Contents

Summary

An output validator is a program that is given the output of a submitted program, together with the corresponding input file, and a correct answer file for the input, and then decides whether the output provided is a correct output for the given input file.

This document describes a standard for Output Validators for Programming Contest Control Systems. It is designed especially with ACM ICPC-style contests in mind. This includes being well suited for use in the ICPC World Finals, but also being useful in other ICPC-related activities (such as regional contests, local training contests, or even evaluation of programming assignments in courses).

Overview

A validator program must be an application (executable or interpreted) capable of being invoked with a command line call. The details of this invocation are described in Section 3 Invocation. The validator program has two ways of reporting back to the Contest Control System that invoked it:

1. The validator must give a judgment (see Section 4 Reporting a judgment).
2. The validator may give additional feedback, e.g., an explanation of the judgment to human judges. This is described in Section 5 Reporting Additional Feedback.

Invocation

The CCS will invoke the validator and passing it at least four command line parameters.

The usage of the validator is as follows:

```
validator input judgeanswer feedbackdir < teamoutput
```

The four parameters are:

input
: a string specifying the name of the input data ?le which was used to test the program whose results are being validated.

judgeanswer
: a string specifying the name of an arbitrary ?answer ?le? which acts as input to the validator program. The answer ?le may, but is not necessarily required to, contain the ?correct answer? for the problem. For example, it might contain the output which was produced by a judge?s solution for the problem when run with input ?le as input. Alternatively, the ?answer ?le? might contain information,

in arbitrary format, which instructs the validator in some way about how to accomplish its task. The meaning of the contents of the answer ?le is not defined by this standard.

feedbackdir
: a string which specifies the name of a ?feedback directory? in which the validator can produce "feedback files" in order to report additional information on the validation of the output ?le. The feedbackdir must end with a path separator (typically '/' or '\' depending on operating system), so that simply appending a filename to feedbackdir gives the path to a file in the feedback directory.

teamoutput
: the output ?le which was produced by the program being validated is given on the validator's standard input pipe.

The two ?les pointed to by input and judgeanswer must exist (though they are allowed to be empty) and the validator program must be allowed to open them for reading. The directory pointed to by feedbackdir must also exist.

Reporting a judgment

A validator program is required to report its judgment by exiting with speci?c exit codes:

- If the output is a correct output for the input ?le (i.e., the submission that produced the output is to be Accepted), the validator exits with exit code 42.
- If the output is incorrect (i.e., the submission that produced the output is to be judged as Wrong Answer), the validator exits with exit code 43.

Any other exit code (including 0!) indicates that the validator did not operate properly, and the contest control system invoking the validator must take measures to report this to contest personnel. The purpose of these somewhat exotic exit codes is to avoid con?ict with other exit codes that results when the validator crashes. For instance, if the validator is written in Java, any unhandled exception results in the program crashing with an exit code of 1, making it unsuitable to assign a judgment meaning to this exit code.

Reporting Additional Feedback

The purpose of the feedback directory is to allow the validator program to report more information to the contest control system than just the accept/reject verdict. Using the feedback directory is optional for a validator program, so if one just wants to write a bare-bones minimal validator, it can be ignored.

The validator is free to create different files in the feedback directory, in order to provide different kinds of information to the contest control system, in a simple but organized way. For instance, there may be a ?judgemessage.txt? file, the contents of which gives a message that is presented to a judge reviewing the current submission (typically used to help the judge verify why the submission was judged as incorrect, by specifying exactly what was wrong with its output). Other examples of files that may be useful in some contexts (though not in the ICPC) are a score.txt file, giving the submission a score based on other factors than correctness, or a teammessage.txt file, giving a message to the team that submitted the solution, providing additional feedback on the submission.

A contest control system that implements this standard must support the judgemessage.txt file described above (I.e., content of the "judgemessage.txt" file, if produced by the validator, must be provided by the contest control system to a human judge examining the submission). Having the Contest Control System support other files is optional.

Note that a validator may choose to ignore the feedback directory entirely. In particular, the contest control system must not assume that the validator program creates any files there at all.

Examples

An example of a judgemessage.txt file:

```
Team failed at test case 14.
Team output: "31", Judge answer: "30".
Team failed at test case 18.
Team output: "hovercraft", Judge answer: "7".
Summary: 2 test cases failed.
```

An example of a teammessage.txt file:

```
Almost all test cases failed, are you even trying to solve the problem?
```

Validator standard output and standard error

A valididator program is allowed to write any kind of debug information to its standard output and standard error pipes.