## NAME

**tail** — extract file trailer, follow changes to file

## SYNOPSIS

**tail** [**-zqv**] [**-n** [**+**|**-**]*lines*|**-c** [**+**|**-**]*bytes*] [**-**{**f**|**F**}] [**--pid**=*pid*] [**-s** *interval*]
[**--max-unchanged-stats**=*max*]] [**--retry**] [*file*]...
**tail -r** [**-zqv**] [**-n** *lines*|**-c** *bytes*] [**--retry**] [*file*]...

## DESCRIPTION

Without **-r**, copies the last *lines* (default **10**) or *bytes* of each *file* (standard input stream if "**-**", the default) to the standard output stream.
If *lines* or *bytes* start with a **+**, the given number of *lines* or *bytes* is skipped from the beginning, then the remainder of the file is copied, instead.

Then, if **-f**|**-F**, waits until more data is written to each *file*, and copies that as well. This stops only when all *file*s hang up/error, or *pid* was specified and died. **-F** additionally reopens *file*s by name when they're replaced.

With **-r**: reverses entire *file*s, line by line. **-nc** limit the range to be reversed.

With more than one *file*, each one is delineated by an empty line and
`==> file <==`
heading.

*lines* and *bytes* are in the mostly-case-insensitive format:
[*base*][**b**|**KMGTPEZY**[**B**]] (with at least one of {*base*, **b**, **KMGTPEZY**, **B**})
Where *base* is an optionally-floating-point number of bytes, defaulting to **1**, which is then optionally multiplied by the relevant unit. **B** sets the unit multiplier to **1000** (from **1024**). **b**(lock) is a unit of **512**. *lines*|*bytes* is equal to $base \cdot unit^{mult}$, if any, or *base*.

### Following

If the standard input stream is a pipe (FIFO), it is not followed. Block devices and directories aren't followed ever.

Regular files that got truncated are output again in their entirety.

**-F** is most useful in log-rotation-style scenarios, where a file, such as, traditionally, `/var/log/messages`, is moved away (to `/var/log/messages-20230703`), and an empty file made in its place to hold new messages for the day. **-f** would continue to fruitlessly read the original file; **-F** detects this scenario (since `/v/l/m` is now a different file) and follows the new file named by the path, copying it from the beginning.

## OPTIONS

**Without -r**:

| | |
|---|---|
| **-n**, **--lines=** *lines* | Copy the last *lines* of each *file*. Default: **10**. |
| **-c**, **--bytes=** *bytes* | Copy the last *bytes* of each *file*. |
| **-n**, **--lines=-***lines* | The same. |
| **-c**, **--bytes=-***bytes* | The same. |
| **-n**, **--lines=+***lines* | Start copying from the *lines*th line of each *file*. |
| **-c**, **--bytes=+***bytes* | Start copying from the *bytes*th byte of each *file*. |

**With -r**:

| | |
|---|---|
| **-r** | Copy each *file* in reverse line order (last line first, then the previous). |
| **-n**, **--lines=** *lines* | Limit to *lines*. |
| **-c**, **--bytes=** *bytes* | Limit to *bytes*. Truncated lines have their starts removed. |

**Generally:**

**-z**, **--zero-terminated** Line separator is NUL, not newline. File headers are still written with newlines.

**-q**, **--quiet**, **--silent** Never write file headers.

**-v**, **--verbose** Always write file headers.

**-f**, **--follow**, **--follow=descriptor**
After *file*s run out, keep trying to read them, and copy new data as it appears.

**--follow=name** The same, but if they haven't seen new data in *max·interval*, check to see if they were replaced by a different file at the same location; if so: output that file in its entirety and follow it going forward.

**-F** **--follow=name --retry**

**--retry** With **-f**|**-F**: if a *file* wasn't found, skip it, process all the other *file*s, then open and follow it when it appears while following.
Without: keep trying to open it synchronously (holding up other *file*s) until it is.

**--pid**=*pid* Exit **0** instantly when *pid* dies when following.

**-s**, **--sleep**=*interval* If *pid* was specified, or any *file* cannot be poll(2)ed, polled, sleep for sleep(1)-style *interval* between checking. Default: **1s**.

**--max-unchanged-stats**=*max*
With **-F**, try reopening *file* every *max*th time it returns empty (or if it hasn't been readable in *max* iterations). No effect otherwise.

**--follow** values are prefix-matched (**--follow**=*n* is equivalent to **--follow=name**, &c.).

## EXIT STATUS

**1** if a *file* couldn't be opened (if **--retry**: for any reason except not existing ; if not **--retry**: for any reason) or read. (With **-F**, this extends to successfully-re-opened *file*s.)

