Crafting GUI Interface to Display Process Control Block and its States

22111058

2022-2024

Guided By:
Prof Nitin.P.Patil

Signature(Student) Signature(Guide)

Contents

1 Introduction

1.1 Abstract

This project was undertaken to first learn about how the linux handles proces management internally, how scheduling is done at lower levels in the kernel and to be able to interact with the kernel using kernel drivers and modules. This project also highlights asoftware design style that is very rare that is part of the system runs inside the kernel space implicitly(on most systems systemcalls are part of the kernel only and thus part of the system runs in the kernel explicitly).One would say that there are many tools like ps and Htop to see what the current state of a process is, well it is true but those tools are not as realtime as we want and they miss almost half of the internal states that the linux kernel scheduler uses.Also one of the primary aim was to check whether we can actually tinker with the linux's default scheduler and add our own states or not. This study and tool will be primarily used for academics purpose. This project reports aim to deliver the knowledge and the journey through which I discovered what works and what not. This report can be further used as guide to devolop this project further if someone is intrested. The project is mainly divided into parts i.e the backend c part and shell scripts that does all the heavy lifting and extracting the information from the kernel.The second part that is frontend, I was going to first build it web based but after discussion with my guide we came to conclusion of a native gui reason for that was it was faster and reduced number of tools required to operate this tool. This tool can serve as an example a starting point for making such gui tools in future.

1.2 Project Overview

This report hightlights all the learning and journey to discover what all was needed to achieve the above mentioned goal.There was no predefined path to acheive what was expected. I have tinkered with two approaches first one was of using systemtaps and second one was writing a custom driver that could interact with the kernel. I have explained both the approaches there pros and cons in following sections. The frontend for the project is done in python3 and raylib. Raylib is versatile wrapper written over opengl that is fast simple and powerfull for creating anytype of ui as it compiles to opengl.

2 Tools/technologies and Domain

2.1 Domain

This project can come under two domains first one is software design and implementation and second being lowlevel kernel or system programming.

2.2 Hardware Used

The initial version of the driver was tested on a virtual machine. The reason for doing so was that while I was learning to write tthe driver if there was any mistake in the logic or i was accessing a wrong address or writing to wrong memmory address, the kernel used to panic and hang up for which the only solution was a reboot to reset kernel,so to avoid crashing my actual machine severla times and avoid possible data loss I implemented the initial driver modules on qemu vm that ran intel pentium based 6th gen cpu with single core and 3.2ghz clock speed, the vm had 2gigabytes of ram and a virtual network interface card that was used to provide network for ssh'ing into the vm.If the kernel crashed i just rebooted the vm and it was back again and i had no fear of losing my data. Once i was confident about the driver I wrote the same driver on my own machine and was able to use it thought there were few crashed sometimes but they were managable. As to what architecture this project will support well any architecture where one can run a modern linux kernel we can run this project(backend) though the frontend is bit tricky to get working on some platform where opengl is not supported.

2.3 Software

- Qemu:QEMU is a generic and open source machine emulator and virtualizer.When used as a machine emulator, QEMU can run OSes and programs made for one machine (e.g. an ARM board) on a different machine (e.g. your own PC). By using dynamic translation, it achieves very good performance.When used as a virtualizer, QEMU achieves near native performance by executing the guest code directly on the host CPU. QEMU supports virtualization when executing under the Xen hypervisor or using the KVM kernel module in Linux. When using KVM, QEMU can virtualize x86, server and embedded PowerPC,

64-bit POWER, S390, 32-bit and 64-bit ARM, and MIPS guests.

- Debian:For this project i have used the latest debian 12 with linux kernel version 6.1. Debian was chosen for its versitatlity and availability on varios architecture easily.Debian is the one of the oldest and most stable operating system based on linux kernel. The debian 12 has linux kernel version 6.1.0 and is pretty stable.

- Gcc: gcc is compiler collection from the GNU project headed by Richard Stallman.Gcc is a very versatile and robust collection of compilers. For this particular os and setup we have used gcc 12 that came preshipped with the os, as that is the best compatible compiler for compiling the kernel.

- Python3: Python3 was used for writing the frontend. Python was chosen for its reliablity and quick prototyping abilities.

