



# 本科生实验报告

实验课程：\_\_\_\_\_操作系统\_\_\_\_\_

实验名称：\_\_\_\_\_编译内核\_\_\_\_\_

专业名称：\_\_\_\_\_计算机科学与技术\_\_\_\_\_

学生姓名：\_\_\_\_\_凌国明\_\_\_\_\_

学生学号：\_\_\_\_\_21307077\_\_\_\_\_

实验地点：\_\_\_\_\_教室\_\_\_\_\_

实验成绩：\_\_\_\_\_

报告时间：\_\_\_\_\_2023. 03. 06\_\_\_\_\_

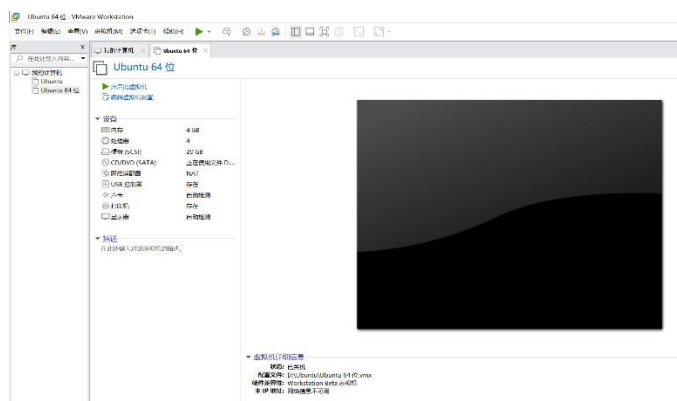
## 1. 实验要求

在本次实验中，同学们会熟悉现有 Linux 内核的编译过程和启动过程，并在自行编译内核的基础上构建简单应用并启动。同时，同学们会利用精简的 Busybox 工具集构建简单的 OS，熟悉现代操作系统的构建过程。此外，同学们会熟悉编译环境、相关工具集，并能够实现内核远程调试。具体内容如下。

1. 搭建 OS 内核开发环境包括：代码编辑环境、编译环境、运行环境、调试环境等。
2. 下载并编译 i386（32 位）内核，并利用 qemu 启动内核。
3. 熟悉制作 initramfs 的方法。
4. 编写简单应用程序随内核启动运行。
5. 编译 i386 版本的 Busybox，随内核启动，构建简单的 OS。
6. 开启远程调试功能，进行调试跟踪代码运行。
7. 撰写实验报告。

