CPE 100 Introduction to Computer Programming
International Sections – August 2020
Laboratory Exercise 9

Objective

This lab gives you practice creating applications that consist of more than one source file, using functions written by someone else. It will also give you more practice with pointer-based function arguments.

Instructions

Summary

I have written some functions that manipulate dates, which are declared in the header file **dateFunctions.h** and implemented in the file **dateFunctions.c**. I have also given you a simple program, **datecalc.c**, which provides an example of how to use these functions.

You will write a new program called **checkFuture.c**, that will use the same functions. This program will run in a loop. Each time through the loop, it will do the following:

- Ask the user for a date in the form **dd/mm/yyyy**
- Make sure the date is valid. If it is not, print an error message.
- If the date is valid, determine whether the date is in the future or not (later than today) and print the result.

Continue looping until the user enters the string “BYE” for the date.

Details

1. Go to the course website and download my code: **dateFunctions.h, dateFunctions.c and datecalc.c** and the file **Makefile**. Open or type the header file (**.h** file) and look at its contents. You will see that it defines a number of symbols to reflect error status values. It also declares several functions that work with dates.

The code that implements these functions is located in **dateFunctions.c,** but *you do not need to look at this code*! The header file explains how to use the functions - what they do, what you should pass to them, and what they return. (You should **not** change anything in either **dateFunctions.h** or **dateFunctions.c**).

2. Create a program called **checkFuture.c**. This program should do the following:

a) Loop, asking for a date in the form ***dd/mm/yyyy***.
b) If the user enters “BYE”, exit the loop.
c) Call the function **checkDate()** in **dateFunctions** to see if the date is in valid format. Note that you will need to pass pointers to day, month and year variables that you declare in your main. If the date you read is not valid, print an error message.
d) If the date is valid, call the functions **dateToday()** and **dateCompare()** to figure out whether the date is in the future or the past. Print the results. (See example below.)

Here is a sample run of the program:

```
testuser@testuser-ThinkPad-L412:~/cpe100/Lab10$ ./checkFuture
Enter date to check (dd/mm/yyyy): 12/12/2013
12/12/2013 is in the past
Enter date to check (dd/mm/yyyy): 10/01/1988
10/01/1988 is in the past
Enter date to check (dd/mm/yyyy): 07/08/2033
07/08/2033 is in the future
Enter date to check (dd/mm/yyyy): hhhhhh
hhhhhh is not a valid date
```

```
Enter date to check (dd/mm/yyyy): 13/19/2000
13/19/2000 is not a valid date
Enter date to check (dd/mm/yyyy): BYE
```

Note that in order to be able to use my functions, you must include my header file in your program. To do this, add the following line after the include commands for the system libraries. Notice that for local header files, you use quotes, not angle brackets.

```
#include "dateFunctions.h"
```

3. Compile and link your program.

```
gcc -o checkFuture checkFuture.c dateFunctions.c
```

4. Run your program on the following list of dates:

```
22/11/2019
31/02/2011
ajajkkklllja
10/01/2014
01/02/1890
02/03/1920
25/01/2100
20/12/2020
2/4/2014
```

The output from your program might look like this:

```
22/11/2019 is in the future.
31/02/2011 is not a valid date.
ajajkkkklllja is not a valid date.
10/01/2014 is in the past.
01/01/1890 is not a valid date.
02/03/1920 is in the past.
25/01/2100 is in the future
20/12/2020 is in the future
2/4/2014 is not a valid date.
```

If you want, you can use the information in the returned status code to print more detailed error messages explaining what is wrong. For an example, see **datecalc.c**.

5. Once you are satisfied that **checkFuture.c** is working correctly, modify the **Makefile** so that it will build your program **checkFuture** as well as building the **datecalc** executable which it does now.

6. Delete any object files and executable files by typing:

```
make clean
```

Notice that the rule with the target "clean" has no dependents. Use the **ls** command to check that there are no .o or executable files in the directory any more.

7. Type:

```
make
```
Or
```
make -f  Makefile
```

The system should compile and link both **datecalc** and **checkFuture**. If this does not happen, you probably have not modified the Makefile correctly.

8. Delete the file **datecalc** (the executable file). Run **make** again. What happens?

9. Edit the file **dateFunctions.c**. Add one blank line, then save the file. Now run **make**. What happens? Do you understand why?

Upload your C file **checkFuture.c** and the modified **Makefile** to the Windu server, using the usual link on the website.