A Simple Judging System for the ACM Programming Contest

Joseph Clifton
Computer Science and Software Engineering
University of Wisconsin – Platteville
Platteville, WI 53818
clifton@uwplatt.edu

Abstract

The University of Wisconsin – Platteville has been a host site for the North Central North America (NCNA) regional of the ACM International Collegiate Programming Contest every year since 2002. We are a small site, hosting a maximum of eleven teams. We hosted a few times earlier in the 1990's, but it wasn't until 2002 that we started looking at mechanizing the submittal and judging process.

At that time, we considered using the Programming Contest Control (PC^2) system developed at California State University, Sacramento. It is a generalized system and hence is necessarily "complicated". We had successfully developed submittal programs for our lower-division programming courses and believed we could create a simpler judging tool tailored for small contest sites. This paper presents our tool for automating the judging for the ACM programming contest for a small contest site that uses Windows-based computers.

1. Background

The University of Wisconsin – Platteville has been a host site for the North Central North America (NCNA) regional of the ACM International Collegiate Programming Contest [1] every year since 2002. This programming contest is sponsored by IBM and has teams participating from around the world. The NCNA region consists of Wisconsin, Iowa, Minnesota, North Dakota, South Dakota, Nebraska, Kansas, Manitoba, and Ontario. Thus, the contest includes all the states that are part of the Midwest Instruction and Computing Symposium. The NCNA regional contest is held each year in late October or early November. Since the region is geographically rather large, the contest is distributed over several contest sites.

Our university hosts a maximum of eleven teams. We hosted a few times earlier in the 1990's, but it wasn't until 2002 that we considered mechanizing the submittal and judging process. At that time, we downloaded the latest version of the Programming Contest Control (PC^2) system developed at California State University, Sacramento. [2] The author doesn't recall the exact details, but does recall struggling with the installation. Perhaps there were campus restrictions on certain required components, or maybe it was an issue of compatibility. In any case, we had developed submittal programs for course work and believed we could create a simpler judging tool. So just prior to the 2002 contest, the author developed a lightweight windows-based tool tailored for small contest sites such as ours.

The tool has gone through a few modifications since then, but the basic principle of simplicity remains the same. The server runs on any windows system and requires minimal setup. Furthermore, the tool allows most tasks to be performed with a minimal number of mouse clicks and no typing. Checking program output is the only time a keyboard is needed during the judging process, and even that could be eliminated by using a GUI-based difference tool.

The server has an audio annunciation when a team submittal arrives. The judges then select which program they wish to judge (in case more than one arrives at the same time). Clicking a single button compiles, links, and runs the program, sending the output to a file. The judges then run a difference program comparing the contestant's output to the judges' output and if necessary, examine the output manually. When finished, the judges select a status to report to the group and click a button to send back the status.

At the start of the contest, teams are issued a team number and a password, and are given the client executable. They use the client to submit their programs and get back the results. There are no user manuals or help files. The interface is so simple, contestants don't need them.

2. Other Contest Programs

Over the years, other programs have been developed to help support programming contests. PC^2 is probably the best known. Mary Richmond, Dave Bosley, Paul Meyers, and Doug Lane created it in 1988 as a senior project. The system has evolved a great deal since that time and continues to evolve. It is a "heavy-weight" solution, very configurable and designed for handling both single and multi-site contests. The version 8.5 administration guide (released in 2003) is 89 pages long! The latest version of PC^2 is 9.1. The versions after 8.5 don't have new administration guides but instead have "What's new" documents that reference the 8.5 administration guide and a Wiki. The current version of the team guide is seven pages long. [2, 3]

Significant improvements have been made to PC^2 since the author first considered using it. The author had also looked at PC^2 three or four years ago, but still wasn't convinced to change. The latest version has simplified setup and includes auto-judging, but still has a few issues. For example, when the author tried logging in with an invalid login name, the system hung. Also, when the author forgot his server password, he spent about half an hour searching through the documentation to find a fix before giving up and later noticing the pc2reset.bat file. Other current bugs are listed in the Version 9.1 "What's New" document.

