

1 实验概述

- 实验任务一: 环境配置
- 实验任务二: 编译 Linux 内核
- 实验任务三: Qemu 启动内核并开启远程调试
- 实验任务四: 制作 Initramfs
- 实验任务五: 编译并启动 busybox
- 实验任务六: 编译、启动和调试 Linux 0.11 内核

2 实验步骤与实验结果

2.1 实验任务一

2.1.1 任务要求

1. 更换下载源为清华源
2. 配置 C/C++ 环境
3. 安装其他工具

2.1.2 实验步骤与结果

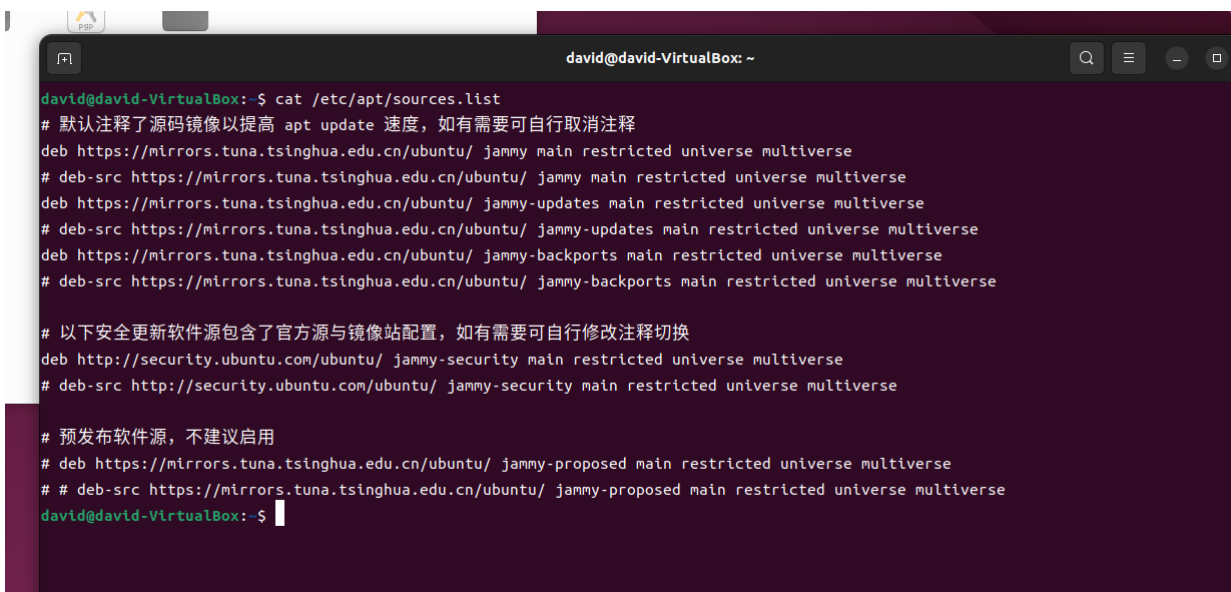
备份原来的下载源:

```
sudo mv /etc/apt/sources.list /etc/apt/sources.list.backup
```

下载清华源, 并将下载源复制进/etc/apt/sources.list 后退出

```
sudo gedit /etc/apt/sources.list  
sudo apt update
```

如图, 已经配置好清华源:



```
david@david-VirtualBox: ~  
david@david-VirtualBox:~$ cat /etc/apt/sources.list  
# 默认注释了源码镜像以提高 apt update 速度, 如有需要可自行取消注释  
deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy main restricted universe multiverse  
# deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy main restricted universe multiverse  
deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy-updates main restricted universe multiverse  
# deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy-updates main restricted universe multiverse  
deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy-backports main restricted universe multiverse  
# deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy-backports main restricted universe multiverse  
  
# 以下安全更新软件源包含了官方源与镜像站配置, 如有需要可自行修改注释切换  
deb http://security.ubuntu.com/ubuntu/ jammy-security main restricted universe multiverse  
# deb-src http://security.ubuntu.com/ubuntu/ jammy-security main restricted universe multiverse  
  
# 预发布软件源, 不建议启用  
# deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy-proposed main restricted universe multiverse  
# # deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy-proposed main restricted universe multiverse  
david@david-VirtualBox:~$
```

