COMPUTING HONOURS PROJECT (COMP10034)

FINAL REPORT & PROJECT MANAGEMENT MARKING SHEET

Student: Kyle Christie (B00415210)

Project Title: Developing Device Drivers in Rust

Supervisor: Paul Keir

Marking Scheme According to Project Specification:

Item	Mark (%)
Background Literature Review Development Experiments Conclusion Reflection	5 13 33 6 6 4
Total *(out of 100)*	67

Comments: *(Please continue overleaf if necessary)*

This project is interesting, technically challenging, and timely. It considers the use of a popular, emerging programming language, and investigates its most notable feature (static memory safety), at a foundational position within the OS stack: device drivers. Kernel-level device driver development using Rust today is a significant challenge, and while the project didn't meet its objective of implementing a USB device driver in Rust, other drivers were developed and tested, and the technical difficulty of the work is recognised. The dissertation itself is fair, though grammar, and minor structural issues interrupt the reading somewhat. The literature review is good, and shows that knowledge is contemporary; though a few more academic and peer-reviewed entries would have been welcomed. The development section could have benefited from more figures or short code excerpts, and the section on reflection is rather short; given the marking scheme. The correspondence with noted industry figures is also commended, and the project was managed very well.

Moderator

The background and reflection sections are, as mentioned above, rather short and grammar (especially the lack of apostrophes) does indeed interrupt the readability of the report.

The literature review contains some unsubstantiated statements (such as "System extensions are said to be much easier to debug..." and an unattributed quote 'it's likely that using Rust has already prevented hundreds of vulnerabilities from reaching production').

Some illustrative diagrams would have helped to clarify some of the more technical descriptions, especially in the development section.

More detailed comments follow:

The introduction is somewhat vague in the precise aims of the project. In particular, the "project goal" section indicates that the sole aim is to develop a Linux device driver in Rust, while the project was in fact broader than this; analysing the literature, and comparing language features of C and Rust.

In the background, the kernel is defined very briefly; and as an interface. The kernel *has* an interface (an API), but is in itself a program. Reference counting is introduced as a mechanism applied in garbage collection; but garbage collection has not been defined (it is defined later in 3.3.1, but would be better in the background section).

With "although" we often have "Although X, Y.", but in the following you have no Y: "Although there are several disadvantages, namely the addition of significant bloat within code as each assignment will see a call which updates the reference count.". Alternatively, "although" can be a final clause in a sentence, in which case it will not follow a full stop. Perhaps "There are though several disadvantages..." would work better here.

Perhaps the background section could also have introduced the reader to basic elements of the Rust language.

Literature review:

Be wary of long sentences, and improper grammar. For example, the following should be split into 2 or 3 smaller sentences:

"That writing and debugging kernel code is difficult, kernel extensions need 2 machines in order to debug which introduces overhead and only has limited debugger support and that kernel extensions are a great risk to security as successful attackers can gain free reign in the kernel while any bug in a kernel extension could also be a critical reliability problem."

You say the Rust compiler "...checks code at compile time so errors can be detected before code is deployed...", but this is true for almost any language+compiler. Perhaps introducing the ownership model first would help.

Figure 3 doesn't show that Rust has no defined memory model, it shows the memory layout of a Rust process.

I might characterise Bjarne Stroustrup's comments on Rust as defences against C++ attacks, rather than (firstly) criticisms of Rust.

"...this is known as a 'user-after-free' vulnerability." - should be 'use-after-free'.

"Garbage collected programs are often faster..." - this is a surprising comment; I think a citation should be used to support it. Indeed in Section 3.6 you later say that it "may even negatively impact performance...", which I take as the common understanding.

Your explanation of driver types in the development chapter, could also have been included in the background chapter.

Given that you consider the driver code in the appendix to a level of detail where you discuss the main `struct` name (i.e. `file_operations`), you could have included some excerpts to help explain (especially as the struct is only 7 lines). I think short code excerpts would also have improved the readability of Section 5.1, and especially Section 5.2.

Figures 5, 7, 8 and 9 are not referenced in the text.

Line spacing seems narrower at page 32.

The name "RustMouse" seems odd within the Rust "Hello World" driver.

"...encapsulated within 'impl' structures which is not as clean as C..." It's not clear to me what is unclean about an `impl` block.

Your conclusion would be strengthened by incorporating thoughts on your own experimental results.

The section on reflection is rather short.

The "Additional Figures" section should also have some commentary (even within that section); but should also be placed within the appendices.

Hyphenation
Rusts suitability
Androids total vulnerabilities
in the kernels interfaces
Rusts enforced variable mutability
within the projects 'cargo.toml'

If not, they design a safe Rust API
->
If not, then design a safe Rust API

Spelling:
abstractiosn

Project Management Mark: **8 out of 10**

Comments:

The project was very well managed, with good use of tools (e.g. via Trello and Github); frequent notes and updates visible on the Github repository; and regular, clear and honest meetings.