The PC^2 "startup checklist for geniuses" from the version 8.7 documentation is: [2]

- Install PC^2 by unzipping the PC^2 distribution to the PC^2 installation directory
- Add the Java *bin* directory and the PC^2 installation directory to the PATH
- Add "**.**", java/lib, and the PC^2 installation directory to the CLASSPATH
- Modify the *sitelist.ini* file as necessary to specify each site server name
- Edit the *pc2v8.ini* file to point servers and clients to the server IP:port and to specify the appropriate site server name; put the modified .ini file on every server and client machine
- Start a PC^2 server using the command "pc2server" and answer the prompted question.
- Start a PC^2 Admin client using the command "pc2admin" and login using the name "root" and password "root".
- Configure at least the following contest items via the Admin:
    - Accounts (generate the necessary accounts)
    - Problems (create one or more contest problems, specifying the problem input data file if there is one)
    - Languages (create one or more contest languages, specifying the language name, compile command line, executable filename, and program execution command line).
- Press the "Start Contest" button on the Admin "Time/Reset" tab
- Start a PC2 client on each Team and Judge machine and log in using the Admin-created accounts and passwords.

- Start a PC^2 client on the Scoreboard machine and log in using the "board1" Scoreboard account/password; arrange for the scoreboard-generated HTML files to be accessible to user's browsers.

In version 9.0, the "sitelist.ini" was no longer necessary, since the values formerly specified in that file were made interactively configurable.

The batch files PC^2 uses to start its various programs are: [2]

- *pc2reset* - Creates a backup zipped archive of all existing PC2 databases, then clears all databases to the state matching a fresh installation
- *pc2server* - Starts a PC^2 Server
- *pc2admin* - Starts a PC^2 Client expecting an Administrator login
- *pc2team* - Starts a PC^2 Client expecting a Team login
- *pc2judge* - Starts a PC^2 Client expecting a Judge login
- *pc2board* - Starts a PC^2 Client expecting a Scoreboard login

Figure 1 shows the PC^2 administrator program. The teams and judges were auto-generated using the "Generate" button. The program has many options for creating and modifying accounts, problems, languages, groups, etc. For example, when editing an account, the display name, password, and group can be specified. In addition, there are 49 different permissions from which to choose! Fortunately, the program's defaults are satisfactory, so most options and features don't need to be used.

