Operating Systems, Fall 2017
Instructions for Databar Exercise 3: C programming

September 14, 2017

1 Introduction

In this third databar exercise you will write a C program to train on c programming. The exercise is more open than previous exercises.

You will report on this exercise as part of the first report. You will there write half a page about the design of your program and your experiences. You will also have to attach your source code. NB: You will write more in the first report than the outcome of this exercise.

Read through this document in entirely before starting working on the exercises!

2 Learning objectives

During this exercise you will be working towards the following learning objectives:

- You can explain how the compiler, assembler and linker are used to create executables.
- You can apply standard programming methodologies and tools such as test-driven development, build systems, debuggers.
- You can explain the role of each component of a compilation toolchain used in system programming and how the components interact.
- Given reference material, you can implement C programs with pointers, pointer arithmetic, arrays, structs, memory management and low level I/O.
- You can explain in your own words how static, automatic and dynamic memory management can be used to handle memory in C programs.

- You can explain in your own words what a C pointer is, its relation to arrays in C and the result of pointer arithmetic.
- You can explain in your own words how memory leaks can occur in systems with dynamic memory management and give examples of two approaches to prevent them.

3 Rules

DTU has a zero tolerance policy on cheating and plagiarism. This also extends to the reports and indeed all your work. To copy text passages or source code from someone else without clearly and properly citing your source is considered plagiarism. See the study hand book for further detail.

4 Reference material

During the work on the exercise you will program in C. Before starting working read the following document:

- ANSI C for Programmers on UNIX Systems by T. Love, `http://www-h.eng.cam.ac.uk/help/tpl/languages/C/teaching_C/`

  An introduction to C for programmers already knowing other languages, e.g. java.

During the work on the exercise you will use several tools. You might find the following manuals helpful and you should browse them:

- GNU Binutils `http://www.gnu.org/software/binutils/`

  This is a large collection of tools for manipulating different executable binary files.
- GCC `http://gcc.gnu.org/`

  GCC is a collection of compilers for many languages including Java, C and C++.
- GNU Make `http://www.gnu.org/software/make/manual/`

  make is a tool that can help you coordinate compilations of larger programs.
- GDB `https://sourceware.org/gdb/current/onlinedocs/gdb/`

  A debugger allowing you to step through the code as you are executing and print out values.

5 Setting up the environment

You use the VirtualBox image you set up the first week.

6 Working on the assignment

6.1 Commands

In this exercise you have to read commands from stdin and perform actions based on them. The commands assume a counter which is initialized to 0. Commands are single characters with the following meanings.

- 'a': Add the current counter value to the collection, then increment the counter.
- 'b': Do nothing except increment the counter.
- 'c': Remove the most recently added element from the collection and increment the counter.
- Anything else: Stop processing commands, print the collection in the order they were added, and exit.

Initially the counter will be set to 0 and the collection will be empty, as shown in figure 1.

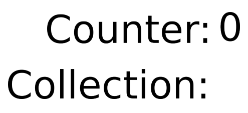

Figure 1: A newly initialized collection

As an example, the string "`abbbaq`" would produce the collection shown in figure 2. Note that we use the 'q' character to terminate command processing, but any character other than 'a', 'b' and 'c' would have the same effect.

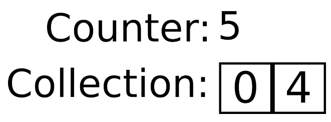

Figure 2: A collection generated by the string "`abbbaq`"

The string "`abcabcaq`" will produce the collection shown in figure 3.

You need to read a command from stdin, process the command, and repeat, using functions from the C library such as `printf` and `scanf`. You will need to create and maintain all necessary data structures yourself. How you implement the collection is

Counter: 7
Collection: $\boxed{6}$

Figure 3: A collection generated by the string "`abcabcaq`"

entirely up to you but keep in mind that you do not know the size of the collection at compile time. Some form of dynamic memory allocation will be required. There are a large number of different ways to accomplish this.

6.2 Printing

You are required to print out the contents of your collection to stdout.

When your program encounters a character other than 'a', 'b', or 'c', you should print your collection, in order, and then terminate by returning from main. The format for printing the collection is as a comma delimited list of integers. The output of the three previous examples, in order, are:

```
<blank>
0,4
6
```

Good Luck!