What is Linux System Calls and Library Functions?

by Himanshu Arora on July 4, 2012

Computer software are developed to either automate some tasks or solve some problems. Either way, a software achieves the goal with the help of the logic that the developer of that software writes. Every logic requires some services like computing the length of a string, opening a file etc. Standard services are catered by some functions or calls that are provided for this purpose only.

Like for calculating string length, there exists a standard function like strlen(), for opening a file, there exists functions like open() and fopen(). We call these functions as standard functions as any application can use them.

These standard functions can be classified into two major categories :

1. Library function calls.
2. System function calls.

In this article, we will try to discuss the concept behind the system and library calls in form of various points and wherever required, I will provide the difference between the two.

1. Library functions Vs System calls

The functions which are a part of standard C library are known as Library functions. For example the standard string manipulation functions like strcmp(), strlen() etc are all library functions.

The functions which change the execution mode of the program from user mode to kernel mode are known as system calls. These calls are required in case some services are required by the program from kernel. For example, if we want to change the date and time of the system or if we want to create a network socket then these services can only be provided by kernel and hence these cases require system calls. For example, socket() is a system call.

2. Why do we need system calls?

System calls acts as entry point to OS kernel. There are certain tasks that can only be done if a process is running in kernel mode. Examples of these tasks can be interacting with hardware etc. So if a process wants to

do such kind of task then it would require itself to be running in kernel mode which is made possible by system calls.

3. Types of library functions

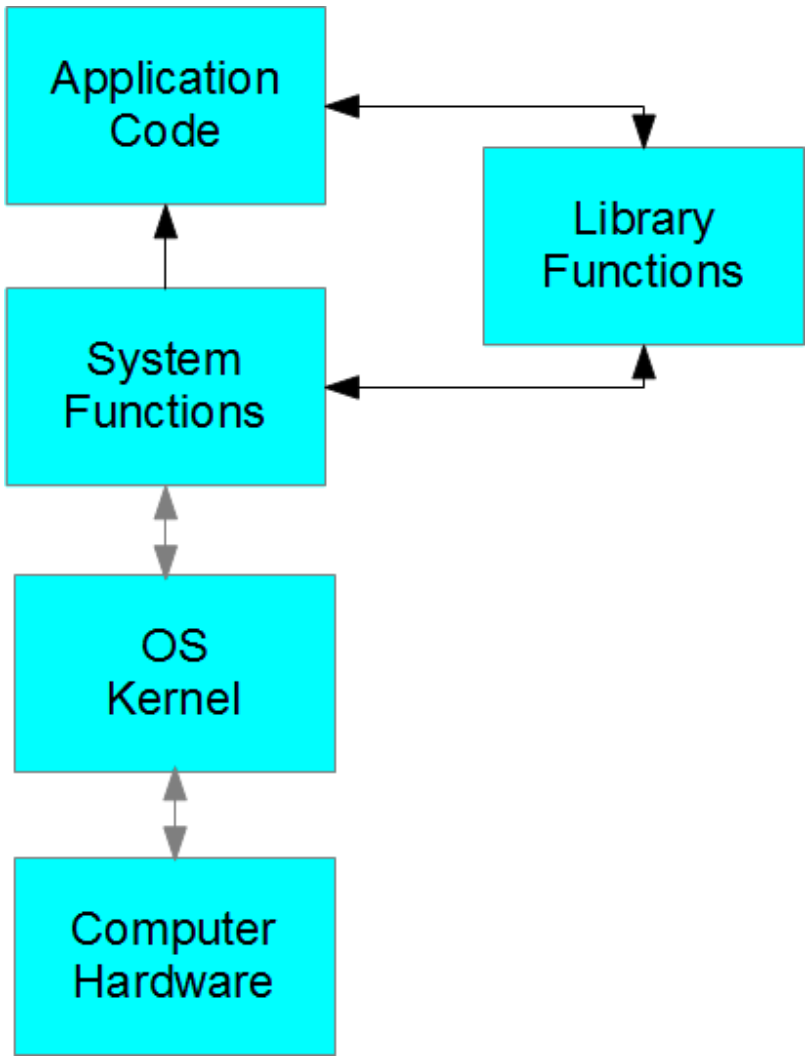
Library functions can be of two types :

- Functions which do not call any system call.
- Functions that make a system call.

