**14057 Savev David**

# Erlang assignment – PROGRAMMING PARADIGMS

My solution is contained inside three different files, named „reader.erl“, „scanner.erl“ and „counter.erl“.

More precisely, the Reader invokes the Scanner processes and a Counter process and sends ciclically to each Scanner a line of text (only whether the file **exists** and is not **empty**).

Inside *Reader*, I have implemented:

- **word_count(FileName, N)** -> This is the „main“ function of the distribuited system; it is enough to call reader:word_count('filename.txt', <nrofscanners>) to launch the execution.
  This function pays also attention to inexisting files, empty files or nr. of scanner <= 0, in that case the execution is not started.
- **createScanner(N, Counter, Registered)**
  This function recursively creates N scanners (how much we need) and puts their id's inside Registered list.
- **sendText(ListScanners, CopyListScanners, File, RowsCount)**
  This function takes line by line from the File, and sends it ciclically (recursively) to the list of registered scanners.
  CopyListScanners is used when we need to restart the cicle: when ListScanners == [], since we have iterated all the elements, it is enough to copy ListScanners = CopyListScanners.
- **waitAllScanners(List, Length)**
  Since we have obtained before the number of lines of the File, in this function we recursively wait the same number of responses by all the Scanners.
- **notifyAll(ListScanners, Counter)**
  This function send a „Finish“ notification to all the Scanners and the Counter that it has finished his functions -> this function is called after waitAllScanners().

The *Scanner* process, is used in order to split the sentence received by the Reader in words and send them to the Counter:

- **receiveCounter()**
  This function is called only once when the Scanner is instantiated: it waits the PID of Counter and after that calls receiveLine(Counter) function.
- **receiveLine(Counter)**
  receiveLine is called recursively until a „Finish“ is not received; this function receives a line of text from the Reader, splits it and puts it inside a list -> after that, the Scanner sends a message of feedback to the Reader and calls the function sendWords(List, Counter).
- **sendWords(List, Counter)**
  This simple function, iterates the list of Words, and sends each Word to the Counter.

The *Counter* process, after receiving all the words from the Scanner processes, inserts them all inside a list and counts the occurence of each of them:

- **receiveWord(Words)**
  receiveWords is called recursively since a „Finish“ is not received; this function receives Words and puts them inside a List.

- **countOcc(List, StillInserted, Count)**
  When a „Finish“ is received, this function is called; I have used here the original list of words called List, which is actually iterated, the list StillInserted that stores only once an encountered Word, and list Count which stores the number of occurences of each Word present in StillInserted.
- **incrementCount(Count, newCount, Position)**
  This function simply gives back a new count-list with the proper Position incremented by 1.
- **printResult(StillInserted, Count)**
  This is the final function of the distribuited system and should print out the occurences of each Word.

  **P.S.** In my implementation, I'm also printing the *sequence* of the sentences that are received by the Scanners: in some cases, the order in which those are printed in the Console is WRONG (maybe because some Scanner is running slower and prints the received row after some time), but you can denote that the lines are assigned in correct cyclic order.