Vrije Universiteit Amsterdam

Bachelor Thesis

Sample Title

Author: Alexander Balgavy (2619644)

1st supervisor: person
daily supervisor: person

A thesis submitted in fulfillment of the requirements for the VU Bachelor of Science degree in Computer Science.

June 24, 2020

Abstract

Abstract goes here.

1 Introduction

The introduction should cover

- why reliability is important
- specifically in filesystems - why do you need a reliable filesystem?
- what's done in this paper (high-level summary)

2 Background information

2.1 C as the implementation language

The choice of an implementation language may affect the bugs or vulnerabilities that are present in the system. The de-facto standard implementation language for operating systems and their components has long been C. Many popular file systems are implemented in C, such as Ext4. **ext4Source**. C is based on typeless languages, BCPL and B, which were developed specifically for operating system programming in early Unix. A design principle of C was to be grounded in the operations and data types provided by the computer, while offering abstractions and portability to the programmer. [1]

Describe studies on issues with C, bugs found, CVEs, etc.

2.2 Possible alternatives

- Restricting C: MISRA-C, Frama-C
- Rust: esp. reference ownership
- D: allows functional contracts
- Coq: allows writing formal specification, proving it, and extracting certified program from constructive proof of its specification in OCaml, Haskell, or Scheme.
- Ada

2.3 Formal verification

Explain what it is, particularly Hoare triples. Why is it useful?

2.4 FUSE

Why use it for development, what are its limitations?

3 Related work

3.1 FSCQ

3.2 Yxv6 & Yggdrasil

3.3 Argosy

3.4 COGENT

4 Design & implementation

- Filesystem design: why was MINIX chosen as the basis? Which parts of my filesystem are similar to MINIX? What changes/compromises are made?
- Language features: strong typing, lack of pointers (access types vs in-out parameters), modularisation using packages and private parts
- FUSE & the FUSE driver
- Verification: contracts (functional and data). Also - what can't be verified (at the moment)?

5 Results & analysis

- Include the formal verification report, what's verified in relation to CWE numbers.
- Add some performance analysis. Maybe timing? Try to do some common filesystem tasks & see how it performs?

6 Conclusion

Summary and concluding remarks, including possible future work.

References

[1] D. M. Ritchie, "The development of the c language," in *The Second ACM SIGPLAN Conference on History of Programming Languages*, ser. HOPL-II, Cambridge, Massachusetts, USA: Association for Computing Machinery, 1993, pp. 201–208, ISBN: 0897915704. DOI: 10.1145/154766.155580. [Online]. Available: https://doi.org/10.1145/154766.155580.