配置 C/C++ 环境:

```
sudo apt install binutils
sudo apt install gcc
gcc -v
```

如图, 已经配置好 C/C++ 环境:

```
david@david-VirtualBox:~$ gcc -v
Using built-in specs.
COLLECT_GCC=gcc
COLLECT_LTO_WRAPPER=/usr/lib/gcc/x86_64-linux-gnu/11/lto-wrapper
OFFLOAD_TARGET_NAMES=nvptx-none:amdgc-nvptx-none:amdhsa
OFFLOAD_TARGET_DEFAULT=1
Target: x86_64-linux-gnu
Configured with: ../src/configure -v --with-pkgversion='Ubuntu 11.4.0-1ubuntu1~22.04' --with-bugurl=file:///usr/share/doc/gcc-11/README.Bugs --enable-languages=c,ada,c++,go,brig,d,fortran,objc,obj-c++,m2 --prefix=/usr --with-gcc-major-version-only --program-suffix=-11 --program-prefix=x86_64-linux-gnu- --enable-shared --enable-linker-build-id --libexecdir=/usr/lib --without-included-gettext --enable-threads=posix --libdir=/usr/lib --enable-nls --enable-bootstrap --enable-clocale=gnu --enable-libstdcxx-debug --enable-libstdcxx-time=yes --with-default-libstdcxx-abi=new --enable-gnu-unique-object --disable-vtable-verify --enable-plugin --enable-default-pie --with-system-zlib --enable-libphobos-checking=release --with-target-system-zlib=auto --enable-objc-gc=auto --enable-multiarch --disable-werror --enable-cet --with-arch=32=i686 --with-abi=m64 --with-multilib-list=m32,m64,mx32 --enable-multilib --with-tune=generic --enable-offload-targets=nvptx-none=/build/gcc-11-XeT9lY/gcc-11-11.4.0/debian/tmp-nvptx/usr,amdgc-nvptx-none:amdhsa=/build/gcc-11-XeT9lY/gcc-11-11.4.0/debian/tmp-gcn/usr --without-cuda-driver --enable-checking=release --build=x86_64-linux-gnu --host=x86_64-linux-gnu --target=x86_64-linux-gnu --with-build-config=bootstrap-lto-lean --enable-link-serialization=2
Thread model: posix
Supported LTO compression algorithms: zlib zstd
gcc version 11.4.0 (Ubuntu 11.4.0-1ubuntu1~22.04)
david@david-VirtualBox:~$
```

安装其它工具:

```
sudo apt install nasm
sudo apt install qemu
sudo apt install cmake
sudo apt install libncurses5-dev
sudo apt install bison
sudo apt install flex
sudo apt install libssl-dev
sudo apt install libc6-dev-i386
```

已配置完成:

```
david@david-VirtualBox:~$ qemu-system-i386 --version
QEMU emulator version 6.2.0 (Debian 1:6.2+dfsg-2ubuntu6.24)
Copyright (c) 2003-2021 Fabrice Bellard and the QEMU Project developers
david@david-VirtualBox:~$
```

2.2 实验任务二

2.2.1 任务要求

下载并编译 Linux 内核

2.2.2 实验步骤与结果

下载 Linux-5.10.19 内核并编译成 i386 32 位版本:

```
make i386_defconfig
make menuconfig
make -j8
```

如图, 内核已经编译完成:

```
david@david-VirtualBox:~/lab1/linux-5.10.19$ make -j8
CALL    scripts/atomic/check-atomics.sh
CALL    scripts/checksyscalls.sh
CHK     include/generated/compile.h
Kernel: arch/x86/boot/bzImage is ready (#2)
```

Linux 压缩镜像 linux-5.10.19/arch/x86/boot/bzImage 和符号表 linux-5.10.19/vmlinux 均已经生成:

```
david@david-VirtualBox:~/lab1/linux-5.10.19$ ls -lh arch/x86/boot/bzImage
-rw-rw-r-- 1 david david 7.9M  2月 26 20:15 arch/x86/boot/bzImage
david@david-VirtualBox:~/lab1/linux-5.10.19$ ls -lh vmlinux
-rwxrwxr-x 1 david david 25M  2月 26 20:15 vmlinux
```

