6th International Conference on Applied Informatics
Eger, Hungary, January 27–31, 2004.

An Evaluating Tool for Programming Contests

Márk Kósa, János Pánovics, Lénárd Gunda

Institute of Informatics, University of Debrecen
email: {mkosa | panovics}@inf.unideb.hu, leonardo@gizi.dote.hu

Abstract

Students of the University of Debrecen majoring in informatics have been participating in regional ACM international collegiate programming contests since 1995. In earlier times arrangement of the local rounds was difficult because we had to check the contestants' submissions by hand. Beyond the discomfort, this hindered the efficient work of the jury and involved a number of possibilities of making mistakes.

The Programming Contest Result Manager (PCRM) program developed in the past two years gives the solution to the above problems. The program automates the evaluation of submissions and provides both the jury and the contestant with a user interface. This application can help the jury not only in ACM type but also in other kinds of practical programming contests.

1. Introduction

In this paper we will introduce you to the Programming Contest Result Manager (PCRM) program. We will discuss what the program is good for, even if the name may seem obvious.

PCRM is a program that can be used to help judging a programming contest, where the contestants must solve problems on a computer, and then send the solution to the online jury. It can also be used to judge an offline competition as well. The PCRM program can automatically check the sent solutions – it compiles them, runs them, and checks the output they give. It can work with programs that produce an output file and also with programs that read from the standard input and write to the standard output.

PCRM also includes a POP3 client program that can accept incoming solutions from the contestants via email, and can fully automate the entire judging process.

2. About programming contests

What is a programming contest anyway? What kind of contests can PCRM judge?

During a programming contest, contestants (which can also be teams) compete against each other by solving problems. There are several problems in a competition. For each problem, the contestants create a solution which is a source file written in a programming language. This source file, when compiled, generates a program, which then can read input data from either the standard input or from a file. It writes the generated output either to the standard output or to a file.

When the jury receives the source file from a contestant for a problem, it compiles it with the appropriate compiler and runs the program with test cases. For each problem, there is one or maybe more (but at least one) test case file(s), which the program must process, and generate a correct output. An output is correct, if it is the same as a pre-generated one, or if a program that verifies outputs find that the output is correct.

When using PCRM, there are test case files for each problem. However, each physical file can of course contain more real test cases - this is the case in ACM mode, where all test cases must actually be in one file. This usually means that the input is built so that it can contain several test cases following one another.

PCRM helps you automating the judging process by receiving the solutions (automatically), compiling them, running them, evaluating the result, and keeping a track of events and generating (or showing) the complete result.

2.1. Evaluation modes

The program can work in one of three modes:

- *Problem based:* In this mode, the only measure is the number of problems solved. The individual test cases do not count, i.e. a problem is only solved if all test cases are solved correctly. You can have as many problems as you wish and each problem can have as many test cases as needed.

- *Score based:* In this scoring mode, the winner is also determined by the number of correctly solved problems. Every test case file that is evaluated is equal to a score of 1 (or more, if a solution checking program is used. In this case, the solution checking program determines the score for a test case file). That is, if we have 5 problems and 3 test case files each, the maximum score is 15 and the minimum score is 0. You can have as many problems as you wish and each problem can have as many test cases as needed.

- *ACM based:* PCRM was originally created for ACM style competitions. Here, the contestant with the most correctly solved problems wins. If there are two or more contestants with the same number of solutions, a score is calculated, and the contestant with the less score wins. The score is calculated as follows: For each problem, the contestant gets as many points as many minutes passed

since the beginning of the competition. For every unsuccessful entry (compile error, run time error, wrong answer, etc.) he also gets penalty points. But penalty is only given, if the problem is accepted in the end. So, if I sent in problem A after 80 minutes, got a wrong answer, sent it in after 84 minutes again (I am fast at finding the error), then again at 92 minutes, which is finally accepted, I will get $92 + 2 \times$ penalty points (assuming penalty is 20, that sums up to 132). If someone else sent in a correct solution after me (say at 118 minutes in the competition), that contestant still beats me. In ACM mode, there can only be one test case file for each problem. This does not mean one test case because these problems usually have a test case format that many test cases can actually be put into a single file. Because of the way the score is calculated and maintained, it is necessary to have this restriction. Even if you specify more test cases, PCRM will only work with 1.

