

设计与实现自定义Linux 系统调用及内核编译

操作系统

设计与实现自定义Linux 系统调用及内核编译

准备材料

基本命令

1. 打开文件夹
2. 用vi/vim/gedit打开文件
3. 进入root权限
4. 使用管理权限做某事（用gedit打开文件）
5. 拷贝文件
6. 用gcc编译c文件
7. 执行编译好的目标文件
8. 为了使make命令执行并行处理

实验步骤

1. 解压内核源码
2. 修改内核源码
3. 添加函数声明
4. 添加系统调用
5. 编译
 - 步骤一 进入/usr/src/linux-3.10.4/目录下
 - 步骤二 调用make menuconfig命令配置内核中需要编译的代码功能
 - 步骤三 调用sudo make bzImage编译内核
 - 步骤四 调用make modules编译模组
 - 步骤五 调用make modules_install安装模组
 - 步骤六 调用 cp arch/x86/boot/bzImage /boot/vmlinuz-3.10.4 进行拷贝
 - 步骤七 调用cp .config /boot/config-3.10.4
6. 创建开机镜像
7. 修改开机启动项
8. 重新启动并查看版本号

标题

9. 利用代码对自定义系统调用进行测试

实验报告要求

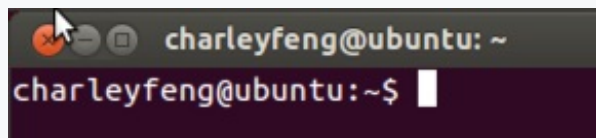
友情提示

准备材料

1. 虚拟机
2. Ubuntu 版本：12.04 32位 **教程实践环境为12.04 32位Ubuntu**
3. Linux kernel 版本：3.10.4
4. [32位Ubuntu和实验内核下载地址](#)

基本命令

本次实验基本所有命令都需要使用sudo，不允许进入root，因为需要看你的主机名。



1. 打开文件夹

```
cd 路径
```

2. 用vi/vim/gedit打开文件

```
vi/vim/gedit 文件名
```

3. 进入root权限

```
su
```

4. 使用管理权限做某事（用gedit打开文件）

```
sudo gedit sys.h
```

5. 拷贝文件

```
cp 被拷贝文件 目标地址
```

6. 用gcc编译c文件

```
$ gcc 文件名 -o 文件名（不带有.c）
```

7. 执行编译好的目标文件

```
./文件名
```

8. 为了使make命令执行并行处理

```
-j 选项可以用来指定作业数，如果你的电脑是多核 cpu，j后面的数字最大可以是 核数*2  
make -j4
```

实验步骤

1. 解压内核源码

1.使用cd命令进入内核所在的文件夹

```
charleyfeng@ubuntu: ~/Desktop
charleyfeng@ubuntu:~$ ls
Desktop  Downloads  Music  Public  Videos
Documents  examples.desktop  Pictures  Templates
charleyfeng@ubuntu:~$ cd Desktop/
charleyfeng@ubuntu:~/Desktop$ ls
linux-3.10.4.tar.xz  Parallels Shared Folders
charleyfeng@ubuntu:~/Desktop$
```

2.将文件移动至/usr/src目录下

```
charleyfeng@ubuntu:~/Desktop$ mv linux-3.10.4.tar.xz /usr/src/
mv: cannot move 'linux-3.10.4.tar.xz' to '/usr/src/linux-3.10.4.tar.xz': Permission denied
charleyfeng@ubuntu:~/Desktop$ sudo mv linux-3.10.4.tar.xz /usr/src/
[sudo] password for charleyfeng:
charleyfeng@ubuntu:~/Desktop$
```

3.进入/usr/src目录

```
charleyfeng@ubuntu:~/Desktop$ cd /usr/src/
charleyfeng@ubuntu:/usr/src$ ls
linux-3.10.4.tar.xz  linux-headers-3.11.0-15-generic
linux-headers-3.11.0-15  parallels-tools-12.0.2.41353
charleyfeng@ubuntu:/usr/src$
```

4.解压文件

4.1 用命令xz -d 文件名.tar.xz解压成tar。

```
charleyfeng@ubuntu:/usr/src$ xz -d linux-3.10.4.tar.xz
xz: linux-3.10.4.tar: Permission denied
charleyfeng@ubuntu:/usr/src$ sudo xz -d linux-3.10.4.tar.xz
charleyfeng@ubuntu:/usr/src$ ls
linux-3.10.4.tar  linux-headers-3.11.0-15-generic
linux-headers-3.11.0-15  parallels-tools-12.0.2.41353
```

4.2 用命令tar -xvf 文件名.tar解压成功。

如果出现permission denied，请加sudo

```
charleyfeng@ubuntu: /usr/src
linux-3.10.4/usr/Kconfig
linux-3.10.4/usr/Makefile
linux-3.10.4/usr/gen_init_cpio.c
linux-3.10.4/usr/initramfs_data.S
linux-3.10.4/virt/
linux-3.10.4/virt/kvm/
linux-3.10.4/virt/kvm/Kconfig
linux-3.10.4/virt/kvm/assigned-dev.c
linux-3.10.4/virt/kvm/async_pf.c
linux-3.10.4/virt/kvm/async_pf.h
linux-3.10.4/virt/kvm/coalesced_mmio.c
linux-3.10.4/virt/kvm/coalesced_mmio.h
linux-3.10.4/virt/kvm/eventfd.c
linux-3.10.4/virt/kvm/ioapic.c
linux-3.10.4/virt/kvm/ioapic.h
linux-3.10.4/virt/kvm/iodev.h
linux-3.10.4/virt/kvm/iommu.c
linux-3.10.4/virt/kvm/irq_comm.c
linux-3.10.4/virt/kvm/irqchip.c
linux-3.10.4/virt/kvm/kvm_main.c
charleyfeng@ubuntu: /usr/src$ ls
linux-3.10.4      linux-headers-3.11.0-15      parallels-tools-12.0.2.41353
linux-3.10.4.tar  linux-headers-3.11.0-15-generic
charleyfeng@ubuntu: /usr/src$
```

2. 修改内核源码

1. 进入目录/usr/src/linux-3.10.4/kernel

```
charleyfeng@ubuntu: /usr/src$ ls
linux-3.10.4      linux-headers-3.11.0-15      parallels-tools-12.0.2.41353
linux-3.10.4.tar  linux-headers-3.11.0-15-generic
charleyfeng@ubuntu: /usr/src$ cd linux-3.10.4/
charleyfeng@ubuntu: /usr/src/linux-3.10.4$ cd kernel/
charleyfeng@ubuntu: /usr/src/linux-3.10.4/kernel$
```

2. 打开文件sys.c，在sys.c中添加一个函数，该函数如下，使用vim编辑器打开，使用shift+g跳转到文件末尾，使用i插入代码，使用:wq保存并退出。

要求：打印内容必须为你姓名的全拼，例如：fengxiachong

```
1.  #include <linux/linkage.h>
2.  asmlinkage int sys_helloworld(void){
3.      printk(KERN_EMERG "hello fengxiachong!");
4.      return 1;
```

5. }

注意事项：

- 不要把这段函数实现放到某个**预编译命令**之中，例如 `#ifndef #endif` 或者 `#ifdef #endif`。
- 添加代码时，函数没有参数需要加一个 `void`。
- 注意包含**头文件**。
- 前任助教：我比较喜欢用 `gedit` 来编辑，因为图形界面更亲切，虽然 `vim` 更有逼格...在用 `gedit` 打开本次实验所有文件时，需要在前面加上 `sudo`。

```
charleyfeng@ubuntu:/usr/src/linux-3.10.4/kernel$ sudo vim sys.c
[sudo] password for charleyfeng:
charleyfeng@ubuntu:/usr/src/linux-3.10.4/kernel$
```

```
charleyfeng@ubuntu: /usr/src/linux-3.10.4/kernel
#include <linux/posix-timers.h>
#include <linux/security.h>
#include <linux/dcookies.h>
#include <linux/suspend.h>
#include <linux/tty.h>
#include <linux/signal.h>
#include <linux/cn_proc.h>
#include <linux/getcpu.h>
#include <linux/task_io_accounting_ops.h>
#include <linux/seccomp.h>
#include <linux/cpu.h>
#include <linux/personality.h>
#include <linux/ptrace.h>
#include <linux/fs_struct.h>
#include <linux/file.h>
#include <linux/mount.h>
#include <linux/gfp.h>
#include <linux/syscore_ops.h>
#include <linux/version.h>
#include <linux/ctype.h>
#include <linux/linkage.h>

-- INSERT --                                45,26                                0%
```

```
charleyfeng@ubuntu: /usr/src/linux-3.10.4/kernel
__put_user(s.loads[1], &info->loads[1]) ||
__put_user(s.loads[2], &info->loads[2]) ||
__put_user(s.totalram, &info->totalram) ||
__put_user(s.freeram, &info->freeram) ||
__put_user(s.sharedram, &info->sharedram) ||
__put_user(s.bufferram, &info->bufferram) ||
__put_user(s.totalswap, &info->totalswap) ||
__put_user(s.freeswap, &info->freeswap) ||
__put_user(s.procs, &info->procs) ||
__put_user(s.totalhigh, &info->totalhigh) ||
__put_user(s.freehigh, &info->freehigh) ||
__put_user(s.mem_unit, &info->mem_unit))
    return -EFAULT;

    return 0;
}
#endif /* CONFIG_COMPAT */

/*my own system call implement -- by fxc*/
asmlinkage int sys_helloworld(void){
    printk(KERN_EMERG "hello fengxiachong!");
    return 0;
}
```

