Digital Systems
SCC150
PRACTICAL WEEK 20

Overview

In the lecture (Lecture Friday Week 19) we have discussed string operations (*strlen*, *strcmp*, ...) as provided by the c library *string.h*. The aim of this practical is to create your own implementation of a string operations library which we call *mystring.h*. *mysting.h* should provide the functions *mystrlen, mystrcmp, mystrncmp, mystrcpy, mystrncpy* and *mystrdup*. The functions should be equivalents to the *string.h* library functions *strlen, strcmp, strncmp, strcpy, strncpy* and *strdup*. The prefix *my* is used to avoid name space clashes with the standard library.
To learn about input parameters, behavior and expected return values of the functions you can use the linux man pages (e.g. type *man strlen* to obtain a description of function *strlen*). The lecture slides from Friday Week 19 provide some implementation examples of string functions which may be used as a starting point. Furthermore, the web provides a vast number of implementation examples which can be used as inspiration.

Provided Files

mystrtest.c For testing purposes the file *mystrtest.c* is provided. This file contains a simple c program which makes use of the functions that the library should implement. At the start of the file you find the statement

```
#define MYSTRING 1
```

When this line is present the test program uses *mystring.h*. If the line is deleted or commented out

```
//#define MYSTRING 1
```

the program uses *string.h*. Thus, by simply changing this line the test program switches between your implementation of string operations and the system libraries. Thus, testing is simplified and you can compare results of your function implementations against results obtained by the the system libraries. In *mystrtest.c* the library functions are addressed via macros (e.g. *STRLEN(...)*). Depending on the library used these macros are replaced at compile time by the appropriate function calls (e.g. *STRLEN(...)* becomes *mystrlen(...)* when we use our implementation and *strlen(...)* when we use the system libraries.

mystring.h This is the include file a program such as *mystrtest.c* is using. This header file provides the function prototypes of the functions *mystrlen, mystrcmp, mystrncmp, mystrcpy, mystrncpy* and *mystrdup*.

mystring.c This file contains the implementations of the functions *mystrlen, mystrcmp, mystrncmp, mystrcpy, mystrncpy* and *mystrdup*. These functions are currently only placeholders and it is your task to fill these functions with life.

Environment

We will be using VirtualBox as in the Practical of Week 8 to run Linux within MSWindows. VirtualBox is already installed on your machine. The virtual machine image is installed in all lab computers but before you can run Linux, you need to configure the Linux virtual machine as follows:

1. Download the `build-nat-vm.vbs` configuration script. To do this, open a Internet Explorer browser, go to `http://andrew-scott.eu/docs/lancs-vm` and then right click the `build-nat-vm.vbs` link, select Save target as..., and download it to your Desktop.

2. Double click `build-nat-vm.vbs` that you have just downloaded to your Desktop. This script will create a virtual machine appropriately configured for you and will create a file called `lancs-vm` in your Desktop.

3. Double click `lancs-vm` to start the virtual machine. You should be automatically logged on as the user `lancs`.

Compiling

To compile the test program you have to compile *mystrtest.c* and *mystring.c* and assemble both objects into a single executable file. All compilation and linking steps can be carried out with a single command line:

```
gcc -g mystring.c mystrtest.c -o mystrtest
```

The program can then be run by using the command

```
./mystrtest
```

or

```
gdb ./mystrtest
```

to run the program in the gdb debugging environment.

Debugging

You should use gdb to verify that string operations such as *mystrncpy* modify strings in memory the same way as the functions provided by the system libraries.