

A RUST-BASED, MODULAR UNIKERNEL FOR MICROVMS

RustyHermit @ FOSDEM 2023

Stefan Lankes, Jonathan Klimt, Martin Kröning

Stefan
@stlankes

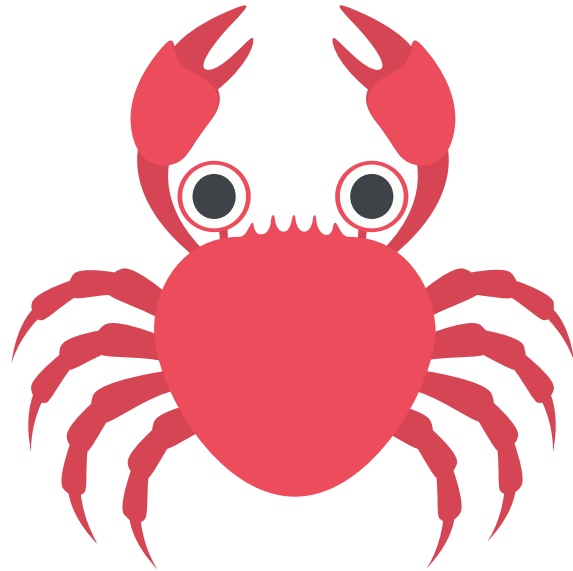
Jonathan
@jounathaen

Martin
@mkroening



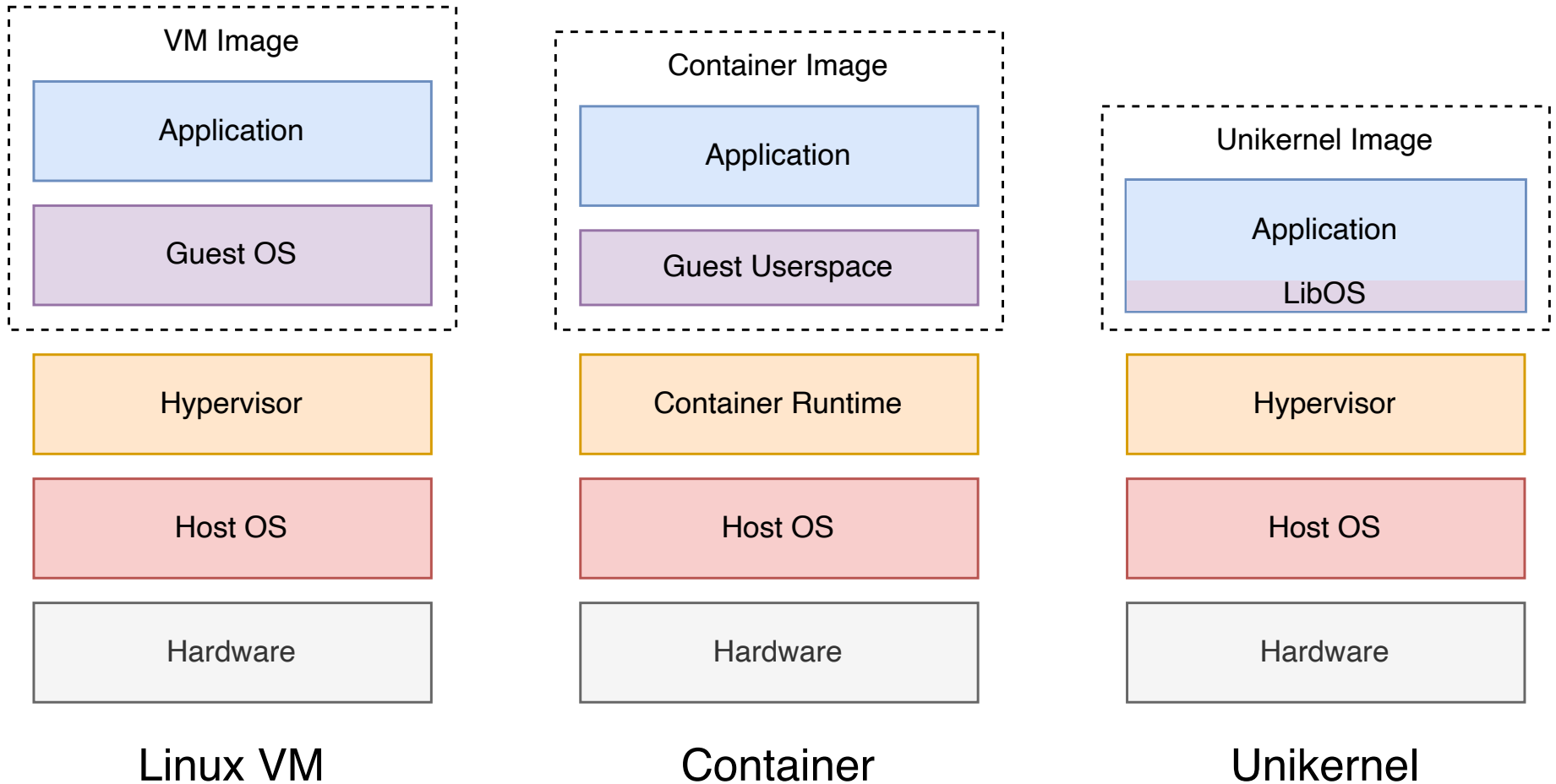
This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957246 and the Horizon Europe project NEMO under grant agreement No 101070118.

