Linux 内核修改与编译图文教程

麦洛克菲内核底层安全培训

www.mallocfree.com

1、实验目的

针对 Ubuntu10.04 中，通过下载新的内核版本，并且修改新版本内核中的系统调用看，然后，在其系统中编译，加载新内核。

本实验在 Ubuntu 10.04 +编译 **linux-2.6.32.65 内核版本成功**

2、任务概述

2.1 下载新内核

http://www.kernel.org/
linux-2.6.32.65

2.2 修改新内核添加一个系统调用：sys_mycall

添加新的系统调用函数，用来判断输入数据的奇偶性。

2.3 进行新内核编译

通过修改新版内核后，进行加载编译。最后通过编写测试程序进行测试

3、实验步骤

3.1 准备工作

查看系统先前内核版本：

(终端下)使用命令：**uname -r**

3.2 下载最新内核

我这里使用的内核版本是

3.3 解压新版内核

将新版内核复制到**“/usr/src”**目录下

在终端下用命令：cd　/usr/src 进入到该文件目录

解压内核：**linux-2.6.32.65.tar.xz，在终端进入 cd　/usr/src 目录输入一下命令：**

cp linux-2.6.32.65.tar.xz /usr/src/
cd /usr/src/
xz -d linux-2.6.32.65.tar.xz
tar xvf linux-2.6.32.65.tar

文件将解压到/usr/src/linux-2.6.32.65 目录中

使用命令：**ln -s linux-2.6.32.65 linux**

```
zyr@ubuntu:~$ cd /usr/src
zyr@ubuntu:/usr/src$ ll
total 24
drwxrwsr-x  6 root src  4096 2015-02-21 23:10 ./
drwxr-xr-x 10 root root 4096 2010-04-29 20:17 ../
lrwxrwxrwx  1 root src    16 2015-02-21 23:10 linux -> linux-2.6.32.65//
drwxr-sr-x 24 root src  4096 2015-02-22 01:43 linux-2.6.32.65/
drwxr-sr-x 24 root src  4096 2015-02-21 23:07 linux-3.18.3/
drwxr-xr-x 24 root root 4096 2010-04-29 20:44 linux-headers-2.6.32-21/
drwxr-xr-x  7 root root 4096 2010-04-29 20:45 linux-headers-2.6.32-21-generic/
zyr@ubuntu:/usr/src$
```

3.4 安装必要的工具

在终端下输入一下命令：
sudo apt-get install build-essential kernel-package libncurses5-dev fakeroot
sudo aptitude install libqt3-headers libqt3-mt-dev libqt3-compat-headers libqt3-mt

3.5 内核修改

3.5.0 asm、linux 和 scsi 等链接是指向要升级的内核源代码

 # cd /usr/include/
 # rm -r asm linux scsi
 # ln -s /usr/src/linux/include/asm-generic asm
 # ln -s /usr/src/linux/include/linux linux
 # ln -s /usr/src/linux/include/scsi scsi

3.5.1 添加新的系统调用

在文件：/usr/src/linux/arch/x86/kernel/syscall_table_32.S 最后增加一个系统表项：
.long sys_mycall

```
.long sys_fallocate
.long sys_timerfd_settime       /* 325 */
.long sys_timerfd_gettime
.long sys_signalfd4
.long sys_eventfd2
.long sys_epoll_create1
.long sys_dup3                  /* 330 */
.long sys_pipe2
.long sys_inotify_init1
.long sys_preadv
.long sys_pwritev
.long sys_rt_tgsigqueueinfo     /* 335 */
.long sys_perf_event_open
.long sys_mycall
```

3.5.2 添加系统调用号

x86:
/usr/src/linux/arch/x86/include/asm/unistd_32.h 中添加：
#define _NR_mycall 337
#define NR_syscalls 338 //原来 NR_syscalls 是 337，现在由于加了一个系统调用，所以应该加 1

```
#define __NR_pwritev                334
#define __NR_rt_tgsigqueueinfo      335
#define __NR_perf_event_open        336
#define __NR_mycall                 337

#ifdef __KERNEL__

#define NR_syscalls 338
```

x64:

