command output ps, dig, ls, ...
↓ text
jc --<parser> or magic
↓ JSON
jq / script filter, select, format
jc sits between text tools and JSON tools

RUNNING JC

`jc <command> [args]`
Magic syntax: run and parse in one step

`<command> | jc --<parser>`
Standard syntax: parse piped text

`cat <file> | jc --<parser>`
Parse a saved file or capture

`jc /proc/<path>`
Parse a /proc file

`jc <start>:<end> --<parser>`
Parse only a slice of input lines
Zero-based, end excluded, negatives count from the end. Use it to skip headers or footers.

`jc -s --<parser>`
Slurp many one-line inputs into an array

`jc dig <host> | jq -r '.[].answer[].data'`
Magic syntax feeding jq

`ps aux | jc --ps`
Same, with the standard syntax

OUTPUT OPTIONS

`jc -p <command>`
Pretty-print JSON

`jc -y <command>`
YAML output

`jc -r <command>`
Raw output: strings, no int or date conversion

`jc -M <command>`
Add _jc_meta: parser, timestamp, exit code

`jc -m <command>`
Monochrome output

`jc -C <command> | less -R`
Force colour in a pipe

OUTPUT OPTIONS (cont.)

`jc -q <command>`
Suppress warnings

`jc -d <command>`
Debug a parse failure (-dd verbose)

`jc -u --<parser>`
Unbuffer output for live streams

`JC_COLORS=<k>,<kw>,<n>,<s> jc -p ...`
Custom colours: key, keyword, number, string

COMMON PARSERS

`jc ps aux`
Processes (--ps)

`jc ls -l <dir>`
Directory listing (--ls)

`jc df -h`
Disk usage per filesystem (--df)

`jc dig <host>`
DNS answers and timings (--dig)

`jc ss -tulpn`
Sockets (--ss)

`jc ip route`
Routing table (--ip-route)

`jc lsblk`
Block devices as a tree (--lsblk)

`jc systemctl list-units`
Units and their state (--systemctl)

`jc stat <file>`
File metadata (--stat)

`jc free`
Memory use (--free)

`jc uptime`
Load averages and uptime (--uptime)

`jc mount`
Mounted filesystems (--mount)

`jc env`
Environment variables (--env)

`jc netstat -an`
Connections (--netstat)

`jc lsof`
Open files (--lsof)

`jc ping -c 3 <host>`
Round-trip stats (--ping)

`jc find <dir>`
Paths from find (--find)

COMMON PARSERS (cont.)

`jc who`
Logged-in users (--who)

FILE AND STRING PARSERS

`jc --csv < <file>.csv`
CSV rows to objects

`jc --yaml < <file>.yml`
YAML file to JSON

`jc --toml < <file>.toml`
TOML file to JSON

`jc --ini < <file>.ini`
INI file to JSON

`jc --xml < <file>.xml`
XML file to JSON

`jc --kv < <file>`
key=value text to JSON

`jc --jwt <<< <token>`
Decode a JWT without verifying

`jc --x509-cert < <cert>.pem`
Inspect a certificate

`jc --asciitable < <table>.txt`
Any aligned text table to JSON
The fallback when no dedicated parser exists. Use -m variant for multi-line cells.

`jc --timestamp <<< <epoch>`
Epoch to date fields

`jc --url <<< <url>`
Split a URL into parts

`jc --ip-address <<< <ip>`
Analyse an IP address

`jc --semver <<< <version>`
Split a semantic version

`jc --syslog < <file>`
RFC 5424 syslog lines

`jc --git-log`
Parse git log output
Needs the default log format; git log --pretty=... is not supported.

STREAMING AND HELP

`<command> | jc --<parser>-s`
Streaming parser: one JSON line per input line
Names end in -s (ls-s, ping-s, vmstat-s). Cannot use magic syntax.

`ping <host> | jc --ping-s -u | jq .`
Live ping as it arrives

STREAMING AND HELP (cont.)

`jc -qq --<parser>-s`
Keep streaming past bad lines

`jc -h`
List parsers

`jc -hh`
Also list hidden parsers

`jc -hhh`
List parsers by category

`jc -h --<parser>`
Parser docs, schema and examples

`jc -a`
About jc and parser versions

`jc -v`
Show version

`jc -B | sudo tee /etc/bash_completion.d/jc`
Bash completion (-Z zsh, -X xonsh)

CUSTOM PARSERS

Plugin folder by platform

Platform	Path
`Linux`	~/.local/share/jc/jcparsers
`macOS`	~/Library/Application Support/jc/jcparsers
`Windows`	%LOCALAPPDATA% \jc\jc\jcparsers

A plugin is a Python file named like a module (letters, digits, underscore) with a parse() function. Copy jc/parsers/foo.py as the template; a plugin with a built-in's name overrides it.

`echo '<text>' | jc --<plugin>`
Run a custom parser like a built-in

RECIPES

Biggest memory users

```
jc ps aux \
  | jq -r 'sort_by(-.rss)[:5][]
| "\(.rss) \(.command)"'
```

Disks above 80% full

```
jc df | jq -r '.[]
  | select((."use%"|tonumber) >
80) | .mounted_on'
```

Listening ports and processes

```
jc ss -tlnp \
  | jq -r '.[] |
"\(.local_port)
\(.process_name)"'
```

Failed systemd units

```
jc systemctl --failed \
  | jq -r '.[].unit'
```