RUSTYHERMIT



A Rust-based, modular libOS for creating Unikernels

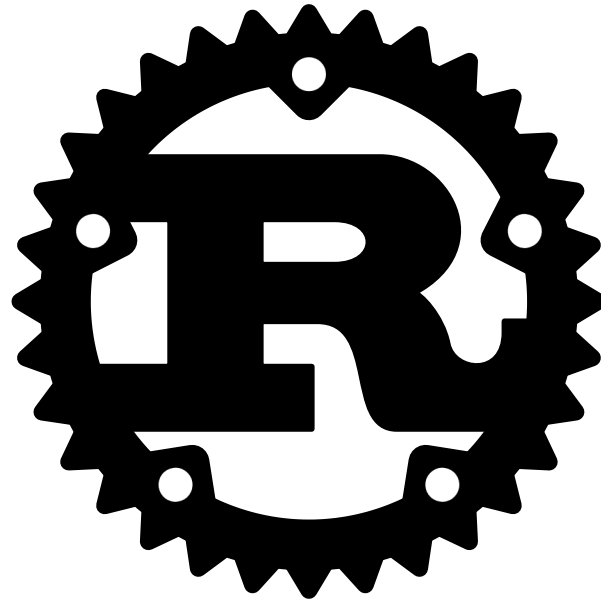
UNIKERNELS 🙇



MICROVMS




- Might not have PCI or ACPI support
- Firecracker (AWS)
- QEMU `microvm` platform
- Uhyve

RUST



CC-BY

WHY RUST?

- Rust is productive. 
- Rust is fun. 
- Rust is safe. 

PROOF OF COOLNESS 🧐

- Sum types
 - aka tagged unions
 - Enumerations can contain data

```
enum Option<T> {  
    None,  
    Some(T),  
}
```

```
match option {  
    None => println!("We got nothing. 🙄"),  
    Some(value) => println!("Hurray, some {value}! 🥳"),  
}
```


SAFE RUST

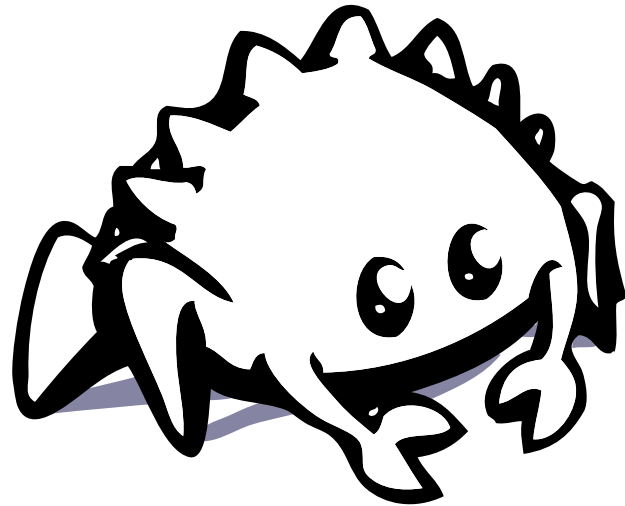




Image from @aldeka

SAFETY 🙄



This is undefined behavior and can't happen:

- Accessing invalid pointers ✨
 - use-after-free
 - double-free
 - out-of-bounds
- Data races 🏎️
- ...

