AUTOMATED TEST HARNESS INTERFACES

This document specifies the interfaces to the automated tester harness. The harness is composed of two parts: the **top-level tester** and the **individual test handlers**, The name of the file to load for this is `AUTOTEST.LCOM` in the top level of the `{$MEDLEY}internal/test` directory.

The top-level tester is set up similarly to the package FileBrowser. Items are selected in the same manner as FileBrowser, and are displayed similarly. The portions of the display are as follows (from top to bottom):

1. A prompt window for displaying messages and getting new input.
2. A command menu with the following commands:

TEST Tests sequentially each of the items selected in the test files window. Testing consists of loading the file containing the test suite, calling a function which has the same name as the NAME field of the filename (this function must return NIL iff the test suite is not successful), then undoing (as best as possible) the side-effects of loading and running the test suite. The function which is called is passed one argument: the name of the directory that the test suite came from (including the host name). If this item is selected with the middle button, then first it asks for the name of the file to direct output to (selecting this item with the left button will direct output to T, the process TTY display stream), before running the test suites. All output directed to NIL, the default output stream, will go to this file, including all error messages generated by the automated test harness and by TEST-MESSAGE (see below). It is assumed that no other activity is being performed while testing is in progress.

ABORT Aborts any tests in progress. Confirmation (via clicking the left mouse button) is required. New tests can be selected, tests can be re-run, etc. after an abort.

PAUSE Temporarily pauses any tests in progress. Any pause time does not count in the computation of timeouts (see below).

RESUME Resumes PAUSEd testing.

DIRECTORY Does a directory of files (the directory pattern is prompted for in the prompt window) and puts them in the test files window in order to have a new set of test suites to select from.

PRINT Prints the results of testing the selected files. Selecting this item with the left button will print on the default printer. Selecting this item with the middle button will put up a menu asking whether to print to a printer or a file. If a printer is selected, then a menu asking for the printer to print to (gotten from DEFAULTPRINTINGHOST plus the selection "Other"; the latter will ask for the name of a new printer to print to) is put up. Otherwise, if a file is selected, then the user will be prompted for the name of a file to print to (also, if the type of output is not obvious, i.e. PRESS or INTERPRESS, then the user will be prompted for the type of output). When the Hardcopy item of the right button menu is selected for this window, then the command is performed (except that selecting the main item does the default, while selecting either the printer or the file sub-item starts the sequence of questions at the intuitive place).

SUMMARIZE Similar to PRINT, except that it prints only those tests (out of the selected tests) which failed.

QUIT Quits testing, closing the window and throwing away all test results, test names, etc. stored in the window. If any tests are currently in progress, then confirmation (via clicking the left mouse button) is required in order to quit (in this case an ABORT is performed before quitting). When the tester window is closed, this command is performed.

3. A status window, which has the following fields:

Suite The name of the test suite currently running.

ID The ID of the current test being performed by SINGLE-TEST.

Start The time that the current test was started.

End The time that the current test will time out at, or blank if none.

4. A summary window, which has the following fields:

Files The number of files in the test files window.

Selected The number of files (test suites) selected in the test files window.

Completed The number of test suites completed.

Successful The number of test suites which were successful.

5. The directory pattern used to select the test suite files. Unless otherwise overridden, the directory pattern by default only selects the latest version of each test suite file. Also, unless otherwise overridden, the directory pattern by default only selects .LCOM files (if a source file is more recent than the corresponding compiled file, then an error message is displayed).

6. A heading line which identifies each column in the test files window.

7. The test files window which has a line for each test suite file which matches the directory pattern. The left button on an entry selects only that entry. The middle button on an unselected entry adds that entry to the selected entries. The middle button on a selected entry removes that entry from the selected entries. The right button in the left portion of an entry will extend the current entries to include this entry and all the entries inbetween (the mouse cursor will change to a right pointing arrow when this action is enabled). This window is also scrollable (both vertically and horizontally). When each test is completed, a line is drawn through the entry. This window has the following columns:

Result: The result of testing using the corresponding test file. The following can appear in this column:

? The test suite has not been completed or possibly even initiated, so no results are know
n

pass The test suite completed successfully.

FAIL The test suite did not complete successfully. This could be because a test in the test sui
t returned bad results, a test in the test suite aborted, a test in the test suite timed o
u etc..

