CS 502 Operating Systems — WPI, Fall 2003
Craig E. Wills — Project 5 (20 pts)
Assigned: Monday, December 1, 2003 — Due: Thursday, December 18, 2003 (5:00pm)

Introduction

An interesting approach for studying the behavior of an application is to trace the set of system calls invoked during the lifetime of the application. The *strace* command in the Linux Operating System executes an application given as arguments to it and in prints all system calls and their arguments used by the application. In this project you will use *strace* to analyze the behavior of different applications.

To begin the project, you should use *strace* to execute a few commands and observe the output. The system call trace printed by *strace* is printed to stderr rather than stdout. This distinction allows it to be redirected to an alternate location. A small shell script in `/cs/cs502/public/bin/dostrace` is available for you to use. This script executes the command given to it with *strace* and redirects the trace of system calls to a log file. Use of the script also allows interactive commands to be more easily traced. For example, using the script with *ls* yields:

```
% dostrace ls
<output of ls command>
strace output in ls.slog
```

The resulting log file can then be analyzed. You should create a program *traceanal* to analyze a log trace file (or the output of *strace* directly). Your program should read the trace of system calls from stdin so it can be invoked either as "`traceanal < ls.slog`" using the log file generated above or "`strace ls |& traceanal`". Your *traceanal* program should strip out any system call arguments and examine only the sequence of system calls. It will also need to ignore command output if you use the latter invocation.

Your *traceanal* program should determine the frequency and sequence of system calls used by a command. Your program should first determine the count for each system call in the trace. If you use a programming language such as C or C++, you will need to create a hash function to map system call names to a hash table. You are also welcome to use a scripting language, such as Perl, for creating *traceanal*. Once your program can determine a count for each system call, you should modify it to also determine the next system call that is used. Your program should print the count for each system call as well as the count for each subsequent system call (indented) as follows:

```
% traceanal < ls.slog
open 17
    mmap 6
    read 4
    write 2
    close 5
...
```

indicating that the *open()* system call is used 17 times in the trace. It is followed 6 times by a call to *mmap()*, 4 times by a call to *read()*, 2 times by a call to *write()*, and 5 times to a call to *close()*. Your program should print similar information for all system calls used in a trace.

Analysis

Once you have your analysis program working, you need to use it to analyze the behavior of different programs on a system. You should use it to investigate the following issues:

1. Observation of program execution behaviors shows that many system calls are invoked as part of starting up a program. To examine this start-up behavior, construct a simple program that makes no system calls and analyze it using *traceanal*. Do all programs exhibit a similar start-up behavior in terms of which system calls are used and their relative sequence?

2. Researchers have proposed using the system call sequence of a program as a "signature" for that program as a means to detect if a copy of a program is substituted by an intruder. Investigate the validity of this idea by checking if the signature of different executions of the same program are the same. The particular counts of system calls may vary, but are the sequences similar? What if different command line arguments are used for a command? Is there variation in the sequence? Does the sequence change if the amount of data or duration of execution varies for a program?

3. How much variation and commonality do you observe from invocations of different commands? You should try to separate out the start-up behavior common to all commands and the command-specific portion.

Submission of Project

Use `/cs/bin/turnin` to turn in your project using the assignment name "proj5". You should submit the source code for your *traceanal* program. A hard-copy report on the analysis using *traceanal* should be turned in to the instructor.