# System Programming in C – Homework Exercise 2

**Publication date:** *Wednesday, April 3, 2024*
**Due date:** *Sunday, April 21, 2024 @ 21:00*

**General notes:**

- A piped sequence of commands is an ordered sequence of Linux commands, each connected to its preceding command using the pipe symbol ('|'). Do not use any other way to combine commands (e.g., *process substitution, etc.*).
- You should only use commands we saw in weeks 1-2:

  **`cp cat head tail cut paste wc sort uniq tr`**
- Do not use commands **`grep`** and **`sed`**, which we see in week 3.
- As always, make sure to use relative paths and not absolute paths. Your solution scripts should be executable from any location in the file system. The only exception to this rule is to use `/share/ex_data/ex2/` when copying files from the shared directory.

## Problem 1:

This question involves parsing a file named `planets.txt` with information about the planets in the solar system. Every line in file `planets.txt` has the following format:

**`<planet>`**`<comma><space>`**`<year>[`**`<space>`**`<status>]`**

- **`<planet>`** consists of an uppercase letter followed by any number of lowercase letters (e.g., `Mars`).
- **`<year>`** is either a four-digit number representing the year of discovery or -1 to indicate year of discovery unknown (or not applicable).
- **`<status>`** is an optional field and is not specified for every planet. When it is specified, it contains text describing the planet. This text may contain white spaces, but you may assume it does not contain any punctuation marks.
- `<space>` and `<comma>` represent the characters `' '` and `','`.

Write a piped sequence of commands for each of the tasks specified in 1-5 below. Specify each piped sequence in a separate line in a solution file named `ex2.1.sh`. The file should contain exactly five lines, each containing a single command or a single piped sequence of commands.

1. Copy the file `planets.txt` from directory `/share/ex_data/ex2/` to the current directory.

2. Write a single command that takes the file `planets.txt` and generates a file named `planets2.txt` that contains the same lines except sorted according to year of discovery (increasing order). Planets with the same year of discovery should be sorted in reverse alphabetical order.

   **Validation:** File `planets2.txt` should contain the same number of lines as `planets.txt`, and its first two lines should be:
   ```
   Mercury, -1 closest to the sun
   Earth, -1 our planet
   ```

3. Write a piped sequence of commands that takes the file `planets.txt` and generates a file named `planets3.txt` that contains the name and year of discovery of the planets in lines 4-7 in file `planets.txt` in the following format:

   **<PLANET>**<colon>**<year>**

   Where **<PLANET>** is the planet's name in upper-case letters, <colon> is the colon character (`':'`), and **<year>** is the year of discovery as in the original file. The optional status should not be printed into `planets3.txt`.

   **Validation:** File `planets3.txt` should contain exactly four lines, and the first two lines should be:
   ```
   MERCURY:-1
   NEPTUNE:1846
   ```

4. Write a piped sequence of commands that takes the file `planets.txt` and appends to file `planets2.txt` (generated in Problem 1.2 above) the number of planets for which the discovery year is unknown (`-1`).

   **Validation:** The piped sequence of commands should add a single line with the number `2` to file `planets2.txt`.

5. Write a piped sequence of commands that takes the file `planets.txt` and generates a file `planets5.txt`. that specifies the optional status of each planet in the following format:

   **<pl>[**<space>**<STATUS>]**

   Where **<pl>** are the first two letters in the planet's name in lower case, <space> is the space character (`' '`), and **<status>** is optional status as in the original file. The year of discovery should not appear in `planets5.txt`.

   **Validation:** File `planets5.txt` should contain the same number of lines as `planets.txt`, and the first two lines should be:
   ```
   ea our planet
   ma
   ```

   **Hint:** To implement this, you may wish to pipe in the output of a piped sequence of commands as one of the two input arguments for command `paste`. We show you how to do this in Recitation (תרגול) #2.

## Final validation and testing for Problem 1:

1. Verify that file `ex2.1.sh` has exactly five lines (use `wc`) and follow the specific validation steps specified for each item in 2-5.
2. Execute the commands in file `ex2.1.sh` as scripts (using `source`) and confirm that the following four files are created in the current directory (use `ls`):
   ```
   planets.txt    planets2.txt
   planets3.txt   planets5.txt
   ```
3. The script should **not print anything to the screen** (errors or standard output).
4. Run the script from **various locations** (not just `~/exercises/ex2/`) to ensure that it does not contain unwanted absolute paths.
5. Use the script `/share/ex_data/ex2/test_ex2.1` to test your solution. Execute the following command from directory `~/exercises/ex2/`:
   ```
   ==> /share/ex_data/ex2/test_ex2.1
   ```
   The script produces a detailed error report to help you debug your code. It tests your code on the original input file (`planets.txt`) as well as a slightly modified version to make sure that your code follows various requirements.

