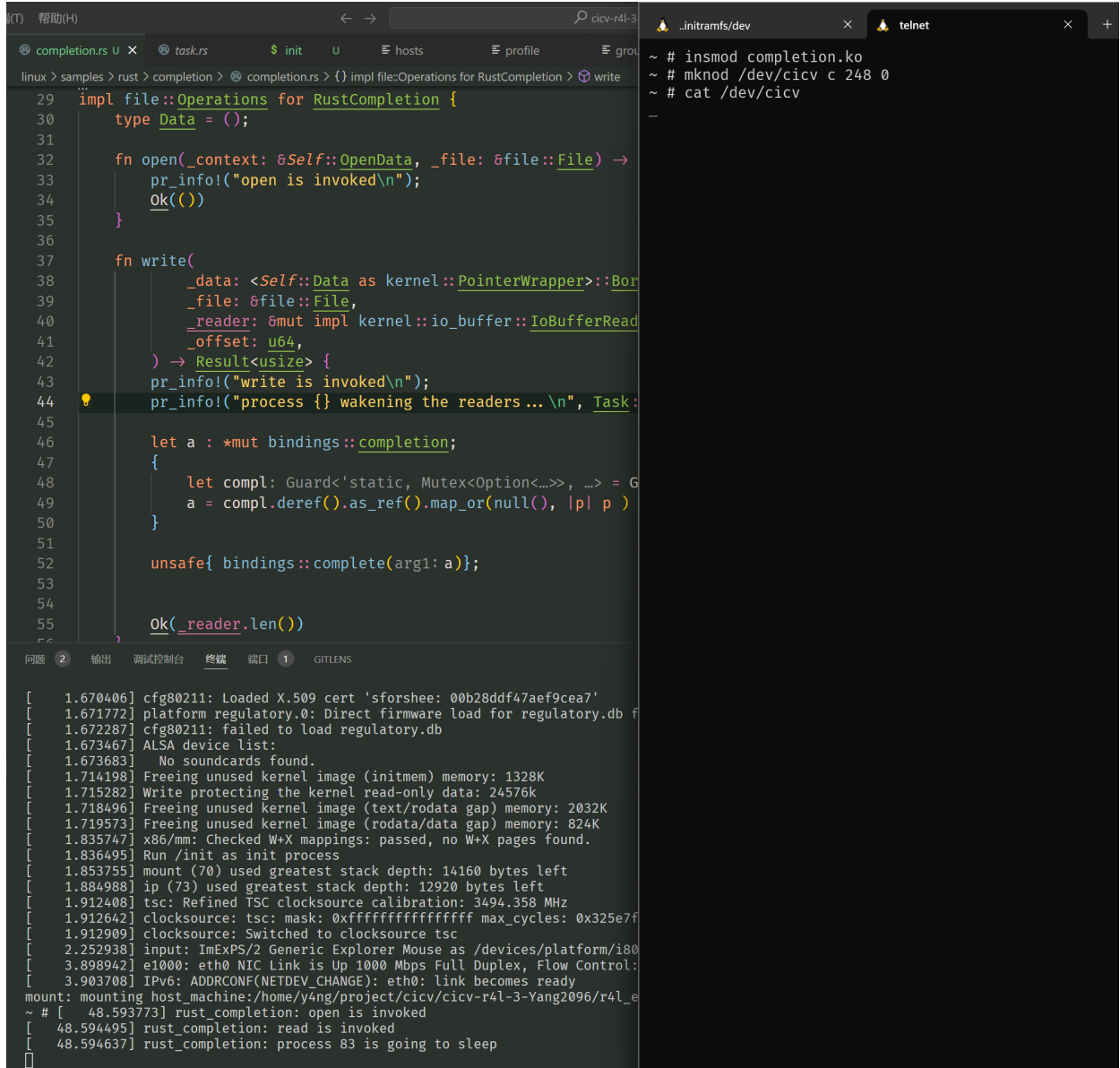


Experiment

1. 在右侧 telnet 连接的终端中装载内核模块，创建 `chrdev`，读取文件



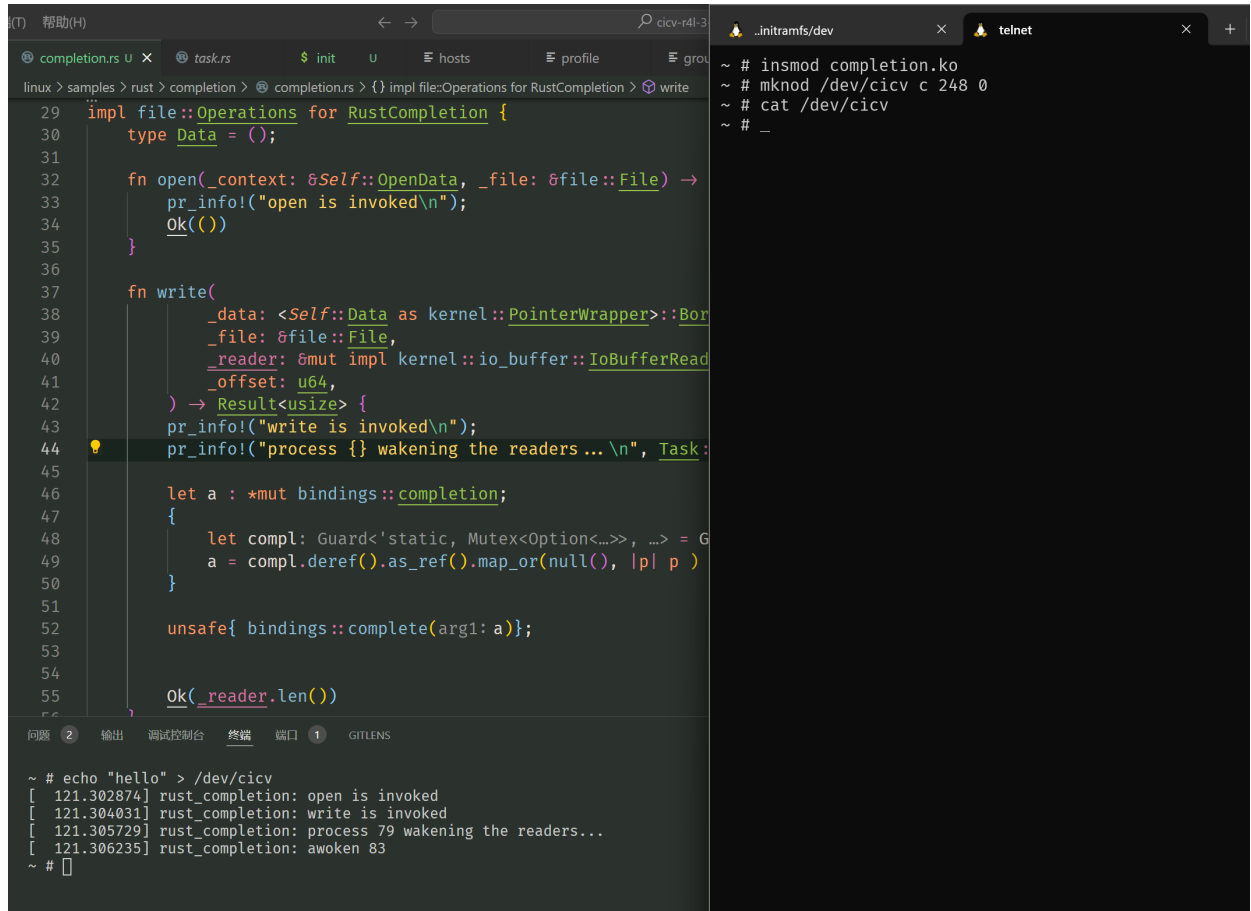
The screenshot displays a development environment with two main panels. The left panel shows a Rust source file named `completion.rs` with the following code:

```
29 impl file::Operations for RustCompletion {
30     type Data = ();
31
32     fn open(_context: &Self::OpenData, _file: &file::File) ->
33         pr_info!("open is invoked\n");
34         Ok(())
35     }
36
37     fn write(
38         _data: <Self::Data as kernel::PointerWrapper>::Bor
39         _file: &file::File,
40         _reader: &mut impl kernel::io_buffer::IoBufferRead
41         _offset: u64,
42     ) -> Result<usize> {
43         pr_info!("write is invoked\n");
44         pr_info!("process {} wakening the readers ... \n", Task:
45
46         let a : *mut bindings::completion;
47         {
48             let compl: Guard<'static, Mutex<Option<...>, ...> = G
49             a = compl.deref().as_ref().map_or(null(), |p| p )
50         }
51
52         unsafe{ bindings::complete(arg1: a));
53
54
55         Ok(_reader.len())
56     }
```