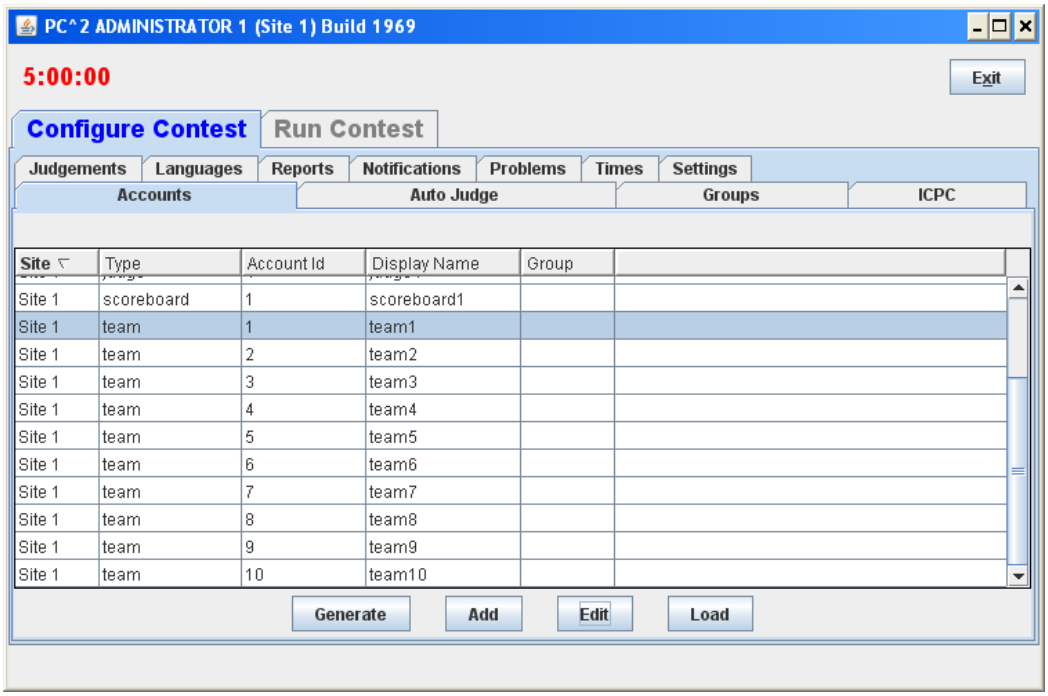

Figure 1: PC^2 Administrator Program

Figure 2 shows the PC^2 Judge program. It allows programs received from the teams to be executed and judgments to be returned to the teams. It also receives clarification requests from teams and allows the judges to respond.

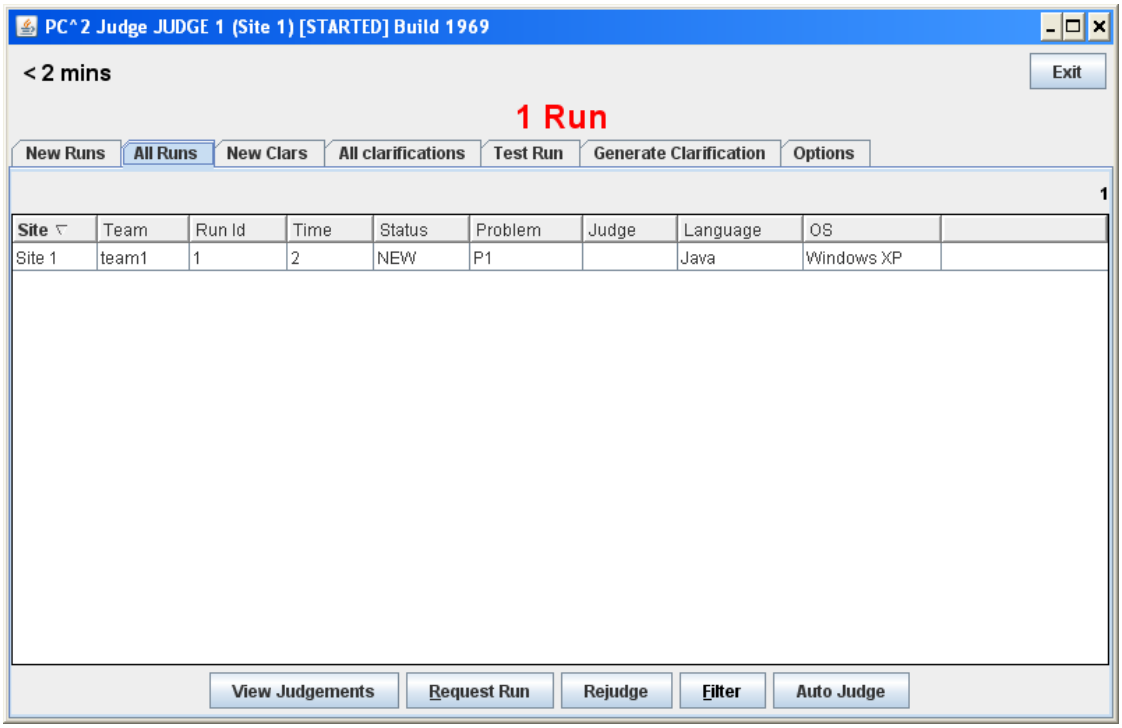

Figure 2: PC^2 Judge Program

Figure 3 shows the PC^2 Team program. It allows contestants to submit programs and view the judges' responses. It also allows teams to request clarifications and view all the clarifications for all the problems. In addition, it shows the remaining contest time. The interface is intuitive and simple to use.

The major differences between PC^2 and the judging tool that we present in Section 3 are that our tool requires less setup and has almost no options. Another way of saying this is that our program is very "lightweight", whereas PC^2 is "heavyweight". One piece of functionality in PC^2 that isn't in our tool is the ability to handle problem clarifications. We still use student runners for that. In addition, our tool currently doesn't support auto-judging, although it could easily be added. The reason it doesn't exist now is that typically the problems in the NCNA contest allow latitude in formatting and auto-judging generally works best with strict output formatting requirements. Therefore, we have semi-automated that function, as will be discussed in Section 3.

