Notes 8

File manipulation

cat

Definition

Concatinate files and print the output as standard text.

Usage

Useful in quickly viewing the raw contents of files, but may not be adequate for more precise uses or for larger files.

Examples

```
cat users_list.txt
```

> prints the contents of `users_list.txt`

```
cat errors??.log
```

> concatinates each `error??.log` file, allowing for viewing of multiple files as defined by the wildcard.

tac

Definition

Concatinates file and prints the output starting with the end line going backwards to the front of the file.

Usage

`tac`, followed by a `FILE`. Useful when wanting to see the latest additions to a file, allowing for easier log and list parsing.

Examples

```
tac users_updated_2024-05-14.txt
```

> prints a list of users starting from the last entry to the first.

head

Definition

Prints the start of a file, ten lines by default, user input allows for a defined amount of lines to print.

Usage

Useful when needing to see the setup of a file, such as checking for python libraries, bash starter, or any number of file starting lines.

Examples

```
head update-2025-7-15.log
```

> Will print the first ten lines of the file `update-2025-7-15.log`

```
head -n 4 verification.sh
```

> Checks the first four lines of the shell script

tail

Definition

Prints the file's last ten lines by default, user defined number will produce from that last line up.

Usage

Seeing the end of logs, the last additions to the file in order. Allows for live viewing of log files as they are written into.

Examples

```
tail -f
```

> `-f` follows the live output, printing the tail of a file as it is being written.

```
tail -n 20 -f requests.log
```

> last 20 lines, then follows in that window

```
tail -5 logs.txt
```

> Views the end of logs, `-5` is BSD syntax, `-n5` and `-n 5` are same

cut

Definition

Slices files by their **delimiters**, with spaces being the tool's default.

Usage

Used when needing to see one field in a file's set.

Examples

```
cut -d',' -f3 users.csv
```

> Prints the third field in a `.csv` file

```
cut -c5-15 log.txt
```

> Prints from the 5th to 15th character in position

```
cut -d' ' -f1 access.log
```

> Takes the first field from an access log

sort

Definition

Writes out a **concatinated**, sorted output, given arguments and a file.

Usage

`sort`, a flag, then the name of the file.

Examples

```
sort employees.csv
```

> Prints each line of `employees.csv` in alphabetical order.

```
sort -n logins.log
```

> Sorts and prints the contents of `logins.log` in numeric order.

wc

Definition

Print newline, word, and byte counts for a file or multiple. A word is any non-empty sequence of non-white spaces, bounded by white space characters at the beginning and end of the input.

Usage

Useful for comparing file word and line counts.

Examples

```
wc -l ~/projects/scraper/scraper.py
43 scraper.py
```

> Prints a count of lines in `scraper.py`.

```
wc -w notes4.md ../notes5/notes5.md
 268 notes4.md
 766 ../notes5/notes5.md
1034 total
```

> Comparison of the word count of notes4.md and notes5.md, with a total tally.