

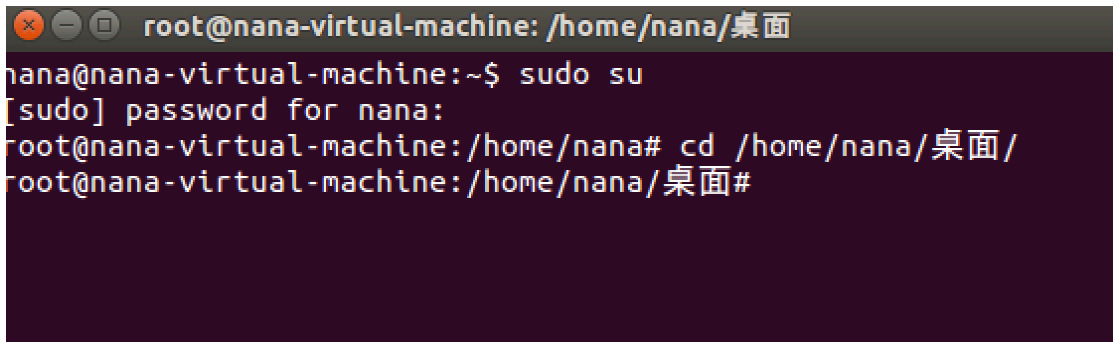
操作系统实验二

向 Linux 内核增加一个系统调用

实验步骤：

注：我的 linux 系统为 ubuntu14.04 版本，新增内核版本为 2.6.33.3

- 1、开启电脑，拷贝内核源码文件 `linux-2.6.33.3.tar.gz`
- 2、把文件从 windows 拷贝到 Linux 系统中。(以拷入桌面为例).
- 3、拷贝到 linux 桌面后，打开终端，输入 `sudo su`，输入你登录系统时的那个密码，获取 root 权限；`cd` 命令进入桌面，



```
root@nana-virtual-machine: /home/nana/桌面
nana@nana-virtual-machine:~$ sudo su
[sudo] password for nana:
root@nana-virtual-machine:/home/nana# cd /home/nana/桌面/
root@nana-virtual-machine:/home/nana/桌面#
```

4.增加系统调用

之后在终端输入 `tar -xzvf linux-2.6.33.3.tar.gz -C /usr/src` 解压文件到 `/usr/src` 目录下，
我的文件是 `linux-2.6.33.3`，具体根据自己的压缩包名字进行修改；

