System Programming in C – Homework Exercise 2

Publication date: *Thursday, March 10, 2022*
Due date: *Sunday, March 27, 2022 @ 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 the first two weeks of the semester:

  `cp cat head tail cut paste wc sort uniq tr`
- Do not use commands **`grep`** and **`sed`**, which we see in the third week.
- Make sure to use relative paths and not absolute paths. Your solution scripts should be executable from any location in the file system. In particular, your scripts should not contain absolute paths for specific directories, other than the shared data source directory `/share/ex_data/ex2/`.

Problem 1:

This question involves parsing a file (`nba_roster.txt`) with information about the nine tallest players on the 2022 New York Knicks basketball team (https://www.basketball-reference.com/teams/NYK/2022.html). Every line in the file has information about one player, in the following format:

`<last>``<comma><space>`**`<first>`**`<dash><space>`**`<position>`**`<space>`**`<uniform>`**

- **`<last>`** consists of an uppercase letter followed by any number of lowercase letters (e.g., `Sims` or `Noel`).
- **`<first>`** consists of an uppercase letter with the player's first initial.
- **`<position>`** has one or two upper-case letters, signifying the player's position.
- **`<uniform>`** is the player's number, as written on their shirt, specified by one or two digits.
- `<comma>`, `<space>`, and `<dash>` represent the characters `','`, `' '`, and `'-'`.

Write a piped sequence of commands for each of the tasks specified in 1-7 below. Specify each piped sequence in a separate line in a solution file named `ex2.1.sh`. The file should contain exactly seven lines, each containing a single command or a single piped sequence of commands.

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

2. Write a single command that takes the file `nba_roster.txt` and prints a sorted version of this file into the file `nba_roster_sorted.txt`. Lines should be sorted based on the player's position, and players with the same position should be sorted based on their last name in reverse alphabetical order (including the comma).

   Validation: File `nba_roster_sorted.txt` should contain the same number of lines as the original file, and its first four lines should be:
   ```
   Robinson, M- C 23
   Noel, N- C 3
   Gibson, T- C 67
   Toppin, O- PF 1
   ```

3. Write a piped sequence of commands that appends to the file `nba_roster_sorted.txt` information about the number of lines, words, and characters (bytes) in the original file (`nba_roster.txt`). This operation should add a single line at the end of `nba_roster_sorted.txt` with three numbers. There should be single asterisk character (`'*'`) before each of the three numbers, and other than asterisks and numbers there should be no other character.

   Validation: The line added to `nba_roster_sorted.txt` should be:
   ```
   *9*36*150
   ```

4. Write a piped sequence of commands that prints the second letter of the two-letter position code for the players on lines 4 and 5 of the file `nba_roster.txt`. If the positions have only one character, then nothing should be printed for that line.

   Validation: The following should be printed to the screen:
   ```
   G
   F
   ```

5. Write a piped sequence of commands that takes the file `nba_roster.txt` and replaces all dashes with commas and all commas with dashes. The output should be placed into the file `nba_roster_reformatted.txt`.

   Validation: `nba_roster_reformatted.txt` should begin with:
   ```
   Sims- J, PF 45
   ```

6. Write a piped sequence of commands that generates a sorted list of all player positions with the counts associated with these positions. The different positions should be printed from the most frequent one to the least frequent one. The list should be put into the file `nba_positions.txt`.

   Validation: File `nba_positions.txt` should contain the following text:

   ```
    4 PF
    3 C
    1 SF
    1 SG
   ```

   Note that there is a single space character before and after every number in the file.

7. Write a piped sequence of commands that reassigns players with new shirt numbers based on the following procedure. The first player mentioned in the file (Sims) is assigned with the smallest number (1), the next player (Samanic) is assigned the second-to-smallest number (3), and so on, until the last player mentioned in the file (Robinson) is assigned with the largest number (91). Print the new list of players with their numbers in the file `nba_roster_reassigned.txt`. Each line in this file should contain a number, followed by a colon (`':'`) and the player's name (last name and initial). See example below.

   Validation: The first three lines in file `nba_roster_reassigned` should be:

   ```
   1:Sims, J
   3:Samanic, L
   13:Noel, N
   ```

   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 seven lines (use `wc`) and follow the specific validation steps specified for each item in 2-7.
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`):
   ```
   nba_positions.txt
   nba_roster_reassigned.txt
   nba_roster_reformatted.txt
   nba_roster_sorted.txt
   ```
3. The script should **print <u>only</u> the following output** to the screen:
   ```
   G
   F
   ```
4. The script should not **print any error message**.
5. Run the script from **various locations** (not just `~/exercises/ex2/`) to ensure that it does not contain unwanted absolute paths.
6. 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 (`nba_roster.txt`) as well as slightly modified versions to make sure that your code follows all requirements.

Problem 2:

This question involves parsing a story file called `story.txt`, containing the text of *The Jungle Book* by Rudyard Kipling taken from the Gutenberg project (https://www.gutenberg.org/files/236/236-0.txt). The file contains free text in no particular format. Write a piped sequence of commands for each of the tasks specified in 1-4 below. Specify each piped sequence in a separate line in a solution file named `ex2.2.sh`. The file should contain exactly four 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 prints all numbers mentioned in `story.txt` to the screen. A number in this context is defined as a consecutive sequence of digits (0-9). This sequence of digits does not have to be immediately flanked by white spaces. For example, the following line: “There were 34 people on the 3rd floor at 10:30pm” contains four numbers (34, 3, 10, and 30). Numbers should be printed in increasing value order. Additionally, you should void printing empty lines, and for this purpose you may assume that the file does not start with a number.

   Hint: use `tr` with option `–c` (see guide with Linux commands for information).

   Validation: you should get three numbers (in separate lines): 39 39 1842.

3. Write a piped sequence of commands that prints to the screen the letter that is most frequently used in `story.txt` in upper-case as a first letter in a word. For this purpose, count each appearance of an upper-case letter that immediately follows a white space (or is the first character in the file). For example, in the sentence “My name is Mark SMITH”, the letter M appears twice in upper-case as a first letter in the word (The ‘M’ in Smith is not counted). Your piped sequence of commands should print a single character to the screen (the letter T in the case of the specific file `story.txt`).

4. 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`.

   - A separating character between words for this purpose is defined as either a white space (space, tab, newline, etc.), or all punctuation marks other than the single quote ('). For example, the sentence “I'm a hard-working person!!” contains exactly five words (I'm , a , hard , working , person).
   - Words should be printed in lower-case letters in alphabetical order.
   - The same word should not be printed more than once.

   Note: punctuation marks are indicated by the `[:punct:]` tag. Treating all of them other than the single quote as separating characters requires a tricky application of `tr`. Figuring out this trick might take you a while …

   Validation: file `story-words.txt` should contain exactly 4691 lines, and its first and last four lines should be:

```
1842
39th
a
aaa
. . .
youth
you've
zaharrof

zaharrof's
```

   A copy of the file you should expect to get is given in `/share/ex_data/ex2/story-words-expected.txt`. Note that none of the lines should be empty. Use the `diff` command (which we see in Recitation #3) to compare your file `story-words.txt` to the expected file.

Final validation and testing for Problem 2:

1. Verify that file `ex2.2.sh` has exactly four lines (use `wc`) and follow the specific validation steps specified for each item in 2-4.
2. Execute the commands in file `ex2.2.sh` as scripts (using `source`) and confirm that it creates a single file `story-words.txt` in the current directory.
3. The script should **print the following output** to the screen (and only this output):

```
39
39
1842
T
```

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 all 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 (**27/3 @ 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.