Kernel Modification

Institutional Affiliation

Vinay Krishna Pillalamarri (Pvkrishna2199@gmail.com)

Narendra Reddy Gangireddy (narendrareddy.gang@email.saintleo.edu)

Team-6

**The goals of your project**

The goal of this project is to come up with modifications to the Linux kernel. This will be achieved through different methods, where Linux kernel modules will be extended and modified. It will also involve changing some functionalities in the Linux Kernel modules, which will ultimately influence the whole Linux Operating System. This project will therefore add two specific functionalities to different modules in the Linux Kernel. The functionality that we need to add is the process weights, where each process in the Kernel is assigned, a weight based on how heavy it is. The Kernel modification will however leave out some of Kernel such as the effect that forking a single process has on its child. In this project, the child modification will be similar to that of the father that will be maintained is the init process weight, which starts at 0. The Kernel modification process will also involve system calls. The system calls will lead to various implementations such as setting the current weight of the current process and getting the total weight of the current process iteratively (Lim et al, 2021).

**The core technical challenges**

One of the core challenges experienced in the project is getting started with the project. We had to look for the right kind of modifications that we would include in the kernel, and make sure we monitor to the end. The initialization part of the project required that one had prior knowledge on how to set up the environment on which the Linux kernel modules would be modified. There were various prerequisites that had to be installed in the Linux operating system, to make sure that different changes and extensions would be made. While performing this task, there were errors that were encountered, such as administrative locks by the operating system. I, therefore, had to remove the lock files and redo the installations. Other challenges included downloading and setting up the Linux source code, which would be the basis of the performance expected for the different modifications that would be done on the Linux Kernel. This step of the project also required a stable internet connection, to make sure that the download process was taking place as expected without any interruptions. I dealt with this challenge by making sure that the internet connection was stable (Deride et al, 2020).

**Details of your approach**

Some of the files that were modified during the project include, the /etc/apt/sources. Lis, which needed to be installed on the Linux operating system, to make sure that the project was working well. One of the extensions added to the project is the Linux source code, using the apt source Linux command, which made sure that Linux source code was available for compilation of kernel modules. Some other permissions and folders were also modified, such as the ~/Linux-4.15.0/ file. It was moved and changed to a different file called, /Linux-4.15.18-custom. Such changes were done before the Kernel build-up process. During the kernel build-up configuration process, other changes took place, such as the /etc/default/grub file. This file was set to the modifications and permissions changed in the /Linux-4.15.18-custom file.

**What did or did not work out from your original proposal, and how did your approach evolve as you worked on the project.**

The original proposal of the project included different goals, which were to come up with new functionality of adding weight to each process in the kernel modules. This is one of the proposals that worked. We managed to rewrite this program by coming up with different structured tasks in the file,~/Linux-4.15.18-custom/include/Linux/sched.h that we had formed using the Geaney functionality that we downloaded. We had also planned to install the different kernel modules. We installed the modules as well, and we also made the necessary changes to the grub file. We did configurations of the downloaded modules, in the files that came with them, through accessing the files in the Linux 4.15.18-custom file. We made different changes in different files such as the shed, h file, the syscall\_64.tbl, and the make file in the kernel. We also created a new file syscalls\_weight.c that we used to implement new system calls. These files were what we used to make major modifications to the Linux kernel. However, there are modifications to the kernel that we had planned to make that were not possible. This is where we lacked a system call signature since the file needed for this modification was not available in the file. This forced us to forego this functionality, and modification since it would result in several errors during the normal performance of the Linux kernel. The missing file was the sisals’ that was supposed to be stored in the syscalls directory.

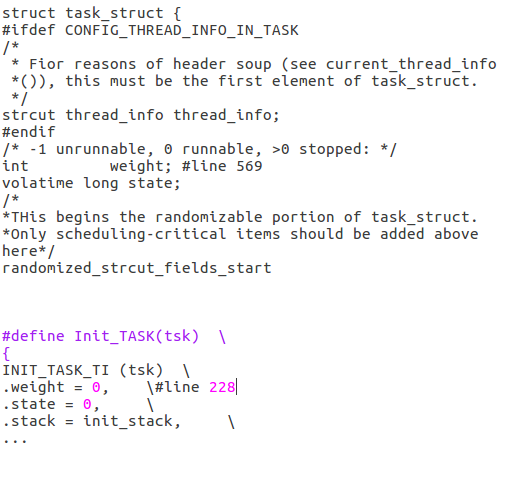
**Techniques from other labs and studios**

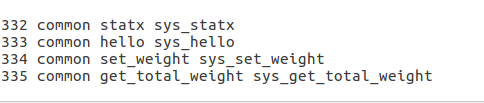
Some of the techniques that were relevant in carrying out our project include synchronization. This is where the correctness of a program is based on the timing of errors. Modules that share memory are accessed and manipulated concurrently. This helped us debug the modifications made to the kernel easily. The use of locks was also helpful in synchronization. Some of the locks we used include the /var/lib/ that helped in locking the administrative directory in the Linux Operating System. We also used spinlock methods that were written in assembly language. These spin locks assisted us in defining the code that was dependent on the architecture. Examples of these locks were used in the new file that we created. Other techniques we used are locks that are used to restrict access to data. These locks make sure that sensitive data is completely out of reach from the different users of the operating system, and applications built based on the OS. This was included in the syscall.64.tbl, where additional syscalls numbers were added to the file to add restrictions to access of sys calls.