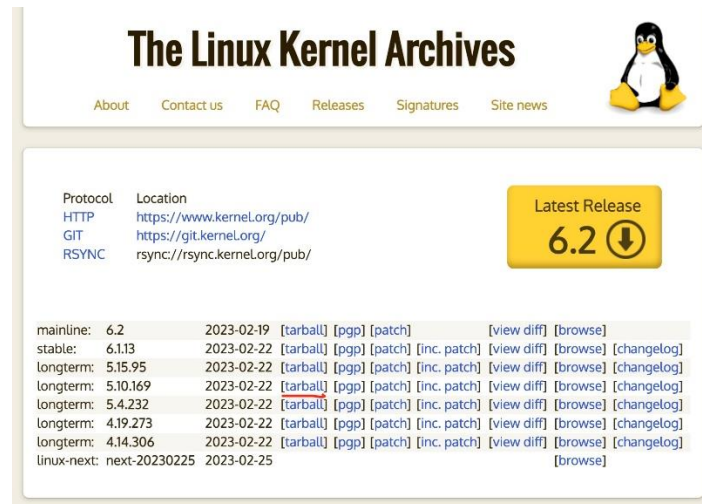
## 2. 实验过程

- 1) 环境配置，下载 Vmware，安装 Ubuntu 虚拟机，换源。

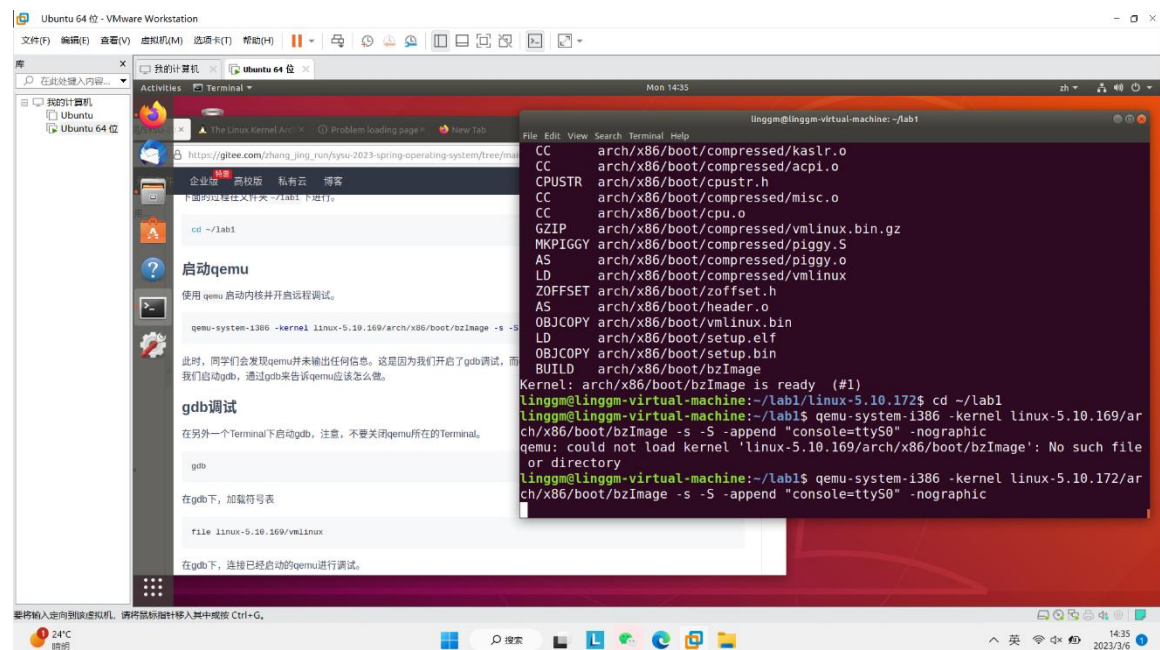


2) 配置 C/C++环境，安装其他工具

3) 下载了 Linux5.10.172，并编译 Linux 内核



4) 启动内核并调试，使用 qemu 启动内核，开启远程调试



5) 第 4 步完成后 qemu 没有输出任何信息，这是因为开启了 gdb 调试，qemu 需等待 gdb 的 c 指令才能继续执行。

第五步是进行 gdb 调试：新开一个终端，启动 gdb，此时不应关闭原有终端，先加载符号表，然后连接 qemu 进行调试，设置断点后，输入 c 运行 qemu

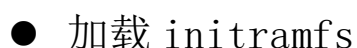
```
linggm@linggm-virtual-machine:~/lab1/linux-5.10.172$ cd
linggm@linggm-virtual-machine:~$ cd lab1
linggm@linggm-virtual-machine:~/lab1$ gdb
GNU gdb (Ubuntu 8.1-0ubuntu3.2) 8.1.0.20180409-git
Copyright (C) 2018 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.htm
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word".
(gdb) file linux-5.10.172/vmlinux
Reading symbols from linux-5.10.172/vmlinux...done.
(gdb)
```

```
File Edit View Search Terminal Help
GNU gdb (Ubuntu 8.1-0ubuntu3.2) 8.1.0.20180409-git
Copyright (C) 2018 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.htm
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word".
(gdb) file linux-5.10.172/vmlinux
Reading symbols from linux-5.10.172/vmlinux...done.
(gdb) target remote:1234
Remote debugging using :1234
0x0000ffff in ?? ()
(gdb) break start_kernel
Breakpoint 1 at 0xc20db82e:
(gdb) c
Continuing.

[ 3.074524] Please append a correct "root=" boot option; I
[ 3.076369] le partitions:
[ 3.076423] 0b00 1048575 sr0
[ 3.077083] driver: sr
[ 3.078177] Kernel panic - not syncing: VFS: Unable to mou
[ 3.078564] wn-block(0,0)
[ 3.078564] CPU: 0 PID: 1 Comm: swapper/0 Not tainted 5.10
[ 3.078564] Hardware name: QEMU Standard PC (i440FX + PIIX
[ 3.079341] 2-lubuntu1 04/01/2014
[ 3.079341] Call Trace:
[ 3.080338] dump_stack+0x54/0x68
[ 3.080620] panic+0xb1/0x25a
[ 3.080965] mount_block_root+0x133/0x1b3
[ 3.081328] mount_root+0xd3/0xec
[ 3.081619] prepare_namespace+0x116/0x141
[ 3.082017] kernel_init_freeable+0x1cd/0x1da
[ 3.082370] ? rest_init+0xa0/0xa0
[ 3.084454] kernel_init+0x8/0xf0
[ 3.084615] ret_from_fork+0x1c/0x28
[ 3.085765] Kernel Offset: disabled
[ 3.086316] ---[ end Kernel panic - not syncing: VFS: Unab
on unknown-block(0,0) ]---
```