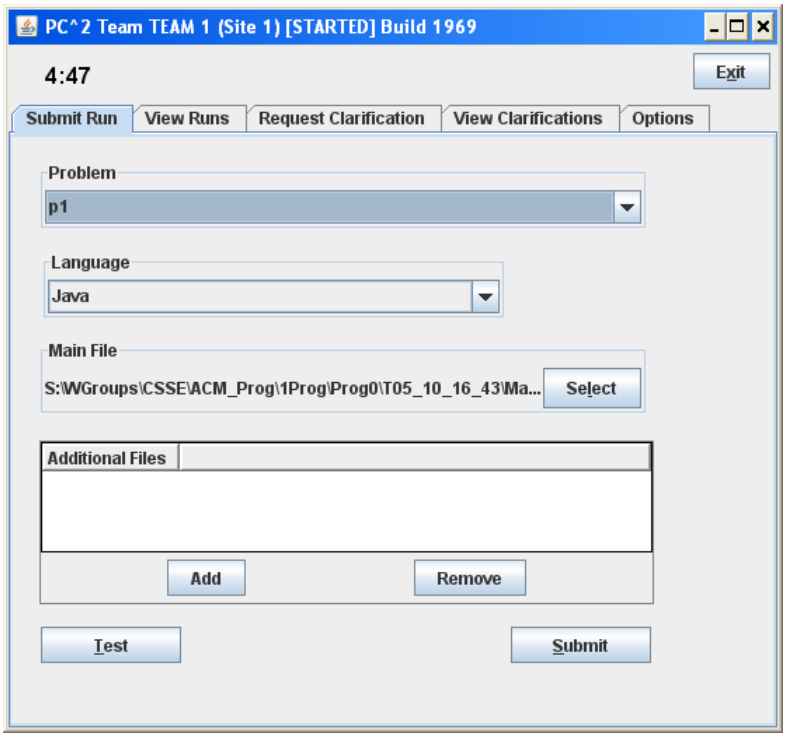

Figure 3: PC^2 Team Program

DOMJudge is another example of a tool that supports programming contests. It has been used in the ACM ICPC Northwestern European Programming Contest for the past three years. It has a 39-page administrators' manual, eleven-page judges' manual, and an eight-page team manual. The "cheat-sheet for those who've already installed DOMjudge before and need a few hints" is: [4]

External software:

- Install the MySQL-server, set a root password for it and make it accessible from all judgehosts.
- Install Apache, PHP and (recommended) phpMyAdmin.
- Make sure PHP works for the web server and command line scripts.
- Install necessary compilers on the judgehosts.
- See also an example command line for Debian GNU/Linux.

DOMjudge:

- Extract the source tarball and run ./configure [--enable-fhs] --prefix=<basepath>.
- Run make domserver judgehost docs or just those targets you want installed on the current host.
- Run make install-{domserver,judgehost,docs} as root to install the system.

On the domserver host:

- Install the MySQL database using bin/dj-setup-database -u root -r install on the domserver host.

- Add etc/apache.conf to your Apache configuration, edit it to your needs, reload web server: sudo ln -s .../domserver/etc/apache.conf /etc/apache2/conf.d/domjudge.conf && sudo apache2ctl graceful
- Check that the web interface works (/team, /public and /jury) and check that the jury interface is password protected.
- Add useful contest data through the jury web interface or with phpMyAdmin.
- Run the config checker in the jury web interface.

On the judgehosts:

- RedHat: useradd -d /nonexistent -g nobody -M -n -s /bin/false domjudge-run
  Debian: useradd -d /nonexistent -g nogroup -s /bin/false domjudge-run
  (check specific options of useradd, since these vary per system)
- Start the judge daemon: bin/judgedaemon

This is more work than we would be willing to do for our small site!