2496,9-16 Bot

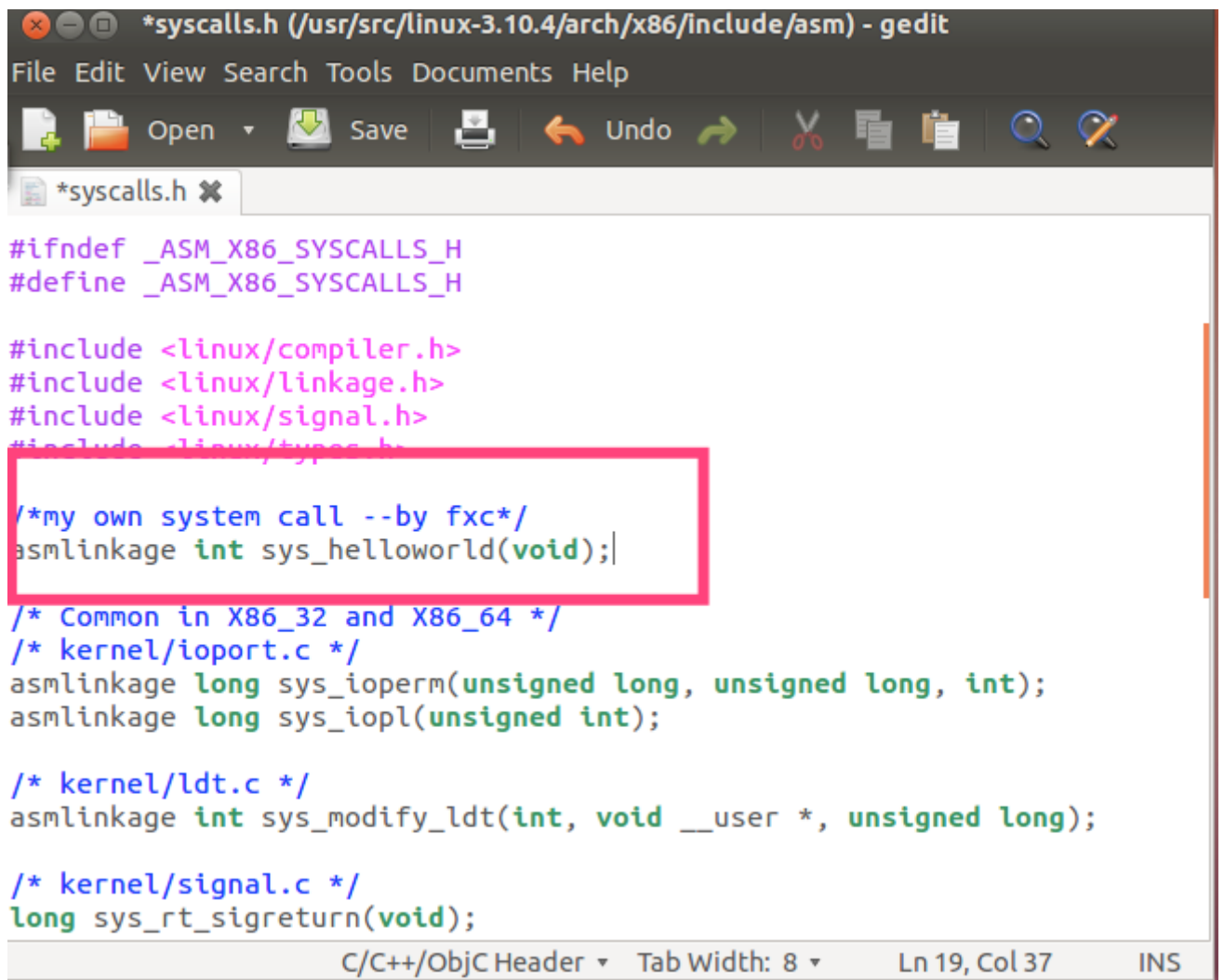
3. 添加函数声明

刚刚只是在sys.c文件里面进行了函数实现，我们还需要在一个名为syscalls.h的头文件里对该函数进行声明。

1. 进入/usr/src/linux-3.10.4/arch/x86/include/asm/目录。（使用cd ..返回上一级目录）

```
charleyfeng@ubuntu: /usr/src/linux-3.10.4/kernel$ cd ..
charleyfeng@ubuntu: /usr/src/linux-3.10.4$ ls
arch      Documentation  init          lib           README        sound
block     drivers       ipc           MAINTAINERS  REPORTING-BUGS tools
COPYING   firmware      Kbuild       Makefile     samples       usr
CREDITS   fs            Kconfig      mm           scripts       virt
crypto    include       kernel       net          security
charleyfeng@ubuntu: /usr/src/linux-3.10.4$ cd arch/
charleyfeng@ubuntu: /usr/src/linux-3.10.4/arch$ cd x86/include/asm/
```

2. 用你喜欢的编辑器开文件syscalls.h进行编辑，加入我们的声明：如图所示



```
*syscalls.h (/usr/src/linux-3.10.4/arch/x86/include/asm) - gedit
File Edit View Search Tools Documents Help
Open Save Undo
*syscalls.h x

#ifndef _ASM_X86_SYSCALLS_H
#define _ASM_X86_SYSCALLS_H

#include <linux/compiler.h>
#include <linux/linkage.h>
#include <linux/signal.h>
#include <linux/types.h>

/*my own system call --by fxc*/
asm linkage int sys_helloworld(void);

/* Common in X86_32 and X86_64 */
/* kernel/ioport.c */
asm linkage long sys_ioperm(unsigned long, unsigned long, int);
asm linkage long sys_iopl(unsigned int);

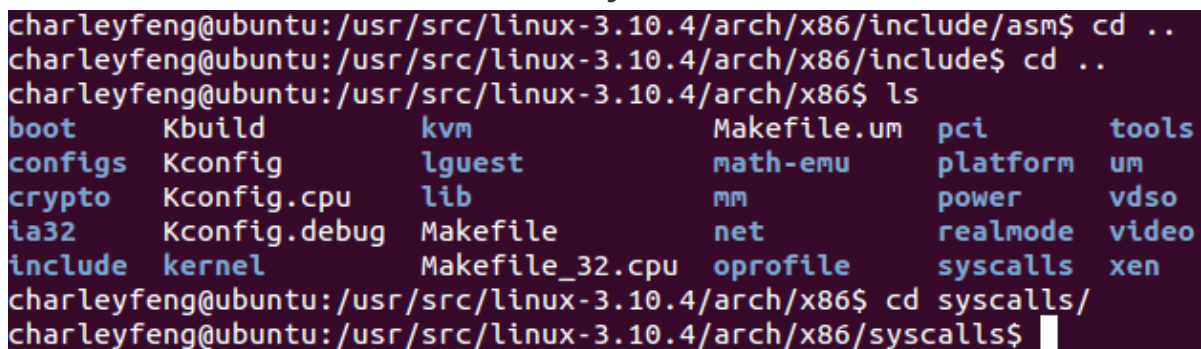
/* kernel/ldt.c */
asm linkage int sys_modify_ldt(int, void __user *, unsigned long);

/* kernel/signal.c */
long sys_rt_sigreturn(void);

C/C++/ObjC Header Tab Width: 8 Ln 19, Col 37 INS
```