- Hello World

```
gcc -o helloworld -m32 -static helloworld.c
```



The screenshot displays a virtual machine interface with two windows. The left window shows a file editor with a C program named `hello.c`. The code includes `<stdio.h>`, defines `main()` which prints "Hello World!", and calls `system("initramfs")`. Comments indicate saving the file, loading `initramfs`, starting the kernel, and adding `initramfs`. The right window shows a terminal running `gcc -o helloworld -m32 -static helloworld.c`, followed by `qemu-system-i386 -kernel linux-5.10.172-vmlinux`. The terminal output shows the loading of GNU GPL v3 license, GDB configuration details, and the execution of `init/main.c`, which prints "Hello World!" before reaching a breakpoint.

Thunderbird Mail | zhang\_jing\_run/sysu-2023-spring-operation-system-free/main/hello.c  
linggmg@linggmg-virtual-machine: ~\$  
linggmg@linggmg-virtual-machine: ~\$

企业版 高校版 私有云 博客

File Edit View Search Terminal Help

File Edit View Search Terminal Help

main.c:  
[ 2.531597] Loading compiled-in iLicense GPLv3+: GNU GPL version 3 or later <<http://gnu.org/licenses/gpl.html>>  
[ 2.547322] PM: Magic number: This is free software; you are free to change and redistribute it.  
[ 2.548353] printk: console [neThere is NO WARRANTY, to the extent permitted by law. Type 'showconfig' to show configuration details.  
[ 2.548997] netconsole: network and "show warranty" for details.  
[ 2.552653] cfg80211: Loading cThis GDB was configured as "x86\_64-linux-gnu".  
Type 'show configuration' for configuration details.  
For bug reporting instructions, please see:  
[ 2.583895] cfg80211: Loaded X.<<http://www.gnu.org/software/gdb/bugs/>>.  
[ 2.586167] platform regulatoryFind the GDB manual and other documentation resources online at<<http://www.gnu.org/software/gdb/documentation/>>.  
[ 2.587176] cfg80211: failed to For help, type "help".  
[ 2.588642] ALSA device list: Type "apropos word" to search for commands related to "word".  
[ 2.589165] No soundcards found(gdb) file linux-5.10.172/vmlinux  
[ 2.629198] Freezing unused kernReading symbols from linux-5.10.172/vmlinux...done.  
[ 2.633775] Write protecting ke(gdb) targ  
[ 2.634603] Run helloworld as iArgument required (target name). Try `help target'  
[ 2.650731] tsc: Refined TSC cl(gdb) target remote:1234  
[ 2.651439] clocksource: tsc: mRemote debugging using :1234  
Breakpoint 1 at 0xc20db82e: file init/main.c, line 850.  
[ 2.651984] clocksource: Switch(gdb) break start\_kernel  
[ 3.107846] input: ImExPS/2 Gen(gdb) c  
/seriol/input/input3 Continuing.  
^[\_

上述文件保存在 ~/lab1/helloworld.c中

gcc -o helloworld -m32 -static helloworld.c

加载initramfs

开始打包initramfs。

echo helloworld | cpio -o --format o --list | dd if=/dev/null of=initramfs.img bs=1M count=1