/usr/src/linux/arch/x86/include/asm/unistd_64.h 中添加：

#define __NR_mycall 299

__SYSCALL(__NR_mycall, sys_mycall)

```
#define __NR_pwritev                                296
__SYSCALL(__NR_pwritev, sys_pwritev)
#define __NR_rt_tgsigqueueinfo                      297
__SYSCALL(__NR_rt_tgsigqueueinfo, sys_rt_tgsigqueueinfo)
#define __NR_perf_event_open                        298
__SYSCALL(__NR_perf_event_open, sys_perf_event_open)
#define __NR_mycall                                 299
__SYSCALL(__NR_mycall, sys_mycall)

#ifndef __NO_STUBS
#define __ARCH_WANT_OLD_READDIR
#define __ARCH_WANT_OLD_STAT
```

3.5.3 添加系统调用的处理函数

在/usr/src/linux/kernel/sys.c 中添加以下处理函数：

函数源码如下：(判断奇偶数)

asmlinkage int sys_mycall(int n)
{
 	
  if(n%2==0)
  return 1;
  else
  return 0;
}

3.6 清除从前编译内核时残留的.o 文件和不必要的关联（如果从前没有进行内核编译的话，则可以省略这一步）

终端下切换至：cd /usr/src/linux

输入以下命令：make mrproper

3.8 配置内核，修改相关参数

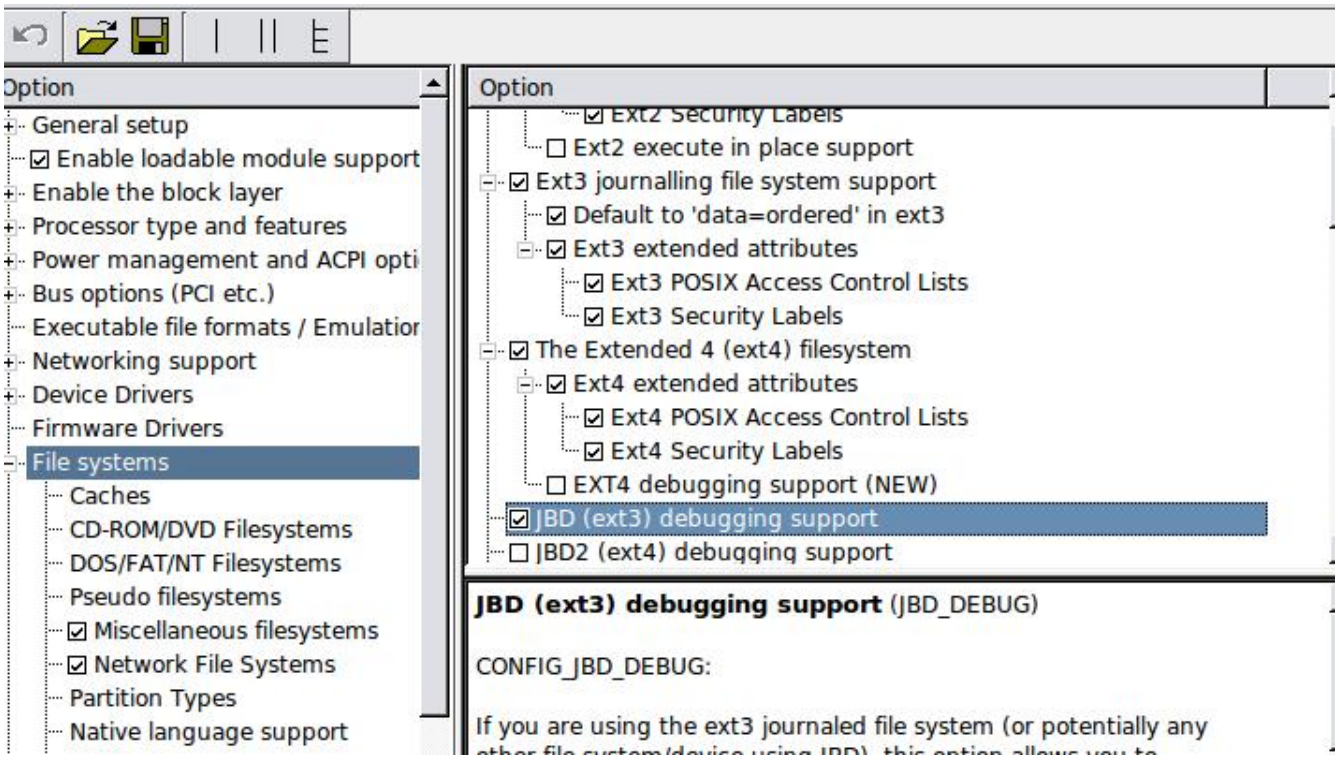
3.8.1 如何配置内核参数？

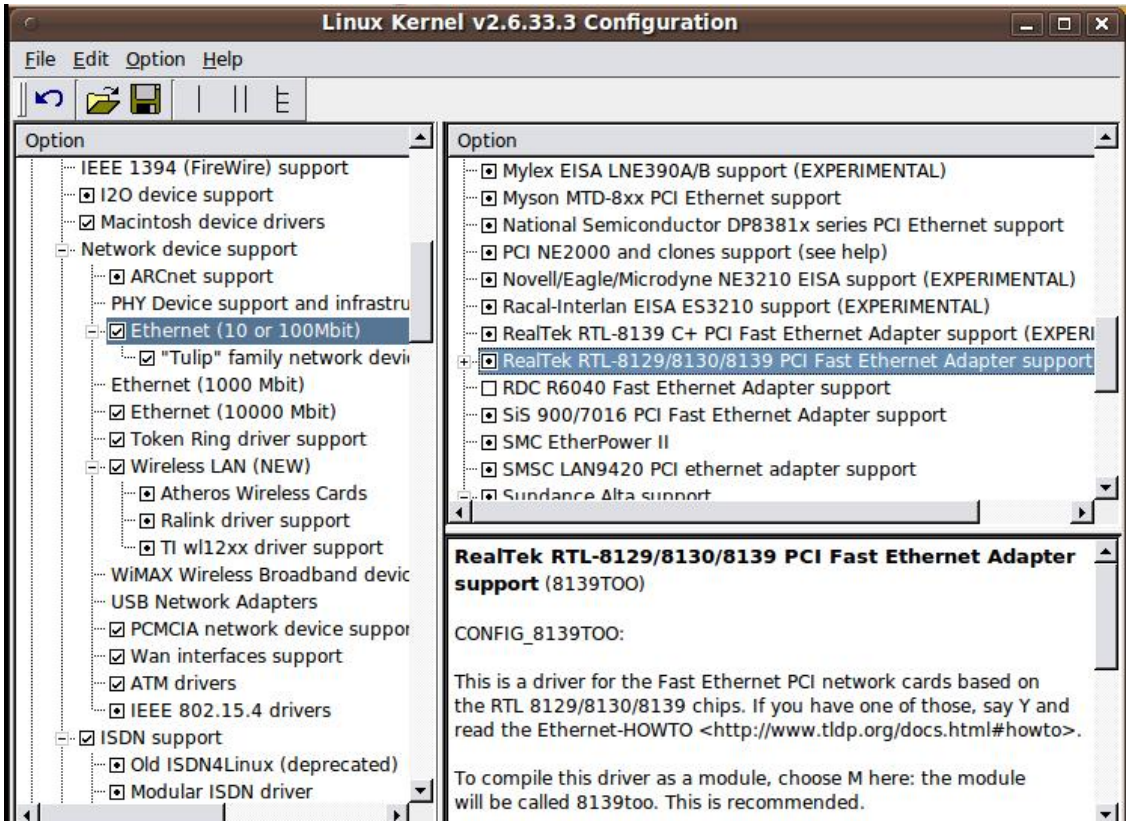
y：将该功能编译进内核。
n：不将该功能编译进内核。
m：将该功能编译成在需要时动态插入到内核中的模块。
单击“ Main Menu”按钮，返回主配置窗口；
单击“ Next”按钮，配置下一个配置项；
单击“ Prev”按钮，配置上一个配置项。
选择的原则是将与内核其它部分关系较远且不经常使用的部分功能代码编译成为可加载模块，有利于缩减内核，减少内核消耗的内存。与内核关心紧密而且经常使用的部分功能代码直接编译到内核中。

输入以下命令：cd /usr/src/linux

输入以下命令：make xconfig

CONFIG_FRAME_POINTER





字符界面下，make menuconfig

设置编译配置选项时，有 3 种选择：

Y--将该功能编译进内核

N--不将该功能编译进内核
M--将该功能编译成可以在需要的时候动态插入到内核中的模块

对于 make menuconfig:
空表示：N
*表示：Y
其它如 M 表示：M

对于 make xconfig:

大多数 条目都可以有三种状态：打上对勾，打上圆点，无勾选，这三种状态分别对应.config 文件中的=y，=m，=n，也即直接编译进内核镜像，编译为 LKM，不定义宏或不编译。

选择相应的配置时，有三种选择，它们分别代表的含义如下：
Y－-将该功能编译进内核
N－-不将该功能编译进内核
M－-将该功能编译成可以在需要时动态插入到内核中的模块
如果使用的是make xconfig，使用鼠标就可以选择对应的选项。如果使用的是
make menuconfig，则需要使用空格键进行选取。你会发现在每一个选项前都有个括号，但
有的是中括号有的是尖括号，还有一种圆括号。用空格键选择时可以发现，中括号里要么
是空，要么是*，而尖括号里可以是空，*和M。这表示前者对应的项要么不要，要么
编译到内核里；后者则多一样选择，可以编译成模块。而圆括号的内容是要你在所提供的
几个选项中选择一项。

3.8.2 修改内核范围(更新后能否启动至关重要，下面显示的是使用 make menuconfig 的结果,make xconfig 对应的是*→√，M→.)

1)文件系统
请务必要选中 ext3 文件系统,
File systems--->
[*] Ext3 journalling file system support
[*] Ext3 Security Labels
[*] JBD (ext3) debugging support
(一定要修改)
以上三项一定要选上,而且要内建(即标*). 这个非常重要,在配置完后一定要检查一下.config 文件有没有"CONFIG_EXT3_FS=y"这一项. 如果不是"CONFIG_EXT3_FS=y"而是"CONFIG_EXT3_FS=m",你在运行内核时就会遇上以下错误: pivotroot: pivot_root(/sysroot,/sysroot/initrd) failed

2)网卡驱动(在 Ubuntu10.04 中已默认选择，可省略这一步)

请务必把自己网卡对应的驱动编译进内核,比较普遍的网卡是 realtek 8139,以下就是这种网卡的配置,以供参考
Device Drivers--->

Networking support--->
Ethernet (10 or 100Mbit) --->
<*> RealTek RTL-8139 C+ PCI Fast Ethernet Adapter support (EXPERIMENTAL)
<*> RealTek RTL-8139 PCI Fast Ethernet Adapter support

3)声卡驱（在 Ubuntu10.04 中已默认选择，可省略这一步）

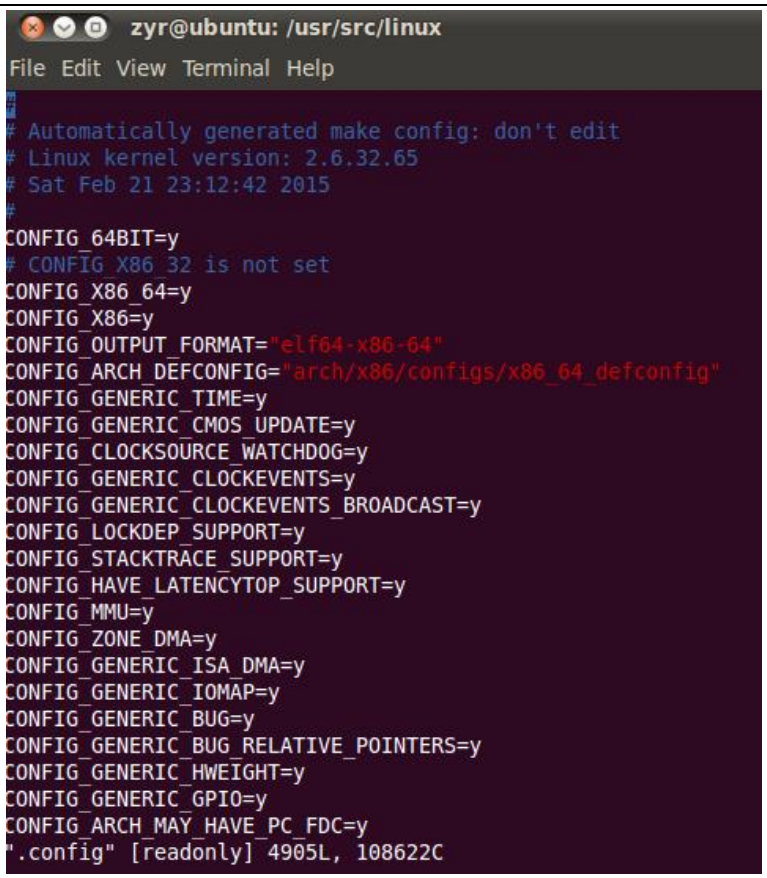
选择自己声卡对应的驱动编译进内核,比较普遍的声卡是 i810_audio,以下就是这种声卡的配置,以供参考
Device Drivers --->
Sound --->
<*> Sound card support
Advanced Linux Sound Architecture --->
<*> Advanced Linux Sound Architecture
<*> Sequencer support
< > Sequencer dummy client
<*> OSS Mixer API
<*> OSS PCM (digital audio) API[*] OSS Sequencer API
<*> RTC Timer support
PCI devices --->
<*> Intel i8x0/MX440, SiS 7012; Ali 5455; NForce Audio; AMD768/8111
Open Sound System --->
< > Open Sound System (DEPRECATED)

