Project Proposal: Visualizing strace

Cynthia Ma - cma1, Sola Park - spark1

October 3, 2013

1 Abstract

The list of system calls that strace outputs is useful information that is difficult to grasp because the language and syntax is unfamiliar for programmers who visualize the computer system from a higher level. Coming from this perspective ourselves, our goal for this project is to parse this information and display it visually in a manner that will be more intuitive to understand. Specifically, we will start by displaying the behavior of a single system call, then expanding to include multiple calls in a trace to grasp a bigger picture. In the process, we hope to find a visualization method that best conveys information about the computer system as a whole.

2 Motivation

When programs crash, or computers run out of memory, or simply run slower, there is a desire to understand why. What goes on underneath code and the interface of a program, however, is easy to neglect and hard to understand because it is not as tangible and the information abstract. strace is an example that demonstrates this problem. It is a tool used to see what happens between a program and an operating system's kernel, but its output is simply a list of system calls. As people who are interested in computer science but unfamiliar with operating systems, we find it meaningful to try different visualization schemes in search for a method that would help users better digest this information. We expect our project to be mostly used by those who are experienced with software programming in general but not so much experts on computer systems.

3 Background

We haven't found a previous attempt at making strace output easier to understand through visualization, although there is code named vistrace that creates a circular graph of strace output in a manner that appears to emphasize aesthetics over comprehension. Researching the broader topic of software visualization gave us references to previous efforts in visualizing traces in general. Specifically, we found two papers that discuss visualizing traces of events and provide inspiration for different visualization schemes that can be applied for our project. First, Zinsight [1] visualized large event traces in multiple displays, two of which seem usable for our project. The first is a two dimensional graph where time is mapped along the y-axis, event-associated information is mapped along the x-axis, and events are organized by color. The second is displaying the events in a tree-like structure. In the first, it is easier to see patterns like loops, while in the second, it is easier to see paths leading from one event to the next. A third possible display was found in the second paper about visualizing Windows system traces [2] using a binary DotPlot display to visualize event similarity between two traces.

4 Idea

Our main idea is to parse the output data returned by strace and display it visually in a comprehensive manner for our users. We will start out by first visualizing strace in a narrow scope by displaying information about a single trace, such as its path or general statistics. We hope to then expand into a wider scope of viewing these system calls as events in a larger context of the program and kernal, and also compare multiple traces.

5 Goals

Milestone 1: In general, we will read papers to better understand strace and the syntax of its output. By the first milestone meeting, we hope to have implemented a basic visualization of a single system call, with code that can take a file of an strace output and parse the information into seperate system calls.

Milestone 2: By this milestone meeting, we hope to demonstrate a visualization of a single strace, likely by using the tree method or some user-interactive step-through graph between program and kernel.

Milestone 3: By the third milestone meeting, we will expand our visualization to display the kernel architecture in more detail that will hopefully provide a wider context in which the system calls are made, whereas the previous effort would have been about analyzing each system call individually.

References

[1] W. De Pauw and Steve Heisig. Zinsight: A visual and analytic environment for exploring large event traces. *SOFTVIS'10*, pages 143–152, 2010.

[2] Y. Wu, Roland H.C. Yap, and Felix Halim. Visualizing windows system traces. *SOFTVIS'10*, pages 123–132, 2010.