2024.11.28 Syswonder 组会报告

Linux hvc, Rust Code Formal Verification

韩喻泷

School of Computer Science, Peking University

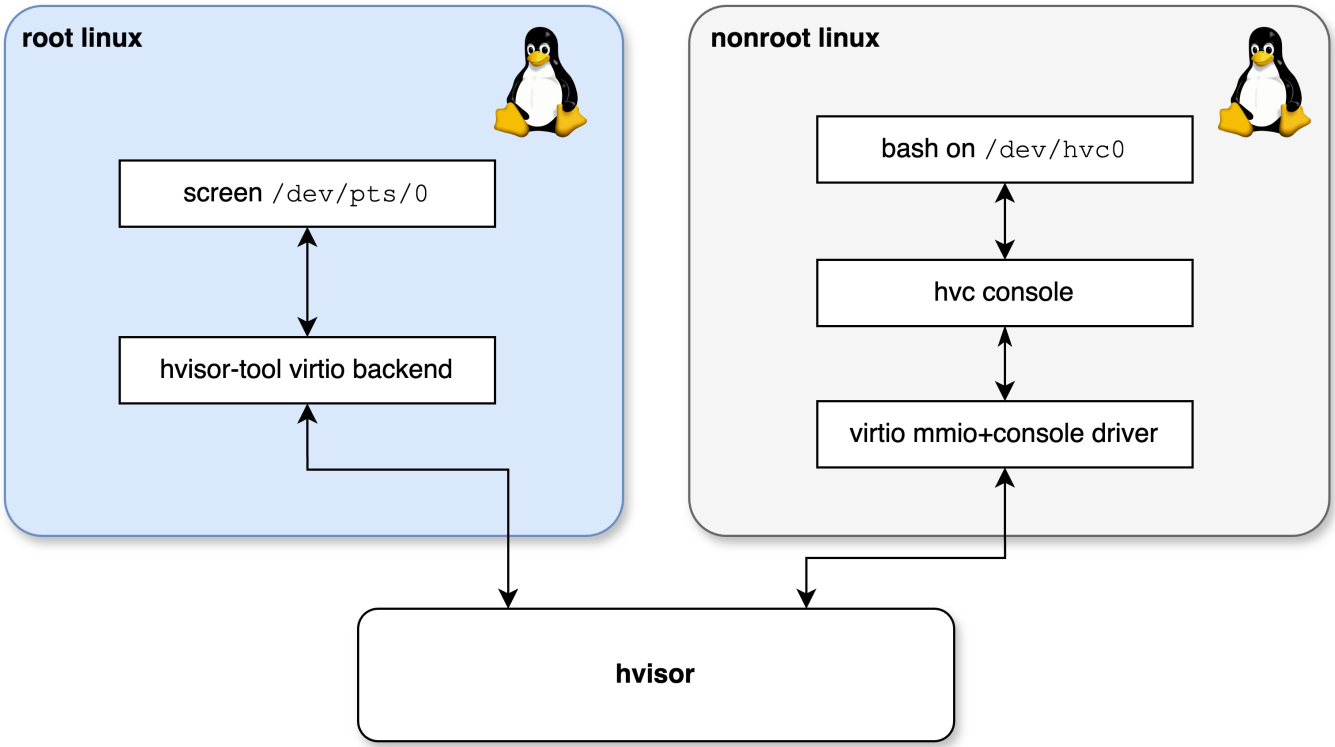
2024 年 11 月 28 日

hvc - Linux Hypervisor Consoles

- `hvc` is a tty driver for linux since linux 2.6
- main code in `drivers/tty/hvc/hvc_console.c`
- originally designed for **IBM Hypervisor Virtual Console Server(HVCS)** running on IBM PowerPC, which support accessing all linux console running on the same machine(partitioned).
- each partitioned OS will be binded a `hvc` device:
  - `hvc0` for the linux 1
  - `hvc1` for the linux 2
  - ...
- then we use screen to connect to the target console.

hvc in virtio console

- virtio console normally has 1 port(`vport0p0`) and two pairs of virtqueues(index = 0 for output and index = 1 for input).
- the data path is shown as below:



hvc in virtio console

- Moreover, hvc_ops defines a set of functions for the hvc device like get_chars, put_chars, etc.
- But theses function should be implemented by the underlying driver, like drivers of virtio console, SBI console, Xen console, etc.
  (eg. `drivers/char/virtio_console.c` set hv_ops to its implemented functions)

last time's problem: what's hvc poll and hvc kick?

- `int hvc_poll(struct hvc_struct *hp)`
  - return a `poll_mask`: 0x1 for poll read and 0x2 for poll write
  - this poll flag's job is to tell the **khvcd** the schedule futher action(eg. write the remaining data later, or try to read data later).
  - first flush hvc buffer data to underlying driver
  - then read input data, and submit it to user space(eg. bash on /dev/hvc0 will use scanf(for example) to read from it).
- `void hvc_kick(void)`
  - call wake_up_process() on the hvc_task(which is a kernel thread **khvcd**) to wake it up.
  - **khvcd**: kthread for HVC daemon, which is a function in hvc_console.c.
  - **khvcd**: if poll mask is 0, then call schedule() to sleep until explicitly waked up. Else it enter a sleep with timeout.
  - the actual "polling" is in the do-while loop in **khvcd**.

Rust Code Formal Verification

coq-of-rust

- translate Rust code to Coq code and refine it for formal verification.
- then it use *specifications* to prove the correctness of the Rust code. For example, never panic, the code behaves as what a formal whitepaper describes, etc.
- It requires to manually write a *simulation* function which is identical to the original Rust code and good for formal verification.[1]
- `https://github.com/formal-land/coq-of-rust`

[1] `https://formal.land/blog/2024/08/19/verification-move-sui-type-checker-1`

Rust Code Formal Verification

RustBelt

- RustBelt introduces λ_{Rust}, a formal version of Rust and proves the soundness of the Rust language itself.
- Ralf Jung et al. 2017. *RustBelt: securing the foundations of the Rust programming language*. POPL[2].
  `https://doi.org/10.1145/3158154`
- However, I cannot find the ways to use RustBelt to verify our Rust code(hvisor for example).
- `https://plv.mpi-sws.org/rustbelt`

[2] Principles of Programming Languages

Rust Code Formal Verification

Verus

- Verus is a SMT(Satisfiability Modulo Theories 可满足性模理论) based verifier for Rust programs.
- Andrea Lattuada et al. 2023. *Verus: Verifying Rust Programs using Linear Ghost Types*. OOPSLA[3].
  `https://doi.org/10.1145/3586037`
- proved double link list, code with interior mutability(Cell, UnsafeCell,...), concurrent code in paper.
- `https://github.com/verus-lang/verus`
- the project is still under active development.

[3] Object-Oriented Programming, Systems, Languages, and Applications

Thanks for watching!

- **TODO**: implement Xilinx's UART driver for hvisor.[4]
- **TODO**: investigate screen re-entry problem in root linux's virtio console pts.

[4] https://github.com/syswonder/hvisor/tree/dev-zcu102