4）支持内核调试的选项：

CONFIG_DEBUG_KERNEL
CONFIG_FRAME_POINTER
CONFIG_DEBUG_INFO
CONFIG_DEBUG_STACKOVERFLOW
CONFIG_DEBUG_STACK_USAGE
CONFIG_IKCONFIG
CONFIG_IKCONFIG_PROC
CONFIG_KALLSYMS
CONFIG_MAGIC_SYSRQ　// #echo"g">/proc/sysrq-trigger

在完成 LINUX 编译的内核配置操作之后，会在/usr/src/linux 目录下生成一个.config 文件，用来记录刚才的配置结果。可以在里面搜索上面的一些编译选项，手动再做修改。

vim .config：



3.8.3 编译二进制内核映像文件

在“/usr/src/linux”目录下建立二进制的内核映像文件，命令如下：
cd /usr/src/linux

sudo make:(此步第一次耗时较长，约 3 个小时)

sudo make modules_install:

sudo make install

sudo mkinitramfs -o /boot/initrd.img-2.6.32.65

sudo update-initramfs -c -k 2.6.32.65

sudo update-grub2 //自动修改系统引导配置，根据/etc/default/grub，产生/boot/grub/grub.cfg 启动文件。

```
zyr@ubuntu:/usr/src/linux$ ll /boot/
total 87616
drwxr-xr-x  3 root root     4096 2015-02-22 01:45 ./
drwxr-xr-x 22 root root     4096 2014-12-22 23:10 ../
-rw-r--r--  1 root root   634929 2010-04-16 19:32 abi-2.6.32-21-generic
-rw-r--r--  1 root root   110365 2010-04-16 19:32 config-2.6.32-21-generic
-rw-r--r--  1 root root   108622 2015-02-22 01:43 config-2.6.32.65
drwxr-xr-x  3 root root     4096 2015-02-22 01:46 grub/
-rw-r--r--  1 root root  8341410 2014-12-22 23:14 initrd.img-2.6.32-21-generic
-rw-r--r--  1 root root 68025884 2015-02-22 01:45 initrd.img-2.6.32.65
-rw-r--r--  1 root root   160280 2010-03-23 17:40 memtest86+.bin
-rw-r--r--  1 root root  2152657 2010-04-16 19:32 System.map-2.6.32-21-generic
-rw-r--r--  1 root root  2096006 2015-02-22 01:43 System.map-2.6.32.65
-rw-r--r--  1 root root     1336 2010-04-16 19:35 vmcoreinfo-2.6.32-21-generic
-rw-r--r--  1 root root  4037888 2010-04-16 19:32 vmlinuz-2.6.32-21-generic
-rw-r--r--  1 root root  4012736 2015-02-22 01:43 vmlinuz-2.6.32.65
zyr@ubuntu:/usr/src/linux$
```

```
### BEGIN /etc/grub.d/10_linux ###
menuentry 'Ubuntu, with Linux 2.6.32.65' --class ubuntu --class gnu-linux --class gnu --class os {
        recordfail
        insmod ext2
        set root='(hd0,1)'
        search --no-floppy --fs-uuid --set d216c6f4-9ece-4cf0-8450-aec4ecaf3a31
        linux   /boot/vmlinuz-2.6.32.65 root=UUID=d216c6f4-9ece-4cf0-8450-aec4ecaf3a31 ro   quiet splash
        initrd  /boot/initrd.img-2.6.32.65
}
menuentry 'Ubuntu, with Linux 2.6.32.65 (recovery mode)' --class ubuntu --class gnu-linux --class gnu --class os {
        recordfail
        insmod ext2
        set root='(hd0,1)'
        search --no-floppy --fs-uuid --set d216c6f4-9ece-4cf0-8450-aec4ecaf3a31
        echo    'Loading Linux 2.6.32.65 ...'
        linux   /boot/vmlinuz-2.6.32.65 root=UUID=d216c6f4-9ece-4cf0-8450-aec4ecaf3a31 ro single
        echo    'Loading initial ramdisk ...'
        initrd  /boot/initrd.img-2.6.32.65
}
menuentry 'Ubuntu, with Linux 2.6.32-21-generic' --class ubuntu --class gnu-linux --class gnu --class os {
        recordfail
```