There are library functions that do not make any system call. For example, the string manipulation functions like strlen() etc fall under this category. Also, there are library functions that further make system calls, for example the fopen() function which a standard library function but internally uses the open() sytem call.

4. Interaction between components

The following diagram to depict how Library functions, system calls, application code interact with each other.



The diagram above makes it clear that the application code can interact with Library functions or system calls. Also, a library function can also call system function from within. But only system calls have access to kernel which further can access computer hardware.

5. fopen() vs open()

Some of us may argue that why do we have two functions for the same operation ie opening a file?

Well, the answer to this is the fact that fopen() is a library function which provides buffered I/O services for opening a file while open() is a system call that provides non-buffered I/O services. Though open() function is also available for applications to use but application should avoid using it directly.

In general, if a library function corresponding to a system call exists, then applications should use the library function because :

- Library functions are portable which means an application using standard library functions will run on all systems. While on the other hand an application relying on the corresponding system call may not run on every system as system call interface may vary from system to system.
- Sometimes the corresponding library function makes the load to system call lesser resulting in non-frequent switches from user mode to kernel mode. For example if there is an application that reads data from file very frequently, then using fread() instead of read() would provide buffered I/O which means that not every call to fread() would result in a call to system call read(). The fread() may read larger chunk of data(than required by the user) in one go and hence subsequent fread() will not require a call to system function read().

6. Is malloc() a system call?

This is one of the very popular misconception that people have. Lets make it clear that malloc() is not a system call. The function call malloc() is a library function call that further uses the brk() or sbrk() system call for

memory allocation.

7. System calls : Switching execution modes

Traditionally, the mechanism of raising an interrupt of 'int $0x80' to kernel was used. After trapping the interrupt, kernel processes it and changes the execution mode from user to kernel mode. Today, the systenter/sysexit instructions are used for switching the execution mode.

8. Some other differences

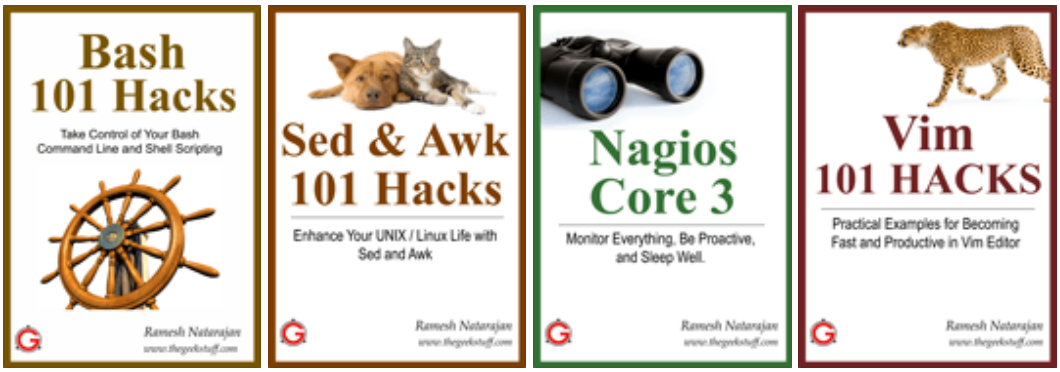
Besides all the above, here are a few more differences between a system and library call :

- A library function is linked to the user program and executes in user space while a system call is not linked to a user program and executes in kernel space.
- A library function execution time is counted in user level time while a system call execution time is counted as a part of system time.
- Library functions can be debugged easily using a debugger while System calls cannot be debugged as they are executed by the kernel.

If you enjoyed this article, you might also like..