启动内核，并加载initramfs。

qemu-system-i386 -kernel linux-5.10.172-vmlinux -drive file=initramfs.img

将上面的gdb的调试过程，可以看到如下输出：

编译并启动BusyBox

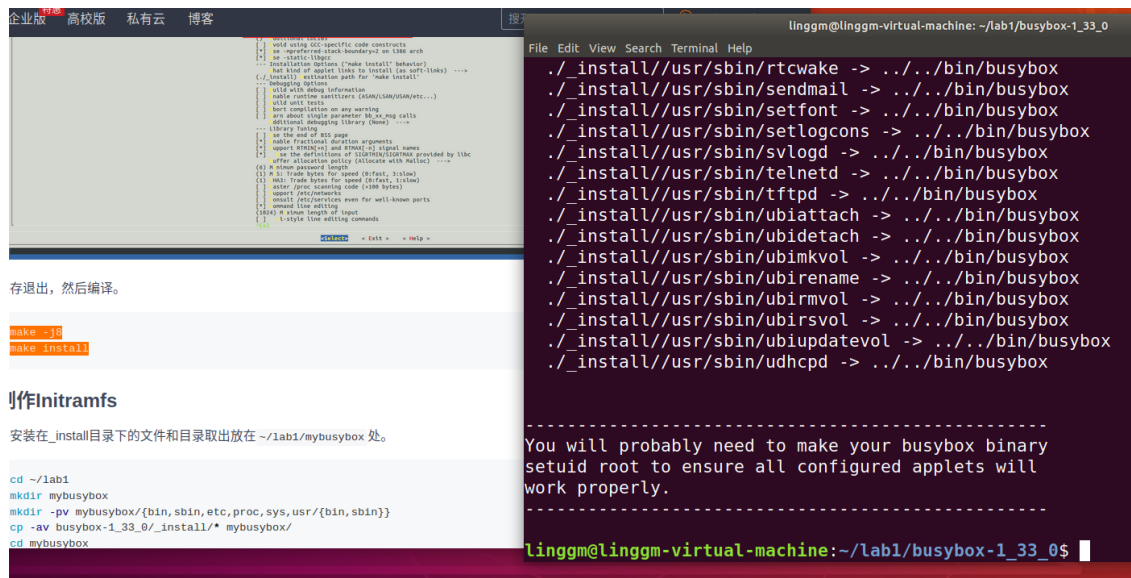
下面的过程在文件夹 ~/lab1下进行。



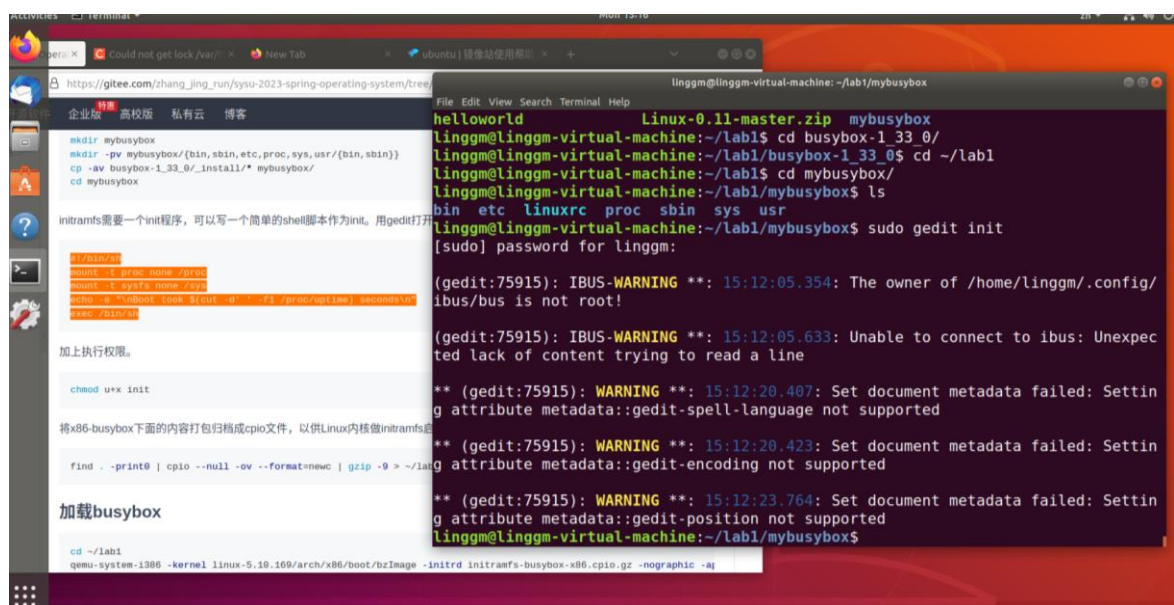
可以看到，按如上步骤，helloworld 程序得以运行。

## 7) 编译并启动 busybox

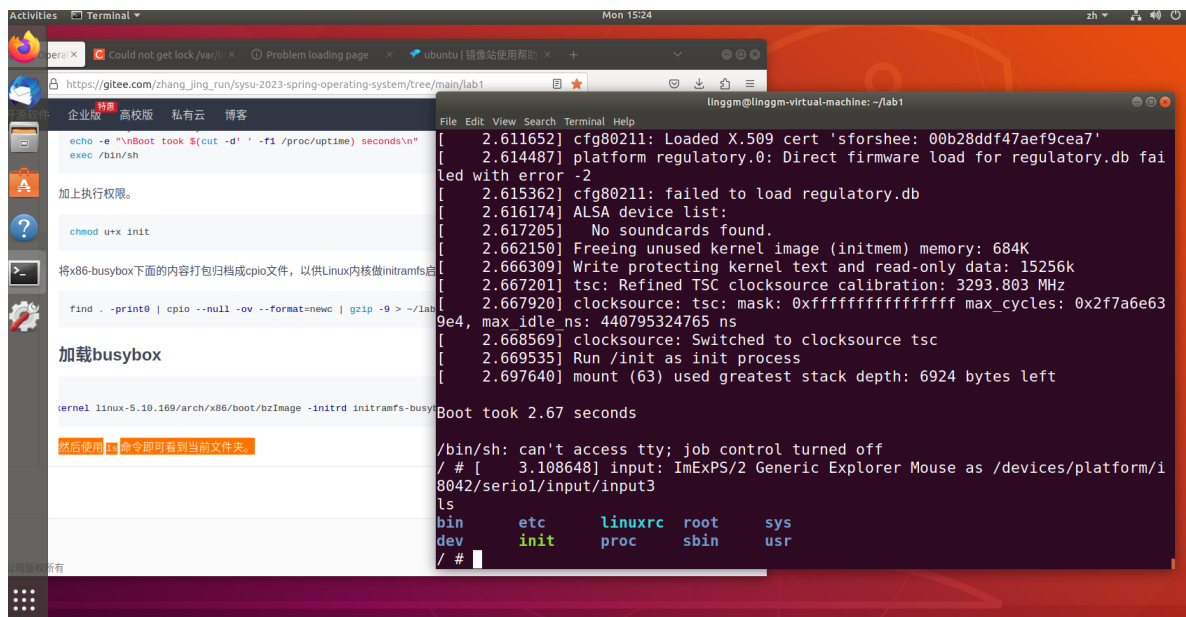
