Comprehensive Guide to the `head` and `tail` Commands in Unix/Linux

`head` and `tail` are two of the most fundamental and frequently used Unix text-processing utilities. They are perfect companions: `head` **shows the beginning of a file/stream**, while `tail` **shows the end**. Together, they allow quick inspection of files without opening them fully — ideal for logs, large datasets, CSVs, or any text output.

This guide covers **every aspect** of both commands: full syntax, all options (GNU coreutils vs BSD/macOS differences), byte vs line mode, advanced features like live following, real-world patterns, pipelines, pitfalls, best practices, and performance notes. Everything is based on modern systems (2026 era): GNU coreutils (Linux) and BSD variants (macOS/FreeBSD).

We'll treat them together since they share many concepts (just mirrored behavior).

---

1. Basic Purpose & Syntax

`head` — Output the first part of files/streams.

`tail` — Output the last part of files/streams.

General Syntax (identical structure):

```
Bash

head [OPTION]... [FILE]...
tail [OPTION]... [FILE]...
```

- If no FILE (or `-` as filename), reads from **stdin** → perfect for pipelines.
- Multiple files → processes each separately, with headers by default (`-v` behavior).
- Output goes to stdout.

Default behavior (no options):

- `head` → first **10 lines**
- `tail` → last **10 lines**

---

2. Core Options — Lines vs Bytes

Both support **line mode** (default) and **byte mode**.

Option	Long Form	Applies To	Meaning	Example
`-n N`	`--lines=N`	Both	N lines (head: first; tail: last)	`head -n 5` → first 5 lines
`-n +N`	(tail only)	tail	From line N to end (skip first N-1 lines)	`tail -n +100` → lines 100–end
`-c N`	`--bytes=N`	Both	N bytes (head: first; tail: last)	`head -c 100` → first 100 bytes
`-c +N`	(tail only, GNU)	tail	From byte N to end	`tail -c +1024` → after first KB

Notes:

- N can be suffixed: `K` (kibibytes), `M` (mebibytes), `G` (gibibytes) — e.g., `-c 10M`.
- Negative N (e.g., `-n -5`) is **GNU extension** for "all but last/first N" (see advanced).

Example file `log.txt` (20 lines):

```
Bash

head -n 3 log.txt          # lines 1–3
tail -n 3 log.txt          # lines 18–20
tail -n +18 log.txt        # lines 18–20 (alternative)
head -c 50 log.txt         # first 50 bytes
```

---

3. Verbose / Quiet Mode (Multiple Files)

When processing **multiple files**, behavior changes:

Option	Long Form	Meaning	Default Behavior
`-v`	`--verbose`	Always print filename headers	No headers (single file)
`-q`	`--quiet / --silent`	Never print headers	Headers for multiple files (default)

```bash
head -v file1 file2 file3   # ⇒ ==> file1 <== \n content... \n ==> file2 <==
head -q file1 file2         # no headers
```

---

4. Advanced `tail` Features (Live Monitoring / Following)

`tail` 's most powerful feature: **real-time following** with `-f` .

Option	Long Form	Meaning	Notes
`-f`	`--follow`	Follow file as it grows (live tail)	Ctrl+C to stop; GNU & BSD
`-F`	`--follow=name --retry`	Same as `-f` + retry if file is rotated/truncated	GNU only; handles log rotation better
`--retry`		Keep trying to open file if inaccessible	GNU only
`--pid=PID`		Stop following when process PID dies	Useful for daemons
`--max-unchanged-stats=N`		(GNU) Check file attributes N times if no growth	Default 5

Classic live log monitoring:

```
Bash
tail -f /var/log/syslog          # watch system log in real time
tail -F /var/log/app.log         # survives log rotation (e.g., logrotate)
tail -f access.log | grep "ERROR" # live grep
```

With PID (stop when service dies):

```
Bash
tail -f --pid=$(pidof nginx) nginx.log
```

---

5. GNU Extensions (Linux) vs BSD (macOS/FreeBSD)

Important portability notes:

Feature	GNU (Linux)	BSD (macOS)	Workaround on macOS
`-n -N` (all but last N)	Yes (`tail -n -5` = omit last 5)	No	`head -n -5` (head has it)
`-c +N` / byte skipping	Yes	Limited	Use `dd` or `sed`
`-F` / `--retry`	Yes	No (`-f` only)	Install GNU tail via `brew install coreutils` → `gtail`
`--max-unchanged-stats`	Yes	No	N/A
Suffixes (K,M,G)	Yes	Yes (but case-sensitive)	

macOS tip: Prefix GNU versions with `g` after Homebrew install: `ghead` , `gtail` .

---

6. Real-World Examples

1. **Quick file preview**:

```bash
head large.csv                    # first 10 rows
tail -n 20 huge.log               # last 20 log entries
```

1. **Pagination / middle of file**:

```bash
# Lines 100–110
sed -n '100,110p' file.txt
# Or combine head/tail
tail -n +100 file.txt | head -n 11
```

1. **Binary files** (e.g., images, executables):

```bash
head -c 16 /bin/ls                # first 16 bytes (magic number)
tail -c 100 binary.file           # last 100 bytes
```

1. **Monitoring multiple logs**:

```bash
tail -f log1.log log2.log         # interleaved output with headers
```

1. **Skip header in CSV**:

```bash
tail -n +2 data.csv | head -n 5  # rows 2–6 (skip header)
```

1. **All but last 10 lines** (GNU):

```bash
head -n -10 file.txt
# Or portable
head -n $(($(wc -l < file.txt) - 10)) file.txt
```

---

7. Common Pitfalls & Best Practices

- **Large files**: Both are extremely efficient — stream without loading full file.
- **Binary data**: Use `-c` for bytes; line mode may stop at first NUL byte.
- **Empty files**: Output nothing (graceful).
- **Trailing newline**: Preserved correctly.
- **Performance**: Faster than `sed` / `awk` for simple head/tail tasks.
- **Portability**: Avoid GNU-only features (`-F` , `-n -N`) in cross-platform scripts.
- **Live following (`-f`)**: Consumes CPU if file doesn't change — combine with `timeout` or tools like `multitail` .
- **Multiple files**: Use `-v` for clarity.
- **Piping**: Perfect — e.g., `dmesg | tail -n 20` (last kernel messages).

Best practice:

- Default 10 lines is often enough → just `head file` or `tail file` .
- For logs → `tail -f` .
- For skipping headers → `tail -n +2` .
- For middle sections → `tail -n +START | head -n COUNT` .

---

8. Quick Reference Table (All Major Options)

Option	Applies To	Meaning	GNU/BSD
`-n N / --lines=N`	Both	N lines	Both
`-n +N`	tail	From line N	Both
`-n -N`	head/tail	All but last/first N (GNU)	GNU only
`-c N / --bytes=N`	Both	N bytes	Both
`-c +N`	tail	From byte N (GNU)	GNU only
`-f / --follow`	tail	Live follow	Both
`-F`	tail	Follow + retry (rotation)	GNU only
`--pid=PID`	tail	Stop when PID dies	Both
`-q / --quiet`	Both	No headers	Both
`-v / --verbose`	Both	Always headers	Both
`-z / --zero-terminated`	Both	NUL-terminated lines (rare)	GNU only

You've now got **complete mastery** of `head` and `tail` — from quick previews to live log monitoring.

What's your use case? Peeking at CSVs? Watching logs? Extracting middle sections? Share a sample command or file type and I'll give you the perfect one-liner! 😊