Task 4 – Automatic checking and grading of C programs

Develop an application for automatically testing and grading C programs. The desired functionality is described below. Save the main application program in a file named project4.c .

Arguments

The main application program accepts as arguments:

(a) The name of a file, in the form <progname>.c , which contains its code program to be graded.

(b) The name of a text file <progname>.args which contains the arguments that are required to run the program <progname>.c, on one line, with ' ' character (space) between successive arguments. You cannot make assumptions about the length of the line or the number of arguments.

(c) The name of a text file <progname>.in which contains the data it will accept program <progname>.c from conventional input.

(d) The name of a text file <progname>.out containing the expected output that must produce the program <progname>.c when run with the arguments that were specified in the file <progname>.args and the input specified in the <progname>.in file.

Calculation of final grade and exit

The final grade that will be assigned by your application to the program <progname>.c results from the sum of the individual points in the following categories:

(a) Compilation: -100 if compilation failed, -10 if compilation succeeded but produced a warning, 0 otherwise.

(b) Similarity of output to expected: the integer quotient <number of similarbytes> * 100 / max(<number of bytes of expected output>, <number of bytes of output>).

(c) Penalty due to incorrect memory access : -15 if the program terminated due to a SIGSEGV or SIGABRT or SIGBUS signal.

Your application finally prints the following messages to the conventional output:

Compilation: X

Output: Y

Memory access: Z

Total: T

with character '\n' ~~before and~~ after each message. The prices X, Y, Z are the degrees that were calculated in the corresponding individual categories as described in the previous paragraph, while T is equal to max(X+Y+Z, 0). Messages are printed even if the corresponding points are zero.

Mode

If the correct number of arguments is not given based on the description above, the application prints an appropriate error message to the error output and terminates returning 2 .

If the correct number of arguments is given, then:

(a) The main application program creates a new process P1 and redirects its error output to a file named <programname>.err . THEP1 produces the executable for program <progname>.c by running the command:

gcc -Wall <progname>.c -o <progname>

The main program waits for P1 to terminate and then checks to see if the file <progname>.err includes (at least once) as a single word the string "error: " which means the compilation failed or, if not, the string "warning:
" which means that the program <progname>.c compiles successfully but with warnings.

(b) If the compilation is successful, the main program creates two new processes P2 and P3 and an anonymous pipeline A, and redirects P2's conventional input to the text file <progname>.in , the conventional output of P2 on the write end of conductor A and the conventional input of P3 on the read end of conductor A.

Process P2 executes program <progname> with the arguments specified in file <progname>.args.

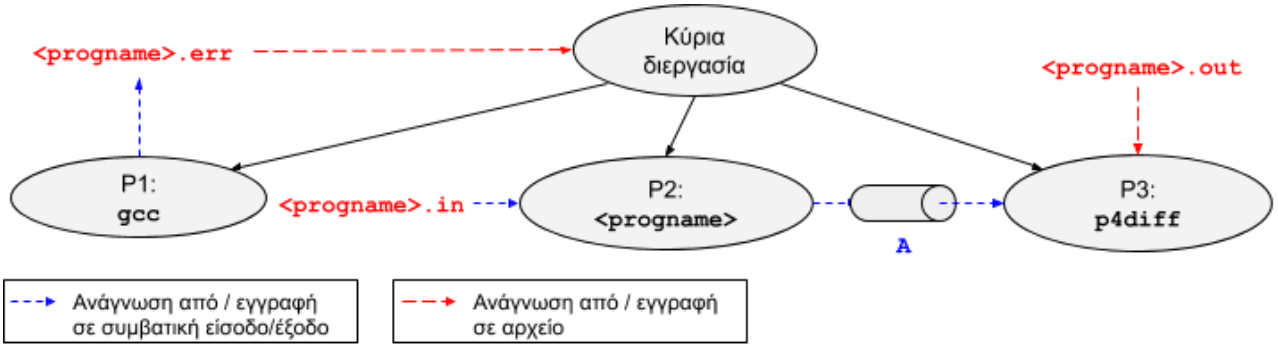
Process P3 executes program p4diff with the filename argument <progname>.out . The p4diff program must be implemented by you so that compares one byte the data it reads from its conventional input with the contents of file <progname>.out , calculate the similarity percentage like described in the previous section and return it as a result via return.

Save the source code of the programp4diff in a file named p4diff.c.

The main program waits for P2 to terminate and checks the termination reason. If P2 terminates due to a SIGSEGV or SIGABRT or SIGBUS signal, the incorrect memory access penalty is calculated as described in the previous section (otherwise the program is assumed to be <progname> ran without a problem so there is no penalty). For for simplicity assume that the program <progname> doesn't have any endless repetition and so process P2 always terminates (either normally or due to a memory access error).

Next, the main program waits for the termination of P3, and receives through its exit status the grade calculated by P3.

(c) Finally, it prints the score messages as described in the previous section.



Requirements/Implementation Assumptions

- Provide appropriate Makefile for your application.
- It is forbidden to use global or static variables, goto and its use command system .
- Processes P2 and P3 run concurrently.
- The program p4diff reads data from file and pipe toblocks of 64 bytes.
- You can assume that the file name <programname>.c he will always have her correct ending (.c) as well as that the file names <progname>.args, <progname>.in and <progname>.out will have the correct format and ending.

Important dates

Tutorial: *Tuesday 19/5/2020* (during class)

Submission deadline: *Sunday 7/6/2020, 23:59*

Packing and shipping work

1. Build a directory named project4submit
2. Copy the project4.c, p4diff.c, Makefile in the project4submit directory
3. Pack and compress the directory project4submit by right clicking and choosing Compress here as .tar.gz
4. Login to autolab, and select project4 of the courseECE116-S20
5. Accept the academic integrity message, then click SUBMIT.
6. In the window that appears locate and select the file project4submit.tar.gz that you built to upload toautolab.
7. After one minute, refresh the page to see your grade on the individual tests.

For those who are in the mood

- Change the main program so that instead of the arguments <progname>.args, <progname>.in , <progname>.out takes the name of a directory as an argument within which are subdirectories, one for each control. Each subdirectory includes the following text files:
  - test.args with the arguments required for this test.
  - test.in with the data that is the conventional input for this test.
  - test.out with the expected program output for this test.

Adjust accordingly the way the final grade is calculated, assuming that all tests are equal and the total sum of points (for the output category) is 100.

- Change the main program so that it takes as an additional argument the maximum time in seconds (real time) that P2 is allowed to run. If P2 has not terminated after so many seconds of execution, the main program sends it a SIGKILL signal and appends to its output messages " Timeout: t sec " where t is the time.