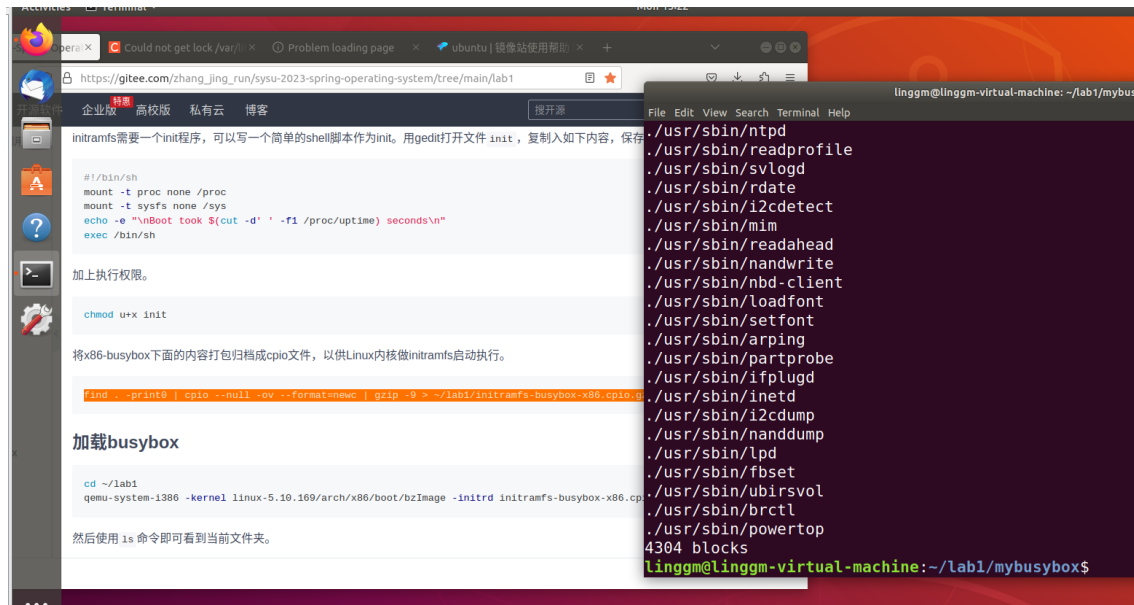
- ## ● 下载，解压，编译



- ## ● 制作 initramfs



## ● 加载 busybox



## 8) 下载 Linux0.11 内核并编译，注意修改 Makefile

```
RAMDISK = #-DRAMDISK=512  
  
# This is a basic Makefile for setting the g  
include Makefile.header  
  
LDLFLAGS += -Ttext 0 -e startup_32  
CFLAGS += $(RAMDISK) -Iinclude -g -m32  
CPP += -Iinclude
```

