# The FreeBSD Kernel Stress Test Suite

Peter Holm
*The FreeBSD Project*
peter@holm.cc

## Abstract

This paper presents the second generation kernel stability test suite. Experience with the previous script based version and reasons to move on to a new one are described.

## 1 Motivation

Uptime is by far one of the more important feature in an operating system. It does not matter how advanced feature an OS provides, if it keeps crashing all time time.

In order to weed out those problems it is important that as many people as possible, on as many different hardware configurations, test the OS before a release.

A test suite must therefore be straightforward to use and easy to extend, to accommodate the widest range of developers.

## 2 Prior Work

### 2.1 FreeBSD

The previous version [1] of the test suite was mostly implemented as shell scripts and has been in use with the FreeBSD Project since June 1999. Since then it has evolved into 27 different tests. In May 2003 logging of problems found, started as a way to promote the test suite.

This is a graph of some 100 problems found:

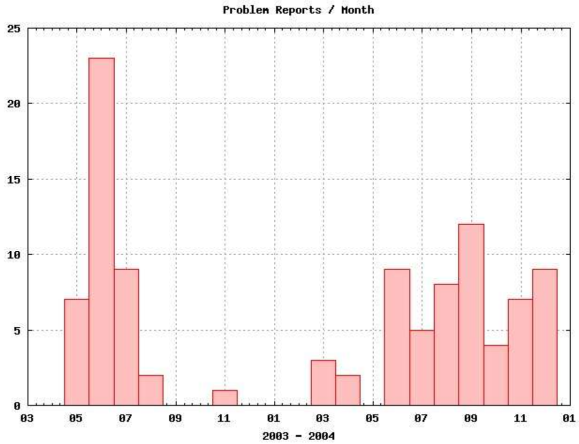

*Illustration 1 Problems found with the script based version of the test suite.*

So as a prototype, the script based version of a Kernel Stress Test Suite seems to have succeeded.

### 2.2 Linux

There is a Linux Test Project [2] that combines both stress- and functionality testing. One of the subtests are also available to FreeBSD: The Open POSIX Test Suite [3]. Another subtest, The Ballista Project [4], implements stress testing of syscalls, but is not straight forward to port.

## 3 The Test Suite

The key functionality of this test suite is that it runs a random number of test programs for a random period, in random incarnations and in random sequence. This very simple idea is implemented in the run test program, which controls the behavior of all the other test programs.

To simplify writing test programs, a test harness is implemented as a library that handles running the test programs. All the creator of new test programs has to implement are three procedures: setup(), cleanup() and test().

### 3.1 The run Program

The run program is the control program that runs all the test programs. All tests are run for a default period of 60 seconds. Each test program sleeps for a random period between 1 and 10 seconds before starting their test. A load factor is used to determine if each test should skip a test cycle. Each program is started in a random number of incarnations.

### 3.2 The Test Programs

Only a few test programs have been implemented so far.

The creat program tests the creat() system call by creating lots of empty files and then deleting them again.

The mkdir program test the mkdir() system call by creating deep directories and then deleting them again.

The syscall program tests system calls by passing random argument values. The primary goal is to ensure that all syscalls validate user arguments.

The swap program uses up memory to ensure that swapping occurs. This is not a test in it selves, but prosuces system load.

## 3.3 Configuration

Each test program accepts a set of predefined options as well as environment variables. The full test relies on a set of environment variables controlling the execution of the tests.

## 3.4 Running

The tests may be run individually in their respective directories or as whole from the top directory by the run.sh shell script. The shell script accepts one optional parameter; the configuration file.

## 3.5 Known Problems

One drawback with the "random all" method is that recreating panics during bug fix testing can be time consuming. Depending on the type of problem it may take any where from minutes to days of testing to get a panic again.

## 4 Conclusion

The new version of the stress test suite should reach a broader range of kernel developers and thus hopefully catch even more kernel bugs.

## 5 Availability

http://www.holm.cc/stress/src/stress2.tgz

## 6 References

[1] The initial version of The Kernel Stress Test Suite is available at http://www.holm.cc/stress/src/stress.tgz
[2] The Linux Test Project test suite http://ltp.sourceforge.net/
[3] The Open POSIX Test Suite http://posixtest.sourceforge.net/
[4] The Ballista Project http://www-2.cs.cmu.edu/afs/cs/project/edrc-ballista/www/index.html