Note 9_9_22

9/9/22 -- Feedback

(12:12)
Here are some feedback notes from the project supervisor;

(Honours Project Update: Early September)

"Not an area that has much focus" - it's not an a field that is often 'in the limelight'

This phrasing feels much more suitable and formal. I might need to adapt a teeny bit further but I'll use something like this going forward.

Potential task: Quantifying number of device driver engineers

This is an interesting task, it would make sense but how exactly could I apply this? Potentially by approaching companies and asking about their driver/systems departments? I suppose that for open-source vendors, I could look back at Git/GitHub commits and specifcally look for commits related to drivers? I'll have to brainstorm exactly how this would be incorporated but I think it would link quite well to the above point - not a field that's in the limelight often

"... also work in a harsh environment" - the word harsh is a little too informal

While writing my reports, project management etc. I'll need to really take care and make sure that I'm writing in a formal matter. I'll also need to make sure that I don't over-explain concepts etc and that I'm as clear and concise as possible.

Research tools don't expect to get much use. It's more about the ideas and conversations.

Yes, I had semi thought that too (especially when I saw that WHOOP was essentially dead in the water) but I think my wider point was that these are all potential routes to fundamentally improving drivers. Maybe I was/am being a bit naive. I *can at least use these tools that I've researched and read about to discuss how we might improve drivers. Such as if Dingo saw widespread use. We have a lot of options.*

Mentioned Valgrind, Helgrind, AddressSanitizer - "I always measure a tool's popularity as to whether Ubuntu has an APT package for it."

These are new tools that I've never heard of before. I'll need to really look into them and how widely used they are etc. I might be able to incorporate them into the project also.

What is Apple's solution? Will Apple's solution be applicable to other Linux flavours? Can you apply it to Linux?

Originally, Apple utilised just Kernel extensions (I believe in a similar way to Linux kernel modules) but in the last few years decided to change this and introduce 2 more categories of 'extensions'. System extensions, used for features that need kernel level co-operation and DriverKit extensions, used for handling custom hardware. These new extensions run in user space and Apple claims there's a lot of benefits to using their new extensions over kernel extensions. Here's the talk I watched as part of my Research https://developer.apple.com/videos/play/wwdc2019/702/ and obviously you'll be able to look at my notes too. The issue with Linux is that it offers a user-space option for writing drivers but it seems to be useless. There are benefits but these seemed to be outweighed by several drawbacks so it's generally better for live drivers to run in kernel space.

The ideal C-to-Rust driver conversion to demonstrate might be one where the original C code shows the ***potential*** for a memory error which Rust would not permit.

I'd been trying to figure out myself how to navigate writing a C and Rust driver and what the comparison would be. This makes sense over the original analysis etc that I'd planned and - I think - noted about somewhere so I think I'll focus more on showcasing Rusts strengths over C and demonstrating this through code and what each language allows etc.

(12:38)
Today I want to work on the following tasks

1. Read into FreeBSD and its drivers

2. Write another Rust Systems lab *or* find a new systems concept that I implement in Rust.
3. Work on the project structure/figure out exactly what it is that I'm going to do.

Right now, I'm going to work on task 1. Later on tonight, I hope to come back and complete tasks 2 and 3. I think task 3 is something I might need to work on with Paul or at least get feedback on.

(22:46)
Earlier today, I took notes on freeBSD. I'm going to continue working on this task and then do the prep for task 2. This way I can then work on task 2 tommorrow morning before work.

(23:02)
I've completed notes from that specific page on FreeBSD drivers, I'm satisfied that I've completed notes and might return to gather in-depth information as necessary. Now, I'm going to look into systems tasks that I could write in Rust. I'd like to find something new or fun to work on.