## 9) 使用 qemu-system-i386 加载，启动内核

```
linggm@linggm-virtual-machine: ~/lab1/Linux-0.11-master
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word".
(gdb) file tools/system
Reading symbols from tools/system...done.
(gdb) target remote :1234
Remote debugging using :1234
do_execve (eip=0x0 <startup_32>, tmp=0, filename=0x0 <startup_32>,
  argv=0x0 <startup_32>, envp=0x0 <startup_32>) at exec.c:192
192      int sh_bang = 0;
(gdb) break *0x7c00
Breakpoint 1 at 0x7c00: file traps.c, line 76.
(gdb) c
Continuing.

Breakpoint 1, 0x00007c00 in die (str=0x6f62 <sleep_on+45> "", esp_ptr=35543,
  nr=0) at traps.c:76
76      printk("%p ",get_seg_long(0x17,i+(long *)esp[3])
  );
(gdb) x 0x7dfe
0x7dfe <do_int3+123>: 0x0000aa55
(gdb) x 0x7dff
0x7dff <do_int3+124>: 0x000000aa
(gdb)
detecting the format is dangerous for raw images, write o
...format explicitly to remove the restrictions.
```

## 10) 使用 gdb 进行远程调试

```
linggm@linggm-virtual-machine: ~/lab1/Linux-0.11-master
192      int sh_bang = 0;
(gdb) break *0x7c00
Breakpoint 1 at 0x7c00: file traps.c, line 76.
(gdb) c
Continuing.

Breakpoint 1, 0x00007c00 in die (str=0x6f62 <sleep_on+45> "", esp_ptr=35543,
  nr=0) at traps.c:76
76      printk("%p ",get_seg_long(0x17,i+(long *)esp[3])
  );
(gdb) x 0x7dfe
0x7dfe <do_int3+123>: 0x0000aa55
(gdb) x 0x7dff
0x7dff <do_int3+124>: 0x000000aa
(gdb) break 0x7dfe
Function "0x7dfe" not defined.
Make breakpoint pending on future shared library load? (y or [n]) n
(gdb) break *0x7dfe
Breakpoint 2 at 0x7dfe: file traps.c, line 114.
(gdb) break *0x7dff
Breakpoint 3 at 0x7dff: file traps.c, line 114.
(gdb) c
Continuing.
```