A paper was presented at the 2005 International Conference on Computer and Information Technology in Bangladesh in which the authors describe online judging software that they created using the .NET frameworks. In the paper, they primarily focus on security and speed, neither of which is an issue for us. They also refer to PC^2. The paper seems to indicate it is still a work in progress. [5]

3. U WP Judging Tool

The UWP Judging tool consists of two stand-alone executables: a client and a server. Both are windows-based GUI programs that don't require installation beyond copying to a folder on the host machine. They support C++, C#, and Java. They require that the program source be in a single file. The tool does not automate the entire judging task, but does automate most of it, and is a good compromise between functionality and ease of setup and use.

3.1 Contest Server

Figure 4 shows the contest server. It is a stand-alone executable and requires no installation. The judges copy it to any folder on a windows-based PC and double-click to execute it. When the program is started, it creates a directory structure under the start-up directory as shown in Figure 5 (assuming that the directories don't already exist). There is one folder for each program and a _Work folder.

The program assumes at most eleven teams and ten programs. These values are easily changed, although that would require a rebuild of the Borland Builder project, i.e., they are not customizable. The team names are fixed as Team1, Team2, etc. The program names are fixed as Prog1, Prog2, etc. These also are not customizable.

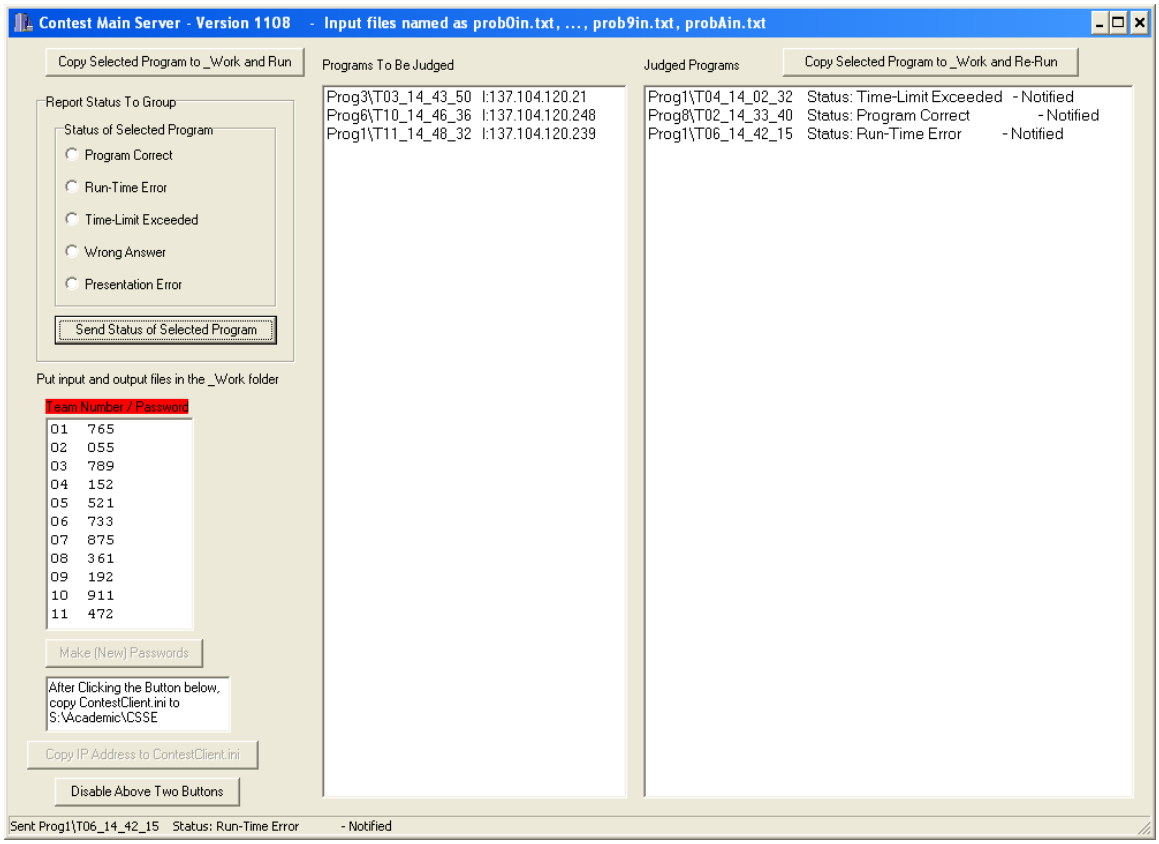

Figure 4: Contest Server

On startup, passwords are generated for each team if they don't already exist. These passwords are also stored to a file called Student.dat and read back from that file on subsequent restarts. A button exists to create a new set of random passwords if desired. The passwords are only three digits long; however, that is sufficient security for our site. It is highly unlikely that any of the teams at our site would do a brute force attempt at using another team's password, at risk of disqualification. Furthermore, the IP address of a team's assigned computer is appended to the submittal information, so the judges can review it if foul play is suspected. The passwords are distributed to the teams during the pre-contest warm-up session.

