ISSN 2319 - 5975

Volume 8, No.3, April - May 2019

International Journal of Networks and Systems

Available Online at http://www.warse.org/IJNS/static/pdf/file/ijns01832019.pdf

https://doi.org/10.30534/ijns/2019/01832019

Linux Kernel Modules

Akhil Gokuldas[1], Akhileswar V[2], Akshay Babu[3], Jishnu Prakasan[4], Mahalingam P R[5]

Muthoot Institute Of Technology and Science(MITS), India

[1] akhilgokuldas11@gmail.com

[2] aakkhhil79@gmail.com

[3] akshaybabu3575@gmail.com

[4] jishnukprakash.98@gmail.com

[5] mahalingampr@mgits.ac.in

ABSTRACT

A kernel is the foundational layer of an operating system (OS). It functions at a basic level, communicating with hardware and managing resources, such as RAM and the CPU. A kernel module is a code that can be loaded into the kernel image at will, without requiring users to rebuild the kernel or reboot their computer. Modular design ensures that you do not have to make a monolithic kernel that contains all code necessary for hardware and situations. We are planning to implement additional kernel modules that will introduce new features as well as supplement some of the existing features. Pausing, auto-shutdown programs, limit background processes, kill not responding processes are the feature we are going to work on.

Key words : Kernel Modules, Linux kernel, Process Management , Pausing File transfer.

1. INTRODUCTION

In the present technology driven world we are constantly searching for innovations for continual development of systems to cope up with the rising competition and trends. Here in the software the scenario, the competition is very tough accounting for the rapid changes and updates involved with software. We've decided to pool in our efforts to develop 4 Linux kernel modules to further supplement the functionality of the Linux Open Source operating system. Through our humble efforts we aim to increase the usability of open source software in the open market henceforth aiding it's popularity and use among the general public as well. Linux operates in two modes–the Kernel mode (kernel space) and the User mode (user space). The kernel works in the highest level (also called supervisor mode) where it has all the authority, while the applications work in the lowest level where direct access to hardware and memory are prohibited. Keeping in line with the traditional Unix philosophy, Linux transfers the execution from user space to the kernel space through system calls and the hardware interrupts. The Kernel code executing the system call works in the context of the process, which invokes the system call. As it operates on behalf of the calling process, it can access the data in the processes address space. The kernel code that handles interrupts, works to the processes and related to any particular process.

1.1 Linux Kernel Modules

The Linux kernel is a monolithic kernel. It is one single large program where all the functional components of the kernel have access to all of its internal data structures and routines. The alternative to this is the micro kernel structure where the functional pieces of the kernel are broken out into units with strict communication mechanism between them. This makes adding new components into the kernel, via the configuration process, rather time consuming. The best and the robust alternative is the ability to dynamically load and unload the components of the operating system using Linux Kernel Modules. The Linux kernel modules are piece of codes, which can be dynamically linked to the kernel (according to the need), even after the system boots up. They can be delinked from the kernel and removed when they are no longer needed. Mostly the Linux kernel modules are used for device drivers or pseudo-device drivers such as network drivers or file system. When a Linux kernel module is loaded, it becomes a part of the Linux kernel as the normal kernel code and functionality and it posses the same rights and responsibilities as the kernel code.

1.2 Life cycle of Linux kernel module

The life cycle of a module starts with the initmodule(). The task of init module is to prepare the module for later invocation. The module is registered to the kernel and attaches its data-structures and functionality to the kernel. The kernel-defined external functions are also resolved. The life cycle of the module ends with cleanupmodule(). It 'de registers' the module functionality from the kernel. We shall later take a more detailed look at the life cycle of the module.[5]

2. PROPOSED PROJECT

A kernel module is a code that can be loaded into the kernel image at will, without requiring users to rebuild the kernel or reboot their computer. Modular design ensures that you do not have to make a monolithic kernel that contains all code necessary for hardware and situations. Common kernel modules are device drivers, which directly access computer and peripheral hardware

Features : Our aim is making the Linux operating system more useful and competing in the international market by adding useful features to it and supplement its usability and reliability above other open source and proprietary operating systems. We are aiming to implement additional kernel modules which will introduce new features as well as supplement some of the existing ones.

2.1 Pausing Feature

Linux does not have the inbuilt feature to pause the transfer of data within the system and between its externally connected memory storage while transferring files, we aim to implement this pausing feature that will enable users to pause the ongoing data transfer and resume it whenever they require, in accordance with their convenience. The same can be extended to any other running processes or programs.

