

# 操作系统实验 1

## 实验一：编译运行 Linux 内核并通过 qemu+gdb 调试实验报告 范睿 PB17111623

### 一、主要步骤和及核心代码

#### 1. 下载并编译 Linux 内核

```
wget https://mirrors.edge.kernel.org/pub/linux/kernel/v2.6/linux-2.6.26.tar.gz  
#将 Linux 内核下载到 home 文件夹中  
tar -zxvf linux-2.6.26.tar.gz  
#将内核解压至~/linux-2.6.26 文件夹中
```

编译:

```
cd ~/linux-2.6.26  
#进入linux-2.6.26文件夹  
make i386_defconfig  
make  
#执行编译指令
```

准备模拟器 qemu

```
sudo apt-get install qemu  
#下载 qemu 的包
```

#### 2. 使用 busybox 生成根文件系统 下载 busybox

```
wget https://busybox.net/downloads/busybox-1.30.1.tar.bz2  
#下载 busybox  
tar -jxvf busybox-1.30.1.tar.bz2  
#将 busybox 解压到~/busybox-1.30.1 文件夹中  
cd ~/busybox-1.30.1  
#进入 busybox-1.30.1 文件夹
```

编译 busybox

```
make defconfig  
#按默认配置编译  
make menuconfig  
#修改 busybox 配置  
make  
make install  
#编译安装
```

## 准备根文件系统

```
cd ~/busybox-1.30.1/_install
#进入 busybox 下的_install 文件夹
sudo mkdir dev
#创建 dev 文件夹
sudo mknod dev/console c 5 1
#在 dev 下创建一个面向字符的设备 console, 主设备 5 个, 次设备 1 个
sudo mknod dev/ram b 1 0
#在 dev 下创建一个面向块的设备 ram, 主设备 1 个, 次设备 0 个
touch init
#创建一个叫 init 的文件
```

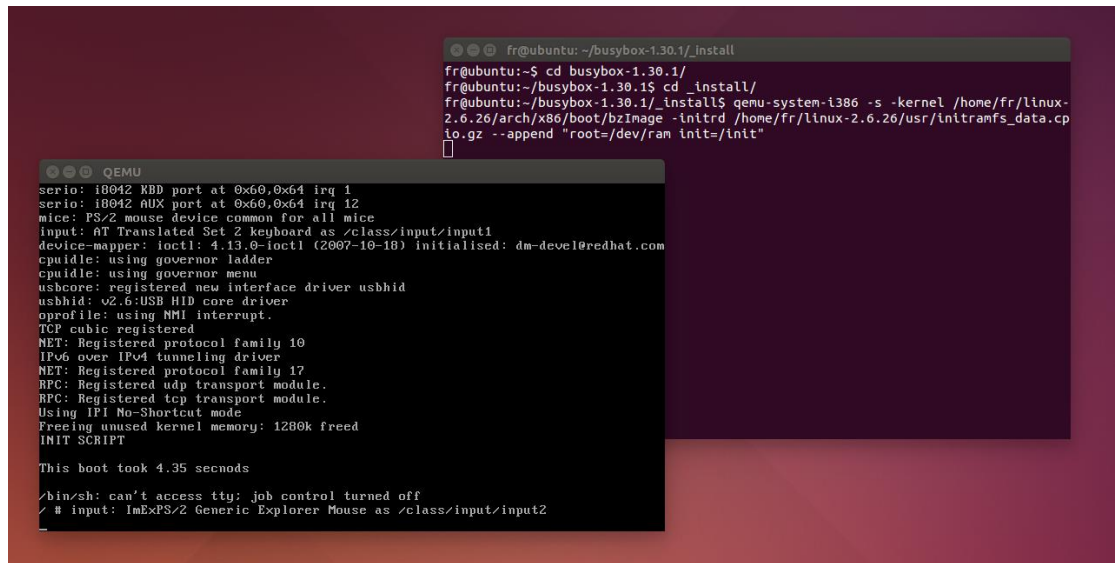
在 init 文件中写入内容:

```
#!/bin/sh
echo "INIT SCRIPT"
mkdir /proc
mkdir /sys
mount -t proc none /proc
mount -t sysfs none /sys
mkdir /tmp
mount -t tmpfs none /tmp
echo -e "\nThis boot took $(cut -d' ' -f1 /proc/uptime) seconds\n"
exec /bin/sh
```

