Regression Evaluation Interface (REI)

Version 1.0
Justin David Smith
September, 2002

Abstract

This document describes the basic use of the Regression Evaluation Interface (REI), which is the regression system used in the Mojave Compiler (MCC). This document provides examples for invoking regression tests, adding test cases and adding test profiles.

Contents

1 Introduction

Note: The current version of this document is not thorough. Note that REI features both command-line and interactive help systems which go into more detail than this document. This document is intended to introduce the concepts of REI and describe basic use, not to provide detailed help about all available commands (yet).

1.1 What is REI?

REI is the Regression Evaluation Interface, and is designed to make it easy to define and run regression test suites for the Mojave Compiler (MCC). REI features both an interactive prompt, and a command-line mode capable of running the regression suite. The interactive mode is designed to make it easy to manipulate test suites (including adding new test cases and new test profiles). The command-line mode is only capable of running the test suite, and is suitable for running regression tests from an automated script.

1.2 What is a test suite?

A test suite is a collection of test cases and test profiles which define the test cases for the Mojave Compiler. For MCC, the standard test suite is in `util/rei/test.suite`. A test suite currently has three components:

- A *suite name*, which is informative text about the test suite.
- A list of *test cases* for the suite.
- A list of *test profiles* (collections of related test cases).

Test profiles are used to group collections of related test cases to simplify testing. When a user wants to perform regression testing, they may choose to run the entire suite, all test cases in a particular test profile, or an individual test case.

1.3 What is a test case?

A test case is a single regression test for the Mojave Compiler, consisting of the following properties:

- A *name* used to uniquely identify the test case.
- An *enabled* flag, which indicates if the test case is enabled. This is used to disable test cases which are broken and expected to remain broken for the long-term future.
- A *target* which must be built for the test case (exactly one target must be specified). The target is built using `cons`.
- The expected *build result*, which indicates the expected result of building the test target. If this is value is 0, we expect that the target will be successfully built; otherwise, we expect the build to fail with the indicated error code.

Additionally, if the test case specifies a build result of 0, then it also defines the following properties:

- An *evaluation command* which will be run (using the `system` call) for the test case, assuming the build was successful.
- The expected *return code* of the evaluation command.

- The expected *output* of the evaluation command.

When a test case is run, first the target will be built. If the expected build result is 0 and the test target is built correctly, then the evaluation command is run and its output is captured. If the evaluation command also runs properly, then the return code and output are checked against expected results to determine if the regression case passed.

When a test case is run and its expected build result is not 0, then the regression case will only pass if the attempt to build its target *fails*. This is used for test cases which check the error-handling of the compiler, e.g. cases which ensure that the compiler raises type errors when expected. Note that the evaluation command is ignored for a test case with nonzero expected build result.

If a test case is disabled, then the test case is never run. Test cases which are disabled include a short description of why they were disabled, such as "Test case needs a feature which is not implemented".

1.4 What is a test profile?

A test profile is a collection of related test cases, designed to make it easier to perform regression testing specific to a particular feature of the compiler. For example, it would make sense to define a test profile of all the atomic test cases; then the user can perform regression related to the transaction system using this profile. A test profile has two properties:

- A *name*, used to specify the test profile when running regression tests.
- A list of *test cases* that should be run when the profile is invoked.

1.5 What are run stats?

Run stats are statistics collected during a regression run. They indicate which test cases passed a particular regression run, as well as the time required for the regression run. Run stats also contain a full log of the associated regression run (including all output from `cons`). Run stats are not saved as part of a test suite, but may be saved separately. Run stats have the following properties:

- The *name* of the test suite that was used for the regression run.
- The *start* and *stop* times for the regression run.
- The complete *log* of the regression run.
- A *count* of the total number of test cases run.
- A list of *successful test cases*. This list includes the total running time of each test case (not including build time).
- A list of *failed test cases*. This list includes a brief description of why each test case failed (e.g. "Build failure"). Details on why a test case failed can be found in the log.

2 Invoking REI

The REI program is stored in `util/rei/`, with the name `rei`. You can invoke it from the `util/rei/` subdirectory, or from the topmost directory of MCC. It must be invoked from somewhere within the *uninstalled* copy of MCC; it cannot work with an installed copy of MCC. The remainder of this document will assume `rei` is in your path, and that you are invoking `rei` from within `util/rei/`.

2.1 Interactive mode

To invoke REI's interactive mode, simply type `rei test.suite`. You can get help at any prompt by typing `help`. If you compiled MCC with GNU readline support, then you can tab-complete command names and filenames at any prompt. Note that there are several prompts in interactive mode, each with their own set of commands; typing `help` will give you help for the current prompt only. The interactive mode is discussed in more detail in section 3.

If you compiled MCC with ncurses support, then you can enable the ncurses interactive mode by running `rei -curses test.suite`. The ncurses mode enables full-screen viewers for output and logs, and a full-screen mode with status line during regression runs. Without ncurses mode, you will get basically the same output, but without the advantage of a status line or arbitrary scrolling features that the viewer offers. The ncurses mode also provides menus in a few locations; however, REI is still primarily controlled by its command prompt.

