CMPE 230

Project-2 Report

Student: Alperen Değirmenci – 2017400255

Instructor: Can Özturan

1- Description and the Solution to the Problem

In this project the task is to write a makefile generator tool called **makefilegenerator** in the Python language. Makefile generator is supposed to generate makefiles for given main, source and header files written in C language. Our program should traverse the directories looking for **.h** and **.c** files. After finding them, it should examine the content of the files and generate a makefile for the functions and header files used in the **main** code.

My program is based on the **traverse.py** code given by instructor Can Öztüran. In the first place, **traverse.py** code traverses all files in a given directory and its sub-directories looking for any type of file and printing their names to the screen. My program is based on this code.

My generator tool starts traversing the directories looking for **".c"** and **".h"** files. Each time it finds a ".c" or ".h" file, it creates an object for the file. Objects are instances of two classes, one is a class for .h files, other one is for .c files. Every time a **".c"** file is found, program checks whether it has **"stdio.h"** line in it or not, if **"stdio.h"** is present in a .c file, program creates the **mainC** object for it. This file is the **main.c** file.

Then program searches the **main.c** file line by line for searching **""** (double quote) pattern, this provides recognizing lines like **#include "addTwoInts.h"** so that we can store include files names in a string.

For the other **.c** and **.h** files, **hClass** and **cClass** are names of the classes. They have fields like ***filepath, filename, onlyName, usedInMain, incInMain*** which happen to be some string and boolean variables. After recognizing types of file while traversing files, these objects are created with defining these fields from file's **directory** and **original file name.** And they are added to lists storing **.h** and **.c** objects.

After shaping the **mainC** and other **.h** and **.c** objects, it's now checking for the warnings and errors. In a for loop traversing the list which stores **.h** objects, our program compares the names of the **.h** files and the **included header files' strings.** If they match, field of the **.h** object **incInMain** which stands for *included in main* is turned to true. Then, they are compared with the lines in the **mainC** to see if they are used in main. If program finds the pattern **e.g. "addTwoInt("** it understands the function from the header file is used in the main file.

Then for the error checking, it compares the included header files and the list objects of the **.h** list, if an included file is not present in the **.h** list, program gives the error message and terminates the makefile generator.

After this step, the program checks the **.h** list one by one again, to check which **header** files are included and their **functions** are used in the **main** file. If a function is **used** in the main file but its **header** file is not **included** in the main file, it again gives an **error** message and terminates the makefile generator. Then, if a **header** file is included in main and its function is **not used** in main **or** a **header file** is present, but it's **not included** in main and its function is **not used** in main it gives a warning with the name of the header file.

Last step is if there's no error, to create the makefile with relevant files. For the header files both included and their function is used in **main** file, make file is generated.

First line of the makefile is in the format of **main#.exe:** where # is the test case's number. Then, in order **main.o** and **headerfile1.o, headerfile2.o …** lines are added to the makefile.

2- How I Run the Code

- I wrote the code in a single file ***makefilegenerator.py***
- I've tried the code in terminal with giving the directory of the test case as the argument
- Like: **makefilegenerator.py \cmpe230fall2017hw2_Testcases_updated\tc1u**
- And makefile is generated in the given directory.
- Tried the makefile in the given directory with the **make** command.
- For the warnings, warning message is shown on the terminal
- For the errors, error message is shown on the terminal and makefilegenerator.py terminates.
- Comments are included in the file.