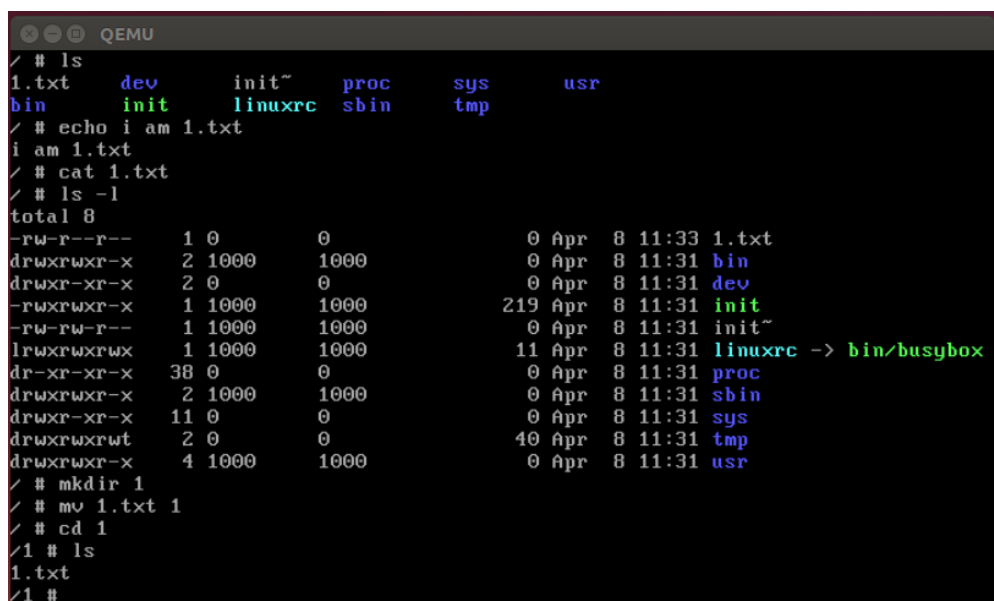
```
chmod +x init
#修改 init 的权限
cd ~/oslab/busybox-1.30.1/_install
#进入_install 文件夹
find . -print0 | cpio --null -ov --format=newc | gzip -9 >
~/oslab/initramfs-busyboxx86.cpio.gz
```

运行 qemu

```
qemu-system-i386 -s -kernel ~/linux-2.6.26/arch/x86/boot/bzImage -initrd
/home/fr/linux-2.6.26/us/initramfs_data.cpio.gz --append "root=/dev/ram init=/init"
```



