Analyzing a Decade of Linux System Calls

An Extended Abstract of a Paper Published in the Empirical Software Engineering Journal

(communicated by Harald Gall)

Mojtaba Bagherzadeh, Nafiseh Kahani, Cor-Paul Bezemer,
Ahmed E. Hassan , Juergen Dingel, James R. Cordy
School of Computing, Queen's University, Kingston, Canada
{mojtaba,kahani,bezemer,ahmed,dingel,cordy}@cs.queensu.ca

EXTENDED ABSTRACT

The Linux kernel provides its services to the application layer using so-called system calls. All system calls combined form the Application Programming Interface (API) of the kernel. Hence, system calls provide us with a window into the development process and design decisions that are made for the Linux kernel. Our paper [1] presents the result of an empirical study of the changes (8,770) that were made to the system calls during the last decade (i.e., from April 2005 to December 2014). The main contributions and most important findings of our study are:

(1) An overview of the Linux system calls. As of December 2014, 396 system calls existed in the Linux kernel. They can be categorized into 10 groups (process management, signal processing, and so on). 76 of the system calls were added over the last decade (new system calls). A new system call is usually not activated for all architectures at the same time. 40 out of 76 (53%) new system calls and 102 of the 393 (26%) existing system calls were sibling calls. A sibling call is a system call that is similar in functionality, and often in name, to another system call.

(2) A study of the evolution of the Linux system calls over the last decade in terms of the size and type of changes that were made to the system calls. With an average growth of 25 LOC per day, the Linux system calls are relatively stable. The commits that are made to system calls are slightly more scattered than kernel commits. There exists a small group of very active system call developers. 8,288 of the 8,770 studied commits (95%) were made to maintain, improve and fix bugs in system calls. 36% of the system call-related commits were bug fixes. 4,498 (50%) of the commits were made to only 25 (6%) of the 393 system calls. 35% of the system call-related commits were made to conduct code restructuring, and 36% of the system call-related commits were made to fix bugs.

(3) A study of the type of bug fixes that were made to the system calls over the last decade. Developers make mistakes in the seemingly trivial activation process of a system call. The steps that are required to activate a system call, such as assigning the unique number and updating the system call table, are performed manually. 58% of the bug fix commits were made to fix semantic bugs. The proportion of bug fixes that fixed memory-related bugs remained constant throughout the last decade.

(4) An analysis of the results and a discussion of their implications.

Generalizability of results. We compared Linux system calls with FreeBSD system calls, to validate that our results are generalizable to other UNIX-based operating systems. Our findings for the FreeBSD operating system confirm that other UNIX-based operating systems use a system call mechanism that is similar to that of Linux. Therefore, we can safely assume that our findings are of value to other UNIX-based operating systems.

Suggestion for automation. First, we suggest the automation of simple, reoccurring tasks in Linux, such as adding and removing system calls. Our study on FreeBSD shows that such tasks can successfully be automated. Second, we suggest that historical information about the evolution of a kernel API should be used to guide the testing process. Finally, we suggest that existing automated testing tools are extended to support testing system calls.

Maintenance Effort. Compared to regular software systems, kernel APIs require an additional type of maintenance that involves adding and removing system calls. Also, approximately 11% of the maintenance effort of a kernel API is assigned to the infrastructure for providing the API.

Overall, the results of our study can be beneficial to practitioners, researchers, and more specifically kernel developers, by providing insights related to the challenges and problems that come with long term maintenance of a kernel API, such as the long-lived Linux kernel API. We have published our classification of 8,870 system call-related changes [1] so that it can be used to conduct further studies.

The full paper is accepted for publication in the Empirical Software Engineering journal, and can be found at: https://link.springer.com/article/10.1007/s10664-017-9551-z.

ICSE '18, May 27-June 3, 2018, Gothenburg, Sweden
© 2018 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-5638-1/18/05.
https://doi.org/10.1145/3180155.3182518

KEYWORDS

Linux kernel - System calls - API evolution - Software evolution

REFERENCES

[1] Mojtaba Bagherzadeh, Nafiseh Kahani, Cor-Paul Bezemer, Ahmed E Hassan, Juergen Dingel, and James R. Cordy. 2017. Analyzing a decade of Linux system calls. *Empirical Software Engineering* (2017). to appear, accepted September 2017.