The right-hand panel of the Windows Explorer screen shown in Figure 5 gives an indication of the server setup required. The judges input files must be put in the _Work folder and must be named as ProbXin.txt, where X is the program number, in hexadecimal. For some years, this was the naming convention used by the contest director and no extra work was required. For other years, each input file needed to be renamed. The files designated 0, 1, 2, etc., are the judges' output files. These can be renamed to whatever the judges want. The comparison between the judges' output and the submittal output is not automated (even though it would be easy to do so) due to the usually lenient output formatting requirements. Instead, the judges use a difference tool, such as diff.exe shown Figure 5. The judge who "drives" for our contest uses command-

line tools, which is also the reason the output files are named as they are. If another judge were to run the server, it is likely a GUI-based tool, such as Beyond Compare, would be used.

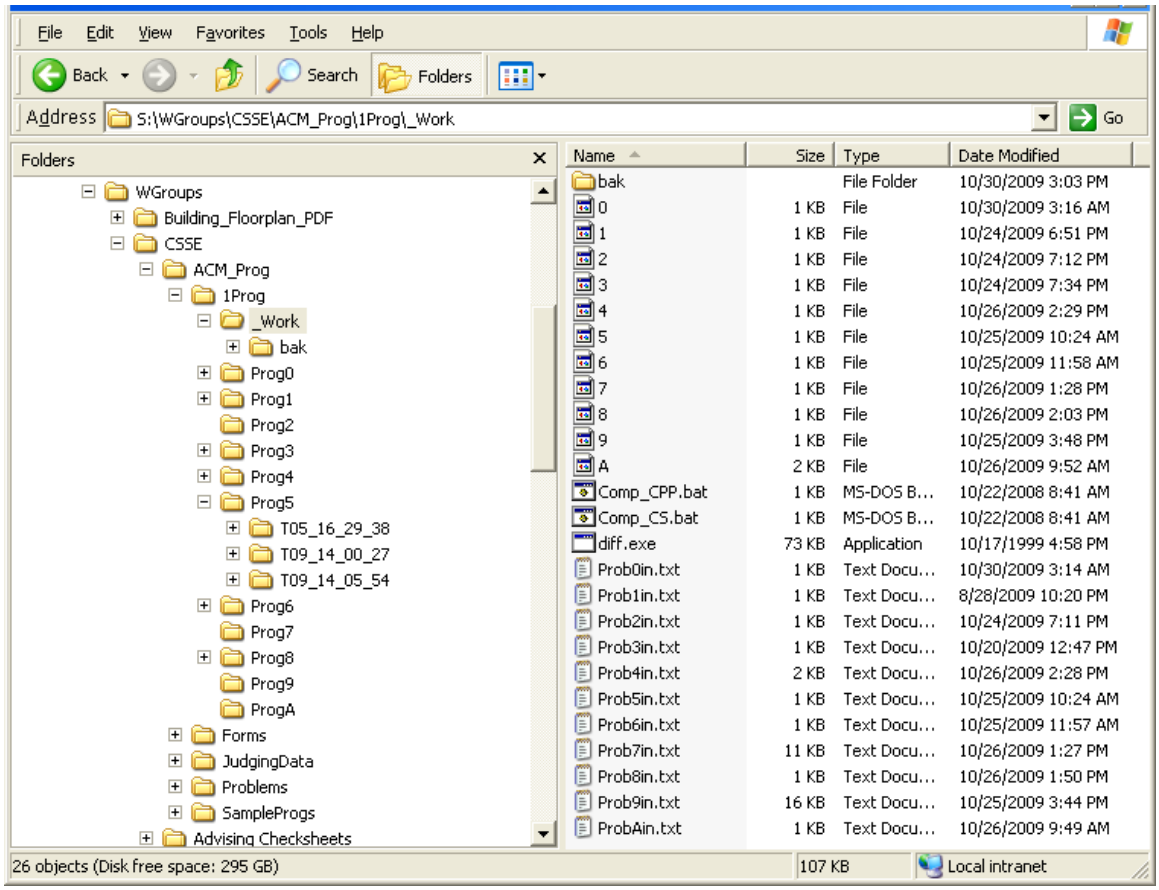

Figure 5: Contest Server Directory Structure

The right-hand panel of Figure 5 also shows two BAT files. These files are used to specify the compiler commands for C++ and optionally, C#. For Java, it is assumed the Java compiler (javac.exe) is in the Windows path. The names shown for these files are the required names and are not configurable. For us, these files are quite short. For example, Comp_CPP.bat invokes the Microsoft Visual Studio compiler as:

```
call "%VS90COMNTOOLS%"vsvars32.bat
cl /FeProg.exe *.cpp
```

It should also be pointed out that it is required that the executable produced by a C++ or C# compiler be named Prog.exe, since the server depends on that when running the program.

As programs arrive at the server, there is an audio annunciation and the submittal appears in the “Programs to be Judged” panel as shown in Figure 4. The entry contains the program number, team number, time of submittal, and IP address of the submittal computer. The program is stored in a folder marked with the team name and arrival time under the appropriate ProgX folder. For example, Figure 5 shows the T05_16_29_38 folder under the Prog5 folder. This folder contains the program submitted by Team 5 at 4:29:38 P.M.

The judge selects an entry in the “Program to Judged” panel and clicks the “Copy Selected Program to _Work and Run” button to execute it. The judge then switches to another program to compare the program’s output to the judges' output. In the case of our contest, the judge switches tasks to a command prompt and runs diff. If the author were a judge instead of the site coordinator, he would use Beyond Compare.