`cd /usr/src` 进入 `/usr/src`

(`ln -s linux-2.6.33.3 linux` 新建一个名字为 `linux` 的快捷方式

`cd linux`) 这两步可以不写，我是为了方便

5.对增加的系统调用添加相关内容

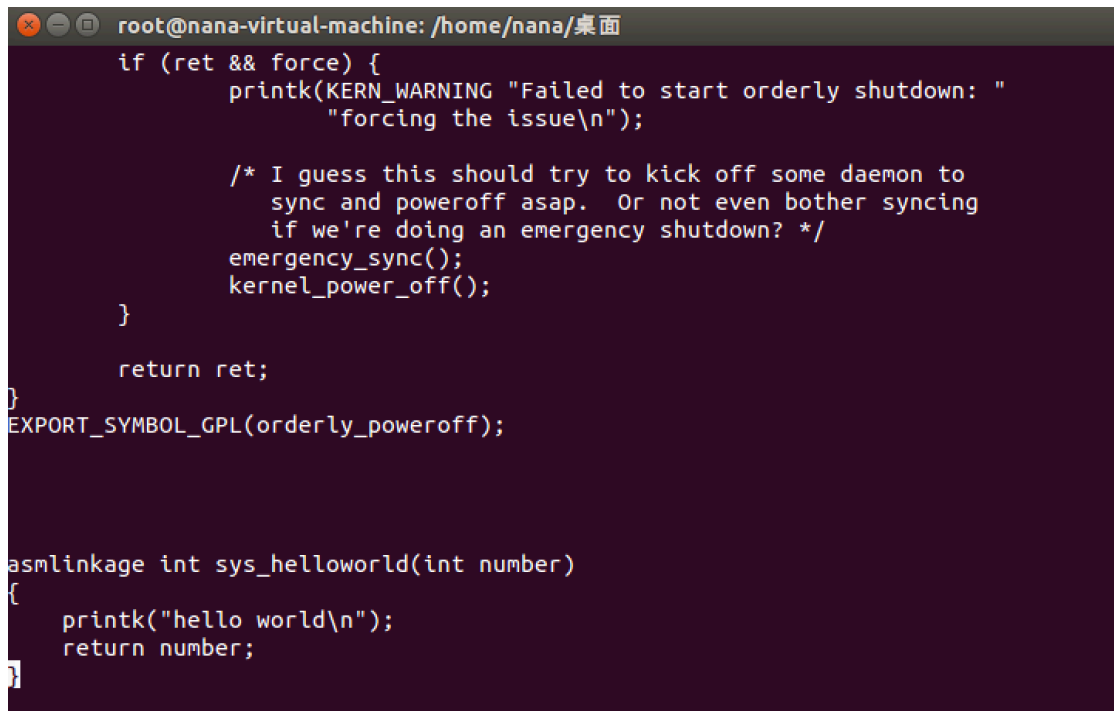
第一步：修改源程序，增加系统调用实现

`# vi /usr/src/linux-2.6.33.3/kernel/sys.c`，在代码末尾加入下面的函数，注意一定不要打错，否则后边 `make` 的时候会各种报错。这里比较常用到的命令有按 `esc` 进入命令模式，`x` 可以删除错误字符，`i` 开始输入文本，最后编写完按 `esc` 进入命令模式后，输入冒号：`wq`，保存退出。

```

asmlinkage int sys_helloworld(int number) // 该函数名中有下划线
{
    printk("hello,world\n"); //printk()函数是系统内核的输出函数，区别 //于 printf()。
    return number;
}

```



```

root@nana-virtual-machine: /home/nana/桌面
    if (ret && force) {
        printk(KERN_WARNING "Failed to start orderly shutdown: "
            "forcing the issue\n");

        /* I guess this should try to kick off some daemon to
        sync and poweroff asap. Or not even bother syncing
        if we're doing an emergency shutdown? */
        emergency_sync();
        kernel_power_off();
    }

    return ret;
}
EXPORT_SYMBOL_GPL(orderly_poweroff);

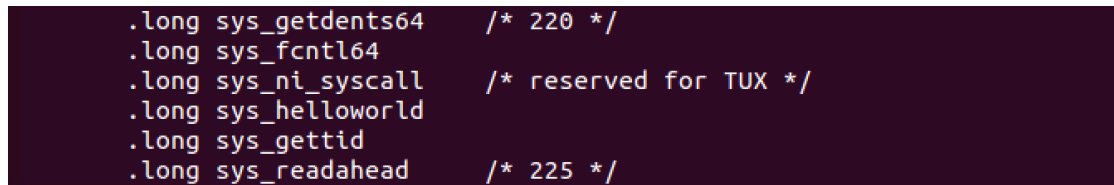
asmlinkage int sys_helloworld(int number)
{
    printk("hello world\n");
    return number;
}

```

第二步：修改头文件，增加系统调用声名

```
#vi /usr/src/linux-2.6.33.3/arch/x86/kernel/syscall_table_32.S
```

将 223 行 `.long sys_ni_syscall` 改为 `.long sys_helloworld`（不要放在其他地方，否则可能出现错误,我曾放在 222 行都不行），依然保存退出



```

    .long sys_getdents64    /* 220 */
    .long sys_fcntl64
    .long sys_ni_syscall    /* reserved for TUX */
    .long sys_helloworld
    .long sys_gettid
    .long sys_readahead    /* 225 */

```

第三步：修改系统调用表，注册系统调用。

```
# vi /usr/src/linux-2.6.33.3/arch/x86/include / asm/unistd_32.h
```

在如图所示的位置，增加一行 `#define __NR_helloworld 223` 对应上面的 223。

```
#define __NR_madvise1      219      /* delete when C lib stub is rem
#define __NR_getdents64    220
#define __NR_fcntl64      221
/* 223 is unused */
#define __NR_gettid        224
#define __NR_readahead     225
```

