Peking University

Google Summer of Code 2015

The FreeBSD Project

Libnetstat Libraries

Author:
Hao Sun

Mentor:
Robert Watson
Gabor Pali

March 27, 2015

1 Introduction

This project is about creating wrapper libraries to support network monitoring and management applications to avoid direct use of kvm(3) and sysctl(3) interfaces. This approach would allow the kernel implementation to change and monitoring applications to be extended without breaking applications and requiring them to be recompiled. For this project, we propose to provide such a library for the network statistics functions (i.e. libnetstat(3)) as a proof of concept, together with methods for building similar libraries for other subsystems.

2 Project Goals

The project goals contain the following aspects: general targets, stretch goals, deliverables, test plans and benefits for the community.

2.1 General Targets

1. Add kernel support to export data in a less ABI-sensitive way using sysctl(3) and kvm(3). The goal is to allow the kernel implementation to change without breaking applications and requiring them to be recompiled, and to allow monitoring functions to be extended without breaking applications.
2. Write a library to present the information gathered by the first target in an extensible way to applications. This is libnetstat(3). It is derived from the sources of netstat(1) with the missing kvm(3) and sysctl(3) interfaces added where needed.
3. Update applications (e.g. netstat(1) and bsnmpd(1)) to use the library instead of reaching directly into kernel memory/consuming sysctls.

2.2 "Stretch Goals"

If there is extra time after finishing the general targets, some "Stretch Goals" are designed as follows.

1. Add manual pages, required documentations and demos of the libnetstat(3) library.

2. Merge the libnetstat(3) library into the base system.

2.3 Deliverables

1. Repository with the libnetstat(3) library.
2. Test cases to make sure the libnetstat(3) library works well.
3. Updated netstat(1) and bsnmpd(1) application using the libnetstat(3) library.
4. Project manual page with detailed descriptions and perhaps some demos.

2.4 Test Plans

The test jobs of the project contain two parts: testing the libnetstat(3) library and testing the modified netstat(1) and bsnmpd(1) application.

On the one hand, we plan to write unit tests for every module of the libnetstat(3) library, and the testing and coding will be conducted simultaneously. The exit criteria of the libnetstat(3) library is finishing all the features and passing all the test cases.

On the other hand, we also modify the netstat(1) and bsnmpd(1) application. This process would be helpful for testing the robustness of the libnetstat(3) library. Also we need to run the test cases of the netstat(1) and bsnmpd(1) to make sure they work well after applying the new library.

2.5 Benefits for the FreeBSD Project

The benefits for the FreeBSD project can be summarized as follows.

1. Robustness for Application Binary Interface (ABI) over time. A library interface, rather than a system call, sysctl(8) variable, or kernel memory interface is the documented interface, and is designed to be robust against common changes to kernel data structures. Some data structures grows frequently as TCP gains new features, but the current sysctl(3) interface is not robust against that growth.

2. Robustness for different word sizes. 32-bit monitoring tools should be able to work on a 64-bit kernel.

3. Improved consistency when multiple sources are available. For example, there used to be a number of bugs involving netstat(1) running on crash dumps, where it will once in a while show a value from the current running kernel instead because a new statistics for sysctl(8) has been added with no crash dump use in mind.

3 Implementation Details

The sysctl() function retrieves system information and allows processes with appropriate privileges to set system information. The information available from sysctl() consists of integers, strings, and tables. Information may be retrieved and set from the command interface using the sysctl(8) utility.

The signature of sysctl() function is as follows.

```
int
sysctl(const int *name, u_int namelen, void *oldp, size_t *oldlenp,
    const void *newp, size_t newlen);
```

The input state is described using a "Management Information Base" (MIB) style name. There are 8 top-level names, respectively `CTL_DEBUG`, `CTL_VFS`, `CTL_HW`, `CTL_KERN`, `CTL_MACHDEP`, `CTL_NET`, `CTL_USER` and `CTL_VM`. Note that the `CTL_NET` is related to networking, which needs us to pay attention to.

