LLVM CLANG - RANDSTRUCT

SOFTWARE VERIFICATION AND VALIDATION PLAN

Author : Jordan Cantrell, Nikk Forbus
Approved by :Tim Pugh
Released by : Tim Pugh

CHANGE RECORD

REVISION	DATE	AUTHOR	SECTIONS/PAGES AFFECTED
	REMARKS		
v0.01	29NOV18	Cantrell, Forbus	All
	Initial creation of verification document.		
v.0.02	30NOV18	All	All
	Inspection and formatting		

1 Introduction

1.1 Purpose

The purpose of this document is to describe the testing, validation, and verification process for the randstruct module for LLVM/Clang.

2 Testing Approach

2.1 Integration Testing

Compile test source(s) with randstruct activated.Test succeeds if compilation completes without randstruct-caused program crashes, and fails otherwise.

2.2 Conversion Testing

Compile test source(s) with randstruct. If conversion runs correctly, resulting in a pseudo-randomized order, test is successful, and fails otherwise.

2.3 Interface Testing

Compile test source(s) with command line arguments (if any). Test succeeds if compilation occurs with appropriate arguments, and fails otherwise.

2.4 Performance Testing

Compile similar/identical objects with randstruct toggled on/off - measure differences in time performance. Test is considered successful based on undetermined arbitrary value, and fails otherwise.

2.5 Regression Testing

Re-verify tests after any changes. Run Clang/LLVM regression tests.

2.6 Acceptance Testing

Clone our code from a repo available to the customer and verify all tests on said copy of code. Acceptance testing is successful if all the tests pass verification and customer accepts, and fails otherwise.

3 Test Description

Disclaimer: Specificity of testing will be determined during the design phase of the software component (TBD).

3.1 Integration Test

Ensures that regardless of whether or not the randstruct component behaves as required, the program itself does not crash for having utilized it.

3.1.1 Functions

TBD

3.1.2 Means of Control

Automatic

3.1.3 Test Data

3.1.3.1 Input Data

No specific data, only the randstruct component being called by the compiler.

3.1.3.2 Output Data

A confirmation of success or fail of randomization.

3.1.4 Test Procedure

Compile test source(s) with randstruct activated. Test succeeds if compilation completes without randstruct-caused program crashes, and fails otherwise.

3.2 Conversion Test

Ensures that when randstruct is called by the compiler, that the pseudo-randomized struct returns a modified struct with appropriately randomized fields.

3.2.1 Functions

TBD

3.2.2 Means of Control

Manual

3.2.3 Test Data

3.2.3.1 Input Data

A set of C++ source files, containing structs to be randomized

3.2.3.2 Output Data

A set of binary files, to be examined via software tools such as gdb, with members rearranged randomly.

3.2.4 Test Procedure

Build input sources with randstruct activated. Examine the structs in the output binaries. Note the order of struct members in the binary files. If this order differs from the order in the source files, the test is successful.

3.3 Interface Test

Ensures that randstruct component is activated when the appropriate compiler flag(s) are used.

3.3.1 Functions

TBD

3.3.2 Means of Control

Automatic

3.3.3 Test Data

3.3.3.1 Input Data

Command line with arguments.

3.3.3.2 Output Data

Upon successful execution, no output data. Otherwise, an error message denoting the use of inappropriate commands, arguments, or arg count.

3.3.4 Test Procedure

Compile test source(s) with command line arguments (if any). Test succeeds if compilation occurs with appropriate arguments, and fails otherwise.

3.4 Performance Test

Ensures that when the randstruct component is called by the compiler, that the time required to compile meets arbitrary requirements for time consumption.

3.4.1 Functions

TBD

3.4.2 Means of Control

Manual

3.4.3 Test Data

3.4.3.1 Input Data

A struct to be randomized.

3.4.3.2 Output Data

A generated performance analysis denoting a comparison between time spent compiling the struct with and without the randstruct component being called.

3.4.4 Test Procedure

Compile similar/identical objects with randstruct toggled on/off - measure differences in time performance. Test is considered successful based on undetermined arbitrary value, and fails otherwise.

3.5 Regression Test

Ensures that new changes to the randstruct code do not break other tests.

3.5.1 Functions

TBD

3.5.2 Means of Control

Automatic

3.5.3 Test Data

3.5.3.1 Input Data

A set of C++ source files, comprising the randstruct module

3.5.3.2 Output Data

A set of test results produced by running the previously defined tests.

3.5.4 Test Procedure

Build randstruct with the newest code changes, and then re-verify all tests after any changes. Run Clang/LLVM regression tests.

4 Environmental Requirements

4.1 Hardware

System running supported platforms (see below).

4.2 Software

Supported Platforms:

- Linux

- Solaris
- FreeBSD
- NetBSD
- MacOS
- Cygwin/Win32
- Windows

LLVM Clang and all its build requirements
Randstruct Component

4.3 Tools

TravisCI - Automated software testing tool integrated with GitHub