1. 50 Linux Sysadmin Tutorials
2. 50 Most Frequently Used Linux Commands (With Examples)
3. Top 25 Best Linux Performance Monitoring and Debugging Tools
4. Mommy, I found it! – 15 Practical Linux Find Command Examples
5. Linux 101 Hacks 2nd Edition eBook Free

- Awk Introduction – 7 Awk Print Examples
- Advanced Sed Substitution Examples
- 8 Essential Vim Editor Navigation Fundamentals
- 25 Most Frequently Used Linux IPTables Rules Examples
- Turbocharge PuTTY with 12 Powerful Add-Ons



Tagged as: System call Vs Function call, System call Vs Library call, System call Vs Library functions

{ 26 comments… add one }

- Jalal Hajigholamali July 4, 2012, 8:51 am

  Hi,

  Very very nice and useful paper…

  Link

- Pierre B. July 5, 2012, 1:08 am

  Thanks, i will be enjoying this article at my lunch break (in 3 hours …)
  Thanks a lot TGS.

  Link
- bob July 5, 2012, 7:18 am

  Excellent article as usual…
  BTW what is the “s” in “sbreak” mean? I sometimes need to know what the word is so that I can remember it later.

  Link
- uvreticent July 5, 2012, 9:48 am

  Nice One ,subtlities of C exposed!!!!!!!!

  Link
- Rajesh Kumar V July 7, 2012, 8:16 am

  every beginner must read this .good Article.

  Big Thanks.

  Rgds,
  Rajesh

  Link
- Rupali Sharma July 16, 2012, 9:50 am

  Excellent article…
  Really nicely explained system calls and library calls. 🙂
  One can use ‘strace’ to determine system calls being used in a c program.

  Link
- jay October 8, 2012, 9:32 am

  please i want to know the critical difference between a system call in linux and windows…
  thank you very much

  Link
- Himanshu January 6, 2013, 7:20 am

  Excellent article . was very useful .
  Thanks 🙂

  Link
- aneesh February 26, 2013, 5:11 am

  relay use full.excellent .thank u

  Link
- Fazlan March 8, 2013, 11:12 am

  very well done! Thanx

  Link
- sooraj s.p July 31, 2013, 10:04 am

  very helpful.

  Link

- jack August 12, 2013, 1:02 am

  thanks..

  Link
- Manu Juma August 15, 2013, 6:35 pm

  Very Good Explanation.

  Link
- shankar N September 15, 2013, 7:59 am

  good one…….

  Link
- Fongoh March 26, 2014, 5:26 am

  Thanks. That reall helped me in my quest.
  I will like to know more of those library routines that tend to make system calls.
  Links or direct responses will be welcomed.
  Thanks again

  Link
- SB April 17, 2014, 11:50 am

  Very simple but excellent article.

  Link
- G,M August 11, 2014, 12:39 am

  Thanks for that

  Link
- Anil November 3, 2014, 12:11 am

  Good, you explain it in very simple manner

  Link
- Mihir November 9, 2014, 9:42 am

  Excellent explanation!

  Link
- sachit January 7, 2015, 10:30 am

  thanks pal helped in my minix project

  Link
- pras January 28, 2015, 3:09 pm

  Very good one!! explaining the subtle differences!

  Link
- harun March 24, 2015, 12:29 pm

  Thanks av appreciated its well explained.

  Link
- Mekuanint Erkie April 20, 2015, 8:49 am

  Excellent Explanation, Thank You!!!

Link

- Jiang October 10, 2015, 5:16 pm

  This can be a good introduction of a computer science lecture.
  BTW, I think the arrow between Application Code and System Functions should be a double-direction one, according to your article.

  Link
- Irene December 14, 2015, 3:38 am

  Helpfull article for understanding the difference between library function and system call

  Link
- Zerg January 19, 2016, 5:16 am

  It really helped me, thank you

  Link

Leave a Comment

Name

Email

Website

Comment

Submit

Notify me of followup comments via e-mail

Next post: wget vs curl: How to Download Files Using wget and curl

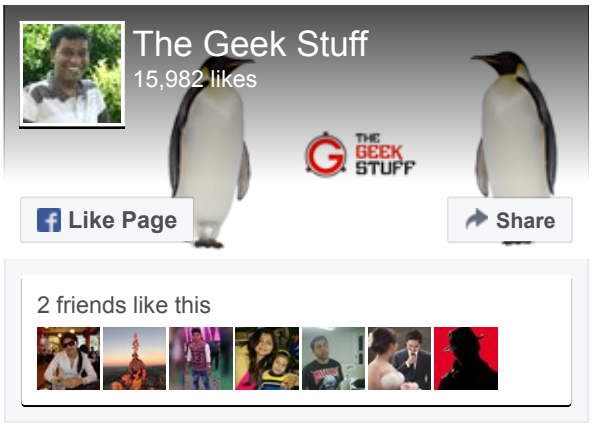
Previous post: Linux ELF Object File Format (and ELF Header Structure) Basics