- Raylib: Raylib is a quite new project by Raman Santamaria who is a computer scientist. Raylib is very fast versatile and powerfull wrapper over opengl which allows us to make robust and dynamic ui. Raylib can be used both with python and c. Also wrappers for many other languages alos exist.

- systemTaps: Eventhough I chose to not use them later, initial design relied on system taps for information extraction.SystemTaps or staps for short is a utility made by redHat for easier access to the kernel and its datastructure.These are simple scripts that are interpreted and compiled into a kernel module by the stap utility and inserted into the kernel.But the project is not supported very well on debian and ubuntu so it was dropped after fighting with for a month and half.

- kernel module: Kernel modules is mechanism provided by linux kernel folks for inserting a driver into the kernel at run time. This single utitliy makes the whole task of writing debugging and testing the kernel modules so streamlined.The whole project relies on the kernel modules functionality.

3 Rationale and Significance

3.1 Relevance

Kernel programming is one of the most niche and difficult and hard to understand job that is very much sought after and there are very few people who understand and can program within the giant kernel like linux or bsd.The main point of project was to devolop a tool that would aid students in analysing program states in realtime and also for me to get familiriased with linux kernel and its working.I chose to implement this project in the way i did for two reasons 1)I dont have a electronics background so as to directly jump on a hardware driver. 2)Hardware is expensive. So keeping in mind those two points i chose to devolop a soft driver, i.e a driver that is implemented for a software component.

3.2 Significance

Understanding how the drivers work and how are operating systems actully interact with the hardware is one the most crucial skill acs student can learn.This tool also visualises the states at realtime that can be helpful in debuggin any running process.

4 Approach methodology

4.1 Initial Phase

In the initial stage I chose to use system Tap devoloped by red hat. In computing, SystemTap (stap) is a scripting language and tool for dynamically instrumenting running production Linux-based operating systems. System administrators can use SystemTap to extract, filter and summarize data in order to enable diagnosis of complex performance or functional problems.SystemTap consists of free and open-source software and includes contributions from Red Hat, IBM, Intel, Hitachi, Oracle, the University of Wisconsin-Madison and other community members.Even thought the tool is amazing and powerfull, the main reason i had to drop this tool was the reason that this is not a well documented tool, and the current version is only compatible with rhel and ibm Linux. Both of which are very rarely used by

students so it didnt made sense to use those tools.Then later i switched to kernel modules directly.

4.2 Final Implementation

In the current implementation I chose to directly use kernel modules as that is the only way we can inject our code into the kernelfor accessing the internal kernel datastructures mainly process control block in this sense.And for the frontend i have used pythong and raylib,even though i could have used a web based gui also but native gui is also important to learn. The kernel module is implented by using the procfs functionality provided by the linux kernel. Initially i was creating a file in normal fs of the os and and implementing the read write abi's on that file by my self. That worked to an extent but later i switched to procfs for it is resilient and didnt crashed or corrupted the fs or os. The module is devoloped following the best practices. The phase of realising that systemtaps wont work took almost a month. For a month i tried making system taps work on my system, but after few fatal os crashes i ditched system tap and moved on to kernel modules.

5 Result/Conclusion

The result Obtain was that the tool actually monitered the process state in realtime and was able to capture even the smallest state changes. This endeavour also allowed us to look at how the kernel works and understand a bit about how kernel manages data and everything. With the devolopement of tool I understood how to design software and troubleshoot big peices of code. Some of the challanges i faced were of how to send the information from the kernel to the userspace. Well getting information in the kernel is very easy and efficient just providing that data in the userspace is a big challenge. Usually there are systemcalls that will provide tha data from the kernel to userspace like read and write. So i took read and modified it in a such a way that it provide a fixed buffer from the kernel to userspace which can be further read by the userspace process running is normal mode. The python program then takes the data provided by the kernel processes it and then displays it to the user.

6 Further devolopment

Further improvements can be done in form of a elaborate read and write system. A system where instead of preparing a seperate and specialised kernel module for each process there is a generic kernel module running in the kernel all the time that will will accept the input from the userspace and extract data on the go instead of reloading the kernel module each time. Also we can make the frontend more interactive and better. Also we can extract more information from the backend and display more information to the user such as memory info and page tables.