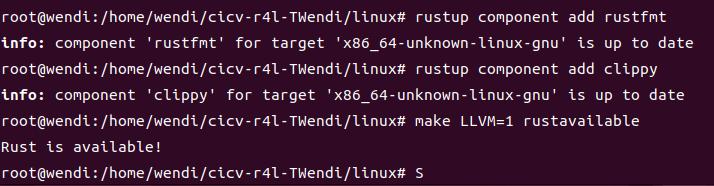
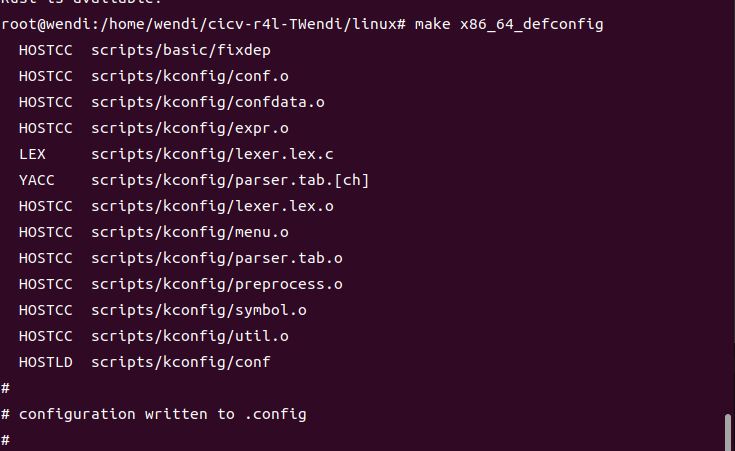
**第二阶段作业-田文迪**

