Rusty Linux: Advances in Rust for Linux Kernel Development

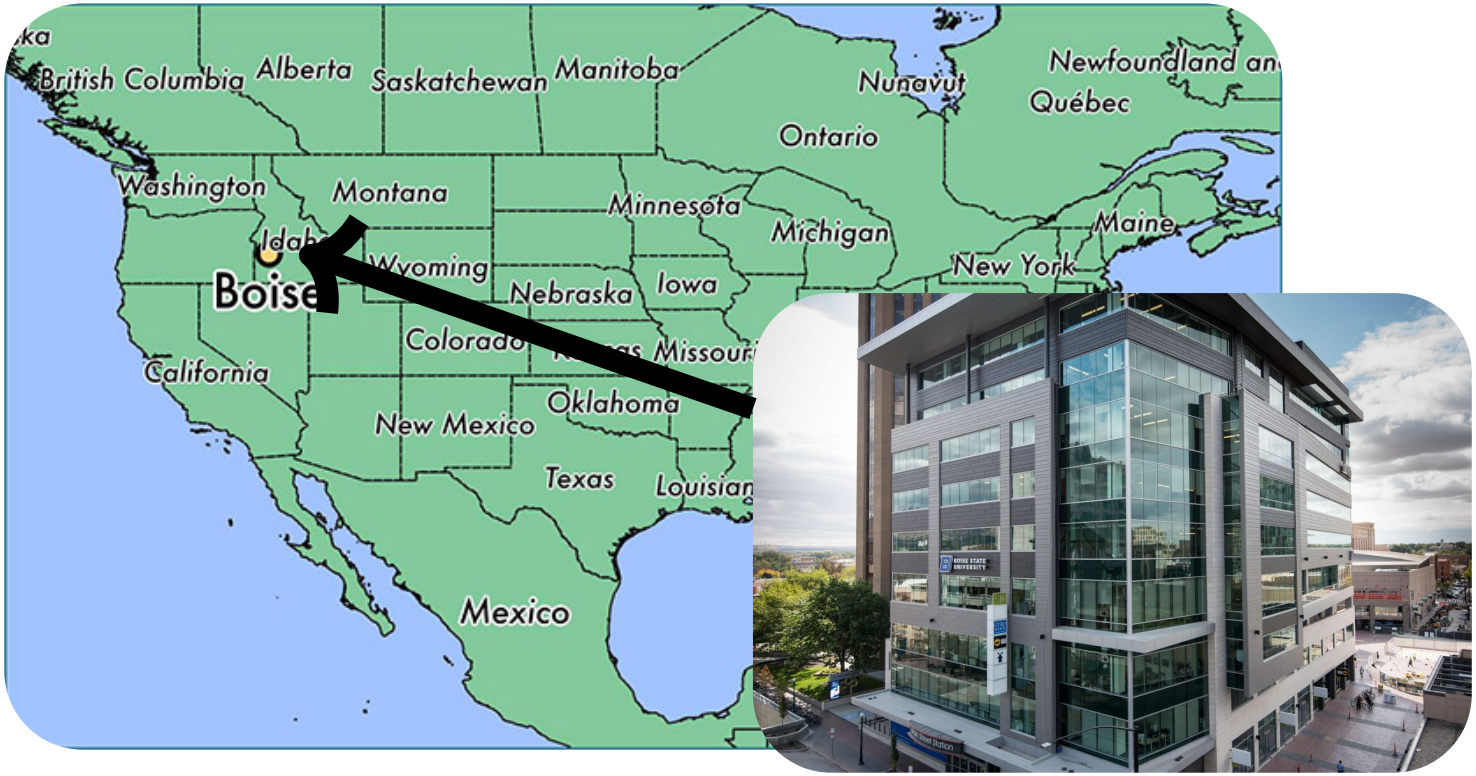
Shane K. Panter[1] Nasir, Eisty[2]

[1]Clinical Assistant Professor
Boise State University

[2]Assistant Professor
Boise State University

International Symposium on Empirical Software Engineering and Measurement, October 2024

Introduction



Boise State University

The Computer Science Department is located in Beautiful downtown Boise Idaho, United States!

Our Research

We aim to find the current advances in using Rust in Kernel development to reduce the number of memory safety vulnerabilities in one of the most critical pieces of software that underpins all modern applications.

