在 zCore 中添加一个 syscall

潘庆霖

2020 年 6 月 15 日

目录

目录

参考资料

- **参考文档**：fuchsia0413/docs/reference/syscalls
- **代码实现**：fuchsia0413/zircon/kernel/lib/syscalls
- VDSO **代码**：fuchsia0413/zircon/system/ulib/zircon

目录

用户线程执行

- 用户线程上下文保存位置：Thread 内核对象
- 调用 context_run：进入用户态
- 从 context_run 返回：用户态发生中断/异常/系统调用

```
// zircon-loader/src/lib.rs:spawn
loop{
    let mut cx = thread.wait_for_run().await;
    kernel_hal::context_run(&mut cx);
    let mut exit = false;
    match cx.trap_num {
        0x100 => exit = handle_syscall(
            &thread, &mut cx.general).await,
        //...
    }
    thread.end_running(cx);
    if exit { break; }
}
```

handle_syscall 函数

- 读取对应上下文的寄存器值，作为系统调用参数
- 创建一个 Syscall 实例
- 执行 Syscall::syscall 函数

```
// zircon-loader/src/lib.rs:handle_syscall
async fn handle_syscall(
    thread: &Arc<Thread>, regs: &mut GeneralRegs
) -> bool {
    let mut syscall = Syscall {
        regs,
        thread: thread.clone(),
        spawn_fn: spawn,
        exit: false,
    };
    syscall.regs.rax = syscall.syscall(num, args).await as
    syscall.exit
}
```

Syscall 结构体

zircon-syscall 库实质上是在为 Syscall 实现各种各样的接口，每一个接口对应一个系统调用。在 Syscall::syscall 函数中进行分发。

```
pub struct Syscall<'a> {
    pub regs: &'a mut GeneralRegs,
    pub thread: Arc<Thread>,
    pub spawn_fn: fn(thread: Arc<Thread>),
    pub exit: bool,
}

impl Syscall<'_> {
    pub fn sys_fifo_create(...) -> ZxResult {
    ...
    }
}
```

目录

获取 object

- 每一次系统调用处理都能接触当前 Thread 对象
- thread.proc() 获取当前进程对象 Arc 指针
- proc.*_object_* 接口用于获取当前进程的内核对象

```
// zircon-syscall/src/fifo.rs: sys_fifo_write
let proc = self.thread.proc();
let fifo = proc.get_object_with_rights::<Fifo>(
    handle_value, Rights::WRITE)?;
let data = user_bytes.read_array(count * elem_size)?;
let actual_count = fifo.write(elem_size, &data, count)?;
```

例程中阻塞

- Thread 提供的阻塞 + 修改线程状态的接口
- 对应内核对象提供的阻塞式接口

```
pub async fn sys_port_wait(
    &self,handle_value: HandleValue,
    deadline: Deadline, mut packet_res: UserOutPtr<PortPack
) -> ZxResult { // get `port` from current process
    port let future = port.wait();
    pin_mut!(future);
    let packet = self
        .thread
        .blocking_run(
            future, ThreadState::BlockedPort,
            deadline.into())
        .await?;
    packet_res.write(packet)?;
    Ok(())
}
```

目录

测试代码位置

- 代码：fuchsia0413/zircon/system/utest/core
- zbi 文件：fuchsia0413/out/default/obj/zircon/system/utest/core/core-tests.zbi

测试方式

- 直接运行 shell，在 shell 中运行 runtests 命令。（部分测例无法测试）
- 重定向 zbi 文件（可测试全部）
  - 修改 rboot.conf 中的 cmdline
  - 修改 zCore/Makefile，重定向 zbi 文件到 core-tests.zbi
  - （可选）设置 zircon-syscall/src/channel.rs: test_args

Thanks!