4. 添加系统调用

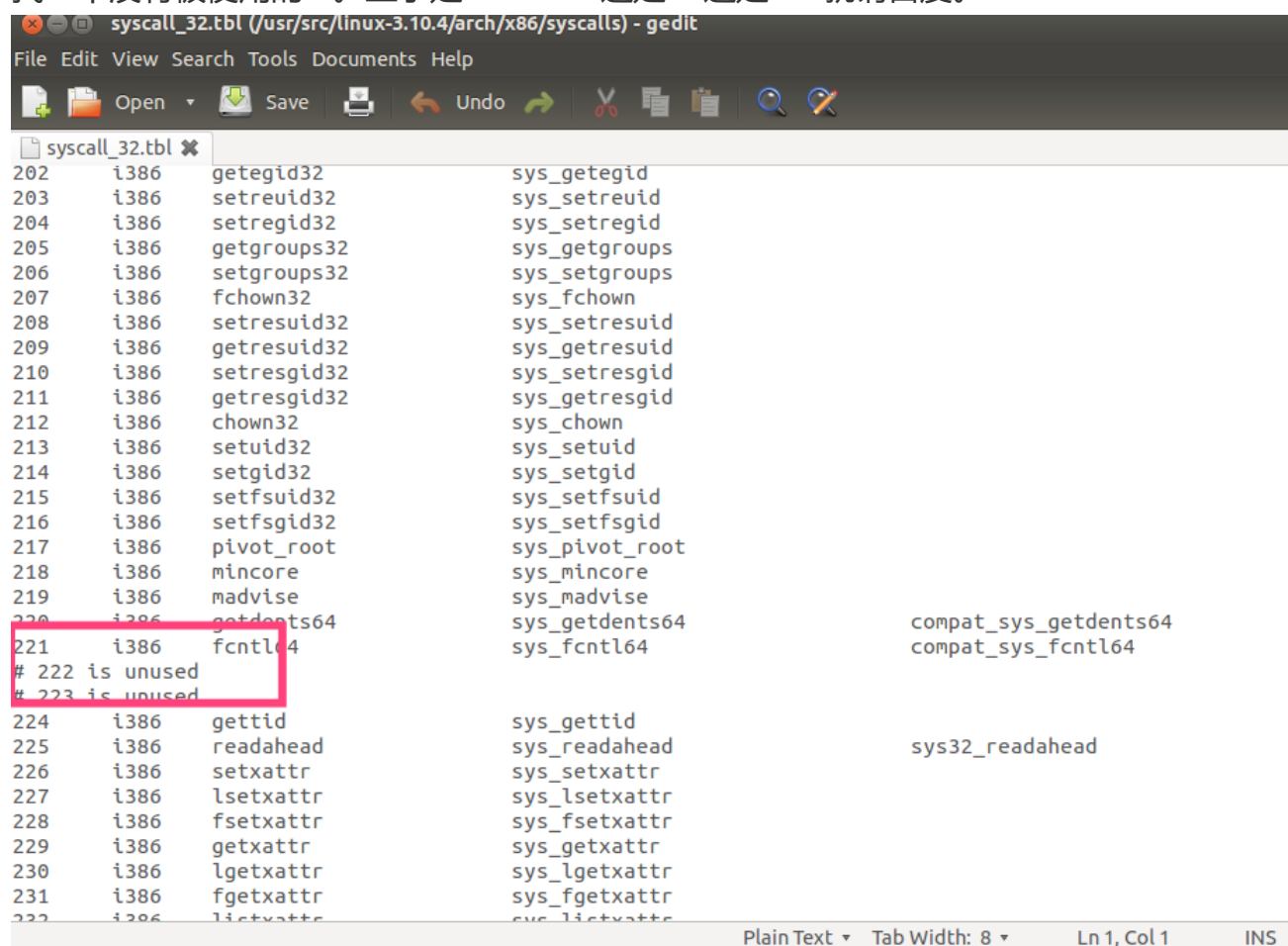
1. 进入 `/usr/src/linux-3.10.4/arch/x86/syscalls` 目录。



```
charleyfeng@ubuntu:/usr/src/linux-3.10.4/arch/x86/include/asm$ cd ..
charleyfeng@ubuntu:/usr/src/linux-3.10.4/arch/x86/include$ cd ..
charleyfeng@ubuntu:/usr/src/linux-3.10.4/arch/x86$ ls
boot      Kbuild      kvm          Makefile.um  pci         tools
configs   Kconfig     lguest       math-emu     platform    um
crypto    Kconfig.cpu lib          mm           power       vdso
ia32      Kconfig.debug Makefile     net          realmode    video
include   kernel      Makefile_32.cpu oprofile     syscalls    xen
charleyfeng@ubuntu:/usr/src/linux-3.10.4/arch/x86$ cd syscalls/
charleyfeng@ubuntu:/usr/src/linux-3.10.4/arch/x86/syscalls$
```

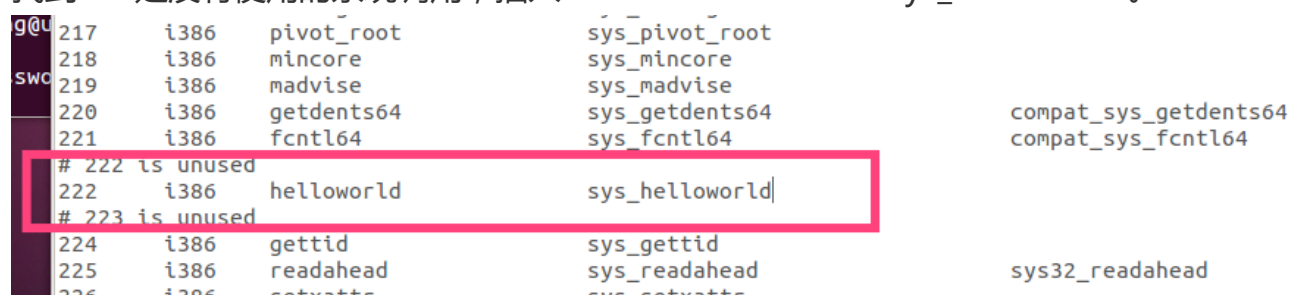
2. 打开文件 `syscall_32.tbl` (该文件有一个系统调用列表, 最前面的属性是id)。这时就有问题了, 有的同学偏偏要用64位的ubuntu来进行这个实验, 显然我们这里打开的文件是适用于32位的。所以, 64位的同学注意啦, 你们打开的是 `syscall_64.tbl`, 而且很显然, 两

个文件的内容不一样，所以下面要加的东西显然也不一样。但是原理应该是一样的，就是找一个没有被使用的id。至于是common还是64还是x32就请百度。



```
syscall_32.tbl (/usr/src/linux-3.10.4/arch/x86/syscalls) - gedit
File Edit View Search Tools Documents Help
Open Save Undo
syscall_32.tbl x
202 i386 getegid32 sys_getegid
203 i386 setreuid32 sys_setreuid
204 i386 setregid32 sys_setregid
205 i386 getgroups32 sys_getgroups
206 i386 setgroups32 sys_setgroups
207 i386 fchown32 sys_fchown
208 i386 setresuid32 sys_setresuid
209 i386 getresuid32 sys_getresuid
210 i386 setresgid32 sys_setresgid
211 i386 getresgid32 sys_getresgid
212 i386 chown32 sys_chown
213 i386 setuid32 sys_setuid
214 i386 setgid32 sys_setgid
215 i386 setfsuid32 sys_setfsuid
216 i386 setfsgid32 sys_setfsgid
217 i386 pivot_root sys_pivot_root
218 i386 mincore sys_mincore
219 i386 madvise sys_madvise
220 i386 getdents64 sys_getdents64 compat_sys_getdents64
221 i386 fcntl64 sys_fcntl64 compat_sys_fcntl64
# 222 is unused
# 223 is unused
224 i386 gettid sys_gettid
225 i386 readahead sys_readahead sys32_readahead
226 i386 setxattr sys_setxattr
227 i386 lsetxattr sys_lsetxattr
228 i386 fsetxattr sys_fsetxattr
229 i386 getxattr sys_getxattr
230 i386 lgetxattr sys_lgetxattr
231 i386 fgetxattr sys_fgetxattr
232 i386 listxattr sys_listxattr
Plain Text Tab Width: 8 Ln 1, Col 1 INS
```

3. 找到222是没有使用的系统调用，插入 222 i386 helloworld sys_helloworld。



```
g@u 217 i386 pivot_root sys_pivot_root
swd 218 i386 mincore sys_mincore
219 i386 madvise sys_madvise
220 i386 getdents64 sys_getdents64 compat_sys_getdents64
221 i386 fcntl64 sys_fcntl64 compat_sys_fcntl64
# 222 is unused
222 i386 helloworld sys_helloworld
# 223 is unused
224 i386 gettid sys_gettid
225 i386 readahead sys_readahead sys32_readahead
226 i386 setxattr sys_setxattr
```

4. 保存。

5. 编译

万一你不是第一次做这一步，你需要注意一个命令.....(我比较谨慎，第一次我也运行了一遍)

运行 `make mrproper`，清理上次编译的结果文件以及配置文件

```
crypto include kernel net security
charleyfeng@ubuntu:/usr/src/linux-3.10.4$ make mrproper
charleyfeng@ubuntu:/usr/src/linux-3.10.4$
```

步骤一 进入/usr/src/linux-3.10.4/目录下

步骤二 调用 `make menuconfig` 命令配置内核中需要编译的代码功能

如果用的和我一样的版本的同学应该此处会出现问题 [参考解决方案](#)

```
charleyfeng@ubuntu:/usr/src/linux-3.10.4$ make menuconfig
HOSTCC scripts/basic/fixdep
scripts/basic/fixdep.c:462:1: fatal error: opening dependency file scripts/basic
/.fixdep.d: Permission denied
compilation terminated.
make[1]: *** [scripts/basic/fixdep] Error 1
make: *** [scripts_basic] Error 2
charleyfeng@ubuntu:/usr/src/linux-3.10.4$
```

解决问题：

1. 运行 `sudo apt-get install ncurses`，结果又出现问题

```
charleyfeng@ubuntu:/usr/src/linux-3.10.4$ sudo apt-get install ncurses
Reading package lists... Done
Building dependency tree
Reading state information... Done
Package ncurses is not available, but is referred to by another package.
This may mean that the package is missing, has been obsoleted, or
is only available from another source

E: Package 'ncurses' has no installation candidate
charleyfeng@ubuntu:/usr/src/linux-3.10.4$
```

2. 运行 `sudo apt-get install libncurses*`，这网速我还能说什么呢.....

```
Get:5 http://us.archive.ubuntu.com/ubuntu/ precise-updates/universe gnat-4.6 4.6.3-1ubuntu4 [11.2 MB]
11% [5 gnat-4.6 3,295 kB/11.2 MB 30%] 214 B/s 2d 7h 27min 1s
```