6、编译安装内核

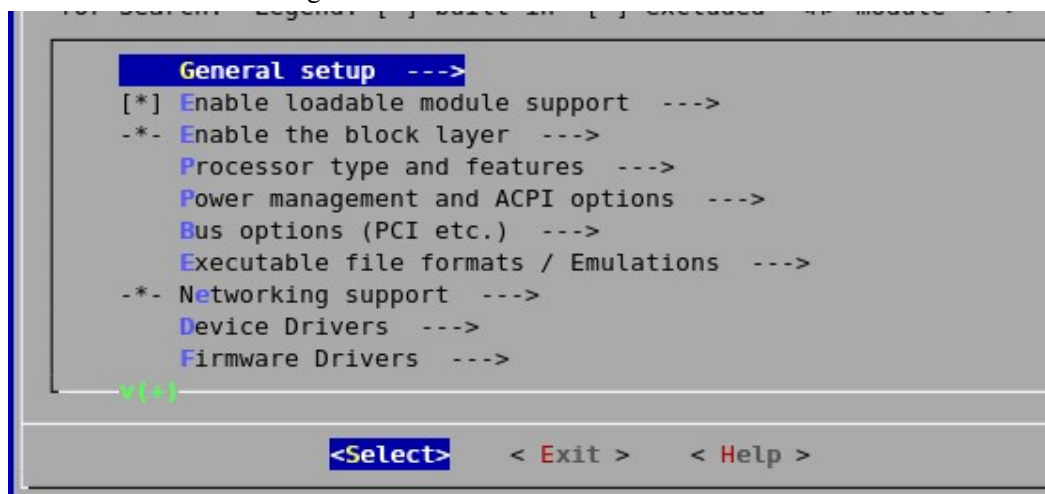
第一步到第四步都是在 `/usr/src/linux-2.6.33.3` 目录下运行。先跳转到该目录下。

第一步：`yum -y install gcc` 此步骤为安装 `gcc`，一般都不需要进行此步骤

第二步：`make mrproper` 清除内核中不稳定的目标文件，附属文件及内核配置文件

第三步：`make clean` 清除以前生成的目标文件和其他文件

第四步：`make menuconfig` 配置内核，采用默认的内核配置即可。（选择 `exit`）



第五步：`make` （该命令等价于 `make bzImage` 编译内核 + `make modules` 编译内核模块）

此步骤最容易出现问題，关于问题的汇总在文档最后，可以参照问题具体内容进行修改，直至编译完成，根据个人电脑配置，等的时候或长或短，最少也要 15 分钟，目前见到的两个成功的大概花了 1 个半小时以上。