Name: The NAME portion of the test suite file name.

File: The full name of the test suite file (except for the host name).

When the tester is loaded, a new entry is added to the background menu, labelled AutomatedTester. When this is selected, an automated tester process is started, which will prompt (in the system prompt window) for a directory pattern which is used to initialize the test files window.

The individual test handler is a function which is called by the top-level function of each test suite (the function which was called by the top-level tester). This function has the following interface (all arguments must be supplied):

Name: SINGLE-TEST (LAMBDA function).

Arguments:

IDENTIFIER The integer identifier of this test. Identifiers are assigned manually and a unique across all tests in all test suites. [We need to set up an index file for this purpose, in the standard test directory.]

EXPRESSION The expression to evaluate (e.g. (PLUS 2 3)). Note that in order to get the right results, this argument would normally be quoted with QUOTE (or ') or be an expression such as (QUOTE (fn)), where fn is a separately defined function (and is therefore compiled code, instead of interpreted code).

PREDICATE The (one argument) predicate to check the result (e.g. (LAMBDA (X) (EQP X 5)) or NULL). This must be NIL iff the result was not correct (non-NIL indicates that the result was correct). If more than one error can occur, then output identifying the specific error should printed (to NIL). Note that this argument would normally be quoted with QUOTE (or ') or FUNCTION in order to get the right results.

TIMEOUT The maximum elapsed (wall) time (in milliseconds) that the expression EXPRESSION should take to complete (NIL implies that no timeout is to be used). With the current Interlisp-D process mechanism, this will only work if the expression (or anything it calls) does a BLOCK, so that another process can check to see whether a timeout has occurred. Also, the timing is not exact, so the actual timeout used will be no less than the value supplied. Time elapsed while the test was PAUSEd is not counted in checking for a timeout.

Result: NIL iff the test was not successful (due to PREDICATE returning NIL, a NOBIND being returned, a timeout occurring, or a deep exit (such as an abort) occurring). Non-NIL indicates success.

Description: This function evaluates the expression EXPRESSION and checks the result with the predicate PREDICATE, returning the result from calling PREDICATE. If NOBIND is returned from either EXPRESSION or PREDICATE, then an error message is printed (to NIL) and a NIL is returned from SINGLE-TEST. If the timeout is exceeded (and timeouts can be checked) then the evaluation of the expression is aborted and an error message is printed (to NIL) and a NIL is returned from SINGLE-TEST. If a deep exit occured in either EXPRESSION or PREDICATE (e.g. from aborting of the expression), then an error message is printed (to NIL) and a NIL is returned from SINGLE-TEST.

Side Effects: A message can be printed (to NIL).

Assumptions: Deep exits completely out of EXPRESSION or PREDICATE are not part of the successful behaviour of either EXPRESSION or PREDICATE (any such exits must be caught internally within EXPRESSION or PREDICATE). Note that deep exits are caught via ERRORSET, so RETFROM, RETTO, RETEVAL, RESUME, etc. are not caught.

There is a function available which prints out an easily identifiable error message in a standard format to the standard ouput. Thisfunction has the following interface (all arguments must be supplied):

Name: TEST-MESSAGE (LAMBDA function).

Arguments:

IDENTIFIER The integer identifier of this test (as given to SINGLE-TEST).

TEXT The text of the error message.

INFO Information specific to this instance of this error.

Result: Not useful.

Description: The error message along with the test identifier and the specific information is printed to NIL in a standard, easy to notice format.

Side Effects: A message is printed (to NIL).

Assumptions: None.

Some side-effects of the automated test harness are:

1. The History List for the Programmer's Assistant is used, therefore old items are lost an
d `REDO`, etc. immediately after testing will redo the last command that the automated test harness performed, not the last item printed in the top level typescript window.
2. The top level value and the value in the Programmer's Assistant of `HELPFLAG` are changed for the duration of running a test suite.
3. Extra processes are run to perform the testing.

Known deficiencies with the implementation are:

1. `ABORTi`ng and PAUSEing can only be done between individual tests.
2. If a test is aborted between individual tests, but not between tests suites, then the effec
s of LOADing and running that test suite are not UNDOne.
3. Some errors are not caught, and some side effects are not undone if errors occur.

Some possible extensions to this package are:

1. Utilities to help with testing for deliberate errors.
2. Utilities to help with automating input which would normally be manual.