Commands

Useful pipe commands:

The command `ls | head` runs ls and head and pipes the standard output of ls into the standard input of head.

The following shell commands are particularly useful in pipes:

- `cat [FILENAME [FILENAME...]]` writes the contents of one or more files to standard output. This is a good way of starting a pipe. If you leave off all the filenames, cat just reads its standard input and writes it to standard output.
- `head [-n N]` reads its standard input and writes only the first N lines (default is 10 if you leave the option off) to standard output. You can also put a minus before the argument e.g. `head -n -2` to *skip the last 2 lines* and write all the rest.
- `tail [-n N]` is like head except that it writes the last N lines (with a minus, it skips the first N ones).
- `sort` reads all its standard input into a memory buffer, then sorts the lines and writes them all to standard output.
- `uniq` reads standard input and writes to standard output, but skips repeated lines that immediately follow each other, for example if there are three lines A, A, B then it would only write A, B but if it gets A, B, A it would write all three. A common way to remove duplicate lines is `... | sort | uniq | ...`.
- `grep [-iv] EXPRESSION` reads standard input and prints only lines that match the regular expression to standard output. With `i` it is case-insensitive, and with `v` it only prints lines that do *not* match the expression.

- `sed -e COMMAND` reads lines from standard input, transforms them according to the command and writes the results to standard output. `sed` has its own command language but the most common one is `s/SOURCE/DEST/` which changes substrings matching the source regular expression into the destination one.
- `wc [-l]` stands for word count, but with `l` it counts lines instead. Putting a `wc -l` on the very end of a pipe is useful if you just want to know how many results a particular command or pipe produces, assuming the results come one per line.

All these commands actually take an optional extra filename as argument, in which case they read from this file as input. For example, to display the first 10 lines of a file called `Readme.txt`, you could do either `cat Readme.txt | head` or `head Readme.txt`.

---

grep

```
ls -1 | grep software    // filters out files without software p


grep -nHi // pattern filenames

grep


FILENAME | grep -i '^[a]' | head -n 1  // this finds the first l
```

grep - command on Unix and Linux used for searching through text in files or *standard input*

- it reads files line by line searching for patterns that match the expression provided to it as an argument
- `grep [-iv] EXPRESSION` reads standard input and prints only lines that match the regular expression to standard output. With `-i` it is case-insensitive, and

with `-v` it only prints lines that do *not* match the expression.

---

sed

- stands for **stream editor** - it can change text using regular expression as it passes from its standard input to its standard output

Common flags

- `sed -e COMMAND` reads lines from standard input, transforms them according to the command and writes the results to standard output. `sed` has its own command language but the most common one is `s/SOURCE/DEST/` which changes substrings matching the source regular expression into the destination one.

This flag allows you to specify multiple commands. It's useful for applying several transformations sequentially on the input stream.

- **Example**: Replace 'apple' with 'orange' and then 'day' with 'night'.
- sed is a 'transformer' for text

```
echo " Hello World | sed -e 's/World/Universe/'


OUTPUT:

Hello Universe
```

- you change the pattern 'World' into 'Universe'

```
echo "apple day" | sed -e 's/apple/orange/' -e 's/day/night/'



Output: orange night
```

sed -i

Modifies files in-place. If a SUFFIX is provided, it backs up the original file by appending the SUFFIX to the filename before making changes.

- **Example**: Change 'hello' to 'hi' in `greetings.txt`, backing up the original file as `greetings.txt.bak`.

```
sed -i'.bak' 's/hello/hi/' greetings.txt
```

y command

---

concatenating files

```
cat "file.txt"
```

- When you use `cat` with a single file, it simply displays the content of that file to the standard output

Else you can use `cat` to concatenate multiple files together:

```

```

Copying and Voving files

copy a file

```
// to copy one file

cp filename

// to copy and save it under new name:

cp filename newfilename
```

move a file

```
mv
```

find a path to current directory

```
pwd // shows path to current directory
```

Shellscript checking

```
sudo apt install shellcheck
```

1. Open Terminal
2. Navigate to the directory where your shell script file is located, or specify the full path of the file in the command.
3. Use the `shellcheck` command followed by the name of your shell script file. For example, if your shell script file is named `myscript.sh`, you would use the following command:

```
shellcheck myscript.sh
```

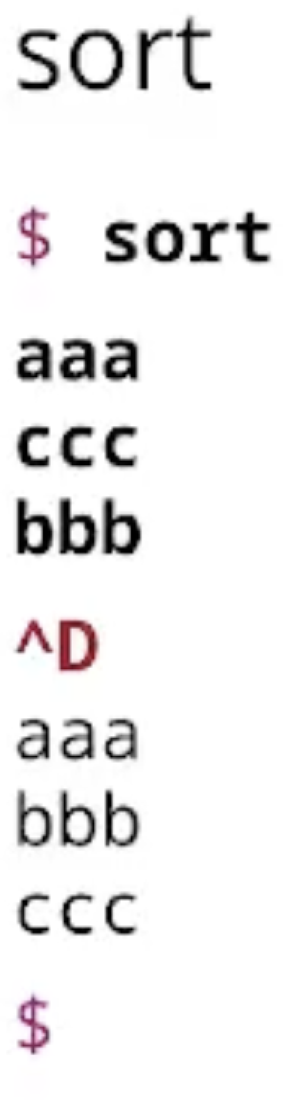
If you want to check a shell script located in a different directory, you can specify the full path to the file, like so:

```
shellcheck /path/to/your/script/myscript.sh
```

sort

```
sort   //sort is a program that reads the lines you pass it, so
             // produces the sorted outcome in the output


sort -r    // sorts them in reverse order
```



```
sort

$ sort
aaa
ccc
bbb
^D
aaa
bbb
ccc
$
```

remove duplicates

- only sequential duplicates, e.g. aba would be fine
- aab would be reduced into → ab

```
// usually used alongside sort (see above):

COMMAND | sort | uniq
```

```
... | sort | uniq | ...
```

redirecting output to a file

```
cat inputfile | sort > output file


cat inputfile | tee output file
```

extract text from PDF

```
pdftotext file.pdf  // this would create file.txt
```

word count

- `wc [-l]` stands for word count, but with `l` it counts lines instead. Putting a `wc -l` on the very end of a pipe is useful if you just want to know how many results a particular command or pipe produces, assuming the results come one per line.