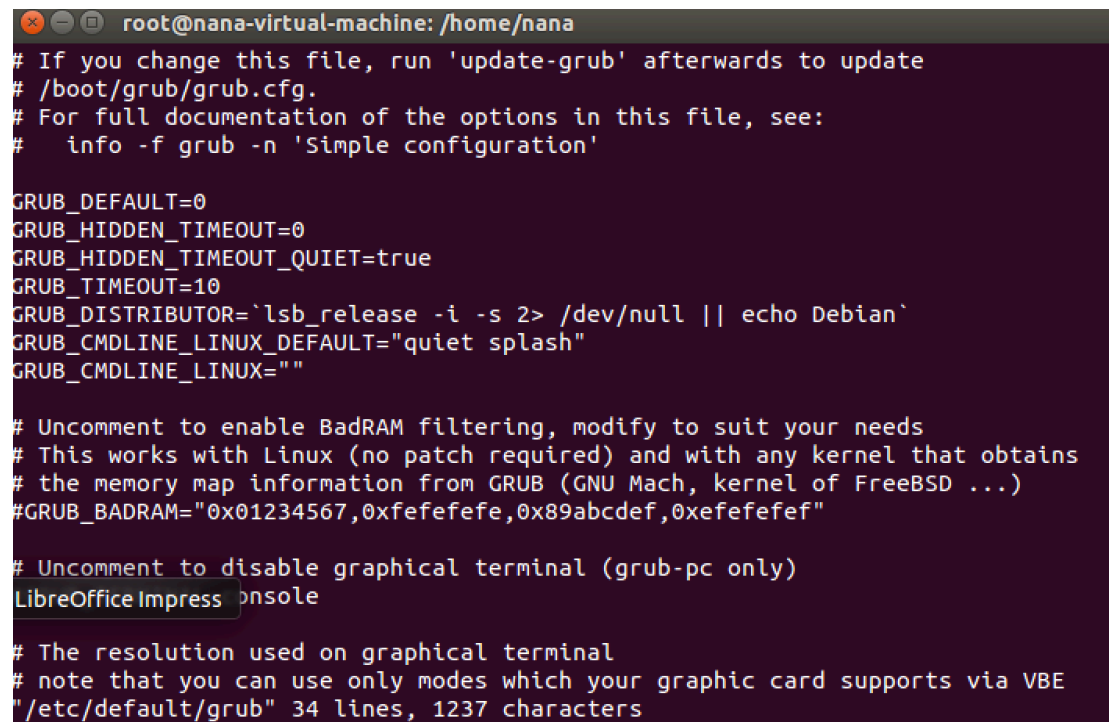
第六步：`make modules_install` 安装模块

安装完后，会在 `/lib/modules/` 目录下生成一个 `linux-2.6.33.3` 目录，下面存放该系统的内核模块。

第七步：make install 安装内核，会自动修改启动文件。（这步可以直接忽略掉）

如果是 ubuntu 系统的话，是没有 conf 的文件，相应的命令行为

```
# vi /etc/default/grub
```



```
root@nana-virtual-machine: /home/nana
# 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=10
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)
LibreOffice Impress console

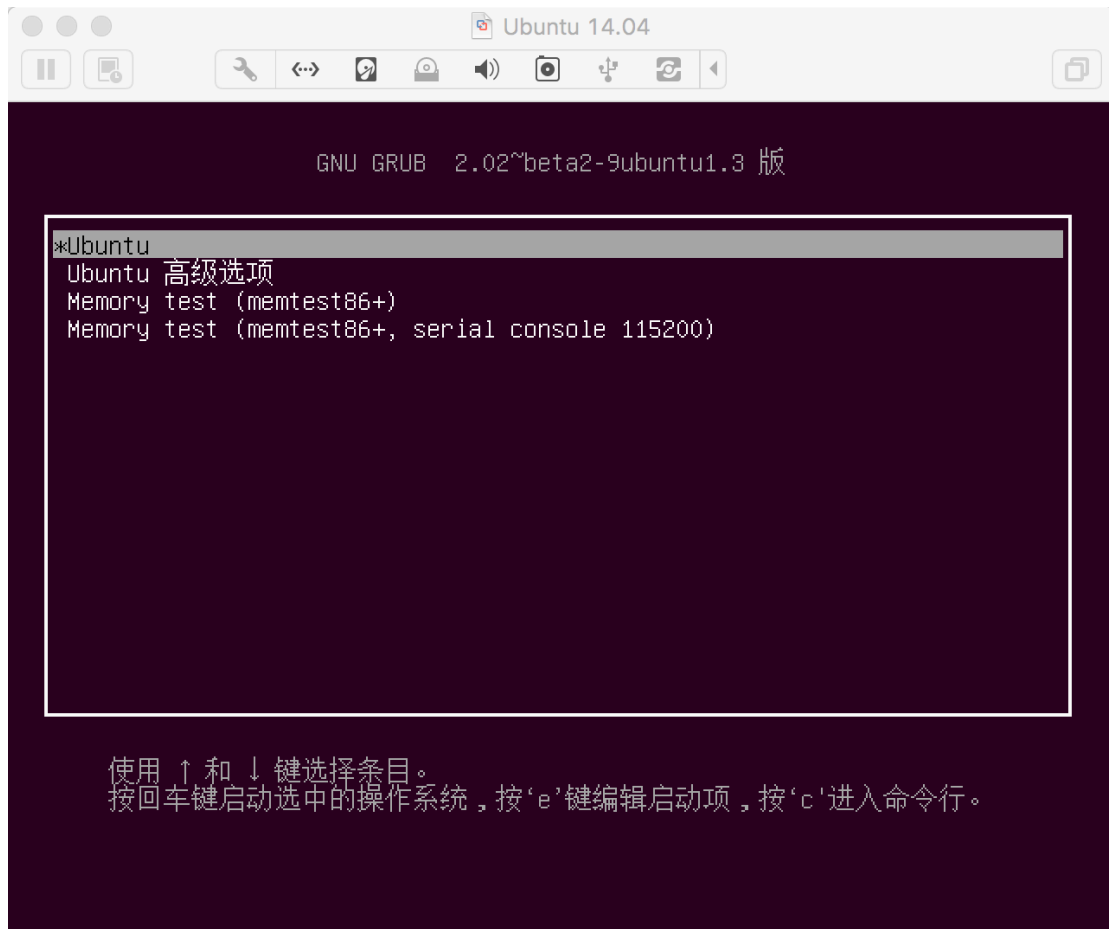
# The resolution used on graphical terminal
# note that you can use only modes which your graphic card supports via VBE
"/etc/default/grub" 34 lines, 1237 characters
```

