VE482 — Introduction to Operating Systems

Lab 6

Manuel — UM-JI (Fall 2019)

Goals of the lab

- Linux kernel source code
- Clean software updates
- Working in Kernel space

1 Introduction

Finally, the weekend is here. Things start to be a bit tense at work with Mr. Frown who is always on your back. In this context a bit of time with your family and a friendly night out with Bob will definitely be much refreshing.

When you arrive at *The Geeks' Tavern* on Friday night, Bob is already seating at the counter sipping a beer. His mood is the total opposite of yours, while you feel disheartened and dejected, Bob seems to be floating in happiness. As soon as he notices you, he walks to you and warmly hug you. Before you can say anything he announces: "You know what? I found the perfect job!". As you silently listens to how great and open his new company is, you cannot stop thinking of your own situation with Mr. Frown. Why on earth did Bob find the perfect job and you are stuck at Lemonion Inc.? You had better grades than him, so why him, not you?

Bob is your friend but sharing his joy is not that simple... As he notices your unusual attitude he inquires about your own situation. A uncontrollable flow of words comes out of our mouth, explaining how mean Mr. Frown is and how unrewarding your job is. Bob tries his best to comfort and encourage you, you can even feel some flattery in his words. While you are thankful for his kindness and empathy, you still would like to have a job like his. Unfortunately his line of work is slightly different from what you are used to and more directly related to Operating Systems. He however promises to let you know as soon as he hear of an opening in his perfect company, CoolZone IT solutions.

Reinvigorated by Bod's words and encouragements, this is with a light heart that you join your parents for Saturday lunch. You just want to relax and enjoy their company. As the discussion shifts to your work at Lemonion Inc. you quickly summarise the current situation and emphasise your desire to do your utmost to keep your job, despite Mr. Frown's abuses. As your parents look concerned you tell them that Bob might be able to help you find a better job, but you first need to refresh your memory on Operating Systems.

The only good side is that although you have been here for already a couple of hours they have not yet asked you when you will buy a flat and get married... Just as this idea draws a smile on your face, your dad appears in the kitchen and orders you to stop drying up the dishes: he has something to show you. A few years back he started a personal project to learn a bit more about filesystems. He designed a very basic and simple inode-based filesystem for the Linux Operating System, and he suggests that you work on it. Your dad seems very excited by the possibility to work with you and help you getting ready for a prospective position at CoolZone IT solutions.

2 Tasks

As you arrive in front of the computer your dad connects to his server[1] through `sftp`, retrieves the archive `/opt/dad/dadfs.tar.bz2`, and opens it. He tells you to start by reading the `README` file and follow the instructions to compile the code. Unfortunately the code appears to be too old to compile with a new kernel. While you feel disappointed, your father just says "Great!". As you gaze at him wondering why the module not compiling is a good news he goes into a great length of explanation on why this is a perfect exercise. You have rarely seen him so excited, even your mum comes wondering what is happening!

From what you understand this is a perfect exercise to understand the structure of the code, how the kernel work, what is a module, how to write backward compatible code, etc. It feels like the benefits are endless. So without any further ado you grab the keyboard and start working, although this feels awkward having both your parents in your back commenting your decisions as if you were still in grade 1.

To guide you in your work you dad asks you a few initial questions:

1. What is a kernel module, and how does it different from a regular library?
2. How to compile a kernel module?

Before going for their nap your father reminds you of the importance of backward compatibility: "Even if you code compiles with a new kernel, it should still compile on an old one, so do not delete all my work!". In a last encouragement before he leaves he blinks his right eye and smile at you before disappearing in the corridor.

After a quick search on internet it appears that the kernel API has changed since the initial code was written. You therefore investigate the problem further hoping to compile it on newer kernels. After a short while you have managed to collect the following information.

- How other projects are doing it:
  - OpenZFS: `https://github.com/zfsonlinux/zfs/blob/master/module/zfs/zpl_file.c`
  - exFAT-nofuse: `https://github.com/dorimanx/exfat-nofuse`
- Useful kernel header files:
  - `linux/fs.h`
  - `linux/uio.h`
- Data structures and functions of interest:
  - `struct kiocb`
  - `struct iov_iter`
  - `copy_to_iter`
  - `copy_from_iter`

Once you have managed to compile the module your are eager to test it, unfortunately your dad only provided simple hints on the commands but no exact command lines. Fully emerged in your work you did your hear your dad coming in your back, so he makes you jump when he warmly congratulates you. He indeed looks pretty impressed by your progress during his nap. He then asks you few more questions.

3. Write and test all the commands that are only hinted in the README file. He seizes the opportunity to emphasise the importance of a complete, precise, and accurate documentation. If he had done so you would not be wasting your time right now…

[1] ji.csproject.org:2482

4. How are `mutex` defined and used? How good is this approach? As he insists on the last question it clearly appears that he expects an honest response, not just "Of course dad your code is perfect, nobody can do better than you."
5. How is information shared between the kernel and user spaces?
6. Document your changes in the README file.

To conclude this nice afternoon your dad talks about the difficulty to write code at the kernel level. "I was sleeping, so I have not idea how many times you crashed the module and the kernel. You were lucky if it did not happen... In general when you crash a module, you're fine, you simply unload and reload it, this is one of the great benefits from modular kernels. However sometimes you're not that lucky, the kernel crashes and so does the whole OS. In that case you have to reboot, adjust your code, compile it, load the module, and hope for the best. This can be very cumbersome and frustrating. The issue lies in the length of the 'Edit-Compile-Run-Crash' loop: when you work at the kernel level it becomes 'Edit-Compile-Run-Crash-Reboot-FindFile'. It often takes minutes between the time you have an idea and the time you can write it down. So be aware that even if there is no Mr. Frown at CoolZone IT solutions, things can be tough, just because of the nature of the work. I really hope you can get a job there but always remember that *the problem is not hard, it simply requires patience*."

After kissing goodbye you leave full of hope and with a strong desire to learn more about filesystems. Therefore you decide to read about them and improve `dadfs` as soon as you feel ready.