









Architecture & Deployment

2025-2026 v0.1.0 on branch main Rev: 03b1cdace14bb0b0720e24be862097b3792214ea

Unix Pipeline

This exercise teaches you how to use Unix command pipelines and stream redirections to analyze and manipulate text files efficiently.






Table of contents

-  [Legend](#)
-  [Setup](#)
-  [The exercise](#)
-  [Example](#)
-  [Your tools](#)
-  [What have I done?](#)



Legend

Parts of this exercise are annotated with the following icons:

-  A task you **MUST** perform to complete the exercise
-  An optional step that you may perform to make sure that everything is working correctly, or to set up additional tools that are not required but can help you
-  The end of the exercise
-  The architecture of the software you ran or deployed during this exercise.
-  Troubleshooting tips: how to fix common problems you might encounter

! Setup

Download the exercise file to your computer with the following command:

```
$> curl -L https://git.io/fAjRa > rainbow.txt
```

Display the file:

```
$> cat rainbow.txt
Somewhere over the rainbow
...
```

! The exercise

Use command pipelines and stream redirections to:

- Count the number of lines and characters in the text
- Print the lines of the text containing the word `rainbow`
- Do the same but without any duplicates
- Print the second word of each line in the text
- Compress the text and save it to `rainbow.txt.gz`
- Count the number of times the letter `e` is used (case-insensitive)
- Count the number of times the word `the` is used (case-insensitive)
- Answer the question: what are the five most used words in the text (case-insensitive) and how many times are they used?

Example

For example, the following command counts the number of words in the text:

```
$> cat rainbow.txt | wc -w
255
```

Your tools

Here are a few commands you might find useful for the exercise. They all operate on the data received from their standard input stream, and print the result on their standard output stream, so they can be piped into each other:

Command	Description
<code>cut -d ' ' -f <n></code>	Select word in column <code><n></code> of each line (using one space as the delimiter)
<code>fold -w 1</code>	Print one character by line
<code>grep [-i] <letterOrWord></code>	Select only lines that contain a given letter or word, e.g. <code>grep foo</code> (<code>-i</code> to ignore case)
<code>grep "^<text>\$"</code>	Select only lines that contain this exact text (e.g. <code>grep "^foo\$"</code>)
<code>gzip -c</code>	Compress data
<code>sort [-n]</code>	Sort lines alphabetically (<code>-n</code> to sort numerically)
<code>tr '[:upper:]' '[:lower:]'</code>	Convert all uppercase characters to lowercase
<code>tr -s '[:punct:]' '[:space:]' '\n'</code>	Split by word

Command	Description
<code>uniq [-c]</code>	Filter out repeated lines (<code>-c</code> also counts them)
<code>wc [-l] [-w] [-m]</code>	Count lines, words or characters

Tip

Remember that if you want to know more about any of these commands or their options, all you have to do is type `man <command>`, i.e. `man cut`.

What have I done?

You have seen that text can be passed through several programs and transformed at each step to obtain the final result you want.

In essence, you have constructed complex programs by piping simpler programs together, combining them into a more powerful whole. You have applied the Unix philosophy.

[↑ Back to top](#)