RSS | Email | Twitter | Facebook | Google+

Google™ Custom Search Search

EBOOKS

- Free Linux 101 Hacks 2nd Edition eBook - Practical Examples to Build a Strong Foundation in Linux
- Bash 101 Hacks eBook - Take Control of Your Bash Command Line and Shell Scripting
- Sed and Awk 101 Hacks eBook - Enhance Your UNIX / Linux Life with Sed and Awk
- Vim 101 Hacks eBook - Practical Examples for Becoming Fast and Productive in Vim Editor
- Nagios Core 3 eBook - Monitor Everything, Be Proactive, and Sleep Well



POPULAR POSTS

- 12 Amazing and Essential Linux Books To Enrich Your Brain and Library
- 50 UNIX / Linux Sysadmin Tutorials
- 50 Most Frequently Used UNIX / Linux Commands (With Examples)
- How To Be Productive and Get Things Done Using GTD
- 30 Things To Do When you are Bored and have a Computer
- Linux Directory Structure (File System Structure) Explained with Examples
- Linux Crontab: 15 Awesome Cron Job Examples
- Get a Grip on the Grep! – 15 Practical Grep Command Examples
- Unix LS Command: 15 Practical Examples
- 15 Examples To Master Linux Command Line History
- Top 10 Open Source Bug Tracking System
- Vi and Vim Macro Tutorial: How To Record and Play
- Mommy, I found it! -- 15 Practical Linux Find Command Examples
- 15 Awesome Gmail Tips and Tricks
- 15 Awesome Google Search Tips and Tricks
- RAID 0, RAID 1, RAID 5, RAID 10 Explained with Diagrams
- Can You Top This? 15 Practical Linux Top Command Examples
- Top 5 Best System Monitoring Tools
- Top 5 Best Linux OS Distributions
- How To Monitor Remote Linux Host using Nagios 3.0
- Awk Introduction Tutorial – 7 Awk Print Examples
- How to Backup Linux? 15 rsync Command Examples
- The Ultimate Wget Download Guide With 15 Awesome Examples
- Top 5 Best Linux Text Editors
- Packet Analyzer: 15 TCPDUMP Command Examples
- The Ultimate Bash Array Tutorial with 15 Examples
- 3 Steps to Perform SSH Login Without Password Using ssh-keygen & ssh-copy-id
- Unix Sed Tutorial: Advanced Sed Substitution Examples
- UNIX / Linux: 10 Netstat Command Examples
- The Ultimate Guide for Creating Strong Passwords
- 6 Steps to Secure Your Home Wireless Network
- Turbocharge PuTTY with 12 Powerful Add-Ons

CATEGORIES

- Linux Tutorials
- Vim Editor
- Sed Scripting
- Awk Scripting
- Bash Shell Scripting
- Nagios Monitoring
- OpenSSH
- IPTables Firewall
- Apache Web Server
- MySQL Database
- Perl Programming
- Google Tutorials
- Ubuntu Tutorials
- PostgreSQL DB
- Hello World Examples
- C Programming
- C++ Programming
- DELL Server Tutorials
- Oracle Database
- VMware Tutorials

Ramesh Natarajan

Follow

About The Geek Stuff

My name is **Ramesh Natarajan**. I will be posting instruction guides, how-to, troubleshooting tips and tricks on Linux, database, hardware, security and web. My focus is to write articles that will either teach you or help you resolve a problem. Read more about Ramesh Natarajan and the blog.

Contact Us

Email Me : Use this Contact Form to get in touch me with your comments, questions or suggestions about this site. You can also simply drop me a line to say hello!.

Follow us on Google+

Follow us on Twitter

Become a fan on Facebook

Support Us

Support this blog by purchasing one of my ebooks.

Bash 101 Hacks eBook

Sed and Awk 101 Hacks eBook

Vim 101 Hacks eBook

Nagios Core 3 eBook