sudo vim /etc/default/grub

注释掉：

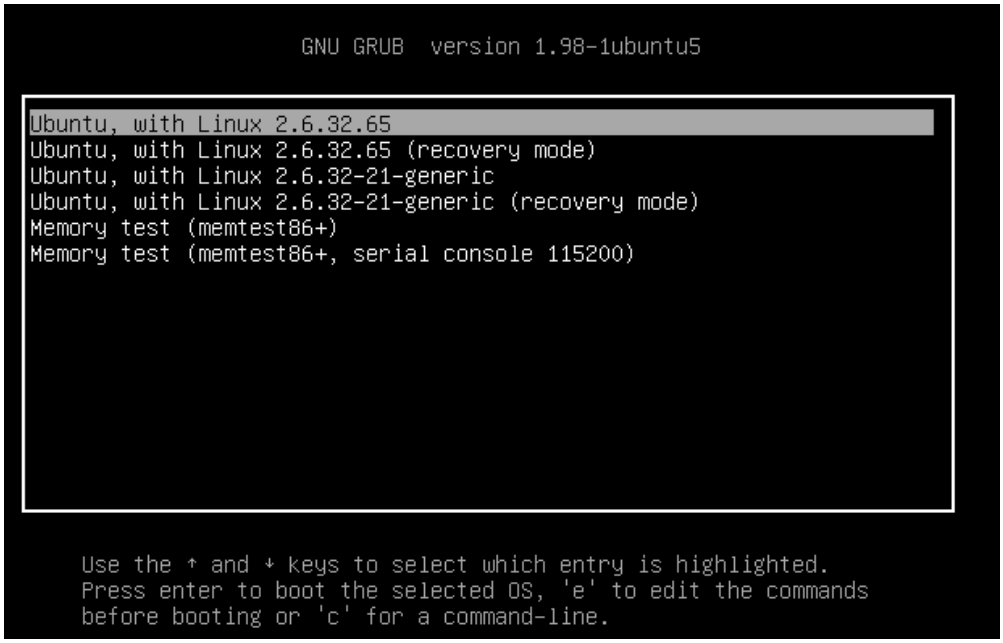
#GRUB_HIDDEN_TIMEOUT=0

```
# If you change this file, run 'update-grub' afterwards to update
# /boot/grub/grub.cfg.

GRUB_DEFAULT=0
#GRUB_HIDDEN_TIMEOUT=0
GRUB_HIDDEN_TIMEOUT_QUIET=true
GRUB_TIMEOUT=10
GRUB_DISTRIBUTOR=`lsb_release -i -s 2> /dev/null || echo Debian`
GRUB_CMDLINE_LINUX_DEFAULT="quiet splash"
GRUB_CMDLINE_LINUX=""

# 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
# you can see them in real GRUB with the command `vbeinfo'
#GRUB_GFXMODE=640x480

# Uncomment if you don't want GRUB to pass "root=UUID=xxx" parameter to Linux
#GRUB_DISABLE_LINUX_UUID=true

# Uncomment to disable generation of recovery mode menu entries
#GRUB_DISABLE_LINUX_RECOVERY="true"

# Uncomment to get a beep at grub start
#GRUB_INIT_TUNE="480 440 1"
```