但修改 default 的值为 1 时，重启依然用的不是新增的内核，2 的时候进入一个无限循环的测试模式，因此建议

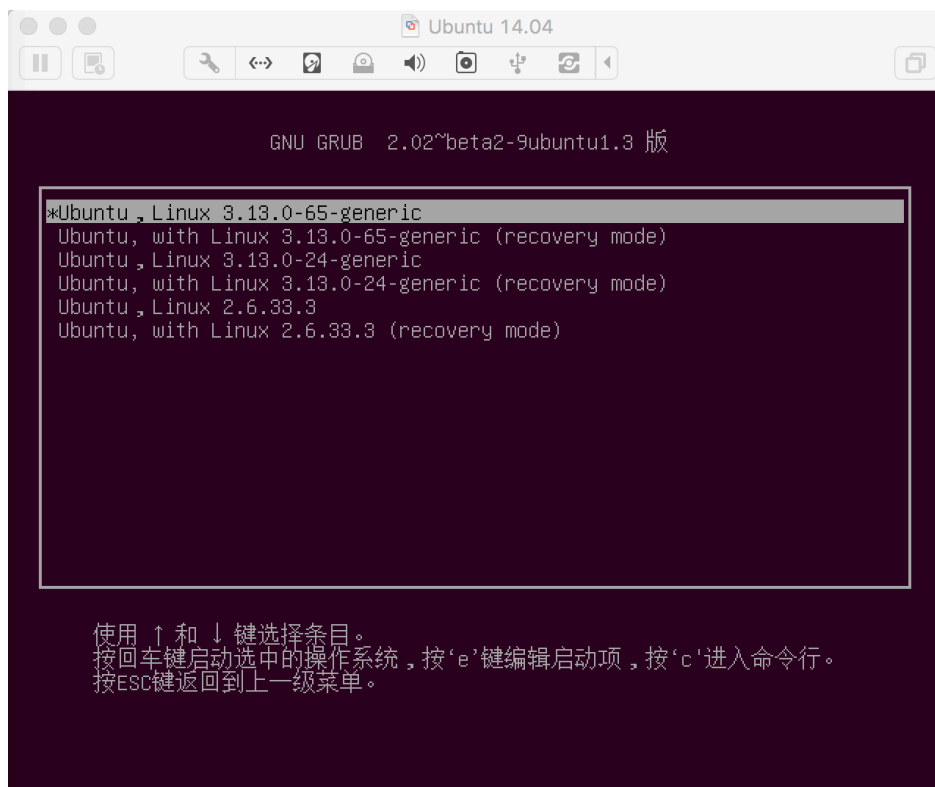
不要在此做修改，直接进入第八步。（不过不同的版本、系统可能会有一些区别）

第八步：进入 linux-2.6.33.3 内核系统

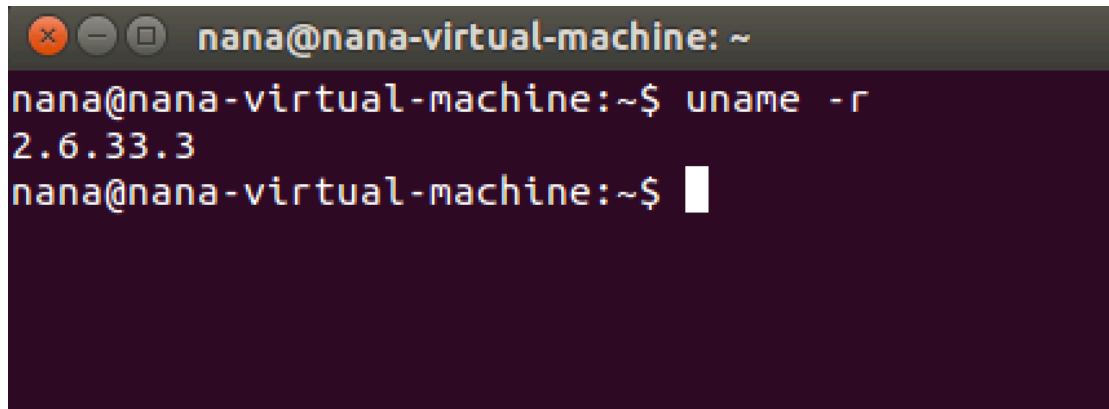
以上已经设置成功，那么下一步就是重启电脑，在重启过程中不断按 shift 键，会出现如图所示的界面：



