

# Introducing Helios

A small, practical microkernel

Drew DeVault

SourceHut

January 9, 2023

# Introduction

---

- Hi
- I'm Drew DeVault
- Project lead for Hare
- Other stuff too
- Moving swiftly along

`https://ares-os.org`

`https://harelang.org`

`https://drewdevault.com`

# What is Helios?

---

Helios is a microkernel, largely inspired by seL4. It is written in Hare and runs on x86\_64 and aarch64; RISC-V is planned.

- $\approx$  8,500 lines of portable code
- $\approx$  3,000 lines non-portable per architecture
- GPL 3.0

Note: Line counts do not include the bootloaders

# What is Helios, ctd

---

What works?

- Capability-based security
- IPC (similar to seL4)
- Preemptive scheduling (single core, no SMP)
- Hardware I/O (ports or mmio), IRQs
- EFI (aarch64) or multiboot (x86\_64)

# Why is Helios?

---

- Kernel hacking is really fun
- Prove if Hare is useful for this purpose
- Can we do better than seL4?
- Can we do better than, dare I suggest, Linux?

# A brief introduction to Hare

---

Hare is a systems programming language designed to be simple, stable, and robust. Hare uses a static type system, manual memory management, and a minimal runtime. It is well-suited to writing operating systems, system tools, compilers, networking software, and other low-level, high performance tasks.

# A brief introduction to Hare

---

- General purpose systems programming language
- 3 years in development
- 18,000 line compiler (C11)
- 12,000 line backend (C99)
- x86\_64, aarch64, riscv64

# What does Hare look like?

---

```
export @noreturn fn kmain(ctx: arch::bootctx) void = {  
    log::println("Booting Helios kernel");  
  
    const pages = init::pages(&ctx);  
    let heap = init::heap_init(&ctx, pages);  
    let task = init::task_init(&heap, ctx.argv);  
    init::load(&task, &ctx.mods[0]);  
    init::heap_finalize(&task, &heap, &ctx);  
    init::devmem_init(&task);  
    init::finalize(&task);  
  
    log::println("Entering userspace");  
    sched::init();  
    sched::enteruser(task.task);  
};
```



# The design of Helios

---

Let's go over the main talking points about its design:

- Capabilities
- Memory management
- Address spaces & tasks
- IPC

And implementation:

- Bootloader
- System initialization
- Runtime API

# Capabilities

---

## CSpace

0x00	Memory
0x01	Device memory
0x02	Task
0x03	I/O port
0x04	IRQ
0x05	null (link 0x06)
0x06	null (link 0x07)
0x07	null (link ...)
	...
0xFF	(null)

- Each task (process) has a CSpace
- Configurable number of capabilities
- Uses MMU to enforce capabilities

Similar to seL4, but:

- **Not** a guarded page table
- Free list for  $O(1)$  capalloc in kernel

# Capabilities

---

- Null
- Memory
- Device memory
- Capability space
- Virtual address space
- Task
- Endpoint
- Notification
- Reply
- ASID control
- ASID pool

Plus, on x86\_64:

- PDPT
- PD
- PT
- Page
- I/O control
- I/O port
- IRQ control
- IRQ handler

# Memory management

---

- Different from seL4: Free list instead of high watermark
- It's not that interesting
- This slide only exists to mention the seL4 thing
- But I feel the need to fill in some of this whitespace
- You just lost the game

# Address spaces

---

VSpace capabilities (like seL4) are capabilities that manage address spaces. Pages can be shared, but not page tables, at least until we know why seL4 doesn't let you share page tables so we can stop cargo-culting that constraint.

# Tasks

---

Tasks have a CSpace (optional) and VSpace and receive CPU time when configured appropriately. Simple round-robin scheduler for now; more sophisticated later.

# IPC: Notifications

---

- Asynchronous: signal never blocks, wait blocks
- Same as seL4

# IPC: Endpoints

---

- Synchronous: send and recv both block, wait for rendesvouz
- Sends registers and/or capabilities
- Supports capalloc
- seL4-style call/reply supported
- Code generation!



# IPC: Poll

---

```
let poll: [_]pollcap = [  
    pollcap { cap = IRQ, events = pollflags::RECV },  
    pollcap { cap = EP, events = pollflags::RECV },  
];  
for (true) {  
    helios::poll(poll)!;  
    if (poll[0].events & pollflags::RECV != 0) {  
        poll_irq();  
    };  
    if (poll[1].events & pollflags::RECV != 0) {  
        poll_endpoint();  
    };  
};
```

# Booting Helios

---

The kernel itself is just an ELF executable, simple as can be. The bootloader's job is to load it, collect information about the environment, and hand it over to the kernel.

Supported: Multiboot on x86\_64, EFI on aarch64; soon: EFI everywhere

The bootloaders are also written in Hare!

# System initialization

---

- Load init from ELF executable
- Prepare capabilities for all system resources, hand them over to init
- Try not to fuck up memory while you're at it
- Uses simple PCI driver (x86\_64) or device tree (aarch64) to enumerate resources
- Please no ACPI please stop adding it to RISC-V please please please
- Jump to userspace

# Runtime kernel API

---

- 14 syscalls; mainly send, recv, yield, etc
- Invoke operations on kernel objects via capabilities
- Uniform kernel/endpoint interface (userspace can pretend to be the kernel)
- Does the fault handler count as the userspace API? Some of it is in uapi/\*
- uapi and kernel API are separated, nice design, much error handling

# Does it work?

---

I am using Helios to present this slide deck.

- Ported system to aarch64 over the last eight weeks(!)
- Simple GPU driver in userspace
- Serial port to switch slides/etc (USB in eight weeks? hah!)
- Slide deck is PNG files in a tarball functioning as the initramfs
- Few hacks or shortcuts!

# Userspace ambitions

---

Plans:

- Helios: Kernel
- Mercury: Driver framework
- Venus: Driver collection
- Gaia: Userspace interface
- Luna: POSIX compatibility layer
- **Ares**: Complete operating system

# What's next?

---

The kernel is mostly "done". Still needs:

- Polish
- About 100 // TODOs
- SMP support
- riscv64 port
- More bootloader options
- Better docs

Next: userspace, drivers, and more!

# Acknowledgements

---

Shoutout to early Hare kernel attempts from Ember Sawady and Alexey Yerin!

Big thanks to the Hare community as well: almost 80 contributors!

We stole a bunch of ideas from seL4 too



# Closing thoughts

---

Kernel hacking is fun! Hare is fun! Let's all have fun together!

<https://ares-os.org>

<https://sr.ht/~sircmpwn/helios>

<https://harelang.org>

Join us: #helios on Libera Chat