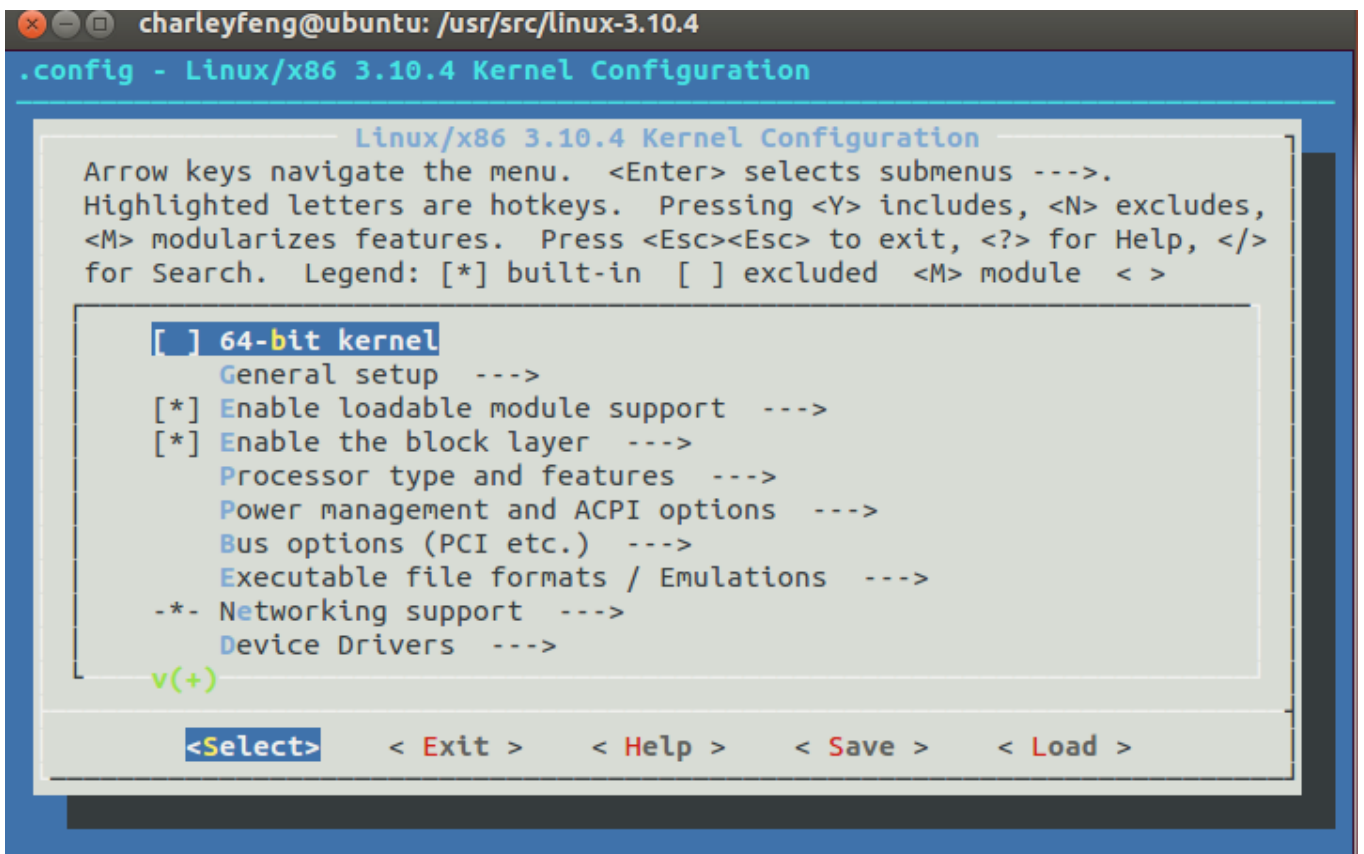
此处还有可能出现的错误

1. 如果在终端执行 `make menuconfig`，显示错误：
2. `Your display is too small to run Menuconfig!`
3. `It must be at least 19 lines by 80 columns.`
4. `make[1]: *** [menuconfig] Error 1`
5. `make: *** [menuconfig] Error 2`

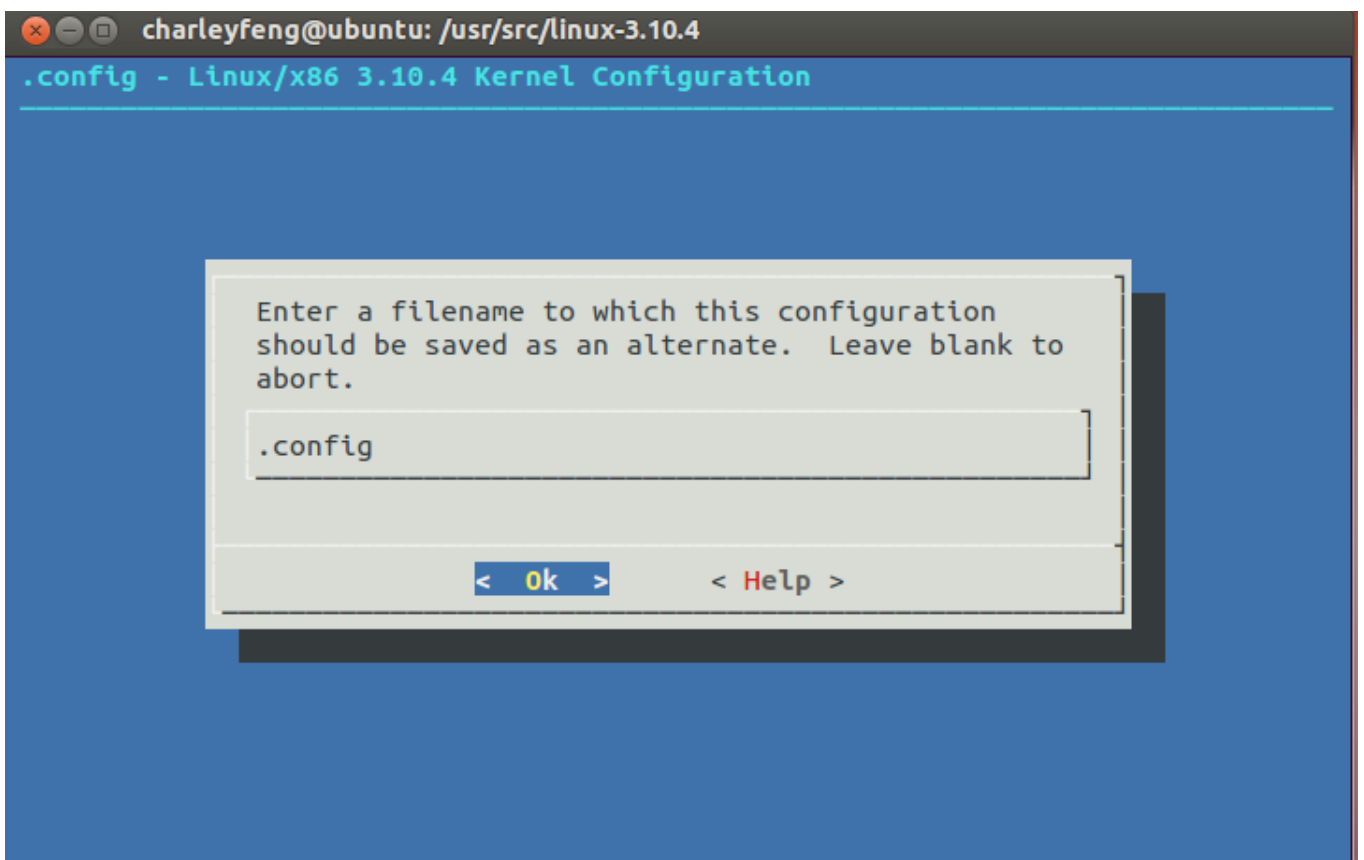
解决方案：把Terminal放大就可以。

3. 执行 `sudo make menuconfig`，如下图所示

选**General setup**，点击save



点击ok、exit



点击exit

```
SHIPPED scripts/kconfig/zconf.hash.c
HOSTCC  scripts/kconfig/zconf.tab.o
HOSTLD  scripts/kconfig/mconf
scripts/kconfig/mconf Kconfig
#
# using defaults found in /boot/config-3.11.0-15-generic
#
/boot/config-3.11.0-15-generic:4853:warning: symbol value 'm' invalid for FB_VESA
*** End of the configuration.
*** Execute 'make' to start the build or try 'make help'.

charleyfeng@ubuntu:/usr/src/linux-3.10.4$
```