**环境配置结果**

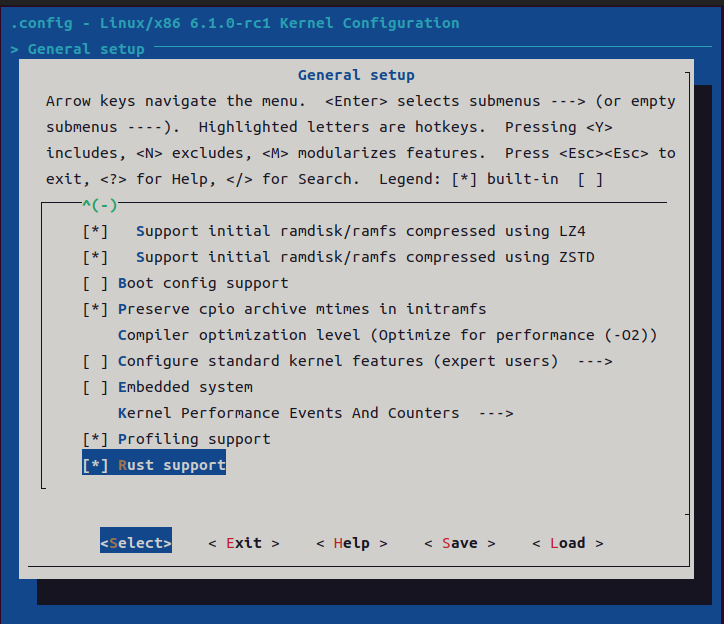


1. **作业1**

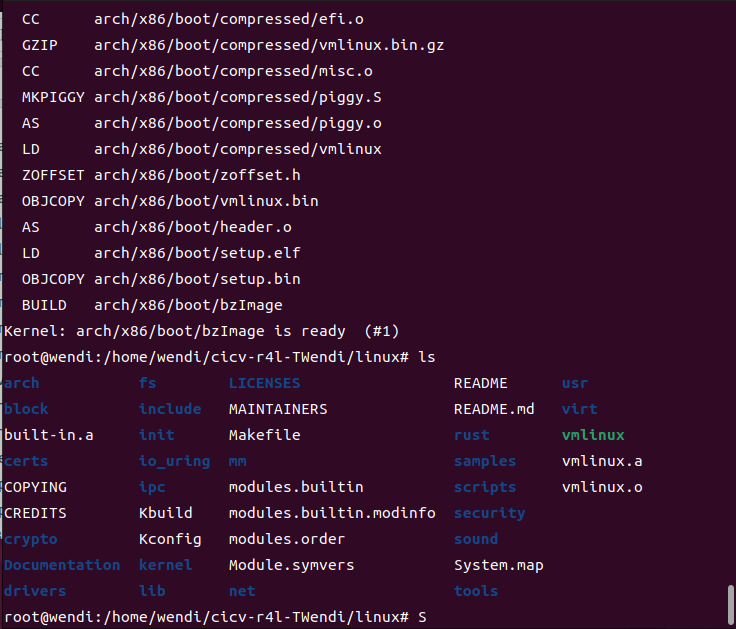
设置x86默认配置：



menuconfig选中Rust Support：

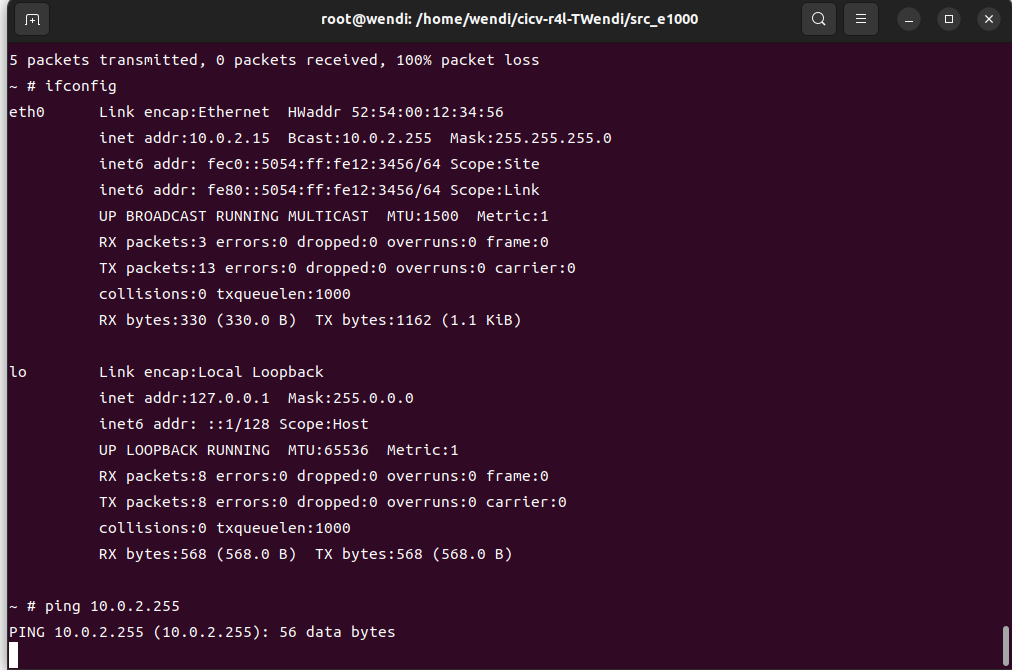


编译生成mlinux：

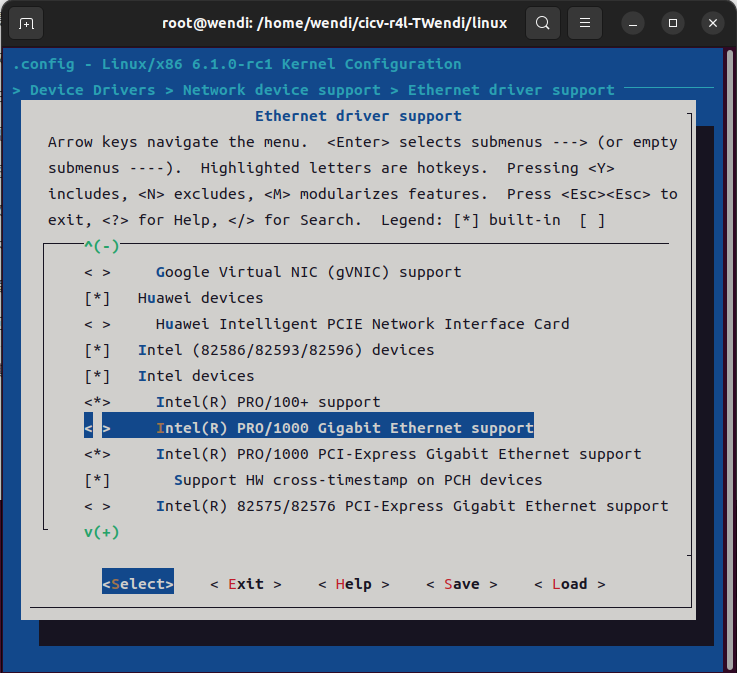


2. **作业2**

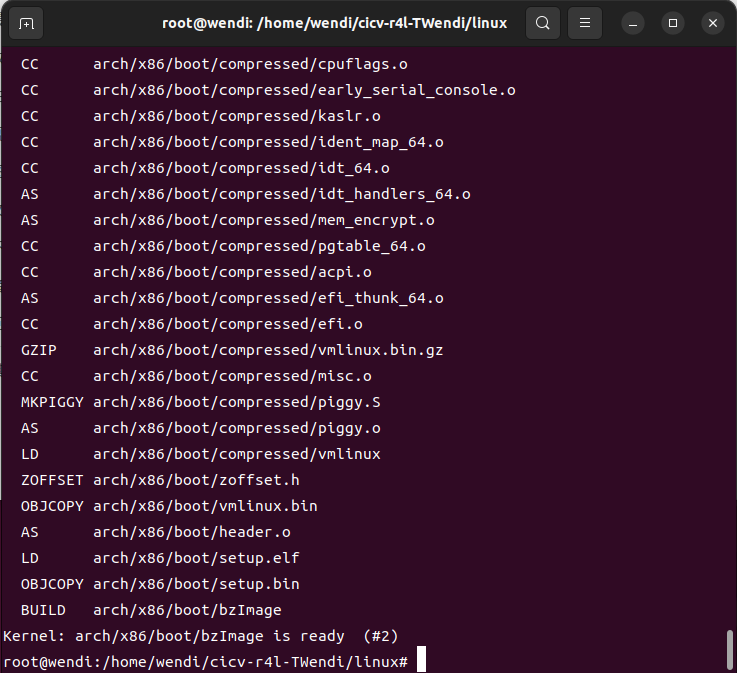
使用ifconfig，出现C版本e1000网卡驱动：



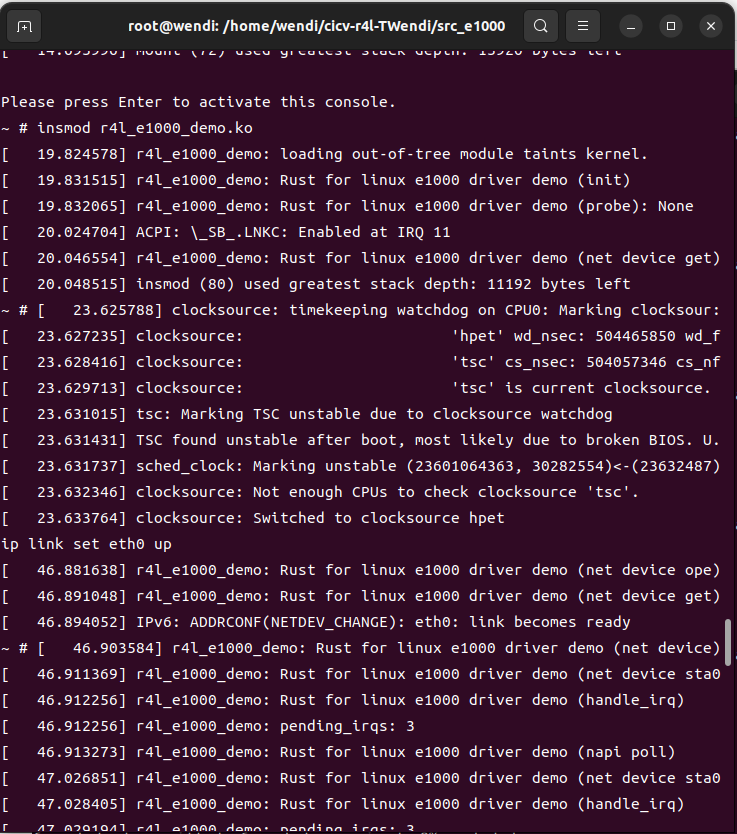
menuconfig中禁用C版本e1000网卡驱动