The right panel shows a telnet terminal window with the following commands and output:

```
~ # insmod completion.ko
~ # mknod /dev/cicv c 248 0
~ # cat /dev/cicv
-
[ 1.670406] cfg80211: Loaded X.509 cert 'sforshee: 00b28ddf47aef9cea7'
[ 1.671772] platform regulatory.0: Direct firmware load for regulatory.db f
[ 1.672287] cfg80211: failed to load regulatory.db
[ 1.673467] ALSA device list:
[ 1.673683]   No soundcards found.
[ 1.714198] Freeing unused kernel image (initmem) memory: 1328K
[ 1.715282] Write protecting the kernel read-only data: 24576k
[ 1.718496] Freeing unused kernel image (text/rodata gap) memory: 2032K
[ 1.719573] Freeing unused kernel image (rodata/data gap) memory: 824K
[ 1.835747] x86/mm: Checked W+X mappings: passed, no W+X pages found.
[ 1.836495] Run /init as init process
[ 1.853755] mount (70) used greatest stack depth: 14160 bytes left
[ 1.884988] ip (73) used greatest stack depth: 12920 bytes left
[ 1.912408] tsc: Refined TSC clocksource calibration: 3494.358 MHz
[ 1.912642] clocksource: tsc: mask: 0xffffffffffffffff max_cycles: 0x325e7f
[ 1.912909] clocksource: Switched to clocksource tsc
[ 2.252938] input: ImExPS/2 Generic Explorer Mouse as /devices/platform/i80
[ 3.898942] e1000: eth0 NIC Link is Up 1000 Mbps Full Duplex, Flow Control:
[ 3.903708] IPv6: ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready
mount: mounting host_machine:/home/y4ng/project/cicv/cicv-r4l-3-Yang2096/r4l_e
~ # [ 48.593773] rust_completion: open is invoked
[ 48.594495] rust_completion: read is invoked
[ 48.594637] rust_completion: process 83 is going to sleep
```

2. 在左侧 vscode 终端中写入 device，可看到右侧阻塞住的进程得以继续运行



The image shows a VS Code editor with two main panes. The left pane displays Rust code for a kernel module named `completion.rs`. The code implements a `RustCompletion` struct with a `Data` type and two functions: `open` and `write`. The `write` function uses `pr_info!` to log messages and `Task::wakeup` to wake up readers. The right pane shows a terminal window with the following commands and output:

```
~ # insmod completion.ko
~ # mknod /dev/cicv c 248 0
~ # cat /dev/cicv
~ # _
```

The bottom status bar of VS Code shows the '终端' (Terminal) tab is active. The terminal output at the bottom of the left pane shows the following log messages:

```
~ # echo "hello" > /dev/cicv
[ 121.302874] rust_completion: open is invoked
[ 121.304031] rust_completion: write is invoked
[ 121.305729] rust_completion: process 79 wakening the readers...
[ 121.306235] rust_completion: awoken 83
~ #
```