2.3 实验任务三

2.3.1 任务要求

用 qemu 启动内核并用 gdb 进行远程调试

2.3.2 实验步骤与结果

在一个 terminal 上使用 qemu 启动内核:

```
qemu-system-i386 -kernel linux-5.10.19/arch/x86/boot/bzImage -s -S -append "console=ttyS0" -
↪ nographic
```

在另一个 terminal 上使用 gdb 调试, 其中调试的步骤为加载符号表、远程连接启动的 qemu、为函数设置断点、开始运行:

```
gdb
file linux-5.10.19/vmlinux
target remote:1234
break start_kernel
c
```

如图, 调试成功:

The image shows two terminal windows side-by-side. The left window is a QEMU terminal showing the boot process of the Linux kernel. It displays various messages such as timer setup, hardware name, and a kernel panic. The right window is a GDB terminal showing the remote debugging session. It includes the GDB license, configuration, and the successful connection to the remote target. The GDB session shows the user setting a breakpoint at the start of the kernel and then continuing the execution.

```
[ 0.113536] ..TIMER: vector=0x30 apic1=0 pin1=2 apic2=-1 pin2=-1
[ 0.131071] ..MP-BIOS bug: 8254 timer not connected to IO-APIC
[ 0.131183] ...trying to set up timer (IRQ0) through the 8259A ...
[ 0.131256] ..... (found apic 0 pin 2) ...
[ 0.145152] ..... failed.
[ 0.145218] ...trying to set up timer as Virtual Wire IRQ...
[ 0.159048] ..... failed.
[ 0.159100] ...trying to set up timer as ExtINT IRQ...
[ 1.573112] ..... failed :(
[ 1.573582] Kernel panic - not syncing: IO-APIC + timer doesn't work! Boot .
[ 1.573928] CPU: 0 PID: 0 Comm: swapper/0 Not tainted 5.10.19 #2
[ 1.574011] Hardware name: QEMU Standard PC (i440FX + PIIX, 1996), BIOS 1.154
[ 1.574180] Call Trace:
[ 1.574643] dump_stack+0x54/0x68
[ 1.574724] panic+0x9e/0x247
[ 1.574780] setup_IO_APIC+0x70c/0x736
[ 1.574830] ? clear_IO_APIC+0x3c/0x60
[ 1.575132] apic_intr_mode_init+0xf4/0xf8
[ 1.575234] x86_late_time_init+0x18/0x29
[ 1.575285] start_kernel+0x3a7/0x45b
[ 1.575329] i386_start_kernel+0x43/0x45
[ 1.575462] startup_32_smp+0x164/0x168
[ 1.576273] ---[ end Kernel panic - not syncing: IO-APIC + timer doesn't wor-
```

License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
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:
<https://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-5.10.19/vmlinux
Reading symbols from linux-5.10.19/vmlinux...
(No debugging symbols found in linux-5.10.19/vmlinux)
(gdb) target remote:1234
Remote debugging using :1234
0x00000000 in ?? ()
(gdb) break start_kernel
Breakpoint 1 at 0xc1fd87d4
(gdb) c
Continuing.

