Kernel Debugging User-Space API Library (KDUAL)

John Livingston
Computer Systems Laboratory
Thomas Jefferson High School for Science and Technology
January 21, 2005

Abstract

The purpose of this project is to create an implementation of much of the kernel API that functions in user space, the normal environment that processes run in. The issue with testing kernel code is that the live kernel runs in kernel space, a separate area that deals with hardware interaction and management of all the other processes. Kernel space debuggers are unreliable and very limited in scope; a kernel failure can hardly dump useful error information because there's no operating system left to write that information to disk.

Kernel development is quite likely the most important active project in the Linux community. Any aids to the development process would be appreciated by the entire kernel development team, allowing them to do their work faster and pass changes along to the end user quicker. This program will make a direct contribution to kernel developers, but an indirect contribution to every future user of Linux.

Introduction and Background

The Linux kernel is arguably the most complex piece of software ever crafted. It must be held to the most stringent standards of performance, as any malfunction, or worse, security flaw, could be potentially fatal for a critical application. However, because of the nature of the kernel and its close interaction with hardware, it's extremely difficult to debug kernel code. The goal of this project is to create a C library that provides the kernel API, but operates in ordinary user space, without actual interaction with the underlying system. Kernel code currently being tested can then be compiled against this library for testing without the risks and confusion of testing it on a live system.

Process

The design of this API has an extremely simple development process: Research, code, debug. Sub-tasks are somewhat difficult to define, as the library cannot do very much of use until complete. However, the rapidly growing source code, along with small demonstrations of sections of the library, is sufficient for progress reporting purposes. Development thus far has simple, no special tools have been needed beyond the vim editor, the GNU C compiler and linker, and a very large amount of work time. Testing of the library with simple functions will be trivial, it is the eventual goal of this project construct a small patch to the kernel using this library both as a demonstration of the library's effectiveness and to solve an existing problem. This patch would allow seamless use of the Andrew File System (AFS) with the 2.6.x kernel, greatly benefiting the lab's workstations by allowing an immediate migration to 2.6, which has large improvements.

On a more detailed level, I have been implementing sections of the Linux VFS, as well as math processing. VFS is necessary to handle “file interaction” in the virtual kernel, while most of the mathematical work has been to optimize basic functions (add, subtract, compare, etc.) using x86 assembly. Because this library attempts to simulate a program that uses hardware directly for computation, its own internal simulation of that computation must be as fast as possible. It will never reach anywhere near the speed of the actual kernel, but the speed difference between the original C syntax for addition and its equivalent in inline assembly is a tenfold increase.

These two sections of the kernel library will be my primary contribution to this project. Current code for the two spans several thousand lines. The majority of the codebase is written; however, minor changes, fixes, and improvements will still require significant effort. This project's success is dependent on efficiency as much as simple functionality.

References

http://www.kernel.org/
The Linux Kernel Archives.

http://www.debian.org/
The Debian distribution of Linux, the best one, not to start a flame war or anything.

http://lkml.org
The archive of the Linux Kernel Mailing List, the primary method of communication for kernel developers

http://www.openafs.org
An open source implementation of the Andrew File System.