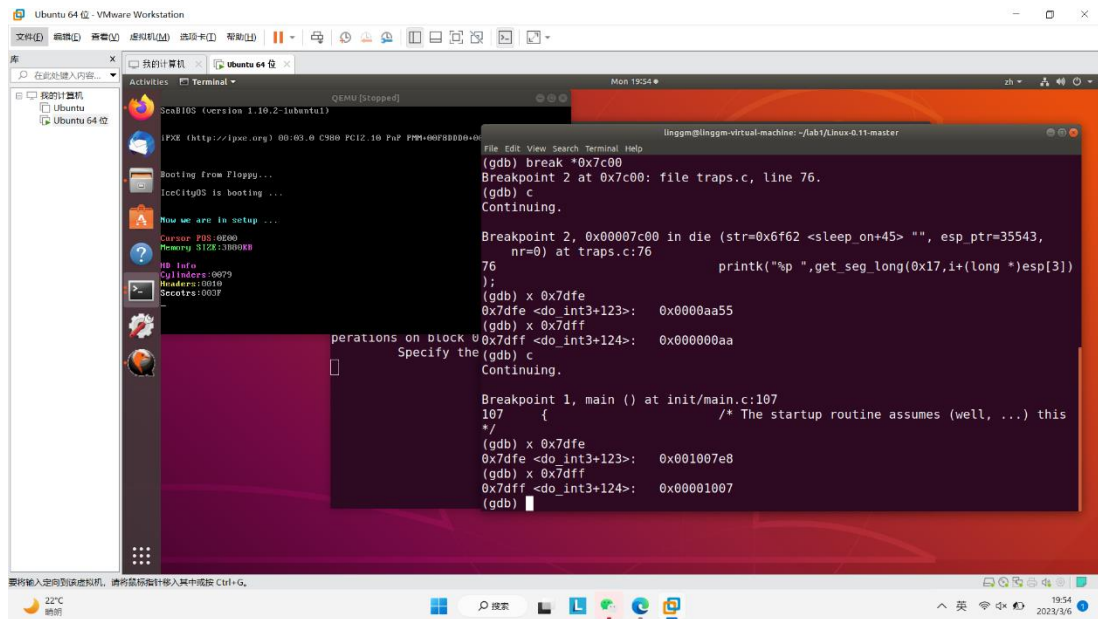
先 file/tools/system 加载符号表

再 target remote :1234 连接 qemu

再 set disassembly-flavor intel

然后 break 设置断点，最后输入 c 运行





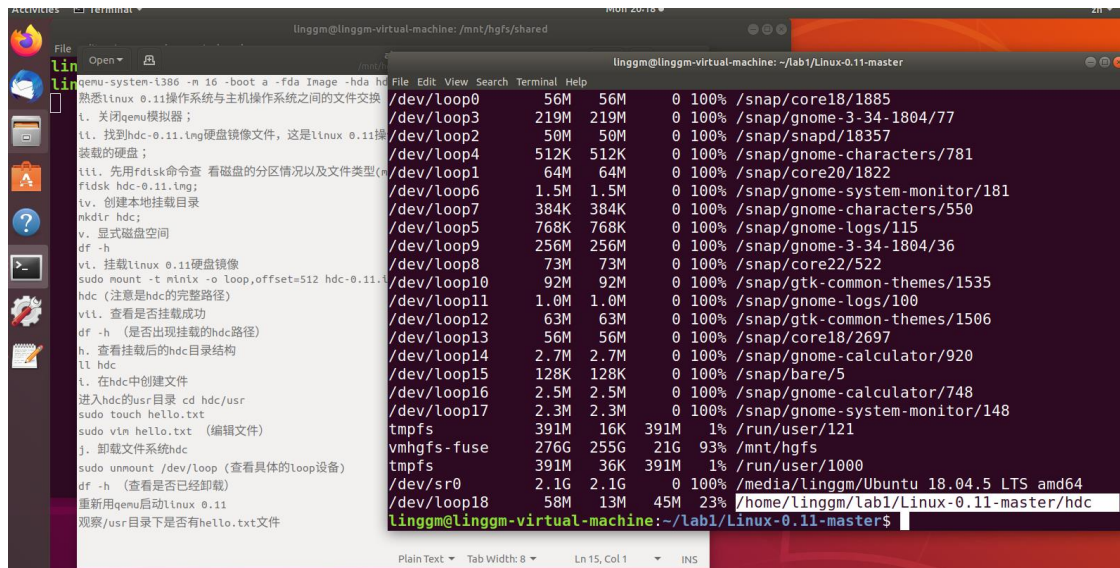
```
(gdb) x 0x7dfe
0x7dfe <do_int3+123>: 0x0000aa55
(gdb) x 0x7dff
0x7dff <do_int3+124>: 0x000000aa
(gdb) c
Continuing.

Breakpoint 1, main () at init/main.c:107
107      {
/* The star
*/
(gdb) x 0x7dfe
0x7dfe <do_int3+123>: 0x001007e8
(gdb) x 0x7dff
0x7dff <do_int3+124>: 0x00001007
(gdb)
```

