

Command-Line Help Text Rubric

This rubric defines principles and guidelines for writing clear, idiomatic, POSIX-aligned help text for command-line programs. It is intended to ensure consistency, readability, and usability across tools while maintaining alignment with Unix and GNU conventions. This rubric applies to program descriptions, positional arguments, option names (short and long), option help text, and epilog/usage text.

Style Definition

Help text should be written in the manpage-style imperative mood used in POSIX/Unix command documentation. This is distinct from prose or narrative writing. It prioritizes directness, concision, structured phrasing, and predictable wording.

Core Principles

- Infinitive Mood (No Subject):
 - Option descriptions must begin with a bare infinitive verb, without an explicit subject; this requirement applies primarily to options (positional arguments may follow argparse's default phrasing).
 - A useful test is whether the description reads correctly when preceded by "To ...".
- Prefer POSIX/Unix Conventions:
 - When possible, adopt terminology and phrasing that aligns with existing Unix tools (e.g., sort, grep, wc, nl, head, tail, uniq, cut). Prefer established vocabulary over invented terms unless a new concept is genuinely required.
- Manpage Economy (Minimal, Compressed Phrasing):
 - Help text should be concise, avoiding unnecessary articles ("a", "the", "an") and filler words.
 - Retain articles when their removal makes the phrase sound unnatural, ambiguous, or awkward.
 - Examples:
 - Preferred: sort lines using dictionary order
 - Avoid: sort the lines using the dictionary ordering method
 - Acceptable: prefix lines by the number of occurrences
 - Acceptable: show all lines, separating groups with an empty line
- Clarity vs. Brevity:
 - Prefer brevity in descriptions, but prefer clarity in option names.
 - When concision conflicts with clarity in help text, prefer clarity.
- Consistent Constraint Notation:
 - When documenting constraints or defaults, place them at the end of the description in parentheses using a consistent format: <action> (default: X; N >= Y)
 - When both defaults and constraints are present, list the default first.

- If argparse automatically appends defaults, avoid restating them manually unless necessary.
- Include Defaults Only When Meaningful:
 - Document defaults when they materially affect behavior or are non-obvious. Avoid documenting defaults that are trivial or self-evident.
- Make Option Interactions Explicit When Necessary:
 - If an option's behavior depends on or modifies another option, explicitly state this in the help text.
 - Example: use PATTERN to split lines into fields (affects --skip-fields)
- Prefer Established Unix Nouns:
 - Use standard Unix terminology whenever possible (file, line, field, pattern, count, width, format, key, separator). Avoid introducing novel terminology unless necessary.
- Keep Mutually Exclusive Options Parallel:
 - When options are mutually exclusive, their help text should be structured similarly for clarity and symmetry.
 - Example:
 - --foo use method FOO to process input
 - --bar use method BAR to process input
- Prefer Clarity Over Brevity in Option Naming:
 - Short option names are useful for frequently used flags, but clarity in long options takes precedence.

Program Description Guidelines

- The program description should ideally be a single, concise sentence.
- Prefer a verb phrase rather than "This program..."
- Avoid marketing language or unnecessary implementation details.
- Example: "Sort lines of text files."

Option Naming Conventions

- Short options should generally be a single letter (e.g., -n, -r, -k) unless there is a strong reason otherwise.
- Long options should use kebab-case (e.g., --max-width, not --max_width or --maxWidth).
- Avoid camelCase in command-line flags.
- Avoid dots in long options (--max.width)

Metavariables and Argument Names

- Prefer standard uppercase metavariables such as FILE, N, PATTERN, SEP, or FMT.
- Use uppercase metavariables consistently in option signatures.
- In tools like argparse, explicitly set metavariables when needed (e.g., metavar="FILE").
- Example: -n, --lines N print first N lines (N >= 1)

Boolean and Negated Options

- When a behavior is enabled by default, prefer providing a `--no-foo` form to disable it.
- Phrase help text primarily in terms of the affirmative behavior when possible.

Epilog and "See Also" Conventions (Optional)

- If an epilog is included, prefer concise references to related tools.
- Example: "See also: `sort(1)`, `uniq(1)`, `cut(1)`."
- Avoid verbose prose in the epilog.

Python and argparse Conventions

- When using Python's `argparse` module, follow `argparse`'s default formatting conventions to maintain consistency with standard Python CLI tools and with `argparse`-generated help output.
- Prefer lowercase for headings and labels as emitted by `argparse` (e.g., positional arguments:, options:), and avoid mixing in sentence case.
- For any new or revised `argparse`-based interface, verify:
 - Headings match `argparse`'s default lowercase style.
 - Option and positional argument descriptions follow the infinitive mood where feasible.
 - Defaults and constraints (if documented) appear in consistent parentheses.
 - Terminology aligns with POSIX/Unix conventions where appropriate.
 - Formatting has not been manually altered in a way that diverges from `argparse`'s standard layout; within that layout, continue to apply this rubric's concision and phrasing principles.

Help Text Review Checklist

- For any new or revised option, verify:
 - Infinitive mood: begins with a verb.
 - POSIX/Unix terminology: reads naturally in a manpage.
 - Concision: no filler words.
 - Constraints/defaults: clear, consistent parentheses.
 - Interactions: documented where relevant.
 - Standalone clarity: understandable without extra docs.
 - Parallel phrasing: especially for mutually exclusive flags.
 - Naming style: long options use kebab-case.
 - Articles: omitted when unnecessary, retained when needed for natural reading.