Linux | DB | Open Source | Web

≡ Menu

- Home
- Free eBook
- Start Here
- Contact
- About

What are Linux Processes, Threads, Light Weight Processes, and Process State

by Himanshu Arora on November 14, 2013

G+1 41 Like 22 Tweet

Linux has evolved a lot since its inception. It has become the most widely used operating system when in comes to servers and mission critical work. Though its not easy to understand Linux as a whole but there are aspects which are fundamental to Linux and worth understanding.

In this article, we will discuss about Linux processes, threads and light weight processes and understand the difference between them. Towards the end, we will also discuss various states for Linux processes.

Linux Processes

In a very basic form, Linux process can be visualized as running instance of a program. For example, just open a text editor on your Linux box and a text editor process will be born.

Here is an example when I opened gedit on my machine :

```
$ gedit &
[1] 5560

$ ps -aef | grep gedit
1000      5560  2684  9 17:34 pts/0    00:00:00 gedit
```

First command (gedit &) opens gedit window while second ps command (ps -aef | grep gedit) checks if there is an associated process. In the result you can see that there is a process associated with gedit.

Processes are fundamental to Linux as each and every work done by the OS is done in terms of and by the processes. Just think of anything and you will see that it is a process. This is because any work that is intended to be done requires system resources (that are provided by kernel) and it is a process that is viewed by kernel as an entity to which it can provide system resources.

Processes have priority based on which kernel context switches them. A process can be pre-empted if a process with higher priority is ready to be executed.

For example, if a process is waiting for a system resource like some text from text file kept on disk then kernel can schedule a higher priority process and get back to the waiting process when data is available. This keeps the ball rolling for an operating system as a whole and gives user a feeling that tasks are being run in parallel.

Processes can talk to other processes using Inter process communication methods and can share data using techniques like shared memory.

In Linux, fork() is used to create new processes. These new processes are called as child processes and each child process initially shares all the segments like text, stack, heap etc until child tries to make any change to stack or heap. In case of any change, a separate copy of stack and heap segments are prepared for child so that changes remain child specific. The text segment is read-only so both parent and child share the same text segment. C fork function article explains more about fork().

Linux Threads vs Light Weight Processes

Threads in Linux are nothing but a flow of execution of the process. A process containing multiple execution flows is known as multi-threaded process.

For a non multi-threaded process there is only execution flow that is the main execution flow and hence it is also known as single threaded process. For Linux kernel , there is no concept of thread. Each thread is viewed by kernel as a separate process but these processes are somewhat different from other normal processes. I will explain the difference in following paragraphs.

Threads are often mixed with the term Light Weight Processes or LWPs. The reason dates back to those times when Linux supported threads at user level only. This means that even a multi-threaded application was viewed by kernel as a single process only. This posed big challenges for the library that managed these user level threads because it had to take care of cases that a thread execution did not hinder if any other thread issued a blocking call.

Later on the implementation changed and processes were attached to each thread so that kernel can take care of them. But, as discussed earlier, Linux kernel does not see them as threads, each thread is viewed as a process inside kernel. These processes are known as light weight processes.

The main difference between a light weight process (LWP) and a normal process is that LWPs share same address space and other resources like open files etc. As some resources are shared so these processes are considered to be light weight as compared to other normal processes and hence the name light weight processes.

So, effectively we can say that threads and light weight processes are same. It’s just that thread is a term that is used at user level while light weight process is a term used at kernel level.

From implementation point of view, threads are created using functions exposed by POSIX compliant pthread library in Linux. Internally, the clone() function is used to create a normal as well as alight weight process. This means that to create a normal process fork() is used that further calls clone() with appropriate arguments while to create a thread or LWP, a function from pthread library calls clone() with relevant flags. So, the main difference is generated by using different flags that can be passed to clone() function.

Read more about fork() and clone() on their respective man pages.

How to Create Threads in Linux explains more about threads.

Linux Process States

Life cycle of a normal Linux process seems pretty much like real life. Processes are born, share resources with parents for sometime, get their own copy of resources when they are ready to make changes, go through various states depending upon their priority and then finally die. In this section will will discuss various states of Linux processes :

- RUNNING – This state specifies that the process is either in execution or waiting to get executed.
- INTERRUPTIBLE – This state specifies that the process is waiting to get interrupted as it is in sleep mode and waiting for some action to happen that can wake this process up. The action can be a hardware interrupt, signal etc.
- UN-INTERRUPTIBLE – It is just like the INTERRUPTIBLE state, the only difference being that a process in this state cannot be waken up by delivering a signal.
- STOPPED – This state specifies that the process has been stopped. This may happen if a signal like SIGSTOP, SIGTTIN etc is delivered to the process.
- TRACED – This state specifies that the process is being debugged. Whenever the process is stopped by debugger (to help user debug the code) the process enters this state.
- ZOMBIE – This state specifies that the process is terminated but still hanging around in kernel process table because the parent of this process has still not fetched the termination status of this process. Parent uses wait() family of functions to fetch the termination status.
- DEAD – This state specifies that the process is terminated and entry is removed from process table. This state is achieved when the parent successfully fetches the termination status as explained in ZOMBIE state.

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

{ 4 comments… add one }

- Jalal Hajigholamali November 15, 2013, 9:54 pm

  Hi,

  Thanks for brief and nice article

  Link
- Chandan November 16, 2013, 8:12 am

  very good article

  Link
- Jitendra December 2, 2013, 1:02 am

  Nice Article.

  Link
- Ariel Ruiz December 19, 2013, 12:47 pm

  Very nice understandable short article.

  Link

Leave a Comment

Name

Email

Website

Comment

Submit

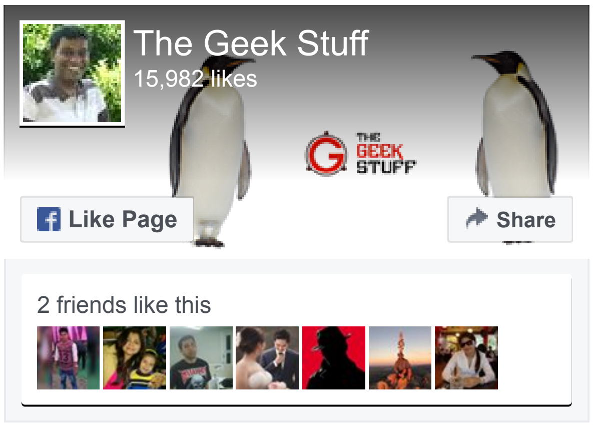
☐ Notify me of followup comments via e-mail

Next post: How to Write an Android App with Activity Lifecycle Function Examples

Previous post: 5 SmartPhone Mistakes to Avoid at All Cost

RSS | Email | Twitter | Facebook | Google+

Google™ Custom Search Search



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