0x7DFE 和 0x7DFF 的不同阶段的内容如上图

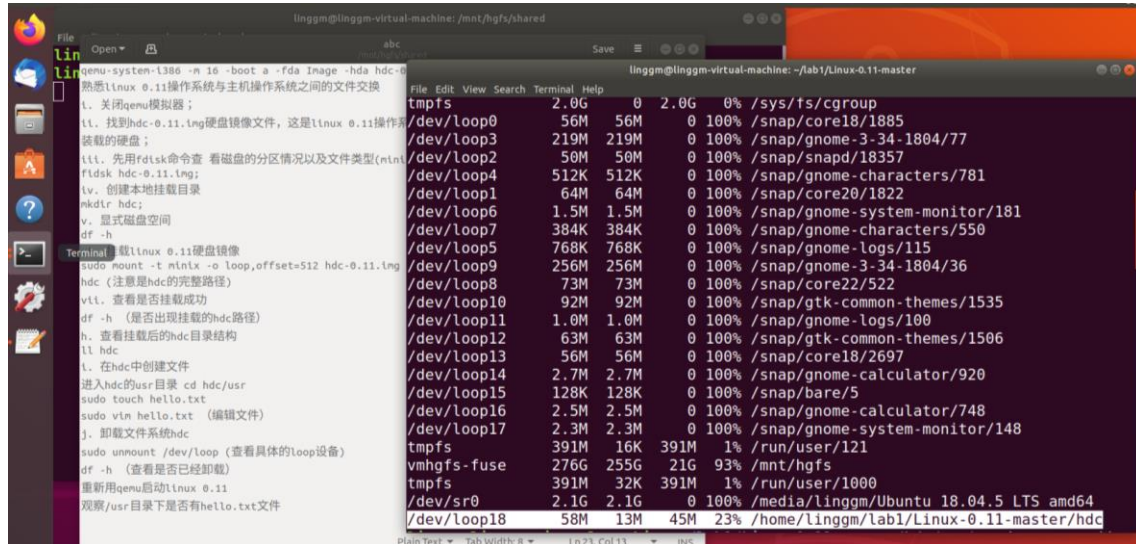
11) 熟悉 Linux0.11 操作系统与主机操作系统的文件交换

- 关闭 qemu, 找到硬盘镜像文件, 创建本地挂载目录

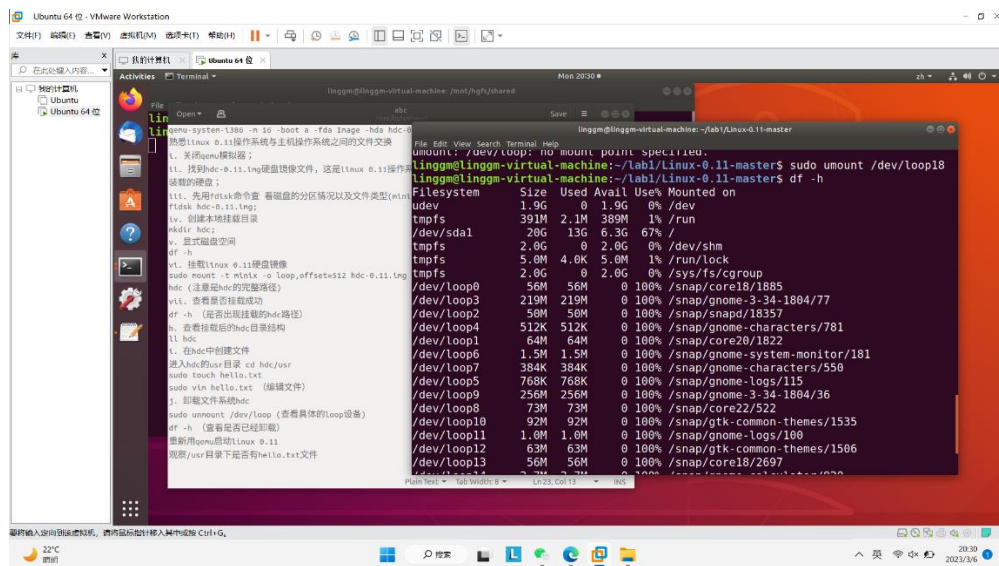


Loop18 中有 hdc 的路径，挂载成功

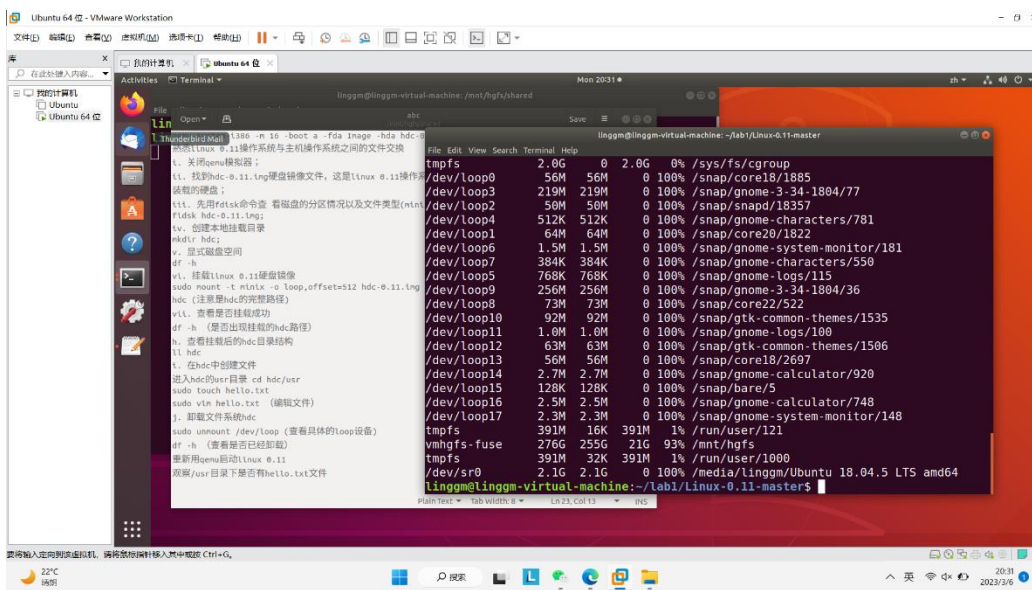
- 在 hdc/src 中创建 hello.txt 文件
- 卸载文件系统 hdc



卸载前如上图



## 输入卸载指令



卸载后，有 hdc 路径的 loop18 消失

- 打开 qemu，启动 gdb，打开 qemu 中的 src