### 3. 熟悉 linux 命令:



### 4. gdb+qemu 调试内核

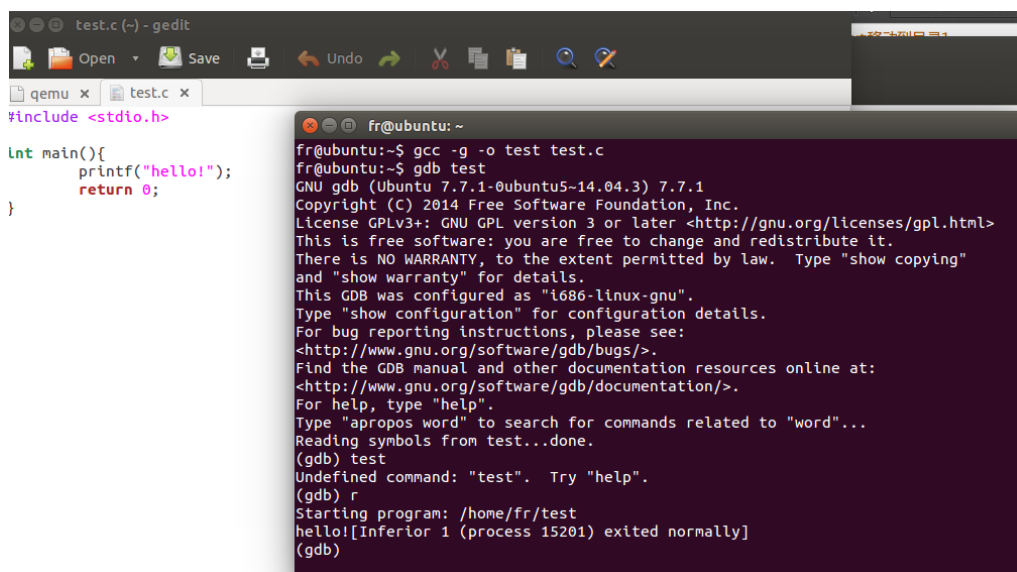
```

sudo apt install gdb
#安装 gdb

```

创建 test.c 文件，写入内容如下图所示：  
效果如下

```
gcc -g -o test test.c
#编译 test.c
gdb test
#用 gdb 调试 test
r
#运行 test
```



##### 5. qemu 中启动 gdb server

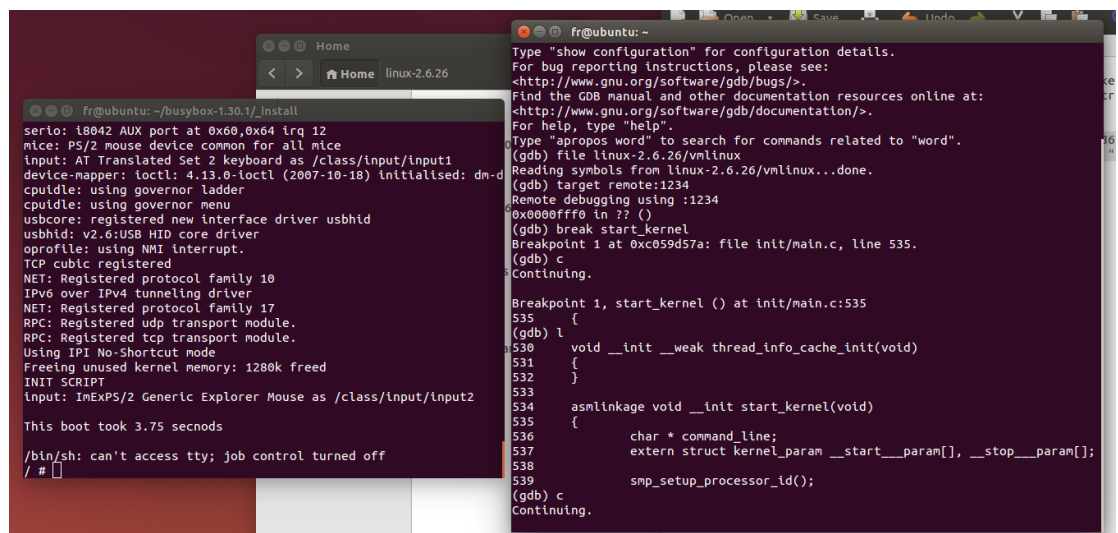
```
qemu-system-i386 -s -S -kernel /home/fr/linux-2.6.26/arch/x86/boot/bzImage -initrd
/home/fr/linux-2.6.26/usr/initramfs-busybox-x86.cpio.gz -nographic --append
"root=/dev/ram init=/init console=ttyS0"
#启动 qemu 运行内核，等待 gdb 连接
```

##### 6. 建立 gdb 和 qemu 之间的连接并设置断点 建立连接

```
gdb
#启动 gdb
file ~/linux-2.6.26/vmlinux
#进入 vmlinux 文件夹
target remote:1234
#建立 gdb 和 qemu 之间的连接
```