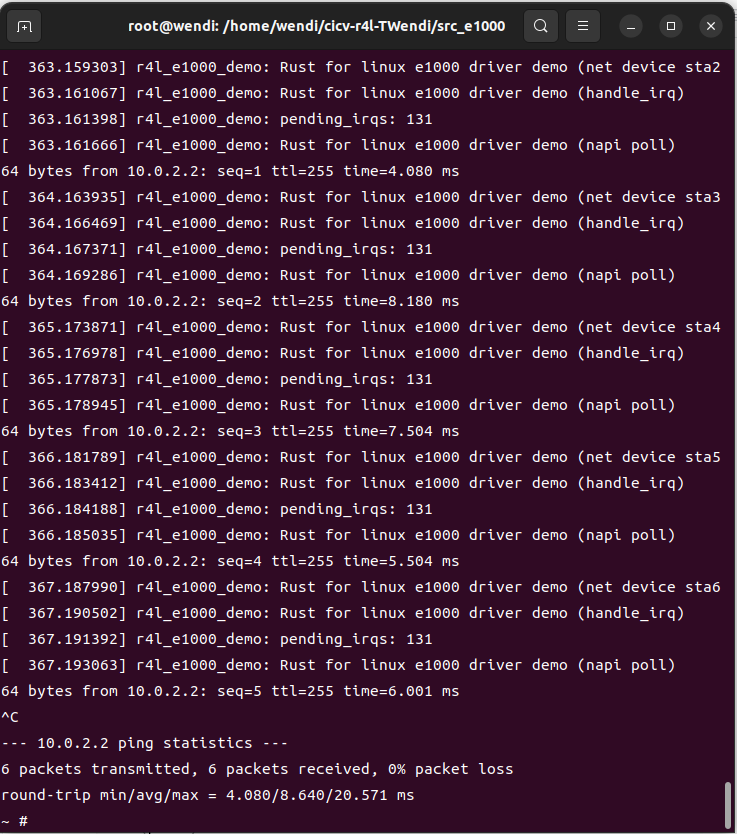


重新编译：

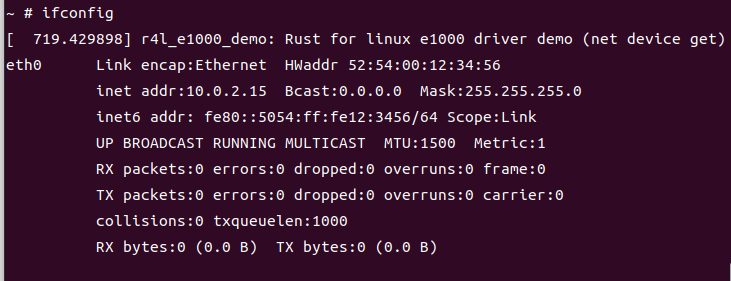


测试结果：





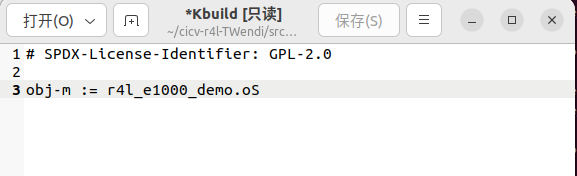
禁用后的ifconfig结果：



* 问题回答：

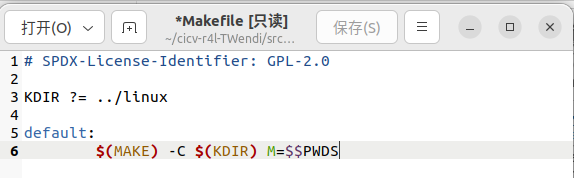
1、编译成内核模块，是在哪个文件中以哪条语句定义的？

在src\_e1000文件夹下的kbuild中定义



2、该模块位于独立的文件夹内，却能编译成Linux内核模块，这叫做out-of-tree module，请分析它是如何与内核代码产生联系的？

makefile中将目录转移到linux目录下，从而与内核代码产生联系。



3. **作业3**

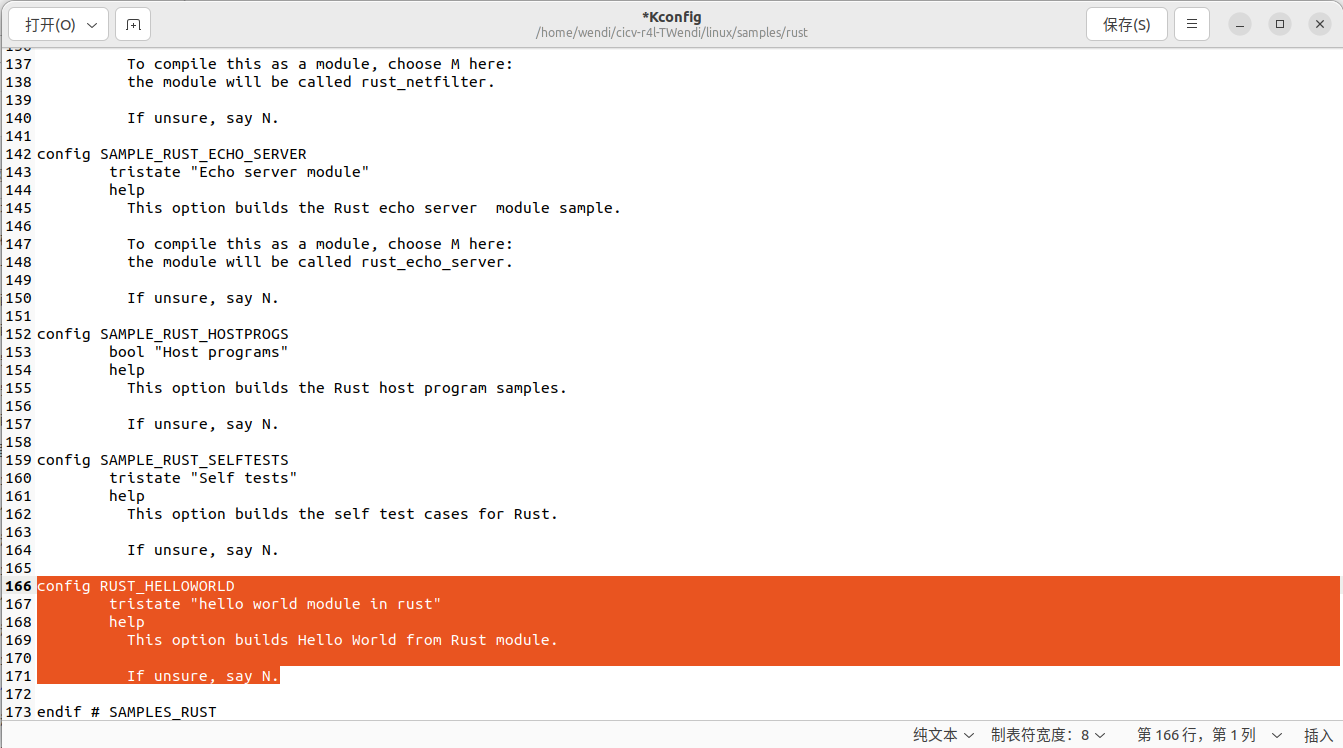