▸ Paper Link

Figure: A rusty computer[1]

[1]AI Prompt: A rusty computer with a penguin next to it

Why Rust

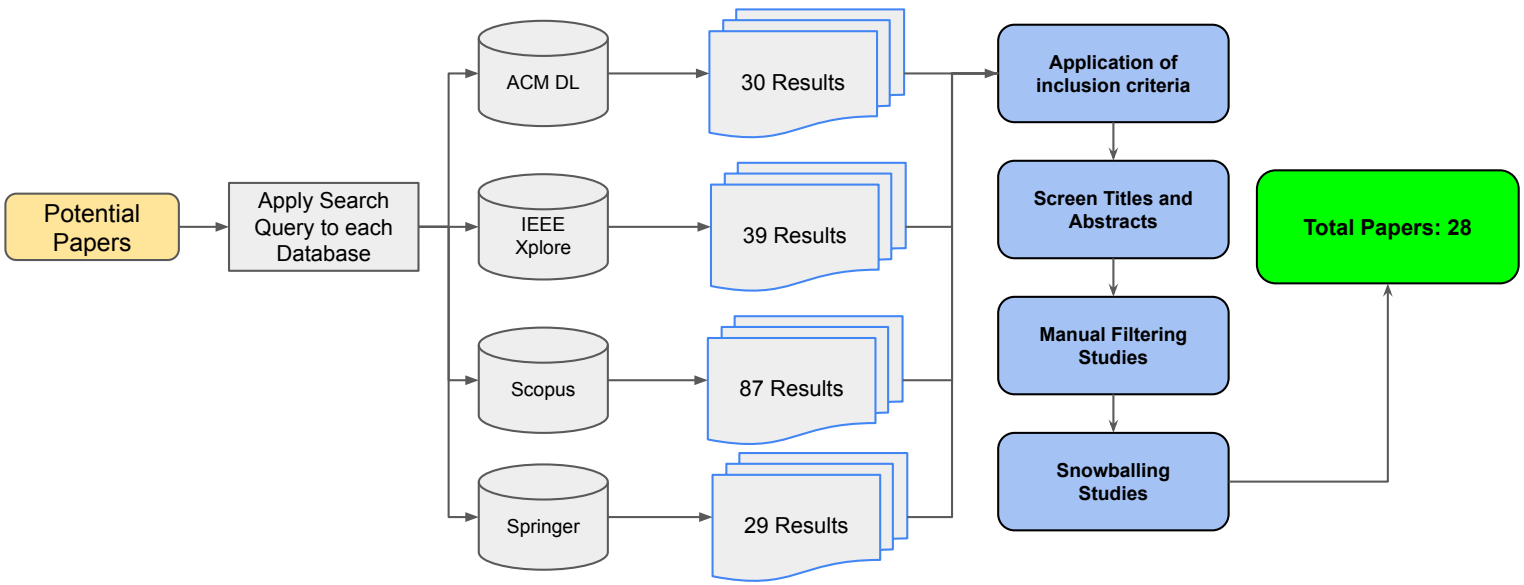
- Low-level **control** like C and C++
- Strong **safety** guarantees
- **Modern**, functional paradigms
- Industrial development and backing
- No garbage collector needed! All checks are performed at compile time

Research Questions

- **RQ1:** What are the existing approaches for implementing operating system kernels in Rust?
- **RQ2:** What are the performance implications of using Rust for operating system kernel development?
- **RQ3:** What are the major challenges and limitations when developing operating system kernels in Rust?
- **RQ4:** What are the lessons learned when developing operating systems kernels in Rust?

Process Diagram

Rusty Linux



Results

Our findings!

Figure: Super happy researcher![1]

[1]AI Prompt: scientist getting research results and is super happy in a cyberpunk universe with lots of computers showing matrix code on them

RQ1

Table: Approaches and Methodologies for Rust in the Kernel

Approach	Papers	Operating System in Rust
Monolithic	4	Linux Kernel v6.1+
Micro-kernel	5	Atmosphere, Redox, Redleaf
Embedded	2	Tock, Hubris, Drone, Bern, HarSaRK
Unikernel	4	RustyHermit, Theseus
Exokernel	1	W-Kernel

RQ2

Table: Performance Implications of Rust in the Kernel

No.	Implication	Studies that Reported the challenge
1	Performance	3
2	Throughput	1
3	Latency	1

RQ3: Challenges and Limitations

- Binary Size - Rust can produce larger binaries
- Missing Features - Rust still evolving and adding features
- Soundness - How to deal with raw memory
- Panics - What happens when things go wrong?
- C Interop - Specific to mixed language kernels

RQ4: Lessons Learned

- Impossible to use 100% rust
- Rust is not as expressive as other formal verification techniques
- Ownership root - An OS provides memory to rust so if the OS is itself written in rust who is the root owner?

Conclusion

- We are still in the early stages of figuring out who to do kernel dev in Rust
- High potential for enhanced security and stability
- Need to address integration issues (FFI)
- Need to expand the body of empirical evidence on Rust's impact!

Questions?

Questions?

Figure: Happy People[1]

[1]

[1] AI Prompt: People attending a conference who all want to ask a question and are really excited!