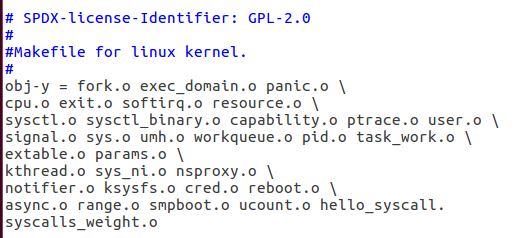
**Evaluations of your solution**

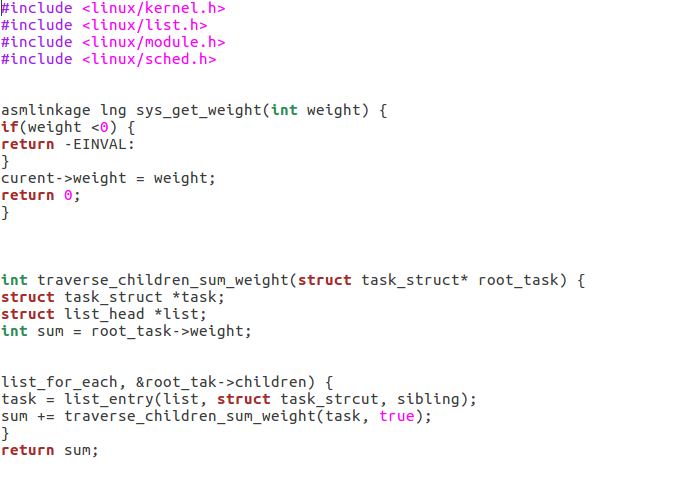
The evaluations of the solution were conducted by rebooting the system, to observe any new features that were added to the system. This approach was practical enough to make sure that any modifications and extensions in the Kernel were made. Some of the observations were also made in the addition in the number of files that were observed in the overall Kernel folder stored in the Linux file that was modified, and new permissions are given. The process of evaluation was important because it helped in the assessment of the achievements made in the project, and hence the Linux Kernel that had to be modified. The evaluation process also made it easy to see the different implementations that worked and the ones that did not work. It also revealed the different files that were modified and the ones that were not available for modification. The hypothesis made at the start of the project was that different processes would have an indication of their weight based on how heavy they were. This hypothesis is the alternate hypothesis rather than the null hypothesis, and it was proved through the results of the different methods arrived at in the system (Song et al, 2020).

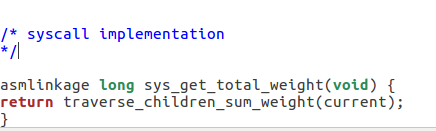
**Results of those evaluations**











**Discussion of how successful your approach was in achieving your goals**

The approach used to arrive at the final project for the presentation was effective. The use of synchronization and locks was very helpful in carrying out the modifications in an easier and much quicker process. This made it possible for the completion of the project in time, as well as the achievement of most of the intended goals of the system. The Kernel was modified, while errors were minimized in the process. This led to the successful completion of the project that meant that most of the tasks that were to be performed were performed. The hypothesis set at the start of the project was also justified. The modules and data structures that had to be changed were changed, as well as the addition of new functionalities in the kernel. The results of the evaluation show that the intended result was arrived at. This, therefore, meant that the intended purpose of the project was arrived at.

**Any insights or questions you may have had while completing this assignment.**

One of the insights that we gained while doing the project, was the possibility of making modifications to an Operating System that is already in operation. This meant that we could contribute to the achievement of software products that work better.

**Any suggestions you have for how to improve the assignment itself.**

The assignment can be improved through the creation of a more defined template and structure of coming up with the project proposal and evaluating their achievement.

**The amount of time you spent on this assignment.**

We took 4 weeks to complete the assignment as a group since each person was assigned a part of the assignment to work on. We then came together on the last week and compiled all the work that we had done.

**References**

Lim, S. Y., Stelea, B., Han, X., & Pasquier, T. (2021). Secure Namespaced Kernel Audit for Containers. In *Proceedings of the ACM Symposium on Cloud Computing* (pp. 518-532).

Oderinde, A. A., Ibikunle, A. A., Bakre, L. G., & Babarinde, N. A. A. (2020). Modification of African breadfruit (Treculia africana, Decne) kernel starch: Physicochemical, morphological, pasting, and thermal properties. *International journal of biological macromolecules*, *153*, 79-87.

Song, D., Hetzelt, F., Kim, J., Kang, B. B., Seifert, J. P., & Franz, M. (2020). Agamotto: Accelerating kernel driver fuzzing with lightweight virtual machine checkpoints. In *29th {USENIX} Security Symposium ({USENIX} Security 20)* (pp. 2541-2557).