设置断点

```
break start_kernel  
#设置断点  
c  
#继续运行
```



```
fr@ubuntu: ~/busybox-1.30.1/_install  
serio: i8042 AUX port at 0x60,0x64 irq 12  
mice: PS/2 mouse device common for all mice  
input: AT Translated Set 2 keyboard as /class/input/input1  
device-mapper: ioctl: 4.13.0-ioctl (2007-10-18) initialised: dm-d  
cpuidle: using governor ladder  
cpuidle: using governor menu  
usbcore: registered new interface driver usbhid  
usbhid: v2.6:USB HID core driver  
oprofile: using NMI interrupt.  
TCP cubic registered  
NET: Registered protocol family 10  
IPv6 over IPv4 tunneling driver  
NET: Registered protocol family 17  
RPC: Registered udp transport module.  
RPC: Registered tcp transport module.  
Using IPI No-Shortcut mode  
Freeing unused kernel memory: 1280k freed  
INIT SCRIPT  
input: ImExPS/2 Generic Explorer Mouse as /class/input/input2  
  
This boot took 3.75 seconds  
  
/bin/sh: can't access tty; job control turned off  
/#  
  
fr@ubuntu: ~  
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-2.6.26/vmlinux  
Reading symbols from linux-2.6.26/vmlinux...done.  
(gdb) target remote:1234  
Remote debugging using :1234  
0x00000000 in ?? ()  
(gdb) break start_kernel  
Breakpoint 1 at 0xc059d57a: file init/main.c, line 535.  
(gdb) c  
Continuing.  
  
Breakpoint 1, start_kernel () at init/main.c:535  
535 {  
(gdb) l  
530 void __init __weak thread_info_cache_init(void)  
531 {  
532 }  
533  
534 asmlinkage void __init start_kernel(void)  
535 {  
536     char * command_line;  
537     extern struct kernel_param __start__param[], __stop__param[];  
538  
539     smp_setup_processor_id();  
(gdb) c  
Continuing.
```

## 7. 携带调试信息再编译

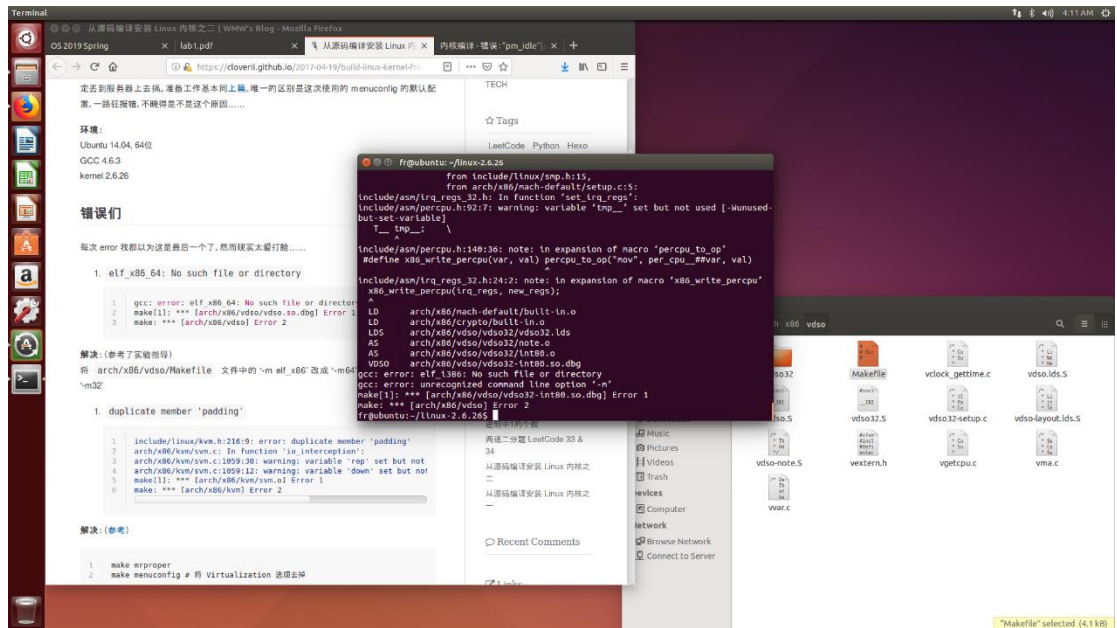
```
cd ~/oslab/linux-2.6.26/  
make menuconfig  
#修改配置信息  
make  
#编译
```