在gdb下, 连接已经启动的qemu进行调试。

2.4 实验任务四

2.4.1 任务要求

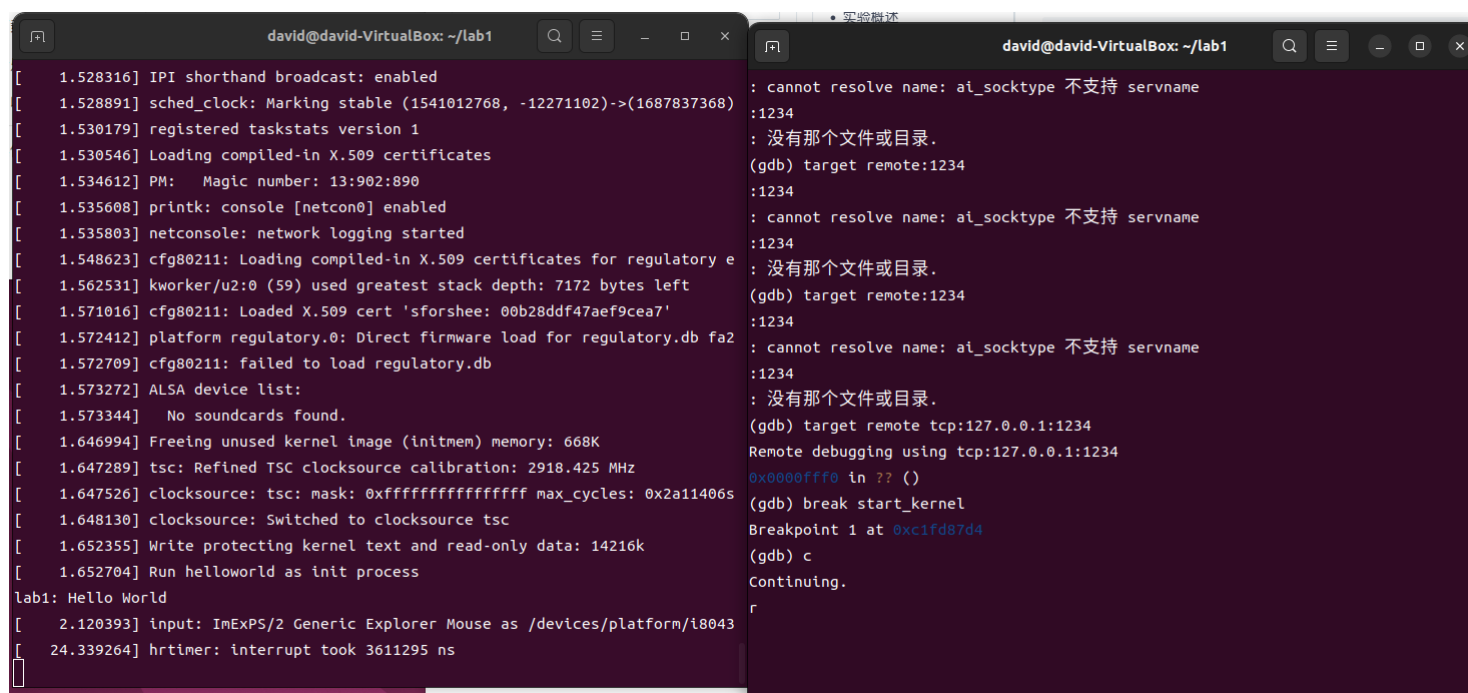
制作一个 Hello World initramfs

2.4.2 实验步骤与结果

编写 helloworld.c 程序, 编译成可执行文件, 并打包 initramfs, 最后启动内核加载 initramfs:

```
gcc -o helloworld -m32 -static helloworld.c
echo helloworld | cpio -o --format=newc > hwinitramfs
qemu-system-i386 -kernel linux-5.10.19/arch/x86/boot/bzImage -initrd hwinitramfs -s -S -append
↳ "console=ttyS0 rdinit=helloworld" -nographic
```

在另一个终端里调试, 最终看到输出 lab1:Hello World:



```
david@david-VirtualBox: ~/lab1
[ 1.528316] IPI shorthand broadcast: enabled
[ 1.528891] sched_clock: Marking stable (1541012768, -12271102)->(1687837368)
[ 1.530179] registered taskstats version 1
[ 1.530546] Loading compiled-in X.509 certificates
[ 1.534612] PM: Magic number: 13:902:890
[ 1.535608] printk: console [netcon0] enabled
[ 1.535803] netconsole: network logging started
[ 1.548623] cfg80211: Loading compiled-in X.509 certificates for regulatory e
[ 1.562531] kworker/u2:0 (59) used greatest stack depth: 7172 bytes left
[ 1.571016] cfg80211: Loaded X.509 cert 'sforshee: 00b28ddf47aef9cea7'
[ 1.572412] platform regulatory.0: Direct firmware load for regulatory.db fa2
[ 1.572709] cfg80211: failed to load regulatory.db
[ 1.573272] ALSA device list:
[ 1.573344] No soundcards found.
[ 1.646994] Freeing unused kernel image (initmem) memory: 668K
[ 1.647289] tsc: Refined TSC clocksource calibration: 2918.425 MHz
[ 1.647526] clocksource: tsc: mask: 0xffffffffffffffff max_cycles: 0x2a114065
[ 1.648130] clocksource: Switched to clocksource tsc
[ 1.652355] Write protecting kernel text and read-only data: 14216k
[ 1.652704] Run helloworld as init process
lab1: Hello World
[ 2.120393] input: ImExPS/2 Generic Explorer Mouse as /devices/platform/i8043
[ 24.339264] hrtimer: interrupt took 3611295 ns

david@david-VirtualBox: ~/lab1
: cannot resolve name: ai_socktype 不支持 servname
:1234
: 没有那个文件或目录.
(gdb) target remote:1234
:1234
: cannot resolve name: ai_socktype 不支持 servname
:1234
: 没有那个文件或目录.
(gdb) target remote:1234
:1234
: cannot resolve name: ai_socktype 不支持 servname
:1234
: 没有那个文件或目录.
(gdb) target remote tcp:127.0.0.1:1234
Remote debugging using tcp:127.0.0.1:1234
0x0000ffff in ?? ()
(gdb) break start_kernel
Breakpoint 1 at 0xc1fd87d4
(gdb) c
Continuing.
r
```

