12. Piping

commands table

commands	application

01. What actually matters in this section?

- `useful`: piping
- `important`: tee command
- `nice to have`: tr command

02. Intro to Piping

- pipe character `|` is used to separate two commands. The output of the first command will be passed to the standard input of the second command.
- example: `command1 | command2`

  `date | rev` will get the date and then reverse it

  `ls /usr/bin -1 | wc -l` will get the list items of the passed directory and then pass to `wc -l` which will print the line counting.

04. Comparing Redirection & Pinging

- Though both the `>` character and the `|` character are used to redirect output, they do it in very different ways
- `>` connects a command to some file
- `|` connects a command to another command
- we can use both together
    - example: `ls /usr/bin -1 | wc -l > count.txt` will list the items and on the directory on a separate line and then pipe with word count to calculate a number of lines then redirects the output to the file name we passed.

05. tr command

- `tr` translate, squeeze, and/or delete characters from standard input writing to standard input
- example: replace each small case `s` with upper case `S`

  command: `cat msg | tr s S` will get the content from `msg` file and then translate `s` to `S`

06. Working with multiple pipes

- example: `cat file | head -7 | tail -5` will feed a file to head, which cuts it down to the first 7 lines of the file and passes it to tail, which then outputs the last 5 lines of the chunk coming to it

07. Working with the tee command

- The tee command reads standard input and copies it both to standard output and AND to a file. This allows us to capture information part of the way through a pipeline, without interrupting the flow.

- example: `command1 | tee file1.txt | command2` will execute the `command1`, redirect the output to `file1.txt`, and also passes the same as an input to `command2`
-