REQUIREMENTS FOR OS DEVELOPMENT

- Raw Memory access 
- Assembly code 

REQUIREMENTS FOR OS DEVELOPMENT

- Raw Memory access 
- Assembly code 

→ Not possible in Safe Rust 

UNSAFE RUST

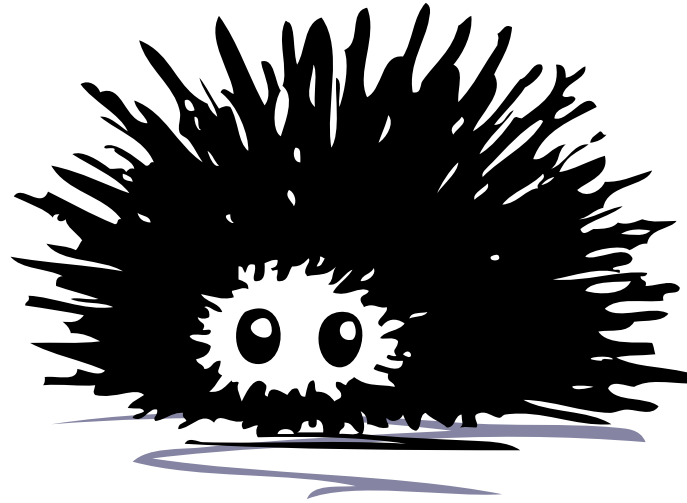


Image from @aldeka

UNSAFE RUST SUPERPOWERS

- Access a raw pointers
 - MMIO
- Call unsafe functions
 - Inline assembly
- ...

```
// Access raw pointers  
unsafe { ptr.read_volatile(value) }
```

```
// Set the highest-level page table  
unsafe { asm!("mov cr3, {}", in(reg) value) }
```

HOW TO 

REQUIREMENTS



- Rustup
- Hypervisor of your choice
 - QEMU
 - Uhyve
- NASM (only for x86_64 with SMP)

Cargo.toml

```
[package]
name = "hello_world"
version = "0.1.0"
edition = "2021"
```

main.rs

```
fn main() {
    println!("Hello World!");
}
```

Cargo.toml

```
[package]
name = "hello_world"
version = "0.1.0"
edition = "2021"
```

```
[target.'cfg(target_os = "hermit")'.dependencies]
hermit-sys = "0.4"
```

main.rs

```
fn main() {
    println!("Hello World!");
}
```

Cargo.toml

```
[package]
name = "hello_world"
version = "0.1.0"
edition = "2021"
```

```
[target.'cfg(target_os = "hermit")'.dependencies]
hermit-sys = "0.4"
```

main.rs

```
#[cfg(target_os = "hermit")]
use hermit_sys as _;
```

```
fn main() {
    println!("Hello World!");
}
```

BUILD

`rust-toolchain.toml`

```
[toolchain]
channel = "nightly-2022-10-19"
components = [ "rust-src" ]
```

BUILD

`rust-toolchain.toml`

```
[toolchain]
channel = "nightly-2022-10-19"
components = [ "rust-src" ]
```

```
$ cargo build \
  --target x86_64-unknown-hermit \
  -Zbuild-std=std,panic_abort \
  --release
```

BUILD

`rust-toolchain.toml`

```
[toolchain]
channel = "nightly-2022-10-19"
components = [ "rust-src" ]
```

```
$ cargo build \
  --target x86_64-unknown-hermit \
  -Zbuild-std=std,panic_abort \
  --release
```

That was easy. 

DEMO 🙄

DEMO 🙄

1. Clone [hermitcore/rusty-demo](#)
2. Compile

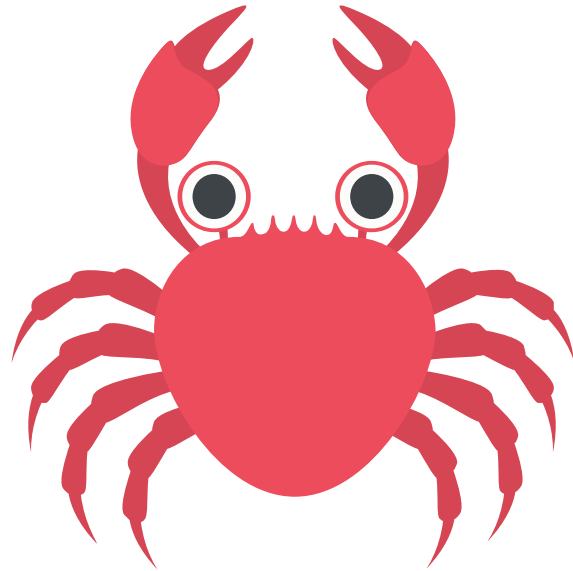
```
$ cargo build \  
    -Zbuild-std=std,panic_abort \  
    --target x86_64-unknown-hermit
```