添加源代码文件：



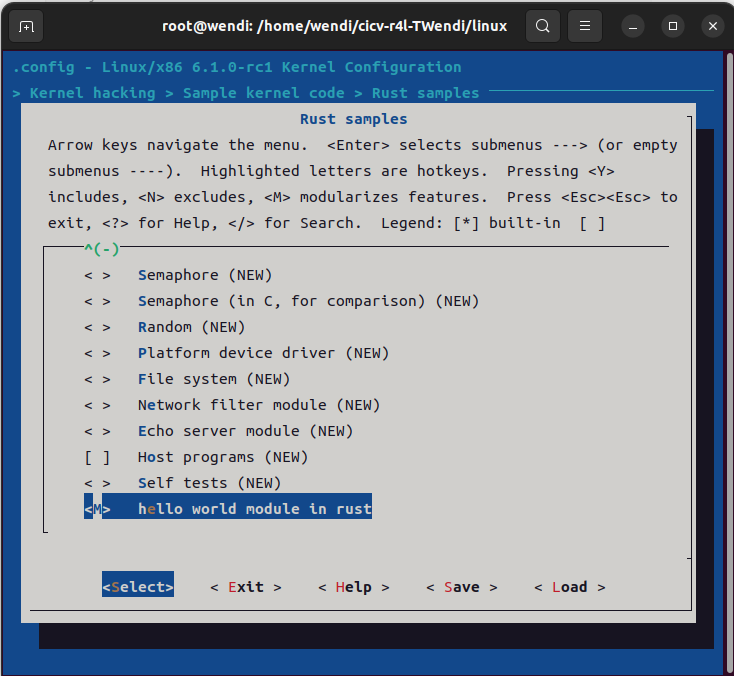
修改Makefile：



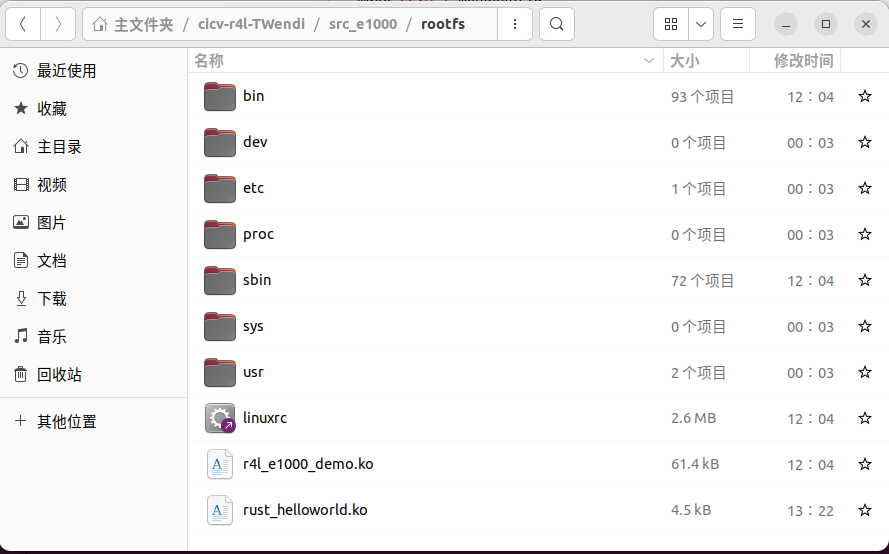
修改Kconfig：



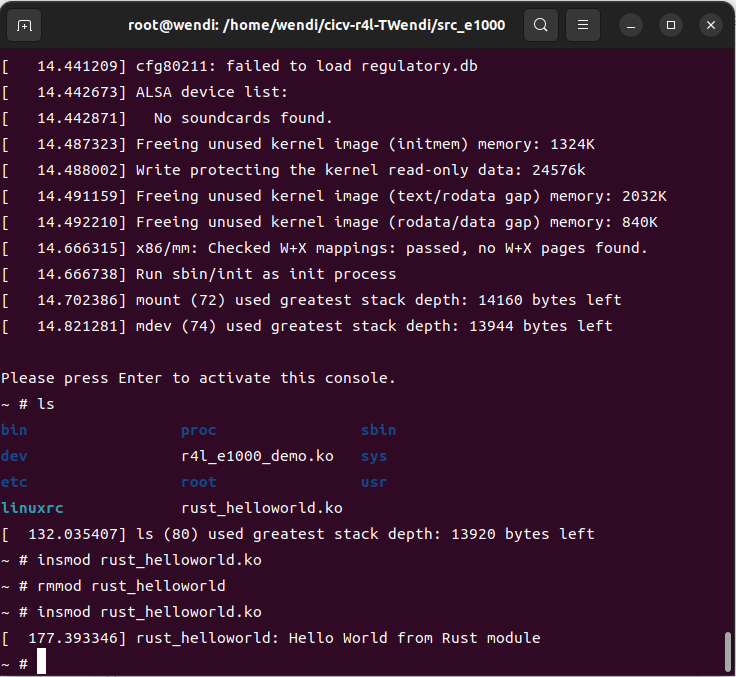
更改menuconfig配置：



复制.ko文件：



Linux shell 下测试，insmod 后没有立即显示字符串，是因为没有在字符串后加上"\n"：

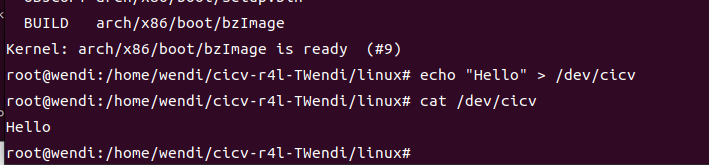


4. **作业5**

在rust\_chrdev.rs中添加如下修改：

|  |
| --- |
| Rust fn write(  this: &Self,  \_file: &file::File,  reader: &mut impl kernel::io\_buffer::IoBufferReader,  offset: u64, ) -> Result<usize> {  let mut buf = this.inner.lock();  let ava\_write\_size = min(GLOBALMEM\_SIZE - offset as usize, reader.len());  reader.read\_slice(&mut buf[offset as usize..offset as usize + ava\_write\_size])?;  Ok(ava\_write\_size) }  fn read(  this: &Self,  \_file: &file::File,  writer: &mut impl kernel::io\_buffer::IoBufferWriter,  offset: u64, ) -> Result<usize> {  let buf = this.inner.lock();  let ava\_read\_size = min(GLOBALMEM\_SIZE - offset as usize, writer.len());  writer.write\_slice(&buf[offset as usize..offset as usize + ava\_read\_size])?;  Ok(ava\_read\_size) } |

重新编译后测试如下：



* 问题回答

作业5中的字符设备/dev/cicv是怎么创建的？它的设备号是多少？它是如何与我们写的字符设备驱动关联上的？

通过/etc/init.d/rcS.sh脚本中的“mknod”指令创建的；

主设备号为248，次设备号为0；

在驱动程序注册设备时指定了设备号和设备名，通过这些信息关联字符设备

