Chronos OS: Phase 2 Report

The Nervous System: Interrupts, Input, and Concurrency

Development Log

January 19, 2026

Abstract

This report details Phase 2 of the Chronos Operating System development. The primary objective was to transition the kernel from a passive loop to a reactive system capable of handling asynchronous hardware events. This required implementing the Interrupt Descriptor Table (IDT), remapping the legacy Programmable Interrupt Controller (PIC), and writing a PS/2 keyboard driver. A key achievement was the implementation of "Interactive Time," where hardware inputs dynamically alter the kernel's scheduling parameters via atomic state sharing.

Contents

1 Overview

A functional operating system must respond to external stimuli (Timer, Keyboard, Disk). In Phase 1, Chronos was a "Brain in a Jar," only capable of internal calculations. Phase 2 focused on connecting this brain to the hardware via the x86 Interrupt System.

2 The Interrupt Architecture

2.1 The IDT (Interrupt Descriptor Table)

The IDT is a data structure used by the CPU to determine the correct response to exceptions and hardware signals. We utilized the `x86_64` crate to define handlers for:

- **Vector 3:** Breakpoint Exception (Debug).
- **Vector 32:** System Timer (IRQ 0).
- **Vector 33:** Keyboard (IRQ 1).

2.2 The 8259 PIC Remapping

A critical legacy issue on x86 architecture is that the 8259 PIC maps hardware interrupts to vectors 0-15 by default. These conflict with CPU internal exceptions (e.g., Double Fault is Vector 8).

Solution: We remapped the Master PIC to offset 32 and the Slave PIC to offset 40. This ensures that a Keyboard Interrupt triggers Vector 33, safely distinct from CPU faults.

```
pub const PIC_1_OFFSET: u8 = 32;
pub const PIC_2_OFFSET: u8 = PIC_1_OFFSET + 8;

pub static PICS: Mutex<ChainedPics> = Mutex::new(unsafe {
    ChainedPics::new(PIC_1_OFFSET, PIC_2_OFFSET)
});
```

Listing 1: PIC Initialization

3 Input Handling and Diagnostics

3.1 The "Deaf CPU" Problem

During initial testing, the interrupt handlers failed to fire. The kernel was executing, but inputs were ignored.

Diagnosis: The PIC lines were masked (muted) by default.
Resolution: We implemented an explicit unmasking routine, writing `0xFC` to the Master PIC data port, which enables lines 0 (Timer) and 1 (Keyboard) while keeping others silenced.

3.2 PS/2 Keyboard Driver

We implemented a driver that listens on I/O Port `0x60`. Upon receiving an interrupt at Vector 33, the handler:

1. Locks the Keyboard Mutex.
2. Reads the scancode byte from Port 0x60.

3. Decodes the scancode into a Rust `char`.
4. Sends an "End of Interrupt" (EOI) signal to the PIC.

4 State Management: Interactive Time

4.1 The Concurrency Challenge

The Main Loop (drawing the screen) and the Interrupt Handler (processing keys) run asynchronously. Sharing data between them usually requires complex locking.

4.2 The Atomic Solution

To modify the system's "Time Budget" in real-time, we utilized `AtomicU64`. This allows the Interrupt Handler to safely modify the budget variable without causing data races or deadlocks.

```rust
// Shared Global State
pub static CYCLE_BUDGET: AtomicU64 = AtomicU64::new(2_500_000);

// In the Interrupt Handler
match character {
    '+' => state::adjust_budget(1_000_000), // Relax Budget
    '-' => state::adjust_budget(-1_000_000), // Tighten Budget
}
```

Listing 2: Atomic State Adjustment

5 Conclusion

Phase 2 is complete. Chronos now possesses a functioning nervous system. It can "feel" time (via the Timer IRQ) and "feel" user intent (via the Keyboard IRQ). The visual feedback loop (Fuel Gauge) now reacts instantly to user input, proving the success of the interrupt architecture.