3. Run with Uhyve

```
$ uhyve -v target/x86_64-unknown-hermit/debug/hello_
```

4. Profit

MODULARITY IN RUSTYHERMIT



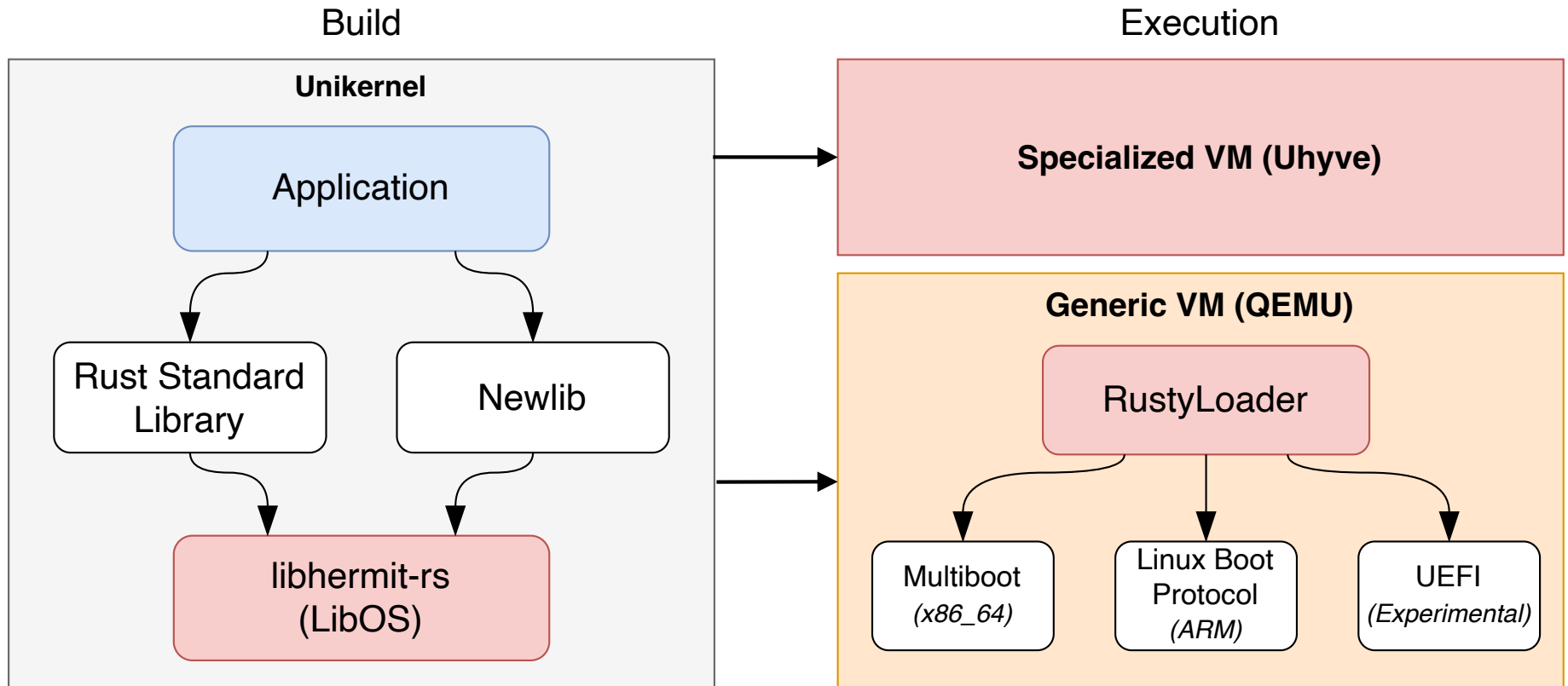
KERNEL FEATURES

```
[target.'cfg(target_os = "hermit")'.dependencies.hermit-sys]
version = "0.4"
default-features = false
features = [
    "smp",
    "tcp",
    "dhcpv4",
    # "acpi",
    # "pci",
    # "pci-ids",
    # "vga",
]
```

KERNEL STRUCTURE

```
libhermit-rs
├── hermit-entry (Entry API)
├── hermit-sync (Synchronization Primitives)
├── linked_list_allocator (Allocation Algorithm)
├── uart_16550 (Serial Device Driver)
├── x86_64 (Architecture-Specific Abstractions)
├── pci-ids (PCI ID Database, optional)
├── smoltcp (TCP/IP Stack, optional)
└── ...
```

THE HERMIT ECOSYSTEM



WORK IN PROGRESS

- Further codebase oxidization
- Miri support
- More modularization
- TCP/IP stack overhaul
- Uhyve network
- Firecracker support
- ARM support

Find us at [github.com/hermitcore!](https://github.com/hermitcore)

WORK IN PROGRESS

- Further codebase oxidization
- Miri support
- More modularization
- TCP/IP stack overhaul
- Uhyve network
- Firecracker support
- ARM support

Find us at [github.com/hermitcore!](https://github.com/hermitcore)

Thanks for listening!

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

