TOL Integrity Tests

4th May 2004

Daniel Rus Morales <danirus@tol-project.org>

v1.0, 28-04-2004

Revision History

Revision 1.0	30-04-2004	Revised by: danirus

TIT are based on a Tol recursive function written on test-it.tol

Abstract

This document provides Tol-Project users and developers with guidelines for understanding and developing Tol Integrity Tests.

Contents

1 Introduction

Since Tol was written until today the language has improved a lot and has changed from older versions to the present ones. Between changes from ones to another versions Tol has increased their complexity and functionality, doing the language more difficult to maintain. When new features are implemented or bugs are resolved some Tol core code has to be changed. Those changes could generates not expected side effects in other functionalities not related with new implemented features or with bugs resolved.

One way to avoid side effects while programming is to get the better knowledge of all Tol code, as a whole. But while this happens, and the beard grows, is necesessary to look for a complementary solution.

Tol Integrity Tests tries to be a mechanism that ensure us, at any moment, that certain functionalities continue being presents and give us valid results.

2 What is Tol Integrity Tests?

Tol Integrity Tests are small Tol programs that check for functionalities. Each small program check for certain functionality and return us the result. Each test program is implemented using a small template. This template has defined three parts; first part is a message that tell us the functionality that is going to be checked, second part is the test block of code, and third part is another message that informs of the results obtained in the second part. The first and third part are very simple, but the second part has to be more complicated, depending on the functionality we want to check.

2.1 Tests based on a template

Lets go to see the template used to check each functionality, that is, the template used to write each file in `tests` directories.

Text **[bold and between brackets]** has to be replaced by real code:

`// Test Description` `Text (descTest := "[Testing ___ ...]");`	first part

`// Test Operation`
`Text (`
`result := {`
`[tol code to do the test]`
 	
`// Test Condition`
`If(`**`[test condition]`**`, "Ok", "Error")`
`}`
`);`
	second part

`// Show Results` `WriteLn(descTest + result);`	third part

As we can see in the table above, the first part only need one line of text explaining the test that is going to be happen. This message are shown when somebody invoke the first test file, located in root directory of Tol source code.

3 How are organized?

Under each directory of source code there is a "tests" directory with test files, in order to have the test files as close to the code files as possible. In that way, test files to prove functions of Set Tol type are under `btol/set_type/tests/`, and test files to prove functionalities of parser are under `bparser/tests/`

3.1 Starting tests: test-it.tol

To make a whole test you have to execute file test-it.tol under root directory of source code. This file use a Tol recurvise function looking for "tests" directories and executing each file found with "tol" extension.

4 How you can contribute?

Majority of tol code in TIT comes from bugs submitted to bugzilla (in http://bugs.tol-project.org), but more complex functionalities, like functions of statistical analysis, need to be written from scratch.

If you are an experienced Tol user, you can help us writing Tol Integrity Tests.