After comparing the outputs, the judge switches back to the contest server, clicks one of the status selections, and then clicks the “Send Status of Selected Program” button. It then sends the status back to the contest client and moves the selected entry to the “Judged Programs” panel. The entry in that panel indicates the program number, team number, status, and whether or not the notification was successful. If the judges have any reason to rerun a submittal, they can click the “Copy Selected Program to _Work and Re-Run” button.

There are two additional “administrative” buttons on the lower-left panel of the server. One button is used to create the ContestClient.ini file, which consists solely of the Server IP address. When that button is clicked, the server’s IP address is retrieved and written to the file. The other button is used to disable the “Make New Passwords” button and “Copy IP Address to ContestClient.ini” button. This can be clicked after setup so these buttons aren’t accidentally clicked during the contest (which happened during the first contest).

3.2 Contest Client

Figure 4 shows the contest client. It is a stand-alone executable and requires no installation. The contestants copy it and the ContestClient.ini file to any folder on their designated computer and double-click to execute it. Teams are issued their team number and password during the pre-contest warm-up session.

When the team wishes to submit a program, they select their team number, the program number, and type in their password. These selections remain until changed. Then they browse to select the program file to submit, and finish by clicking the “Submit the Files” button. The name of the button is a little misleading, since the tool only supports single file submission. The author had expected to allow multiple files at some point, but that

hasn't happened yet! It should be noted that in the "Source Files to Send" panel, the author has selected a program submitted by Team 7 in last year's contest.

The top panel on the right of the contest client shows all the submitted programs. The team number, program number, and time of submittal are displayed. The bottom panel on the right of the contest client shows the judged programs. The program number, team number, submittal time, and judged status are displayed.

There are no other options. There are no user manuals or help files. The interface is so simple, contestants don't need them. One might notice that the language for the program is not specified. The contest server assumes the programming language from the file extension.