sudo update-grub2

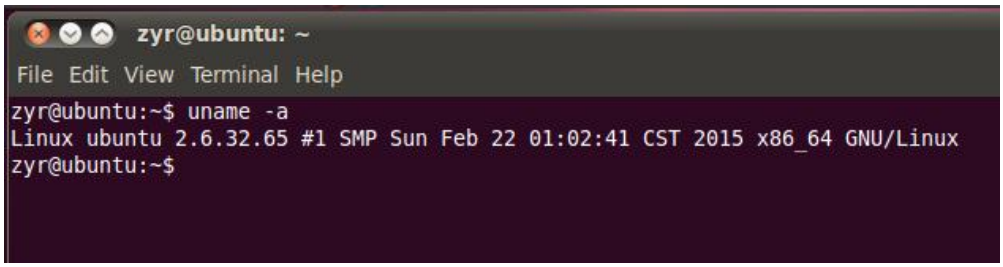
这样在重启系统之后，可以进入启动菜单，选择你想启动的内核。

sudo reboot

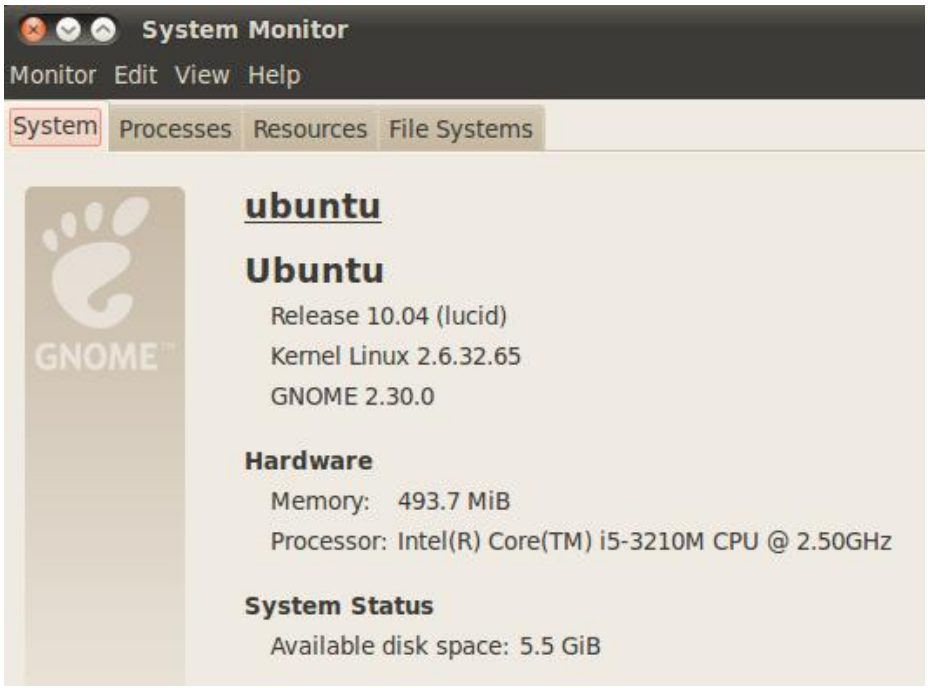


3.9 重启系统

重启系统后在终端下输入一下命令查看内核版本：



通过系统监视器查看内核版本



3.10 修改编译内核完成

4、测试新系统调用

4.1 编写测试程序

下面是 X64 版本：

```
#include <stdio.h>
#include <errno.h>
#include <unistd.h>
#include <sys/syscall.h>
#include <sys/types.h>

#define __NR_mycall 299

int main()
{

        printf("hello worldx\n");
        if(syscall(__NR_mycall, 5))
        {
                printf("even number!\n");
```

```
        }
        else
        {
                printf("odd number!\n");
        }

        return 0;
}
```

```
zyr@ubuntu:/usr/src/linux$ cd
zyr@ubuntu:~$ vim test_syscall.c
zyr@ubuntu:~$ gcc -o test_syscall test_syscall.c
zyr@ubuntu:~$ ./test_syscall
odd number!
zyr@ubuntu:~$
```

如果是 X86，那么可能会需要定义#define __NR_mycall 337.