The kvm(3) library provides a uniform interface for accessing kernel virtual memory images, including live systems and crash dumps. Access to live systems is via sysctl(3) for some functions. That's why we need to create abstractions for the kvm(3) library. Using the kvm(3) library, memory can be read and written, kernel symbol addresses can be looked up efficiently, and information about user processes can be gathered.

After a brief introduction to the subroutines we need to create abstractions. We dive into details and summarize the main abstractions we need to create as follows.

1. Memory access abstractions. This corresponds to the kvm(3) library. We'll

try to use the libmemstat(3) library as many as we can. If there are memory access requirements that cannot be met. We'll create new abstractions for these specific cases.

2. Routing abstractions. This corresponds to the `PF_ROUTE` level of the sysctl(3). The routing module should be able to return the entire routing table or a subset of it. Also it should allow the user to query about the information about multicast group memberships on all interfaces or any specific interfaces.

3. Network protocols abstractions. This corresponds to the `PF_INET` and `PF_INET6` levels of the sysctl(3). In this module, we need to support a batch of network protocols, e.g. IP, TCP, UDP, ICMP, IPv6 etc. We may also add the abstractions of sockets in this module.

4 Time Schedule

We divide the summer project process into 8 stages. The time slot and task description of each stage are shown as follows.

1. (*Apr. 1 - May 25*) Warm up. We'll build the the "world" version of FreeBSD, and then install the system together with setting up the development environment. Hopefully after this step we'll get familiar with the FreeBSD development process and be well prepared for the next stages.

2. (*May 26 - June 3*) Requirement gathering. The main task of our project is to create wrapper libraries to support monitoring and management applications to avoid direct use of kvm(3) and sysctl(3) interfaces. In this step, we plan to read the kvm(3) and sysctl(3) implementations and filter out the interfaces and functions which are related to networks. Then we'll classify all the requirements in order to get a module list for the libnetstat(3) library. After this stage, we should have a vague idea about creating the abstractions for the modules.

3. (*June 4 - June 10*) Case study. In this stage, I want to learn about the libmemstat(3) and existing libnetstat implementations in order to get inspirations and reusable code snippets. After the case study, hopefully I'll be cleared how to make the abstraction.

4. (*June 11 - June 25*) Create abstraction. After Step 2 and 3, We'll start to write code from sketch and the interfaces should be well defined after this step.

5. (*June 25 - July 25*) Core implementation. Based on the kvm(3) and sysctl(3), We'll create the full implementation for the library.

6. (*July 26 - July 31*) Applications update preparation. Now our library is ready! In this step we'd like to read the implementation of netstat(1) and bsnmpd(1) and point out which part could be replaced with our new library.

7. (*Aug. 1 - Aug 10*) Applications update. Update the netstat(1) and bsnmpd(1), also test the new library.

8. (*Aug. 11 - Aug 24*) Documentation and buffer time. Maybe we need to create a manual page for the library. We also reserved some buffer time at this stage.

5 General Information

Name Hao Sun

Email sunhao2013@gmail.com

Phone +86 18001229679

Availability 40 hours per week

Biography My name is Hao Sun, a third year graduate student of Peking University, China. My major is computer science, and I will graduate in July this year and start to work in Facebook in October. Up to now I have 4 internship experiences respectively in Microsoft (twice), Adobe and Multicoreware.

Most of my projects are finished during the internship, including Microsoft Server Performance Advisor (SPA) 3.1, OpenCL Enabled Face Detection Plug-in for IrfanView, and Adobe Content Intelligence Toolkit.

I'm familiar with common programming languages, e.g. C/C++, Python and Java. Also I'm acquainted with operating systems, computer networks and information security, which are my direction of interest at the same time.

This is the first time that I apply to join an open source community. I eager to have intuitive feelings of open source culture and to share my ideas and write codes with you talented programmers. This is why I plan to apply for the projects of GSoC.

Possible Mentor Robert WATSON and Gabor PALI, we've discussed several issues about the project via email and I also shared my ideas and findings of the project with them.