## Problem 2:

This question involves parsing a story file called `story.txt`, containing the text of the novel *Animal Farm* by George Orwell taken from the Gutenberg project (https://gutenberg.net.au/ebooks01/0100011h.html). The file contains free text in no specific format. Write a piped sequence of commands for each of the tasks specified in 1-5 below. Specify each piped sequence in a separate line in a solution file named `ex2.2.sh`. The file should contain exactly five lines, each containing a single command or a single piped sequence of commands.

1. Copy the file `story.txt` from the shared directory `/share/ex_data/ex2/` to the current directory.

2. Write a piped sequence of commands that takes lines 218-237 of `story.txt` and switches between vowel letters according to the following permutation: A→E→I→O→U→A. For example, the letter "E" should be replaced by "I." The transformation should be done in lower-case letters just like in upper-case ones, such that, for example, the letter "u" should be replaced by "a." The transformed text should be printed in file `story-switched-vowels.txt`

   **Validation:** The file `story-switched-vowels.txt` should contain exactly 20 lines, and the first two lines should be:
   `Cheptir V`
   `Es wontir driw un, Mulloi...` (the line continues)

3. Write a piped sequence of commands that takes `story.txt` and prints to the screen the number of times that a single or double quote character (`'''` or `'"'`) appears in the story.

   **Validation:** The number of single and double quote characters in the file is 624.

4. Write a piped sequence of commands that takes `story.txt` and prints to a file named `letter-stats.txt` the four non-space characters that appear most as the 3rd character in a line. For each such character, the file `letter-stats.txt` should contain a line in the following format:

   `<dash>`**`<count>`**`<dash>`**`<char>`**

   Where **`<char>`** is the character, **`<count>`** is the number of times that this character appears as the 3rd character in a line of `story.txt`, and `<dash>` is the dash character (`'-'`). The four characters should be sorted according to their counts (from most frequent to less), and characters with the same count should be sorted alphabetically.

**Validation:** The expected contents of `letter-stats.txt` are:

```
-62-e
-32-o
-28-a
-28-t
```

5. Write a piped sequence of commands that prints a list of all distinct words that appear in `story.txt` into the file `story-words.txt`. For this purpose, a word is defined as a non-empty contiguous sequence of characters that does not contain any white spaces or punctuation marks and before it and after it there is a white space or punctuation mark (use the appropriate `[::]` tags here). Furthermore, your consideration of words should be case-insensitive, meaning that "Hello" and "hello" should count as the same word. The words in file `story-words.txt` should be written in lower-case letters and listed in alphabetical order.

   **Validation:** The expected output file is provided to you in: `/share/ex_data/ex2/story-word-expected.txt`.
   Use the `diff` command to compare your file to this one.

## Final validation and testing for Problem 2:

1. Verify that file `ex2.2.sh` has exactly five lines (use `wc`), and follow the specific validation steps specified for each item in 2-5.
2. Execute the commands in file `ex2.2.sh` as scripts (using `source`) and confirm that the following four files are created in the current directory (use `ls`):

   ```
   letter-stats.txt      story-switched-vowels.txt
   story.txt             story-words.txt
   ```

3. The script should print **one thing on the screen** (the number of quote characters for (3)), and **should not produce any error messages**.
4. Run the script from **various locations** (not just `~/exercises/ex2/`) to ensure that it does not contain unwanted absolute paths.
5. Use the script `/share/ex_data/ex2/test_ex2.2` to test your solution. Execute the following command from directory `~/exercises/ex2/`:

   ```
   ==> /share/ex_data/ex2/test_ex2.2
   ```

   The script produces a detailed error report to help you debug your code. It tests your code on the original input file (`story.txt`) as well as slightly modified versions to make sure that your code follows various requirements.

## Submission Instructions:

1. After you validated and tested your solution, make sure that your `~/exercises/ex2/` directory contains the scripts `ex2.1.sh` and `ex2.2.sh`, and a `PARTNER` file containing the user id of the non-submitting partner. The non-submitting partner should also add a `PARTNER` file containing the user id of the submitting partner. See **Homework submission instructions** for details.

2. Check your solution by running **`check_ex ex2`**. The script should be executed from the account of the submitting partner, and it may be run from any directory. Clean execution of this script guarantees you 80% of the assignment's grade.

3. Once you are satisfied with your solution, you may submit it by running **`submit_ex ex2`**. The script should be executed from the account of the submitting partner and may be run from any directory. You may modify your submission any time before the deadline (**21/4 @ 21:00**) by running **`submit_ex ex2 -o`** from any location.

4. For more information on the submission process, see the **Homework submission instructions** file on the course website.