选择第二个，ubuntu 高级选项，之后进入另一个菜单选择界面：



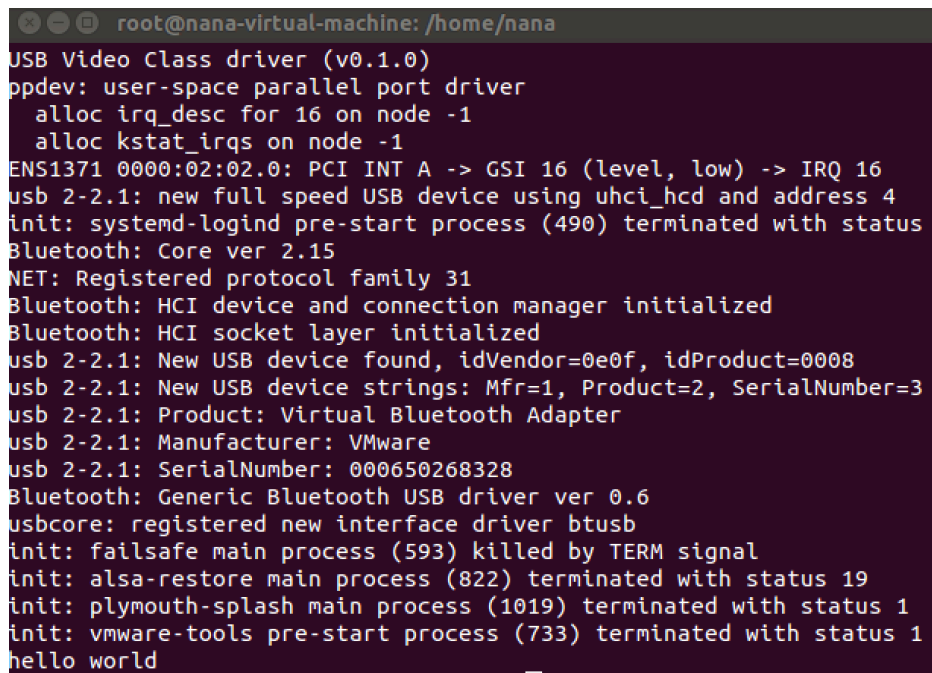
选择 2.6.33.3，进入系统。打开终端，输入 `uname -r` 显示的便是新内核的版本信息,此时为 2.6.33.3 内核版本。



```
nana@nana-virtual-machine: ~  
nana@nana-virtual-machine:~$ uname -r  
2.6.33.3  
nana@nana-virtual-machine:~$
```