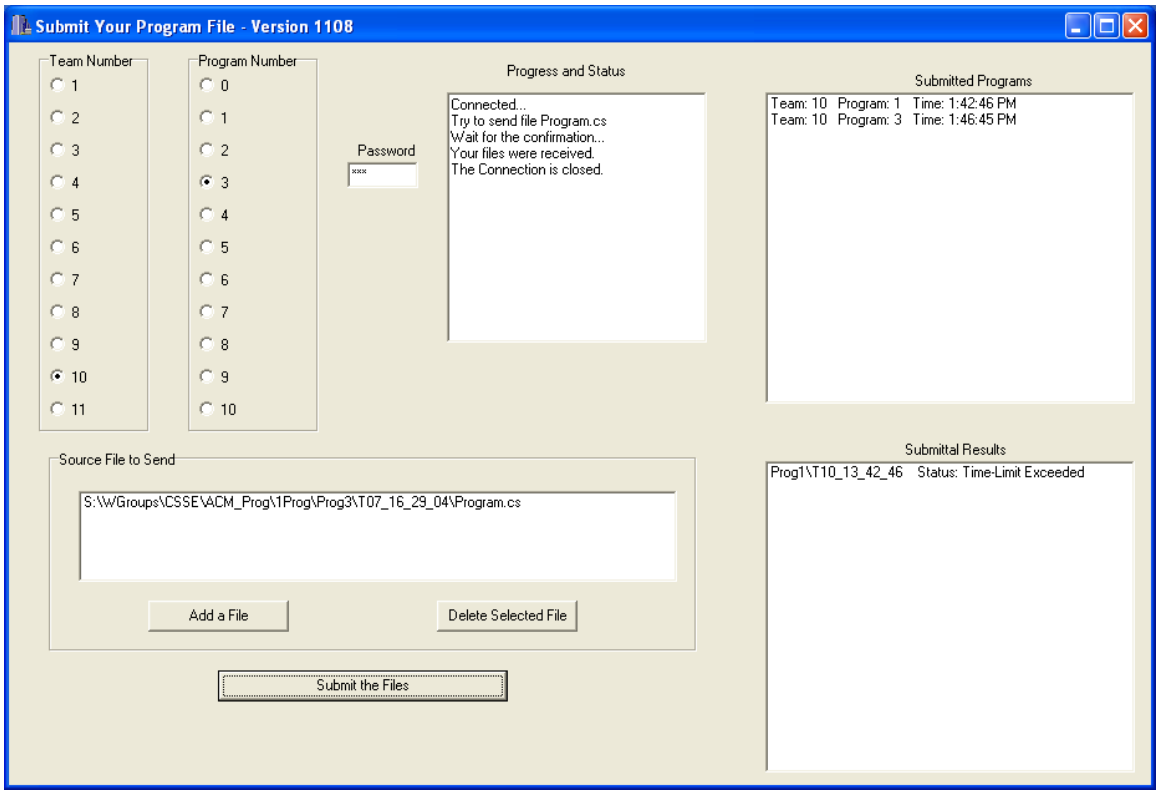

Figure 6: Contest Client

4. Conclusion

The UWP Judging system is a simple windows-based tool. It mechanizes most of the judging tasks. One notable exception is output comparison; however, that is by choice, since adding that feature would be relatively easy. It has less flexibility than a tool such as PC^2. For our tool, there is practically no customization that can be done except for setting up the C++ compiler commands. Team names, passwords, program names,

output file names, etc., are all fixed. However, it is for this reason that it is less complicated. Setup for our tool is very quick. The number of teams allowed is also fixed at a maximum of 11. This could easily be changed but would require a recompilation to change it.

The UWP Judging program has the basic functionality of PC^2 with one major exception: problem clarifications. The UWP Judging system has no provision for handling these.

References

[1] ACM International Collegiate Programming Contest, Available at http://acm.baylor.edu/

[2] Programming Contest Control System developed at California State University, Sacramento, Available at http://www.ecs.csus.edu/pc2/

[3] PC2Wiki, the Wiki dedicated to information about PC^2, Available at http://pc2.ecs.csus.edu/wiki/

[4] DOMjudge, Automated Judge System to Run Programming Contests, Available at http://domjudge.sourceforge.net/

[5] Arefin, A., et al., "Secured Programming Contest System with Online and Real-time Judgment Capability", International Conference on Computer and Information Technology, Bangladesh, 2005