Warning: in interactive mode, REI will change to the topmost directory of the MCC source tree automatically. Files named on the command line are always resolved relative to the working directory when you invoked REI, but files within the interactive shell are ALWAYS relative to the root of the MCC source tree.

2.2 Command-line mode

For information on REI's command-line mode, type `rei -help`. The command-line mode can only be used to run regression suites; it cannot be used to modify them. The output from REI will not contain escape sequences or use ncurses in this mode, so you may redirect REI's output to a file if you wish.

To run an entire test suite, use the command `rei -run test.suite`. To run a particular test profile, or a particular test case, use the command `rei -run-profile` *`name`* `test.suite`, where *`name`* is the name of a profile or test case. Any argument which is valid for the `run` interactive command may be passed to `-run-profile`.

Both `-run` and `-run-profile` can also accept the `-write-stats` *`name`* option, which will write the run stats of the regression run to the named file. This filename is relative to the current working directory when you invoke REI (not necessarily the root of the MCC source tree). This option is necessary if you want to manipulate the run stats later in an interactive session, or if you want to compare the results of this regression run to a later run (see below).

The `-autorun` feature is useful for nightly regression tests that want to compare the results of two consecutive runs. You must have saved the previous regression run stats using `-write-stats` *`name`* option, and you must re-read the run stats using the `-read-stats` *`name`*. If the results of this regression run are the same as the previous run, then no output is produced; otherwise, a report indicating the differences between the previous and current runs is emitted to standard output. This makes the `-autorun` feature ideal for use in nightly `cron` scripts. Typical use of this feature would be `rei -run -autorun -read-stats` *`name`* `-write-stats` *`name`* `test.suite`.

2.3 The `auto-rei` script

The `auto-rei` script is designed for use in nightly regression runs invoked from `cons`. It deals with details such as ensuring the CVS repository is updated and MCC is rebuilt properly; it also handles mailing regression failure reports if the regression results change overnight. To use it, you should checkout a fresh working copy of MCC (**Warning:** do not use the tree you actively develop with!) and add the following command to your crontab:

```
25 2 * * *     cd /path/to/regression-mcc/util/rei && ./auto-rei
```

You must also configure a few parametres at the top of the `auto-rei` script; notably, you should tell `auto-rei` where to send regression reports to.

3 Using the Interactive Mode

This section contains examples on how to use the interactive mode of REI. When you start REI using `rei` or `rei -curses`, you start at the `rei>`, which is the main prompt of the system.

3.1 The `rei>` prompt

The `rei>` prompt is the main prompt of the system. Among other things, you can perform the following actions from `rei>`.

3.1.1 Exiting the program

To exit REI, type `exit`. If the current test suite has any unsaved changes, you will be warned before the program terminates. You will *not* be warned if you have not saved the run stats from a regression run performed in the current session.

3.1.2 Saving and loading a test suite

To save a test suite, type `write` *`filename`*. The filename is relative to the root of the MCC source tree, not the working directory when you started REI. To load a test suite, type `read` *`filename`*. Loading a new test suite will discard the current test suite; you will be warned if you have any unsaved changed to the current test suite.

3.1.3 Running a regression suite

To run the regression suite, type `run` at the `rei>`. This will run all enabled test cases in the test suite. The `run` command takes an optional argument which may limit the number of test cases run.

If you type `run` *`profile_name`*, then all test cases associated with the named test profile are run. You can get a list of all defined test profiles by typing `list -profiles`.

If you type `run` *`n`*, then test case #*n* is run. If you type `run` *`n-m`*, then all test cases from #*n* to #*m* are run. This is useful for running intervals of test cases. To see how the test cases are currently numbered, type `list`. Note that the numerical assignments will change over time as you insert and delete test cases.

If you type `run` *`case_name`*, then the named test case is run. You can get a list of all defined test cases by typing `list`, and you can get detailed information about the test case by typing `list` *`case_name`* `-all`.

If the regression run is successful, then the run stats generated for the regression run are saved in memory. You can manipulate the run stats (including saving the run stats) by typing `stats`; this will take you to the `stats>` prompt, which is described in section 3.3. You can also display the log from the regression run by typing `log`.

3.1.4 Adding a new test case

To create a new, empty test case, type `add`. To create a test case which builds and runs a particular program, type `add` *`target_name`*. You may include arguments to pass to *`target_name`* by appending them to the `add` command. If you specify at least a build target, then REI will attempt to compile and run the program, and will use the results to fill in the expected output fields in the test case. In all cases, you will be taken to the `add case>` prompt where you can edit the test case before saving it. This prompt is described in section 3.2.

3.1.5 Editing a test case

To edit an existing test case, type `edit` *`case_name`*. This takes you to the `edit case>` prompt, which is similar to the `add case>` prompt and described in section 3.2.

3.2 The `add case>` and `edit case>` prompts

The `add case>` and `edit case>` prompts are used to edit the properties of a single test case. They are very similar prompts, and are described together here.

3.3 The `stats>` prompt

The `stats>` prompt is used to display, load and save run stats.