2.2. Jury responses

During an ACM contest, when a solution is sent to the jury, a result is returned after some time to the contestants. Based on this results the contestant has to decide what to do next.

In practical terms, this usually means that the contestant, after sending in the source file, can expect a result on some webpage, which will contain the time of entry, a symbol of the program he sent in and the response he got.

Here is a list and an explanation of the responses that PCRM can return:

- *Accepted:* This is what all contestants want. It means the program passed all tests and all restrictions and was accepted.

- *Compile error:* If the compiler returns an error, this message is sent back to the contestant.

- *Runtime error:* This means that the program has not terminated successfully. This probably means a segmentation fault, access violation, exception or something like that. Please make sure the programs return 0 at the end, because otherwise it might be treated as a runtime error.

- *Time limit exceeded:* For every problem, a time limit can be set. This is the maximum amount of time the program is allowed to run. When exceeded, the program will be terminated, and this error is returned.

- *Memory limit exceeded:* For every problem, a memory limit can be set. The PCRM program checks if this has been exceeded, and if it has, then it will return this error, possibly terminating the program. Note: under Windows a check is only made at the end of a successful run (when none of the above errors occurred), because Windows provides a facility for safely determining the peak memory usage at that time. Under Linux, a more manual and error prone process is used – Windows wins out in process information requests.

- *Wrong answer:* The program finished and produced an output, but the output is wrong.

3. Programming Contest Result Manager

In this section we will discuss the functional schema of the PCRM system and give a short description of some configuration parameters.

3.1. Functional schema

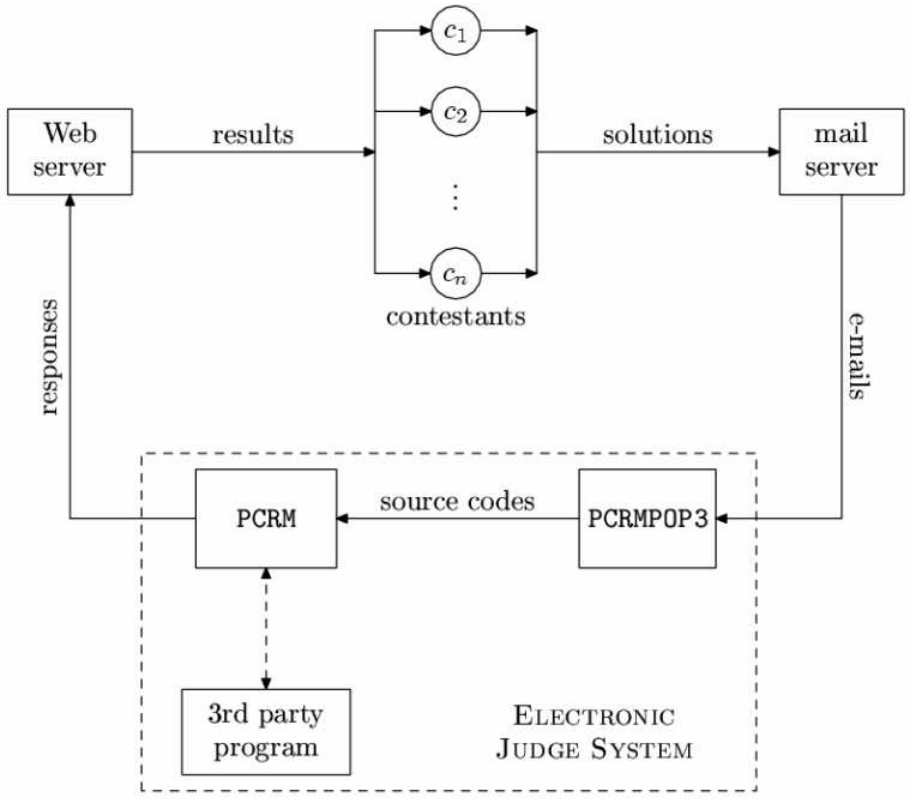

Figure 1: Functional schema of the system.

Figure 1 illustrates the functional schema of the system. In this figure PCRM-POP3 is a simple shell that uses the POP3 protocol to get messages from a server, store them, and then uses a MIME message parser to parse the message and store

the source code received in it. Afterwards it can optionally invoke PCRM to do the checking.

PCRM uses the settings in a configuration file to determine its working mode. It compiles the programs and runs them with the test files. Both the compilation and running command lines can be set for each of the languages, this way you can customize the way programs are handled, and it is also possible to use other programs to run the compiled program (as in the case of Java).