## EXAMPLES

```
$ find tests/ -type f -executable -exec tail -n2 {} +
==> tests/head <==
errstr="$("$head" head.1-1 2>&1 > /dev/full)" && echo "head: /dev/full
[ -n "$errstr" ]                               || echo "head: stderr em

==> tests/env/test <==

"$env" - 2>&3 > /dev/full

==> tests/env/code/env.1-3c <==
#!/usr/bin/env -S PYTHONUNBUFFERED=1 python3 -S
echo "PYTHONUNBUFFERED=$PYTHONUNBUFFERED"


# Extract just the signature generated by Linux sign-file
$ tail -c754 /lib/modules/$(uname -r)/updates/dkms/zfs.ko

$ seq 100 | tail -rc12 | paste -s
100     99      98      7
```

## SEE ALSO

cat(1), dd(1).

head(1) — '**-**'-marked *lines* and *bytes* are symmetric, and '**+**'-marked ones are almost symmetric:

```
{ head -c -10 f; tail -c -10 f; }
{ head -n +10 f; tail -n +11 f; }
```

are equivalent to **cat** f.

`tac`(1) non-portably provides a more general **-r**.

## BUGS

Truncation detection is inherently racy. **echo** `abc > zupa`**; echo** `def > zupa` will, unless under truly pathological load conditions, never cause **tail -f** `zupa` to have produced "def".

### inotify

Consider the scenario where **echo** `'Rejtan je żur.' > f`**; tail -f** `f`, then at some point in the future `{` **sleep** `0.5`**; echo** `'Żupan go uciska.'; } > f` — depending on how the **sleep** is clocked to *interval*, **tail** will either see the file become empty, detect the truncation, then see it grow "Żupan...", or it'll see the file having grown by two bytes (the dot and newline).

Truncation notwithstanding, either responsiveness or system load has to be sacrificed to detect updates to regular files and pipes at closer-to-real-time rate. Similarly, replacement detexion with **-F** suffers greatly from being quantised, effectively being saddled with an up-to-five-second delay by default.

Under Linux, `inotify`(7) is used by default to watch unpollable files, which means **tail** wins the truncation race the vast majority of the time (cf. **BUGS**). However, for the **-F** renaming case, this is less rosy: it works in the base-line log rotation case, but will fail to recognise, for example, **mv** `/var /жар`. This is deemed acceptable — against the pain of the unaccelerated behaviour — for the usual case; however, to achieve "just watch *file*s, no odd quirks" behaviour, specify **---disable-inotify**. (In this case, doing **mv** `/var/log /var/лог` or removing `/var/log` will disable acceleration automatically.)

If `inotify`(7) is available, not disabled, and hasn't failed, *interval* and *max* are effectively meaningless, since **tail** only wakes up when there's data to service.

## STANDARDS

IEEE Std 1003.1-2024 ("POSIX.1"); **-zqvFs**, unit-suffixed *lines* and *bytes*, **--pid**, **--max-unchanged-stats**, **--retry**, and more than one *file* are extensions, originating from the GNU system. With **-r**, **-c** is an extension, originating from the BSD. `sleep`(1)-style suffixes for **-s** are extensions. **-f** is not required to detect truncations (and is only strictly required to work for regular files and pipes (FIFOs)), the illumos gate's **-f**|**-F** behave similarly to this implementation; NetBSD 1.4 and FreeBSD 4.0 only detect truncation with **-F**, but detect appending regardless of file position (in addition to their **-F** being mostly equivalent to this implementation). OpenBSD **-f** is like this implementation's **-F**.

The GNU system disallows *lines* and *bytes* with **B** but without a multiplier and only supports integer *base*s. It also ignores **--retry** without **-f**|**-F**.

A heretofore-unnoted legacy usage of

**tail** ±[*count*][**l**|**c**|**b**][**f**|**r**] [*file*]

is available (so long as it also wouldn't form a valid or invalid normal usage) for compatibility with 4.2BSD (with **b** being ·512**c**, empty decimal *count* equivalent to **10**, and **+0** equivalent to **+1**). Avoid it.

With **-r**, **-nc** also accept ± for compatibility with the aforementioned (and, likewise for compatibility, the value following **+** is reduced by **1**) — this is both odd and forbidden by POSIX.

## HISTORY

### Research UNIX

Appears in the Programmer's Workbench (PWB/UNIX) as `tail`(I) ("deliver the last part of a file") with a **SYNOPSIS** of

**tail** [ ±number[**lbc**] ] [ file ]

which matches the legacy usage described in **STANDARDS**; the default of "`-10l`" matches present day. A *number* must be given. Pipes and character devices are detected and read sequentially, except in **-** mode where only pipes are handled properly. The input is considered a pipe if the standard input stream is a pipe.

The **BUGS** note that

> Tails relative to the end of the file are treasured up in a buffer, and thus are limited in length.

and this means that in **−** mode, **tail** finds the last 4kB and then extracts the request therefrom.

Version 7 AT&T UNIX sees

```
usage: tail +n[lbcr] [file]
```

and a comment of

```
 *      Type 'r' means in lines in reverse order from end
 *       (for -r, default is entire buffer )
```

but the manual is unchanged. *number* can be omitted (in which case it's taken to be **0**, except for **r**), thus **tail −r** is equivalent to present-day. **r** forces **−** mode, is exclusive with the other addressing types (implying **n**) and forcibly adds a newline if the final input line lacks one (i.e. **printf** `'a\nb\nc'` | **tail −*2*r** yields "c", newline, "b", newline; rather than "cb", newline) in contrast to modern behaviour.