步骤三 调用sudo make bzImage编译内核

建议后面再加个参数 j2 或者 j4 好长时间...万一出错了，查看错误信息，回头查看你修改的文件里面是否是按照教程一步一步走的。

是I不是L(这一步不加sudo会有错误)

```
*** End of the configuration.
*** Execute 'make' to start the build or try 'make help'

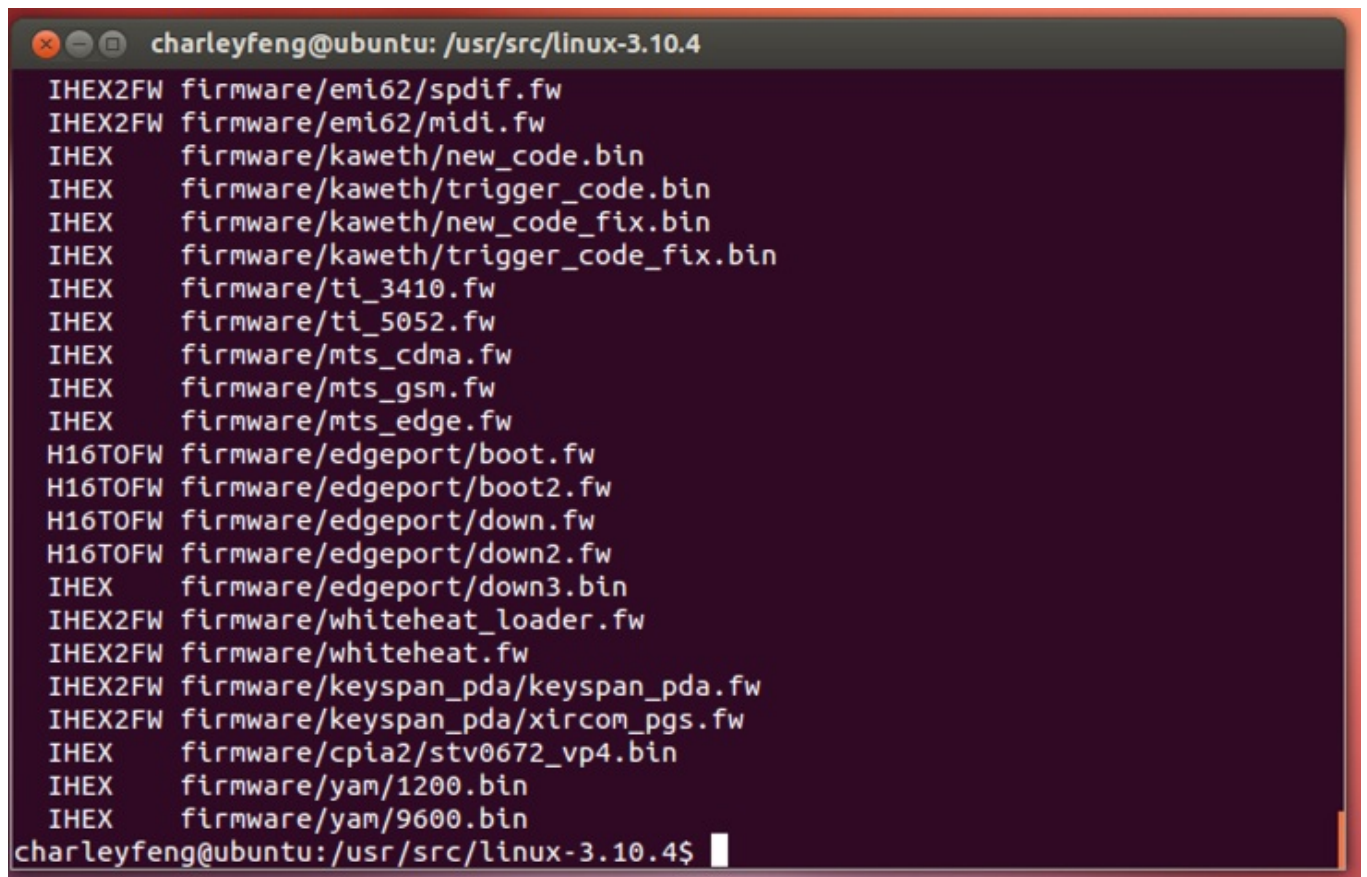
charleyfeng@ubuntu:/usr/src/linux-3.10.4$ make bzImage -j2
```

```
charleyfeng@ubuntu: /usr/src/linux-3.10.4
LD      drivers/xen/xenbus/built-in.o
CC      drivers/xen/xen-acpi-pad.o
CC      drivers/xen/pcpu.o
CC      drivers/xen/biomerge.o
CC      drivers/xen/xen-selfballoon.o
CC      drivers/xen/xen-balloon.o
CC      drivers/xen/sys-hypervisor.o
CC      drivers/xen/platform-pci.o
CC      drivers/xen/swiotlb-xen.o
CC      drivers/xen/xen-acpi-processor.o
LD      drivers/xen/built-in.o
LD      drivers/built-in.o
LINK    vmlinux
LD      vmlinux.o
MODPOST vmlinux.o
WARNING: modpost: Found 1 section mismatch(es).
To see full details build your kernel with:
'make CONFIG_DEBUG_SECTION_MISMATCH=y'
GEN      .version
CHK      include/generated/compile.h
UPD      include/generated/compile.h
CC      init/version.o
LD      init/built-in.o
```

成功了就会出现下图。

```
charleyfeng@ubuntu: /usr/src/linux-3.10.4
CC      arch/x86/boot/video.o
CC      arch/x86/boot/video-mode.o
CC      arch/x86/boot/version.o
CC      arch/x86/boot/apm.o
CC      arch/x86/boot/video-vga.o
CC      arch/x86/boot/video-vesa.o
CC      arch/x86/boot/video-bios.o
HOSTCC  arch/x86/boot/tools/build
CPUSTR  arch/x86/boot/cpustr.h
CC      arch/x86/boot/cpu.o
MKPIGGY arch/x86/boot/compressed/piggy.S
AS      arch/x86/boot/compressed/piggy.o
LD      arch/x86/boot/compressed/vmlinux
ZOFFSET arch/x86/boot/zoffset.h
OBJCOPY arch/x86/boot/vmlinux.bin
AS      arch/x86/boot/header.o
LD      arch/x86/boot/setup.elf
OBJCOPY arch/x86/boot/setup.bin
BUILD  arch/x86/boot/bzImage
Setup is 16876 bytes (padded to 16896 bytes).
System is 5536 kB
CRC 78c2fc87
Kernel: arch/x86/boot/bzImage is ready (#1)
charleyfeng@ubuntu: /usr/src/linux-3.10.4$
```


步骤四 调用make modules编译模组

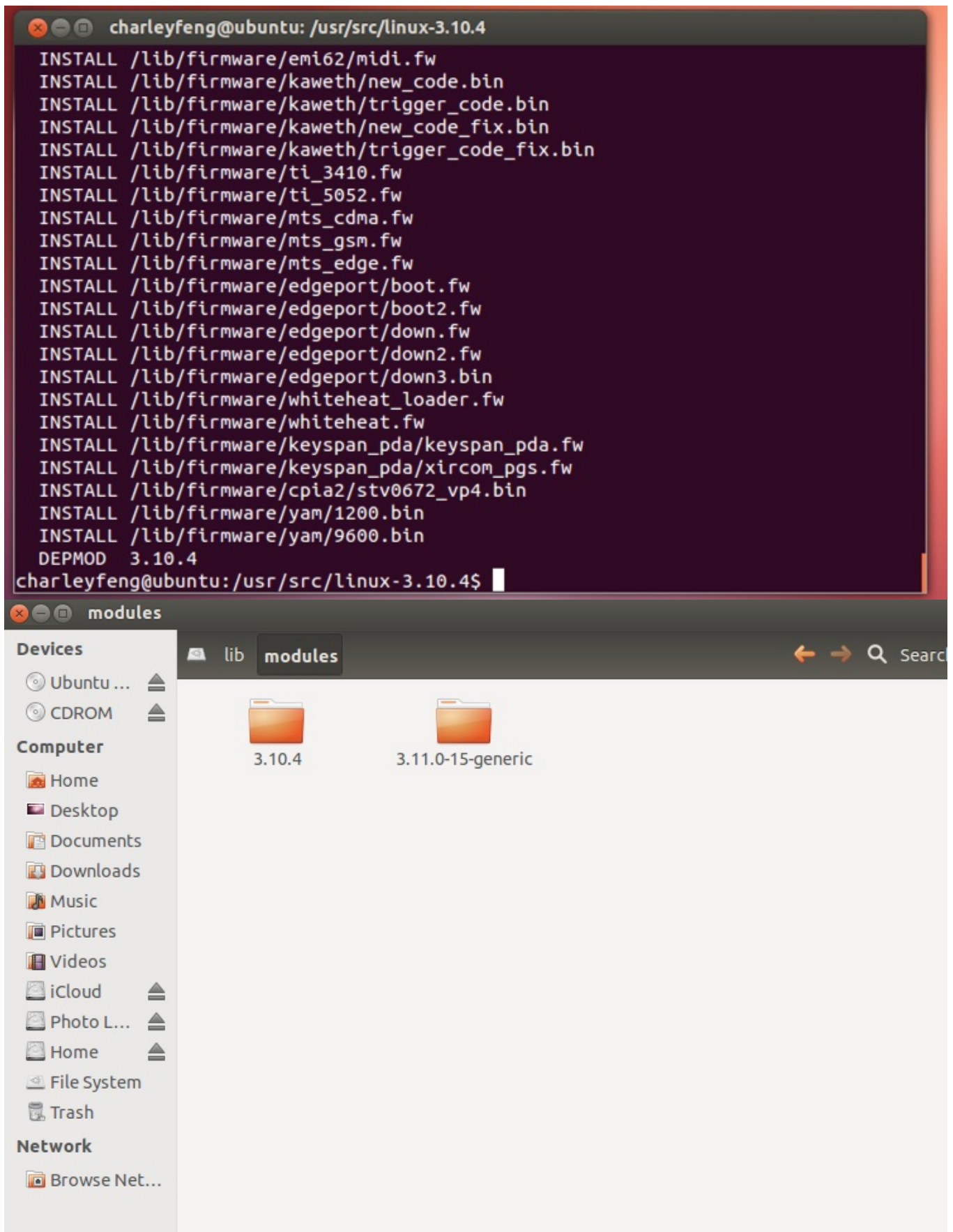


```
charleyfeng@ubuntu: /usr/src/linux-3.10.4
IHEX2FW firmware/emi62/spdif.fw
IHEX2FW firmware/emi62/midi.fw
IHEX    firmware/kaweth/new_code.bin
IHEX    firmware/kaweth/trigger_code.bin
IHEX    firmware/kaweth/new_code_fix.bin
IHEX    firmware/kaweth/trigger_code_fix.bin
IHEX    firmware/ti_3410.fw
IHEX    firmware/ti_5052.fw
IHEX    firmware/mts_cdma.fw
IHEX    firmware/mts_gsm.fw
IHEX    firmware/mts_edge.fw
H16TOFW firmware/edgeport/boot.fw
H16TOFW firmware/edgeport/boot2.fw
H16TOFW firmware/edgeport/down.fw
H16TOFW firmware/edgeport/down2.fw
IHEX    firmware/edgeport/down3.bin
IHEX2FW firmware/whiteheat_loader.fw
IHEX2FW firmware/whiteheat.fw
IHEX2FW firmware/keyspan_pda/keyspan_pda.fw
IHEX2FW firmware/keyspan_pda/xircom_pgs.fw
IHEX    firmware/cpia2/stv0672_vp4.bin
IHEX    firmware/yam/1200.bin
IHEX    firmware/yam/9600.bin
charleyfeng@ubuntu: /usr/src/linux-3.10.4$
```

要是出现错误，请重复上述步骤.....

