Linux Shared Library Profiler Implementation

Edward Ryabikov, Mark Zaslavskiy, Kirill Krinkin
FRUCT LETI Lab
Saint-Petersburg, Russia
{edward.ryabikov, mark.zaslavskiy}@gmail.com, kirill.krinkin@fruct.org

Abstract

Today many different profilers exist for C/C++ shared libraries in Linux-based systems. But most of them have significant disadvantages - usage of special environment, access to sources, recompilation or relinking of target binary, dependency from processor architecture, no possibility for dynamically loaded functions profiling, profiling of all functions a at the same time. These disadvantages cause next problems – inaccurate measurements, big memory consumption, big amount of result data, different behavior of profiled program. Goal of this work was to create lightweight profiler which is free from these issues. Problems were solved by creation of special version for standard Linux dynamic linker and loader – ld-linux.so.

During the development process next tasks were solved – mechanisms for time measurement and statistic collecting were implemented; libdl.so and ld-linux.so sources were modified for receiving profiling options from user, taking measurements, performing output of profiling results; creation of shared library, which allowed to profile dynamically linked and loaded functions at the same time.

Mechanisms of wrapper and redirector functions were implemented for performing profiling. Wrapper is a regular cdecl function, the same for all profiled functions, which performs next actions - registers saving, time measurement of the function starting time, call of the function, measurement of the function end time, registers restoring. Redirector is a generated set of machine instructions, unique for each profiled function, which pass functions address to wrapper and perform transition to wrapper. During relocation and dynamic load process for profiled function binary receive address of corresponding redirector, instead of actual function address; for functions which are not profiled binary receive real function address.

Independence from processor architecture was achieved partly by creation two versions of profiler – for x86 and x86-64 platforms, which differs only by redirector generation code and wrapper code.

Proposed solution was implemented as eglibc patch. For its usage user needs only to build modified version of eglibc and to install it in separated directory. Solution requires minimal modification of profiled binary environment, because in most of Linux-based systems user can specify any other version of ld-linux.so for binary files by modification of Interpreter field in ELF header or directly by running wanted interpreter in command line.

Index Terms: ELF, Profiling, Shared Libraries, Relocations, Cdecl, Dynamic Linking, Dynamic Loading.