2.5 实验任务五

2.5.1 任务要求

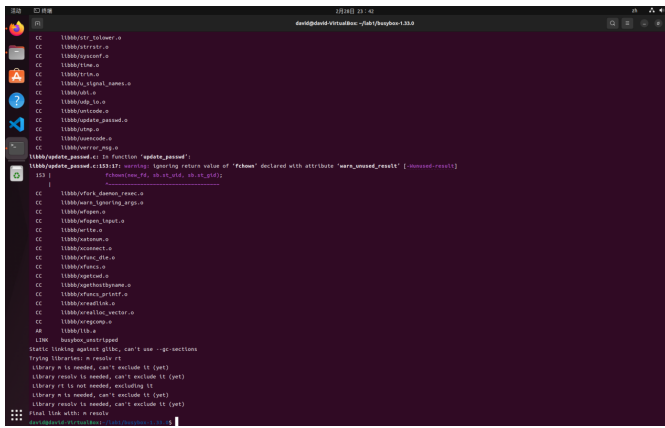
编译并启动 busybox, 其中分为编译 busybox, 制作 initramfs, 加载 busybox

2.5.2 实验步骤与结果

编译 busybox:

```
make defconfig
make menuconfig
make -j8
make install
```

如图, 编译完成:



制作 Initramfs:

```
cd ~/lab1
mkdir mybusybox
mkdir -pv mybusybox/{bin,sbin,etc,proc,sys,usr/{bin,sbin}}
cp -av busybox-1.33.0/_install/* mybusybox/
cd mybusybox
```

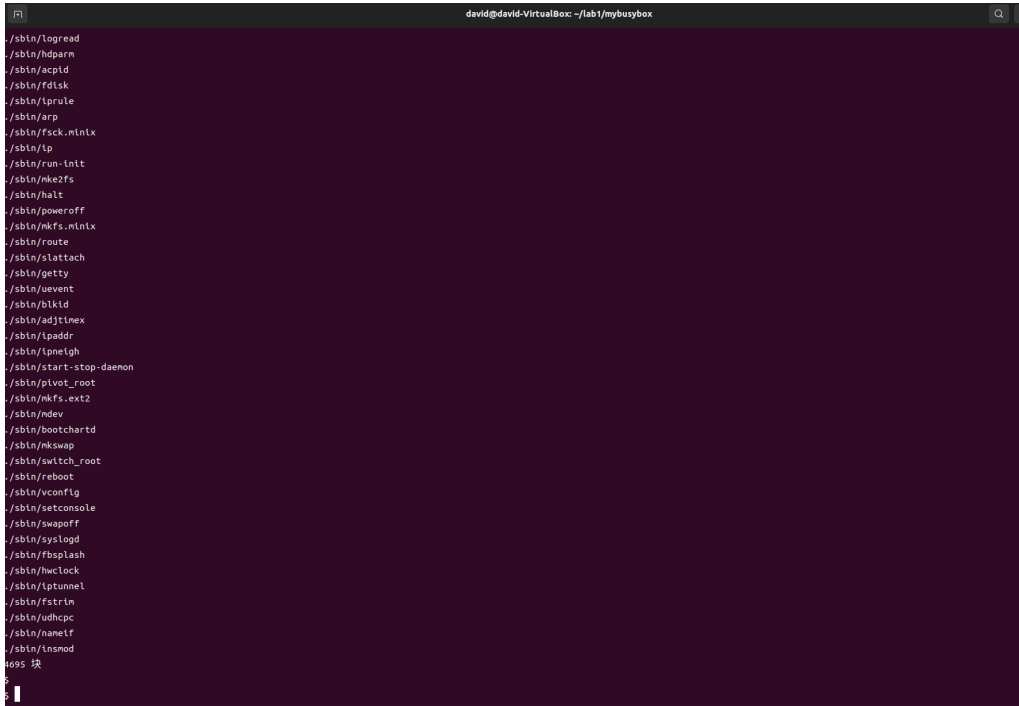
接着写一个 init 程序:

```
mount -t proc none /proc
mount -t sysfs none /sys
echo -e "\nBoot took $(cut -d' ' -f1 /proc/uptime) seconds\n"
exec /bin/sh
```