步骤五 调用make modules_install安装模组

相比于前两个，这个还好。如果进行到这里了，安装完模组会在 lib/modules 下出现



步骤六 调用 `cp arch/x86/boot/bzImage /boot/vmlinuz-3.10.4` 进行拷贝

注意空格。

```
charleyfeng@ubuntu:/usr/src/linux-3.10.4$ sudo cp arch/x86/boot/bzImage /boot/vmlinuz-3.10.4
[sudo] password for charleyfeng:
charleyfeng@ubuntu:/usr/src/linux-3.10.4$
```

步骤七 调用cp .config /boot/config-3.10.4

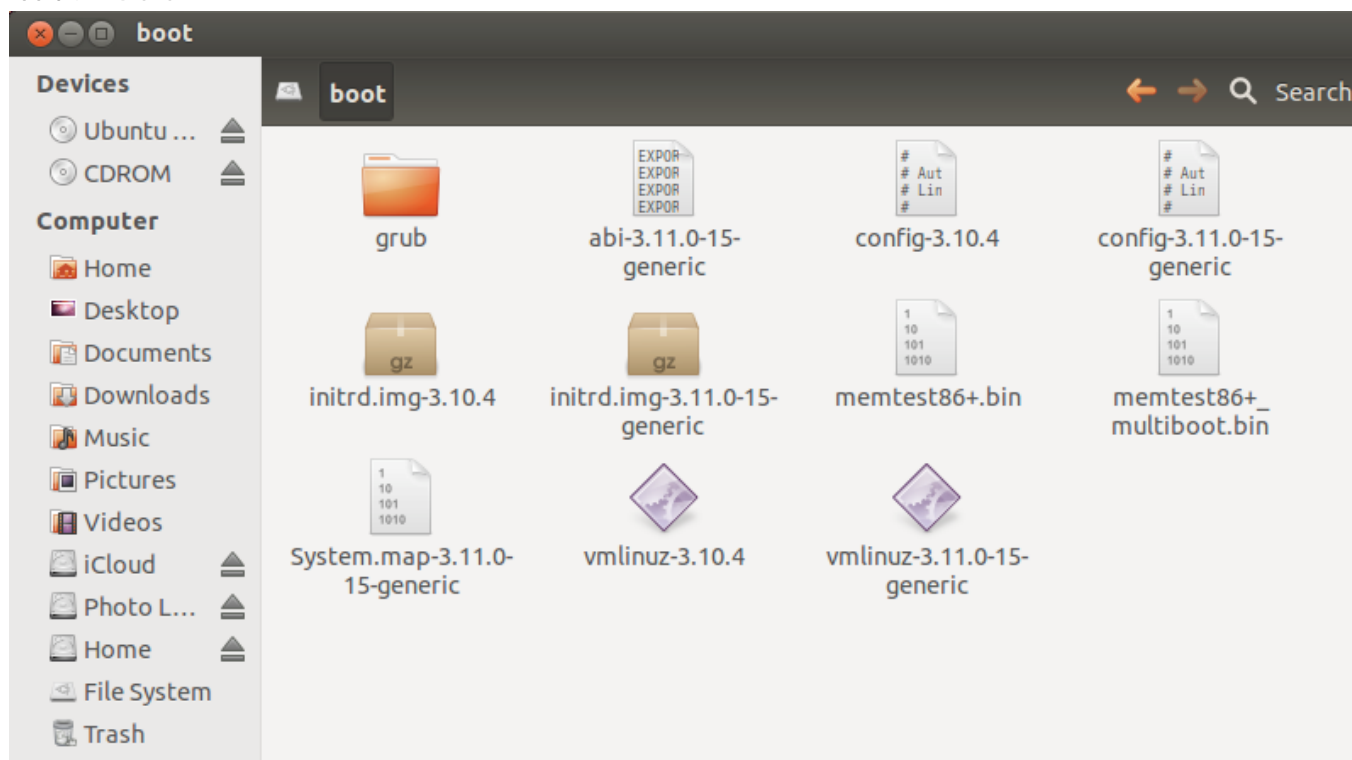
完成编译。

```
charleyfeng@ubuntu:/usr/src/linux-3.10.4$ sudo cp .config /boot/config-3.10.4
charleyfeng@ubuntu:/usr/src/linux-3.10.4$
```

6. 创建开机镜像

用 `mkinitramfs -o /boot/initrd.img-3.10.4 3.10.4` 命令创建

结果如下图

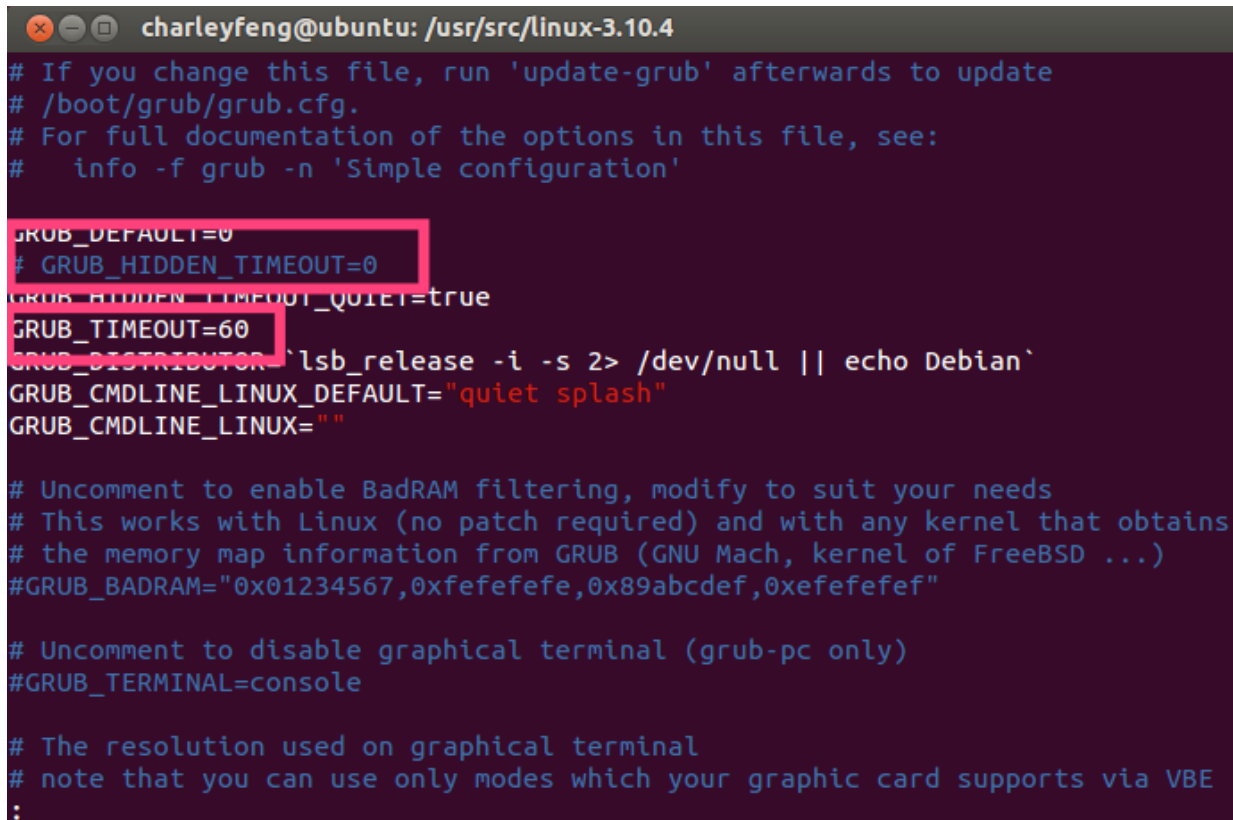


如果出现奇怪错误

- 参考 [Ubuntu initrd分析](#)
- 参考 [mkinitrd mkinitramfs](#)
- 参考 [万能网站](#)

