Assignment 2 – String Vector in C

Computer Architecture Fall 2015

The Data Structure Vector

The Idea of this project is the implementation of the data structure "vector". A vector behaves to the user like an array, thus you can access the elements in a vector by providing the vector an index of an element. Compared to an array, the vector is a dynamic data structure, thus is grows and shrinks during runtime as the number of elements to be stored is not known a priori.

As you are already familiar with linked lists, you should implement the Vector using a **double linked list** in order to store the elements. Note that this is not the most effective way of implementing a Vector (normally, you would use arrays as storage), but for the sake of learning C and handling pointers, we utilize a double linked list. As we are programming in C, you have to take care of allocating memory and, equally important, freeing unused memory again!

Specific Tasks

1. Implement a vector as described above using the provided header file `vector.h`. Name your source file `vector.c`.
2. Write a program utilizing this vector. The Program should read a filename as the only command line argument. This file contains Strings, each separated by a new line (meaning the String may contain spaces and tabs as well). Read the entire file and store each String as an element in your data structure. Name your program `string_reader.c` and the according header file `string_reader.h`.
3. After reading the file, sort this vector lexicographically. You may use the qsort function of the C std library. For that, you have to pass an array to the method.
4. The last part is to output the sorted strings to std-out.
5. Write a makefile (you will learn that in the lectures) providing three targets: **make** (compile and create an executable file), **make clean** (remove all generated files), **make test** (run the test cases).
6. Provide input files containing test cases. You have to think about meaningful test cases, particularly about special cases in order to check your software.

Please be sure you follow exactly the above instructions and implement the features accordingly.

Grading, Submission & Deadlines

Grading

This project will be evaluated with pass/fail marks. **You are allowed to work in team of up to 3 students**. In order to pass, you must provide the following items:

Zip-File containing:

- A README file with your names and a description of your test datasets:
  - What is the content of the file (e.g., contains strings with many special characters)
  - Why have you chosen that file (e.g., making sure that your software can successfully handle non-basic characters)
- All source code and header files of your program in the subfolder `src/`
- All test datasets in the folder `test/`

Your source code:

- Must be able to be compiled as described above and run on the terminal machines
- Has to be commented and you have to explain your steps and decisions in the code. This is important, as you don't have to write a report this time!

Additional Remarks:

- Follow the folder structure and the filename conventions!
- Attend the lab classes as we will work on a lot of useful stuff you will need for the project.

Submission

Please upload the zip-File with the contents exactly as described above to Blackboard using the according SDU assignment tool. We will not accept incomplete zip-Files or files provided by different means than the blackboard system.

Deadline

Upload your zip not later than

Sunday, 13th of December, 23:59:59 Central European Time (UTC+1)

(Be aware: There will be NO extensions to this deadline. You have to hand it in completely and in time! You certainly should also hand in even though your program might not run perfectly!)