最后加上执行权限并将 x86-busybox 下面的内容打包归档成 cpio 文件

```
chmod u+x init
find . -print0 | cpio --null -ov --format=newc | gzip -9 > ~/lab1/initramfs-busybox-x86.cpio.gz
```

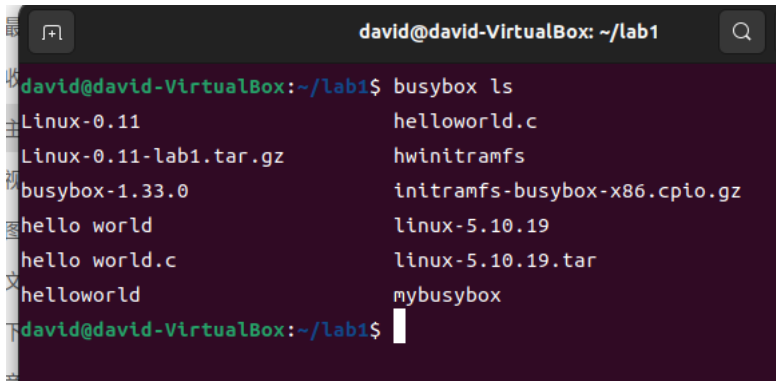
结果如图:



最后加载 busybox:

```
cd ~/lab1
qemu-system-i386 -kernel linux-5.10.19/arch/x86/boot/bzImage -initrd initramfs-busybox-x86.cpio
    ↪ .gz -nographic -append "console=ttyS0" -m size=2048
```

如图, 成功使用 busybox 相关指令, 说明加载成功



```
david@david-VirtualBox: ~/lab1$ busybox ls
linux-0.11                helloworld.c
linux-0.11-lab1.tar.gz    hwinitramfs
busybox-1.33.0            initramfs-busybox-x86.cpio.gz
hello world               linux-5.10.19
hello world.c             linux-5.10.19.tar
helloworld                mybusybox
david@david-VirtualBox: ~/lab1$
```

2.6 实验任务六

2.6.1 任务要求

完成 Linux 0.11 内核的编译, 启动, 调试

2.6.2 实验步骤与结果

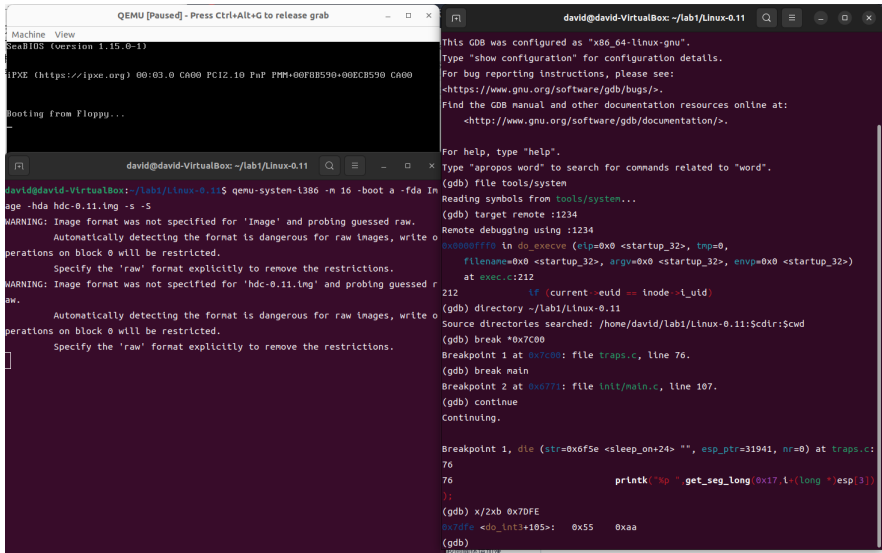
下载, Linux 0.11 内核代码, 并编译 32 位版本的 Linux 0.11 内核, 最后使用 qemu-system-i386 加载、启动内核:

```
make
qemu-system-i386 -m 16 -boot a -fda Image -hda hdc-0.11.img -s -S
```