7. 修改开机启动项

1. 用命令vim /etc/default/grub打开要修改的文件
2. 将GRUB_TIMEOUT改为一个大于0的数,以及注释掉一条语句



```
charleyfeng@ubuntu: /usr/src/linux-3.10.4
# If you change this file, run 'update-grub' afterwards to update
# /boot/grub/grub.cfg.
# For full documentation of the options in this file, see:
#   info -f grub -n 'Simple configuration'

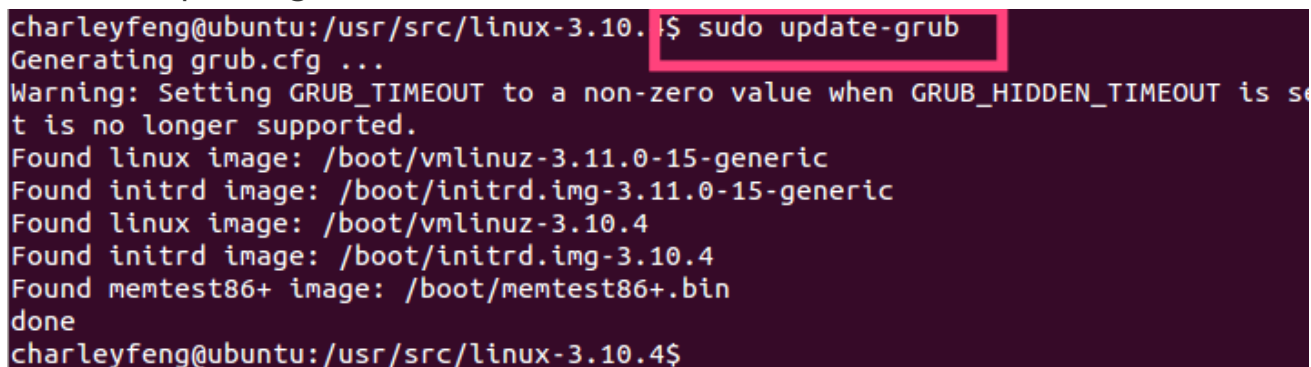
GRUB_DEFAULT=0
# GRUB_HIDDEN_TIMEOUT=0
GRUB_HIDDEN_TIMEOUT_QUIET=true
GRUB_TIMEOUT=60
GRUB_DISTRIBUTOR=`lsb_release -i -s 2> /dev/null || echo Debian`
GRUB_CMDLINE_LINUX_DEFAULT="quiet splash"
GRUB_CMDLINE_LINUX=""

# Uncomment to enable BadRAM filtering, modify to suit your needs
# This works with Linux (no patch required) and with any kernel that obtains
# the memory map information from GRUB (GNU Mach, kernel of FreeBSD ...)
#GRUB_BADRAM="0x01234567,0xfefefefe,0x89abcdef,0xefefefef"

# Uncomment to disable graphical terminal (grub-pc only)
#GRUB_TERMINAL=console

# The resolution used on graphical terminal
# note that you can use only modes which your graphic card supports via VBE
:
```

3. 调用sudo update-grub 命令更新一下开机启动项

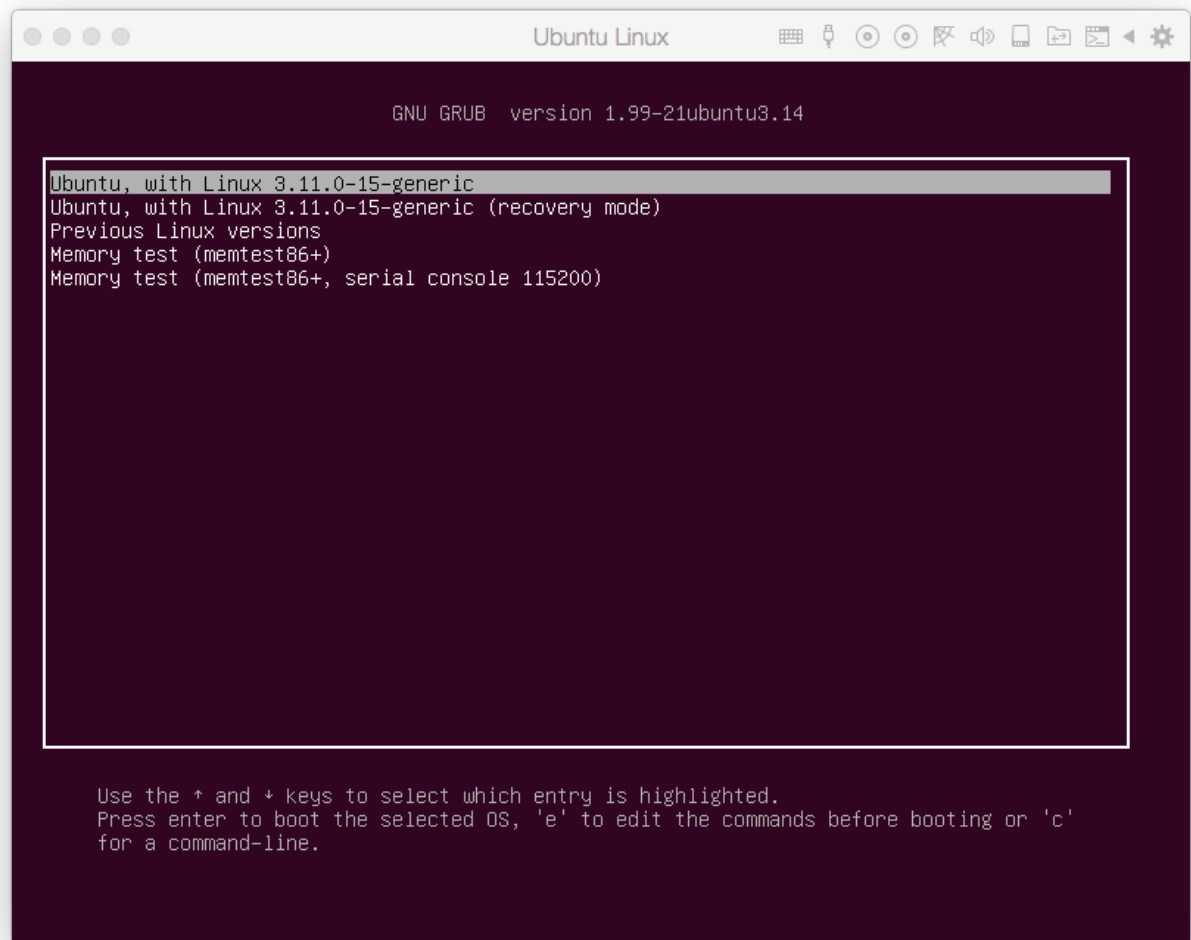


```
charleyfeng@ubuntu: /usr/src/linux-3.10.4$ sudo update-grub
Generating grub.cfg ...
Warning: Setting GRUB_TIMEOUT to a non-zero value when GRUB_HIDDEN_TIMEOUT is set
is no longer supported.
Found linux image: /boot/vmlinuz-3.11.0-15-generic
Found initrd image: /boot/initrd.img-3.11.0-15-generic
Found linux image: /boot/vmlinuz-3.10.4
Found initrd image: /boot/initrd.img-3.10.4
Found memtest86+ image: /boot/memtest86+.bin
done
charleyfeng@ubuntu: /usr/src/linux-3.10.4$
```