If the program reads input from standard input, PCRM redirects standard input for the program so that it receives the required test cases. Standard output can be redirected the same way, if output is written to it.

After the program terminates successfully within the given time limit, not exceeding the given memory limit, the output is compared to the pre-determined output, and if they match, the test case is accepted. PCRM is also able to invoke a third party program, that has to do the checking of the output and inform PCRM with its exit code, whether the solution is correct or not.

3.2. Configuring PCRM

The `pcrm.ini` file is the most important part of PCRM. It contains all the information needed to run a competition. This file can be divided to the following sections: global, HTML, POP3, compiler, run, problem and contestant.

3.2.1. Global section

The *[global]* section contains generic settings for the program and the environment.

- *numcont:* The number of contestants in the competition. This number must be greater than 0, and does not have currently an upper limit.
- *numprob:* The number of problems in the competition. This number must be greater than 0, and currently you cannot have more than 64 problems.
- *mode:* Program operating mode. This basically determines how PCRM will calculate the score after evaluating the results. Currently, there are three operating modes that are available. You can read more about them – especially the third one – in Subsection 2.1.
- *compout:* When PCRM compiles the solutions, you can choose to see the output or you can choose to hide it (send it to limbo?). This setting governs what happens to the output.
- *runout:* Display runtime output for certain programs (should be 0). This includes the program that is run, and standard output is not redirected (because the program writes output to a file). Should not be used, only for testing and debugging.

- *result:* Result calculation method.
- *retest:* Retesting solution setting. This setting tells PCRM what to do when a second or third or more test is requested.
- *pop3head:* Check for pop3 header. PCRMPOP3 adds a small header to all files, which contain information about the file, like contestant, problem and a time stamp. When enabling this option, PCRM will use this information, which of course makes the judging more accurate. It also verifies the integrity of files this way.
- *acmstart:* Competition start time.
- *acmstop:* Competition end time.
- *acmpenal:* Penalty for wrong solution. Default is 20 (ACM standard).

3.2.2. HTML section

PCRM can also generate the output in HTML format. In this case, you can fine tune the settings used to generate it with the *[html]* section.

- *frame:* In case the jury needs to show additional messages, it is possible to insert an <iframe> between the results and the contest history. You can also set the file from which the frame content is included with other settings.
- *framesrc:* Filename to show. This will be written into the `<iframe>`'s `src` property.
- *framewidth:* The width of the frame in pixels.
- *frameheight:* The height of the frame in pixels.
- *showlang:* If the value of this settings is 1, then the programming language a solution was submitted in will also be included in the results table.
- *destfile:* File where to generate the HTML result. Normally, the result is written to a file named `pcrm_res.html` in the same directory where PCRM runs.
- *autorefresh:* If this value is set, the written HTML file will also contain the setting, so the browser will refresh the HTML automatically over a period of time.
- *gfxproblem:* If this value is 1, then instead of the problem name the little circle with a letter is displayed. Only used when using *resgfx* mode in the program.
- *historynum:* Adds numbers to the history list, beginning with 1. Can be used to see how many history rows there are.

- *noresultafter:* After the specified time, the global results will not be generated (this is the result file given in *destfile*).
- *judgeresults:* Sets mode for another type of output generation. If enabled, the program will generate more result files (see *judgefiles* option). These files contain the complete result as if it was generated with normal generation and also individual files for the contestants. This option can be used with noresultafter, so after a certain time only judges can view the complete result and contestants can view only their own results.
- *judgefiles:* Specify path and name of files for *judgeresults*.
- *addtime:* Adds time information to accepted programs (how long it ran).
- *addtime2:* Adds detailed time information to accepted programs (how long it ran). addtime must also be enabled. Currently, this option will not work on Linux systems.
- *addmemory:* Adds memory usage information to accepted programs (how much memory it consumed).

3.2.3. POP3 section

The *[pop3]* section contains settings that are used by the PCRMPOP3 program. These are mostly email settings and options how to launch the PCRM program itself.