接着利用 gdb 进行远程调试, 步骤为: 加载符号表、连接已经启动的 qemu、为 0x7C00 和内核入口设置断点, 最后观察 0x7DFE 和 0x7DFF 地址的内容

```
gdb
(gdb) file tools/system
(gdb) target remote :1234
(gdb) directory /path/to/linux-0.11
(gdb) break *0x7C00
(gdb) break main
(gdb) continue
(gdb) x/4xb 0x7DFE
```

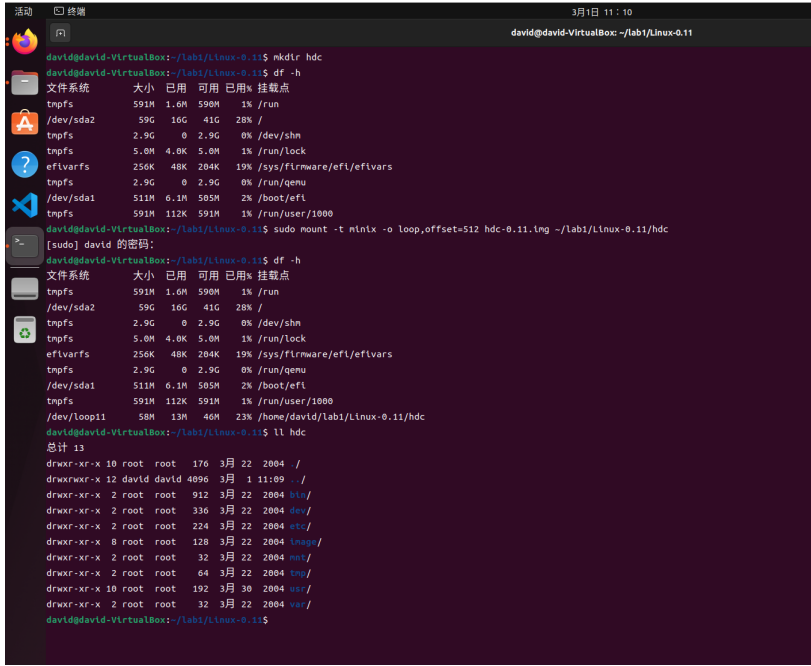
结果显示, 0x7DFE 和 0x7DFF 地址的内容分别为: 0x55 0xaa



接着挂载 hdc, 其中先创建本地挂载目录, 然后挂载 Linux 0.11 硬盘镜像:

```
mkdir hdc
df -h
sudo mount -t minix -o loop,offset=512 hdc-0.11.img ~/lab1/Linux-0.11/hdc
df -h
```

