Programming Paradigms Erlang Homework

Johann Gamper Radityo Eko Prasojo Thomas Tschager

2nd Semester 2018/19

The Government of the City of Erlanger, Kentucky, US are conducting a census of the city population. However, there was no guideline on how to collect and store the data and so the government ended up with a single big unorganized file containing the census data. Each line in the file contains information about the age, gender, marital status, occupation, and yearly income of a citizen, separated by a space. However, the order of the information in each line is random. For example, the file may contain lines such as:

Doctor 35 Married $300,000 Female
16 $6,000 Student Male Single

You are tasked by the Erlanger government to realize a concurrent Erlang program to sort and to summarize the data. The program consists of four types of processes:

1. One **reader**: it reads the file line by line, and dispatches each line to one of the multiply available scanners; in addition, it is the main orchestrator of the entire program, that is, it spawns the other processes and coordinates their termination.
2. Many **scanners**: each scanner receives lines of text; for each received line, the scanner decomposes the line into the fine grained data (age, gender, marital status, occupation, and yearly income), sends the list of data into the sorter, and sends each data to the summarizer.
3. One **sorter**: it receives lists of data from the various scanners and keeps track of the overall lists abd while doing so:
   (a) arrange each list of data into the order of age, gender, marital status, occupation, and yearly income from left to right. For example, the above lines should become:

   35 Female Married Doctor $300,000
   16 Male Single Student $6,000

   (b) sort the lists of data by the age -> marital status -> occupation -> yearly income -> gender, all in ascending order (alphabetically for texts). For example, the above lines should be sorted into:

   16 Male Single Student $6,000
   35 Female Married Doctor $300,000

   Finally, upon completion it prints the content of the ordered list.

4. One **summarizer**: it receives data (a piece of information from the list of data, e.g. a gender or an occupation) from the various scanners, and keeps track of the census data under the following categorization:
   (a) age in the interval of 0-6, 7-12, 13-18, 19-24, 25-30, 31-45, 45-60, and 61+,
   (b) gender in three categories: Male, Female, and Other,
   (c) marital status in four categories: Single, Married, Divorced, and Widowed,
   (d) occupation under all categories found in the data file
   (e) yearly income in the interval of <$10,000, $10,000-$25,000, $25,001-$50,000, $50,001-$100,000, $100,001-$250,000, and >$250,000.

In particular, the reader is invoked as `word_count(FileName,N)`, where `FileName` is the name of the input file to be processes, and `N` is the number of scanners to be created (this determines the degree of concurrency in the overall program).

The flow of the reader is as follows:

1. The reader spawns a sorter and a summarizer.
2. The reader spawns `N` scanners.
3. The reader communicates the PID of the counter to each scanner.
4. The reader reads the file pointed by `FileName` line by line. It then dispatches each line to a scanner, with a rotating policy: the first line goes to the first scanner, the second line goes to the second scanner, . . . , the `N`-th line goes to the `N`-th scanner, the `N`+1-line goes to the first scanner, and so on.
5. When the end of file is reached, the reader informs each scanner that no more lines will be sent.
6. The reader waits the completion of all scanners.
7. The reader informs the sorter and the summarizer that all scanners have completed their execution.
8. The reader closes the file and exits.

The flow of each scanner is as follows:

1. The scanner waits for the PID of the counter.
2. The scanner cyclically waits for a message.
   (a) If the message is about a line of text, the scanner splits the line into a list of data, sends the list to the sorter, and sends each data to the counter.
   (b) If the message is about the end of file reached, the scanner informs the reader that it has completed its execution.

The flow of the sorter is as follows:

1. The sorter cyclically waits for a message.
   (a) If the message is about a list of data, the sorter arrange the data as explained above, and then keep it in a list while maintaining the order of the list also as explained above.
   (b) If the message is about the completion of all scanners, the sorter prints to standard output the list that it keeps.

The flow of the summarizer is as follows:

1. The summarizer cyclically waits for a message.
   (a) If the message is about a data, the summarizer increments a counter that corresponds to the data. For example, if the message contains a "Male", then the counter for gender "Male" should be incremented, and if the message contains a "Doctor", then the counter for occupation "Doctor" should be incremented, etc.
   (b) If the message is about the completion of all scanners, the summarizer prints to standard output the counters it keeps track.

Test the program with the file `data.txt` provided on the web. The result printed by the summarizer should have the following form:

```
Gender
"Female": 49198
"Male": 49129
"Other": 1673

Marital status:
"Married": 47060
"Single": 36702
"Divorced": 8154
"Widowed": 8084

Occupation:
"Professor": 1014
"Pilot": 1065
"Artist": 621
"Chemist": 1111
"Musician": 604
"Janitor": 705
"Firefighter": 1175
"Lawyer": 1122
...

Age:
0-6: 6028
7-12: 6024
13-18: 5874
...

Yearly income:
<$10,000: 47882
$10,000 - $25,000: 10565
...
```