## 二、结果分析

由以上截图可看出，运行结果均正确。

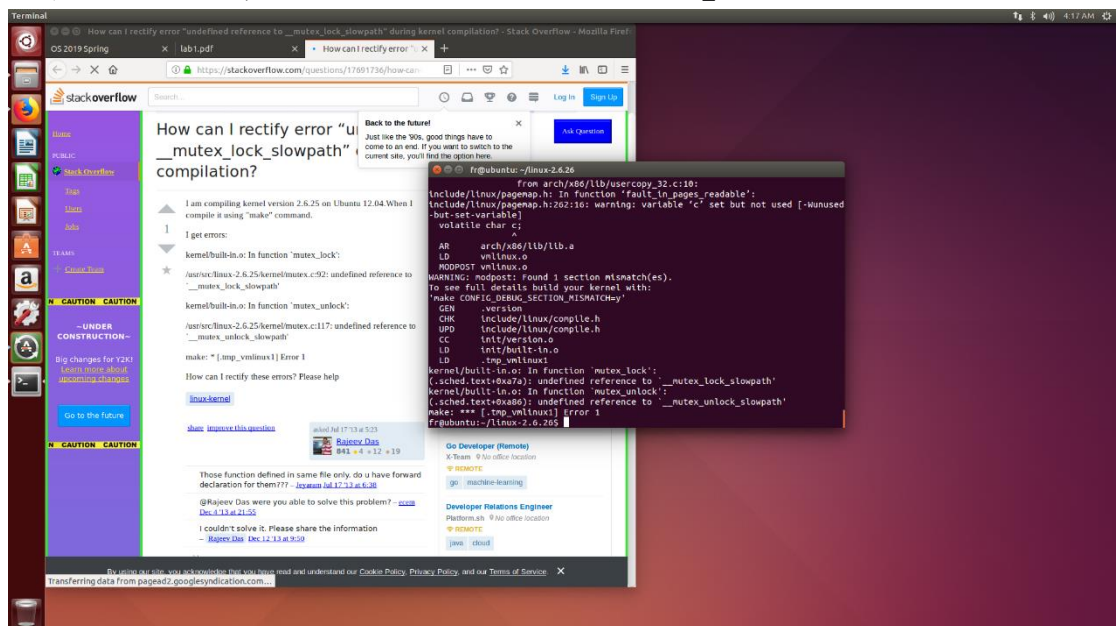
## 三、技术问题

1. 编译 linux 内核，执行 make 时，遇到 elf\_i386:No such file or directory 的错误



解决方法：将 `linux-2.6.26/arch/x86/vdso` 目录下的 `Makefile` 文件中的 ‘-m elf\_x86\_64’ 改成 ‘-m64’， ‘-m elf\_i386’ 改成 ‘-m32’

2. 还是执行 make 时，遇到 undefined reference to ‘mutex\_lock’



解决方法：在 `mutex.c` 中第 61 和 98 行，static 后分别增加 `__used`，保存，重新编译成功

#### 四、实验总结

在本次实验中，我学会了下载并编译 linux 内核，修改其配置，下载并编译 qemu，在 qemu 中进行文件操作，熟悉了 linux 常用指令和常用指令的使用方法，下载 gdb 并在 gdb 中调试自己写的 c 代码，还有 gdb 和 qemu 建立连接，加载符号表，设置断点等操作。