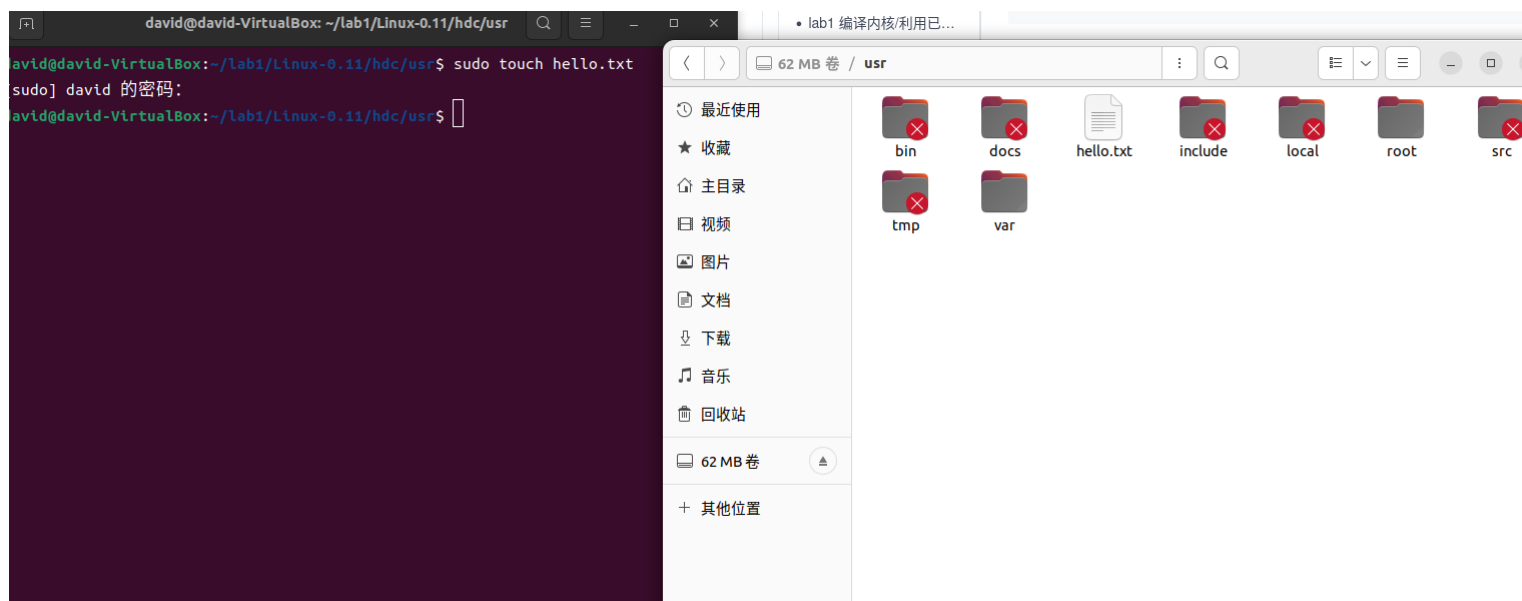
如图显示 hdc 已经挂载成功:



在 hdc 中创建文件:

```
cd hdc/usr
sudo touch hello.txt
sudo vim hello.txt
```

如图, 文件创建成功:



最后卸载文件系统 hdc:

```
sudo umount /dev/loop
df -h
```

已经卸载成功:

```
david@david-VirtualBox:~/lab1/Linux-0.11/hdc/usr$ sudo umount -l /dev/loop11
david@david-VirtualBox:~/lab1/Linux-0.11/hdc/usr$ df -h
```

文件系统	大小	已用	可用	已用%	挂载点
tmpfs	591M	1.6M	590M	1%	/run
/dev/sda2	59G	16G	41G	28%	/
tmpfs	2.9G	0	2.9G	0%	/dev/shm
tmpfs	5.0M	4.0K	5.0M	1%	/run/lock
efivarfs	256K	48K	204K	19%	/sys/firmware/efi/efivars
tmpfs	2.9G	0	2.9G	0%	/run/qemu
/dev/sda1	511M	6.1M	505M	2%	/boot/efi
tmpfs	591M	120K	591M	1%	/run/user/1000

3 实验总结与心得体会

1. 在本次实验中, 对 Linux 操作系统有了初步的了解, 掌握了以下 Bash 的基本用法:

- sudo: 暂时以 root 权限执行命令
- apt: 用于管理软件包的命令, 用于安装、更新、升级、删除软件包
- ls: 列出目录内容
- mkdir: 创建新目录
- cp: 复制文件或目录
- chmod: 更改文件或目录的权限
- tar: 用于打包和解压文件的工具
- cd: 定位
- make: 用于编译和管理大型项目

2. 在内核的调试中, 使用了 gdb 和 qemu 相结合的方法, 其中的原理如下:

QEMU 可以模拟不同架构的计算机系统, 提供高效的虚拟化支持。

在调试的过程中,QEMU 作为 GDB 服务器启动, 并监听 GDB 连接请求, 接着 GDB 连接到 QEMU 并附加到运行中的程序或内核, 最后 GDB 通过 QEMU 操作目标程序。

3. 本次实验熟悉了 Linux 内核的编译、启动、调试过程, 不仅加深了对操作系统原理的理解, 更培养了在复杂系统环境中分析问题、解决问题的工程能力, 为后续深入操作系统领域研究奠定了坚实基础。

4 对实验的改进建议和意见

实验任务六中的文档描述过于精简, 对于初学者来说在探索的过程中遇到了许多困难。如: 挂载 linux 0.11 硬盘镜像, 在 hdc 中创建文件这一步骤并未说明原理, 导致理解起来有一定困难, 影响实验的完成。