Assignment 2 – String Vector in C

Computer Architecture Fall 2017

The Data Structure Vector

The Idea of this project is the implementation of the data structure "vector" or "array list". 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 known a priori.

A very common implementation (the one you should follow) is using an array in order to store the elements of your vector. Normally, you initialize the vector with an array of predefined size (e.g., 10) and use it as storage. During the curse of the program, you might encounter that the initial array will be too small at some point and you need to expand the underlying fixed-length array. This is done by creating an array twice as big as the current array. This strategy is done for strategic reasons, because whenever you create a new array, you have to copy all data from the old array to the new array. If you would increase the size of the array only by 1, every new insertion would require the copying of all data. With doubling the size of the array, you archive the insertion operation to be performed in constant time (amortized; meaning, when you have to double the array size from n to $2n$, the insertion is very expensive but buys you n cheap insertions. As we are programming in C, you have to take care to allocate the memory and equally important, to free 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.
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 a 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. In order to pass, you must provide the following items:

Zip-File containing:

- A README file containing your name 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:

- You are allowed to work in teams of up to 5 students (please stay in the same teams as during the assembly project).
- Please specify all your team-members' names in the README file. One submission per team is enough.
- Follow the folder structure and the filename conventions!
- Attend the lab courses 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, 17th 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!)