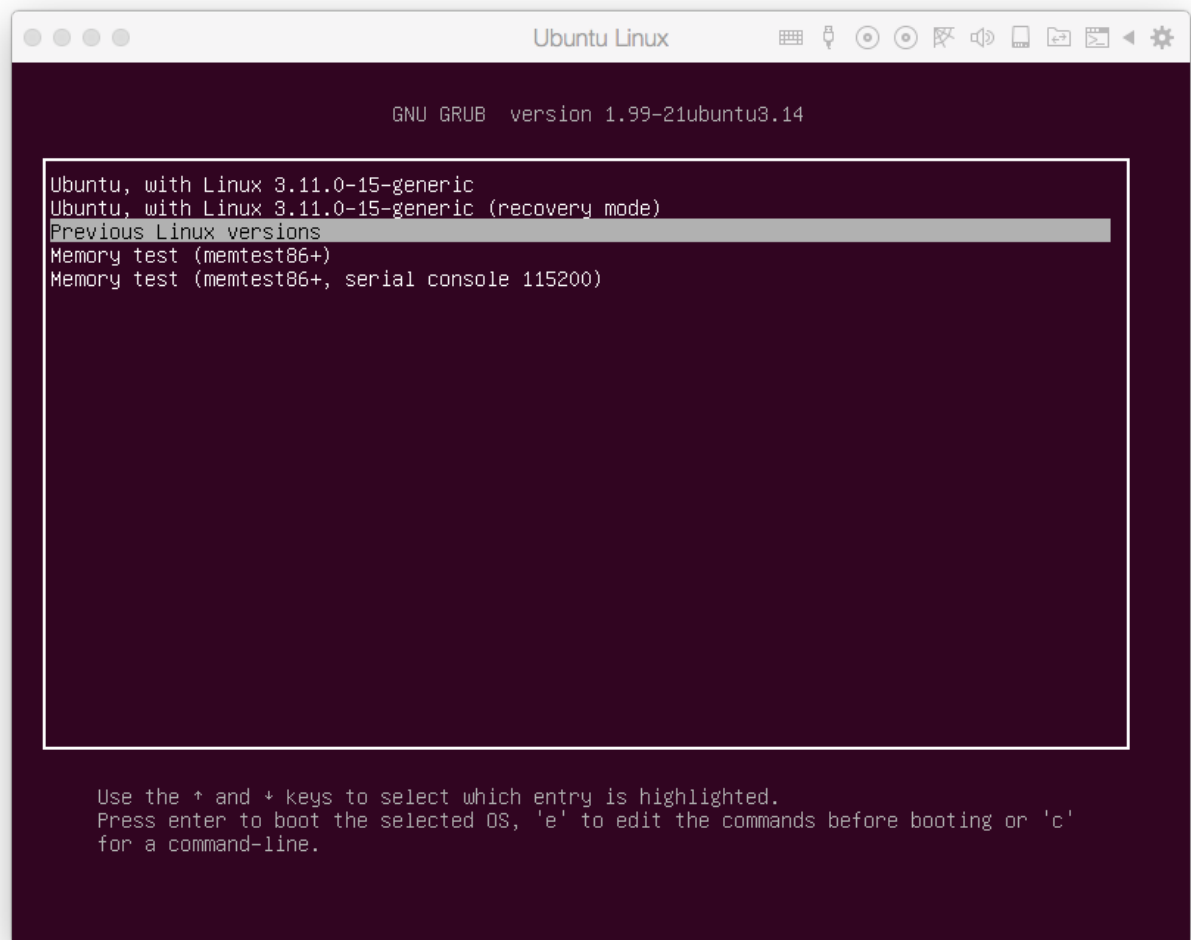
8. 重新启动并查看版本号

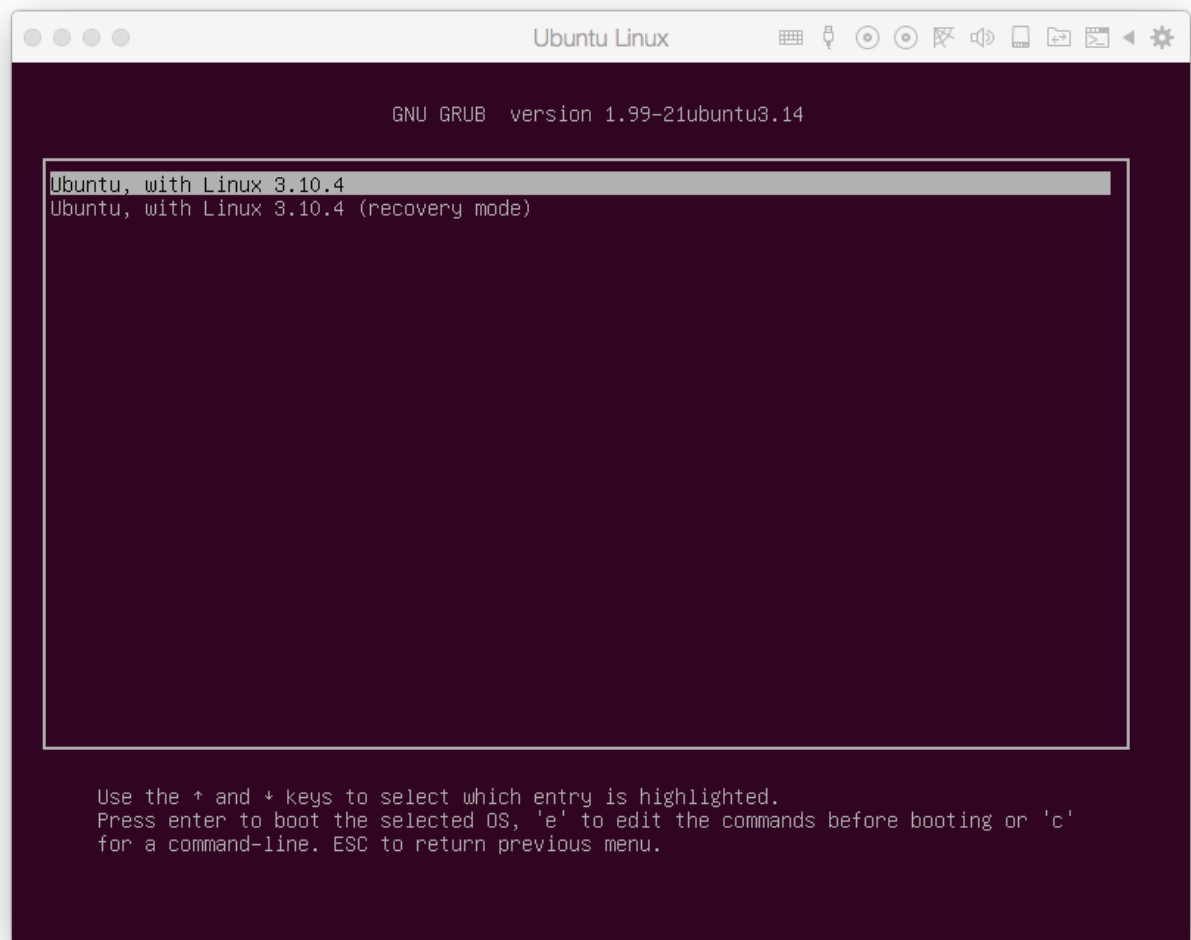
1. 重新启动 (reboot) 在开机界面出现时,有几秒钟的延迟,此时按下ESC键,出现这样

的下面的画面



2. 用命令`uname -a`查看版本号 由于修改的内核版本低于原有的内核，所以我们选择





标题

9. 利用代码对自定义系统调用进行测试

代码如下：

```
1.  #include<linux/unistd.h>
2.  #define _NR_helloworld 222
3.  int main(void){
4.      syscall(_NR_helloworld);
5.      return 0;
6.  }
```

在相应目录下编译为可执行文件 执行该可执行文件 查看输出

```
charleyfeng@ubuntu: ~/Desktop
#include <linux/unistd.h>
#define _NR_helloworld 222
int main(void){
    syscall(_NR_helloworld);
    return 0;
}
~
charleyfeng@ubuntu:~/Desktop$ vim sys_helloworld.c
charleyfeng@ubuntu:~/Desktop$ gcc sys_helloworld.c -o sys_helloworld
charleyfeng@ubuntu:~/Desktop$
```

执行 `sudo dmesg -c`

```
charleyfeng@ubuntu: ~/Desktop
profile_load" name="/usr/bin/evince-preview//sanitized_helper" pid=934 comm="a
pparmor_parser"
[ 146.075406] type=1400 audit(1489462181.021:26): apparmor="STATUS" operation="
profile_load" name="/usr/bin/evince-thumbnailer" pid=934 comm="apparmor_parser"
[ 146.075654] type=1400 audit(1489462181.021:27): apparmor="STATUS" operation="
profile_load" name="/usr/bin/evince-thumbnailer//sanitized_helper" pid=934 comm=
"apparmor_parser"
[ 152.699055] ISO 9660 Extensions: Microsoft Joliet Level 3
[ 152.699934] ISO 9660 Extensions: Microsoft Joliet Level 3
[ 152.712509] ISO 9660 Extensions: RRIP_1991A
[ 152.714889] ISOFS: changing to secondary root
[ 167.966989] type=1400 audit(1489462202.921:28): apparmor="DENIED" operation="
open" parent=1 profile="/usr/lib/telepathy/mission-control-5" name="/usr/share/g
vfs/remote-volume-monitors/" pid=2172 comm="mission-control" requested_mask="r"
denied_mask="r" tsuid=1000 ouid=0
[ 361.144870] hello fengxiachong!
[ 366.425981] hello fengxiachong!hello fengxiachong!
[ 381.160833] hello fengxiachong!hello fengxiachong!
[ 382.233501] hello fengxiachong!hello fengxiachong!
[ 383.274103] hello fengxiachong!hello fengxiachong!
[ 384.265965] hello fengxiachong!hello fengxiachong!
[ 385.194040] hello fengxiachong!hello fengxiachong!
[ 386.098013] hello fengxiachong!
charleyfeng@ubuntu:~/Desktop$
```

实验报告要求

写实验报告！！！不是打印照片！！！

- 根据实验九个步骤分别进行截图和说明（下图只是示例）

- 1. 下载好的压缩包以及文档.PNG
- 1. 移动文件出现权限错误 使用sudo移动成功.PNG
- 2. 解压成tar.PNG
- 3. 解压tar成功.PNG
- 4. 解压后如图.PNG
- 5. 打开sys.PNG
- 6. sys文件内容.PNG
- 7. 添加的函数.PNG
- 8. 添加函数声明.PNG
- 9. gedit添加声明.PNG
- 10. 添加id.PNG
- 11. gedit添加222.PNG
- 13. 出现问题menuconfig.PNG
- 14. 出现错误1.PNG
- 14. 解决问题1.PNG
- 14. 解决问题2 安装依赖.PNG
- 15. 窗口太小错误.PNG
- 17. menucomfig成功.PNG
- 18. 开始编译.PNG
- 19. 编译出现错误.PNG
- 20. 开始编译界面.PNG
- 21. 第一步编译完后.PNG
- 22. 编译模组完成.PNG
- 23. 安装模组.PNG
- 24. 安装模组完成.PNG
- 25. 安装模组完成后出现的.PNG
- 26. 步骤6.PNG
- 27. 步骤7.PNG
- 28. 创建开机镜像.PNG
- 29. 创建开机镜像后的结果.PNG
- 29. 修改开机启动项1.PNG
- 30. timeout修改.PNG
- 31. update出现错误.PNG
- 32. updategrub.PNG
- 33. 问题解决时间设置无效.PNG
- 34. 解决time设置问题.PNG
- 35. 新的boot画面.PNG
- 36. 编译测试程序.PNG
- 37. 测试程序.PNG
- 38. 最后结果.PNG
- 39. 最后执行.PNG

● 实验中遇到的问题及其解决方式

我看实验报告主要看教程中没有的部分，如果和教程一样只是罗列顺序，是不会有高分的，需要自己思考，多查资料，例如：Linux根目录下每个文件夹一般放什么文件.....

- 截图必须带有主机名!

这就是为什么我实验中一直用sudo 因为你用了su 我怎么知道是你做的!

各位注意, 如果没有权限请使用**sudo执行命令**, 所有截图必须带有主机名, 如若不是, 视为抄袭

- 独立完成, 有几份相同, 这几份的总分就是100 (很是人性化)
- 实验报告下周五晚上6点之前
 - 电子版发送至邮箱: hitwh2017systemlab@163.com
 - 纸质版学委收齐以后交至宋健中505后面桌子
 - *注意命名*

友情提示

1. 使用VMware的同学, 在虚拟机上最好提前安装了VMware Tools, 因为需要在win7和ubuntu上来回的复制 粘贴。安装教程请戳[这里](#)。
 2. 大量的时间。
 3. 耐心和细致。
 4. 注意命令之间的空格不要一个劲的闭着眼睛去复制粘贴。
-