2.2 Programs auto shutdown feature

This feature aims to detect programs that will cause system failure by utilizing system resources to a very large extent. An algorithm will be put in place to determine the rate of growth, running time and priority of all user run programs and try to terminate those with the potential of causing trouble after warning the user.

2.3 Limiting background processes

This feature aims to facilitate user the option to decide the maximum number of allowed background processes at any given time. If more processes are attempted to run during this time the user will be prompted to close any of the existing processes before he can continue.

2.4 Slow running processes/Not responding

In this feature the user will be alerted about any slow running process or not responding processes through a pop up dialogue box informing the same so that the user will be prompted to close that program/process or allow Linux to search for an online solution to the same. Like software, hardware is a collective term. Hardware includes not only the computer proper but also the cables, connectors, power supply units, and peripheral devices such as the keyboard, mouse, audio speakers and printers.

3. PROPOSED WORK

3.1 Objectives

The main objectives of the proposed project are:
1. To supplement the growth of the open source Linux operating system.
2. Avoid rebuilding the entire kernel in the event of adding new module.
3. Avoiding the rebooting of the device for loading new modules.
4. Saving the RAM space by loading the module run-time when required, like hot plugging drivers when device is connected.
5. A bug in driver which is compiled as a part of kernel will stop system from loading, whereas module allows systems to load. So we can know which module caused it.

3.2 Proposed Solution

We can tackle all of the above mentioned objectives with the help of kernel modules. As we go on to bring about new features in our operating system with the implementation of kernel modules which can be loaded into the Linux kernel image at will, this will negate all the above mentioned issues in a similar manner.

4. PROJECT DESIGN

Modules : Refer figure 1

4.1 Pausing Module

The Linux OS does not have the inbuilt feature to allow a user to pause the transfer of data either within the different locations in the system or between an externally connected storage device and the system. The pausing feature we are going to implement is structured as a kernel module and it will provide the user inbuilt ability to pause and resume data transfer within the system or between two systems as per their convenience. When the transfer is paused the user may disconnect the externally connected storage device and reconnect it whenever possible and can still resume the data transfer from the point where it was paused in the first place. Refer figure 2.[1]

4.2 Module to handle Not responding processes

Each process has its own identity and requirements. They can only work if those requirements are met. Sometimes they are no met and there could even races for resources. These situations lead to deadlocks, slow running as well as not responding processes. Such processes will have to identified and handled before they could affect the system performance and crash the system. This module discusses the possibility of a handler by regular monitoring implemented as a Linux kernel module. Refer figure 3.[4]

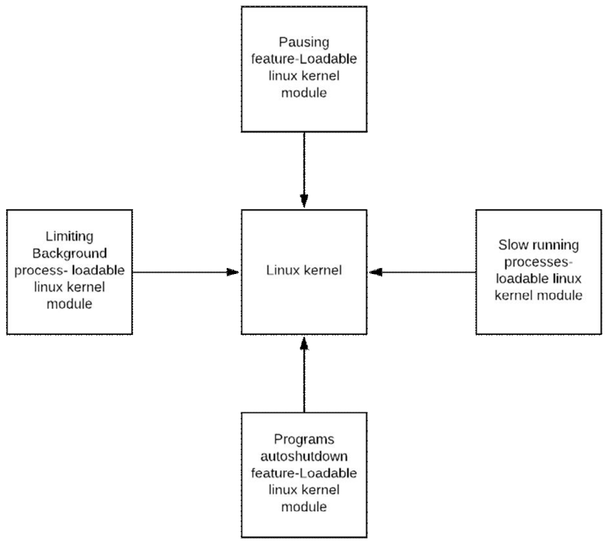
Pausing feature-Loadable linux kernel module
Limiting Background process- loadable linux kernel module
Linux kernel
Slow running processes- loadable linux kernel module
Programs autoshutdown feature-Loadable linux kernel module

Figure 1: Architecture Design of Linux Kernel Modules

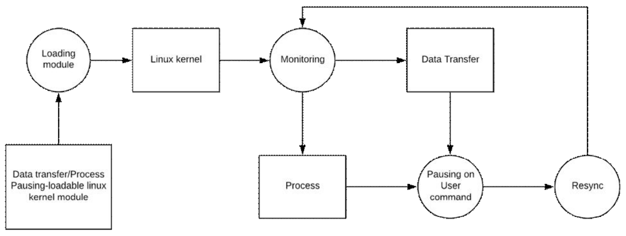

Figure 2: Module to Handle Pausing Feature

4.3 Module to auto-shutdown programs

The programs which use system resources extensively can cause problems in smooth running of the system. Especially on a system with limited resources, it can be a major problem. In an environment where multiple programs are running at the same time all the programs get affected by this. This feature aims to detect programs that will cause system failure by utilizing system resources to a very large extent. An algorithm will be put in place to determine the rate of growth, running time and priority of all user run programs and try to terminate those with the potential of causing trouble after warning the user. Refer figure 4.[2]