- *server:* POP3 Server domain name or IP address.
- *user*: Username to send to server.
- *pass:* Password for user name.
- *checkwait:* If listen or full auto-judge mode, wait time between checks.
- *subject:* Mail subject type and verification.
- *savemail:* Save all messages to `pop3/` folder.
- *genresult:* Generate results after pop3 operation? If set to 1 or 2, then after an auto1 or autoall command is executed, a result generation will also be requested from the PCRM program.
- *genalways:* If set to true then results will be generated in autoall mode after every run, not just ones that update the source files and run PCRM. This can be used to display the time left "constantly".
- *pcrmpath:* Sets where the `pcrm.exe` (or `pcrm` for Linux) executable can be found. This defaults to the current directory, but you can set it to something else like `../debug/windows/pcrm.exe` or something like that.

- *autoend:* Automatically end competition? This too can be used with autoall and is used to end the checking cycle as soon as the end of the competition is reached. This option also checks the acmstop time in the *[global]* section to see when it needs to stop. Will always generate a result according to the *genresult* parameter before exiting. The option can be used to terminate the competition automatically, when time is up, and no more results will be generated. (Note: this also exits from *listen* mode.)

3.2.4. Compiler sections

The *[compilerxxx]* section contains compile lines, which tell the program how to compile the solution source codes. There are two compiler sections, *[compilerwin]* and *[compilerlnx]*, under Windows systems, the first one is used and under Linux systems, the second one is used.

The section has entries in the form: *lang=compileline*, where lang can be `c`, `cpp`, `asm`, `java`, `pas`, `bas`, `cs`. These are for languages C, C++, Assembly, Java, Pascal, Basic, C#, respectively.

3.2.5. Run sections

The *[runxxx]* section contains run lines, which tell the program how the compiled solution programs should be run. There are two run sections, *[runwin]* and *[runlnx]*, under Windows systems, the first one is used and under Linux systems, the second one is used.

They work basically the same way as the compile lines, except this time you specify the run command.

3.2.6. Problem sections

Every problem has it's own *[problem#]* section, where # is replaced with the problem number, counting from 1. Every problem can have some test cases (at least 1). Some parameters:

- *numtest:* Number of test cases.
- *maxtime:* Time limit for each test case, in milliseconds (1/1000 s).
- *maxmemory:* Memory limit for the program, in Kbytes. Default value is 0, which disables the memory checking. Only available under Linux and Windows NT4/2000/XP.

3.2.7. Contestant section

This is where contestant information is stored. There are two types of entries. The first one lists the name of the contestant or team, and the second one gives a passphrase, which is used with the PCRMPOP3 program when handling email messages.

- *#=name*
- *P#=passphrase*

is replaced with the contestant number, starting at 1.

4. Technical information

There are two versions of PCRM available.

The Windows version requires Microsoft Windows operating system. PCRM will run on any of the following systems:
Windows 95/98/Me/NT4/2000/XP/2003Server.
The program runs in console mode. Actually, there are two versions for Windows. The general version cannot measure process memory and execution time because of limitations in the 9x/Me series Windows operating systems. For the version that includes memory and time measurement, you will need NT4/2000/XP/2003Server.

Another version is available for i386 based Linux systems (libc6). Under Linux, only execution time information can be requested, detailed timing information (kernel time, user time) is not available.

Source code is available to registered users. PCRM (and PCRMPOP3) was coded entirely in C++, no other libraries or 3rd party source code was used to create it.

4.1. Windows requirements

To use memory and time information and/or memory limit functionality under Windows, you will need Windows NT4/2000/XP/2003Server. For NT4 you will also need to have a redistributable component installed. This component is called Process Status Helper (PSAPI), and can be found in the redistributable components of the Platform SDK.

More information is available in the MSDN library under Process Status Helper or look at the following address:

`http://www.microsoft.com/msdownload/platformsdk/sdkupdate/psdkredist.htm`

4.2. Further requirements

If you want to use PCRMPOP3, you will need access to a POP3 server. PCRMPOP3 uses the POP3 protocol to receive emails from the POP3 server. If you also wish to read emails there, please make sure your email client does not delete or move emails on the server because in that case, PCRMPOP3 might not be able to find the emails.

If you are using a firewall, then you will have to allow PCRMPOP3 program access the Internet. It will connect to the POP3 server using TCP port 110.

4.3. Downloading the program

The latest version of the PCRM program can be downloaded from the following address:

`http://www.frenzy.hu`

Postal address

Márk Kósa, János Pánovics, Lénárd Gunda
Department of Information Technology
University of Debrecen
H–4010 Debrecen, P.O.Box 12
Hungary