### System V

CB-UNIX 2.1 carries a derivative of Programmer's Workbench (PWB/UNIX) **tail** with a **SYNOPSIS** of

> **tail** [ ±[number][**lbc**] [ **f** ] ] [ file ]

and a missing *number* defaulting to **10**, the "is input a pipe" flag is reset when opening *file*. The argument is better-spaced as ±[*number*] [**lbc**][**f**]: **lbc** are exclusive, **f** may be added (though the order doesn't matter).

**f** is in principle like modern-day **−f**: unless the input was a pipe, instead of exiting **tail** enters an infinite loop of sleeping for 1 second and copying the input file to the standard output stream until end-of-file.

CB-UNIX was, among others, the basis for AT&T System III UNIX, where it first saw light outside of AT&T. However, AT&T System III UNIX was the first one with named pipes (`fifo`(7)s), so the "is input a pipe" flag is once again misconstrued.

AT&T System V Release 1 UNIX changes the **b** units from **512** to <`sys/param.h`> `BSIZE`, dependent on the system's filesystem configuration (**512** or **1024**).

AT&T System V Release 3 UNIX rolls this back by explicitly defining

```
#define BSHIFT 9        /* log2(512)=9, 512 is standard buffer size */
```

AT&T System V Release 4 UNIX patches in 4.3BSD's **r** and *number* parsing (but not **b**) via SunOS.

### The BSD

3BSD carries Programmer's Workbench (PWB/UNIX) **tail** (notably in contrast to being otherwise Version 7 AT&T UNIX).

4.0BSD sees a **SYNOPSIS** of

> **tail** [ ±number[**lbc**][**fr**] ] [ file ]

based on an AT&T System III UNIX foundation. The argument is more accurately transcribed as ±[*number*] [**lbcr**][**f**] inasmuch as all of **lbcr** are exclusive, but may be followed by **f**. If *number* is missing, it still defaults to **10**, except in **r** mode, where it's infinity like in Version 7 AT&T UNIX. **r** overrides **f**. If *number* is **0**, it's treated the same as if it were missing.

4.2BSD fixes the check for the input being a pipe to run *after* opening *file* and differentiates *number* being **0** from being not given.

4.3BSD expands the **−**-mode buffer to **32kB** and turns ±**b** from meaning **10·512** to **1·512** bytes.

4.3BSD−Tahoe exits if a write fails during **f**.

### Standards

System V Interface Definition ("SVID"), and X/Open Portability Guide Issue 2 ("XPG2") include Version 2 AT&T UNIX's **tail**.

IEEE Std 1003.2 ("POSIX.2")'s **tail** ("Copy the last part of a file") sees a **Synopsis** of

`tail` [−f] [ −c *number* ||| −n *number* ] [*file*]
*Obsolescent versions:*
`tail` −[*number*][c|||l][f] [*file*]
`tail` +[*number*][c|||l][f] [*file*]

where **−fcn** are as present-day POSIX, and the **−** mode is allowed to use a buffer, but of size bumped to **10**·`LINE_MAX`. **−f** is allowed to do whatever if the input is neither a pipe nor a regular file. The obsolescent versions are defined in terms of **−fcn**, and *number* defaulting to **10**. The input is required to be a text file (all lines end in newlines, no NUL bytes) unless in **−c** mode; this is more stringent than implementations which were unaffected by NULs, but will avoid needing to opine on the new-line-insertion-on-last-line behaviour in future.

The Single UNIX Specification ("SUS") carries IEEE Std 1003.2 ("POSIX.2") **tail** with the addition of the **b** obsolescent unit, expectedly defined as ·**512c**.

IEEE Std 1003.1-2001 ("POSIX.1") removes the obsolescent usage.

IEEE Std 1003.1-2008 ("POSIX.1") allows **+** to be used as an option delimiter like **−** (i.e. **+n** *10* is equivalent to **−n** *10*), implementations to interpret a *file* of "**-**" as the standard input stream, forbids *number*s of **+0**.

IEEE Std 1003.1-2024 ("POSIX.1") adds **−r**, as present-day.

**The BSD (again)**

4.4BSD includes a new **tail**, with a **SYNOPSIS** of

**tail** [**−f** | **−r**] [**−b** *number* | **−c** *number* | **−n** *number*] [*file* **...**]

citing compatibility with IEEE Std 1003.2 ("POSIX.2"). Naturally, **−br** and allowing more than one *file* are extensions. The latter is as present-day format-wise, but "**-**" is not yet understood. The historical usage is also supported, but **−***4***cr** (equivalent to **−rc***4*) is "last 4 bytes" whereas previously **r** implied **n**, overriding **c**. **−r** has no limit.

The manual says that "The **−f** option is ignored if the standard input is a pipe, but not if it is a FIFO."; this is false, since this is triggered by `lseek`(2) on the standard input stream (i.e. if no *file*s were given) returning `ESPIPE`. This stems from linguistic confusion over the standard distinguishing pipes and `fifo`(7)s which are, in fact, the same. **−f** is simply ignored if reading from the standard input stream and it is a pipe, as present-day.