4.4 Module to limit background processes

This module's function is to limit the number of background processes to a value set by the user. Task vector contains all the data about processes. It is arranged as a linked list. Each element of the list contains the data of a single process. It is usually the task_struct data structure. Module interacts with the kernel to draw information on processes. It checks whether the number of background processes have exceeded the limit set by the user. If yes, it notifies the handler about the event. Handler takes the necessary action to kill or suspend the process. Otherwise, the process will not be intervened and the module continues to monitor coming processes. Refer figure 5.[3]

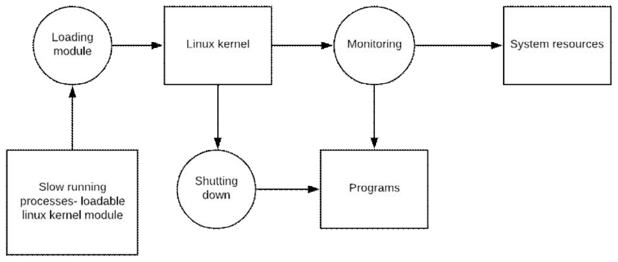

Figure 3: Module To Handle Not Responding Processes

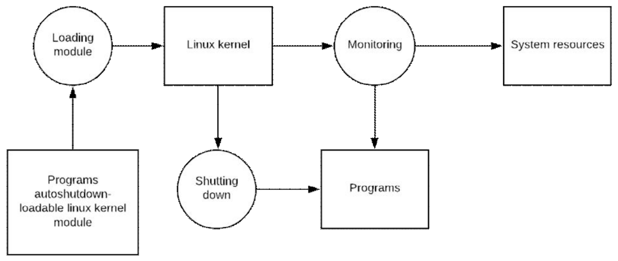

Figure 4: Module to Auto Shut Down Processes

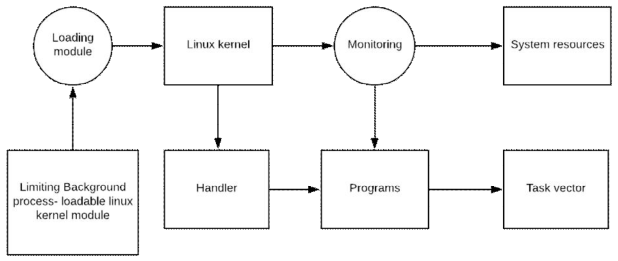

Figure 5: Module to Limit Background Processes

5. CONCLUSION

We are implementing the above mentioned functionalities as kernel modules. After adding these modules, the Linux OS will be enhanced for the user. Kernel modules are loaded by "insmod" command. The "lsmod" command can be used to list all the modules currently installed. Now if the user wants to go back to the standard OS, he/she can remove the module using "rmmod" command. When transferring data to or from an external storage device like a hard disk, USB or DVDs, the user will be able to pause the data transfer using the kernel module functions. The user will be able to restrict the background processes by a predefined limit. It can make the system invulnerable to fork bomb attacks or any other malicious code which infinitely create processes.

REFERENCES

1. Tushar B. Kute and Kabita Ghosh ”**Portable storage device management in linux**”, Computer Science & Engineering: An International Journal(CSEIJ), Vol. 4, No.4, August 2014.
2. G.Keerthi, Dr.R.China Appala Naidu ”**New approach of scheduling algorithms in linux os with goodness function**”, International journel of computer science and information technologies Vol 6, 2006.
3. ZDENEK SLANINA, VILEM SROVNAL ”**Process Monitoring in Operating System Linux**” Department of Measurement and Control VSB Technical University of Ostrava 17. listopadu 15, 708 33 Ostrava-Poruba CZECH REPUBLIC.
4. Henrique Weber, Marco A. Z. Alves, Philippe O. A. Navaux Parallel and Distributed Processing Group, ”**Evaluating Process Scheduling on Linux Operating System**” Informatics Institute, Universidade Federal do Rio Grande do Sul Brasil.
5. Nirav Trivedi, Himanshu Patel, Dharmendra Chauhan, “**Fundamental Structure of Linux Kernel based Device Driver and Implementation on Linux Host Machine**”, International Journal Of Applied Information System(IJAIS), Foundation Of Computer Science FCS, New York, USA, Volume 10-No:4, January 2016.