第九步：在程序中调用自己添加的系统调用

```
# vi test.c  
#include<stdio.h>  
int main()  
{  
    syscall(223,1);    //223 是第五步中宏定义的入口参数，1 为函数的参数，整型变量  
    number  
    return 0;  
}  
# gcc test.c -o test  
# ./test  
# dmesg -c
```



```
root@nana-virtual-machine: /home/nana  
USB Video Class driver (v0.1.0)  
ppdev: user-space parallel port driver  
  alloc irq_desc for 16 on node -1  
  alloc kstat_irqs on node -1  
ENS1371 0000:02:02.0: PCI INT A -> GSI 16 (level, low) -> IRQ 16  
usb 2-2.1: new full speed USB device using uhci_hcd and address 4  
init: systemd-logind pre-start process (490) terminated with status  
Bluetooth: Core ver 2.15  
NET: Registered protocol family 31  
Bluetooth: HCI device and connection manager initialized  
Bluetooth: HCI socket layer initialized  
usb 2-2.1: New USB device found, idVendor=0e0f, idProduct=0008  
usb 2-2.1: New USB device strings: Mfr=1, Product=2, SerialNumber=3  
usb 2-2.1: Product: Virtual Bluetooth Adapter  
usb 2-2.1: Manufacturer: VMware  
usb 2-2.1: SerialNumber: 000650268328  
Bluetooth: Generic Bluetooth USB driver ver 0.6  
usbcore: registered new interface driver btusb  
init: failsafe main process (593) killed by TERM signal  
init: alsa-restored main process (822) terminated with status 19  
init: plymouth-splash main process (1019) terminated with status 1  
init: vmware-tools pre-start process (733) terminated with status 1  
hello world
```

问题解决方法:

1. 在修改头文件编辑 `syscall_table_32.S` 的时候, 找到 223 的时候, 你会发现有两个同名的 `.long sys_ni_syscall` 一个后面有注释, 一个后面没有, 将没有注释的那个替换为 `.long sys_mycall`。

2. 在安装完成之后之后, 有时配置文件中没有添加新的启动项, 这时需要自己添加。

```
cp /usr/src/linux-2.6.30/arch/i386/boot/bzImage /boot/vmlinuz-2.6.30
mkinitramfs -o initrd.img-linux-2.6.30 2.6.30
cp /usr/src/linux-2.6.30/initrd.img-2.6.30 /boot/initrd.img-2.6.30
```

3. `makeconfig` 时, 程序没有出现内核配置的界面, 而是

```
$ make menuconfig
HOSTLD scripts/kconfig/mconf
scripts/kconfig/mconf.o: In function `show_help':
mconf.c:(.text+0x8a4): undefined reference to `stdscr'
scripts/kconfig/lxdialog/checklist.o: In function `print_arrows':
checklist.c:(.text+0x41): undefined reference to `wmove'
checklist.c:(.text+0x61): undefined reference to `acs_map'
.....
menubox.c:(.text+0x3a9): undefined reference to `wrefresh'
scripts/kconfig/lxdialog/menubox.o: In function `print_buttons':
menubox.c:(.text+0x4ef): undefined reference to `wrefresh'
collect2: ld returned 1 exit status
make[1]: *** [scripts/kconfig/mconf] Error 1
make: *** [menuconfig] Error 2
```

原因在于: 缺少必要的 package, 因此出现编译问题。

解决方法: `sudo apt-get install build-essential libncurses5 libncurses5-dev`

即安装 ncurses 库, 此命令中适合 32 位, 64 位可以去网上下载或者查找一下相关指令。

4. make 中出现的问题汇总

错误 1:

在 `make menuconfig` 配置完之后 (选的默认配置), 然后就 `make` 出现如下错误:

```
gcc: 错误: elf_i386: 没有那个文件或目录
make[2]: *** [arch/x86/vdso/vdso32-int80.so.dbg] 错误 1
make[1]: *** [arch/x86/vdso] 错误 2
make: *** [sub-make] 错误 2
```

原因是 `gcc 4.6` 不再支持 `linker-style` 架构。

修改:

1. 在内核目录 `arch/x86/vdso/Makefile` 中, 大约在 28,29 行 找到 `VDSO_LDFLAGS_vdso.lds`
`= -m elf_x86_64 -Wl,-soname=linux-vdso.so.1 \ -Wl,-z,max-page-size=4096`

`-Wl,-z,common-page-size=4096` 把 `"-m elf_x86_64"` 替换为 `"-m64"`

2 然后再继续找, 大约在 72 行左右, 找到 `VDSO_LDFLAGS_vdso32.lds = -m elf_i386 -Wl,-soname=linux-gate.so.1` 中的 `"-m elf_i386"` 替换为 `"-m32"`

然后继续编译, 就可

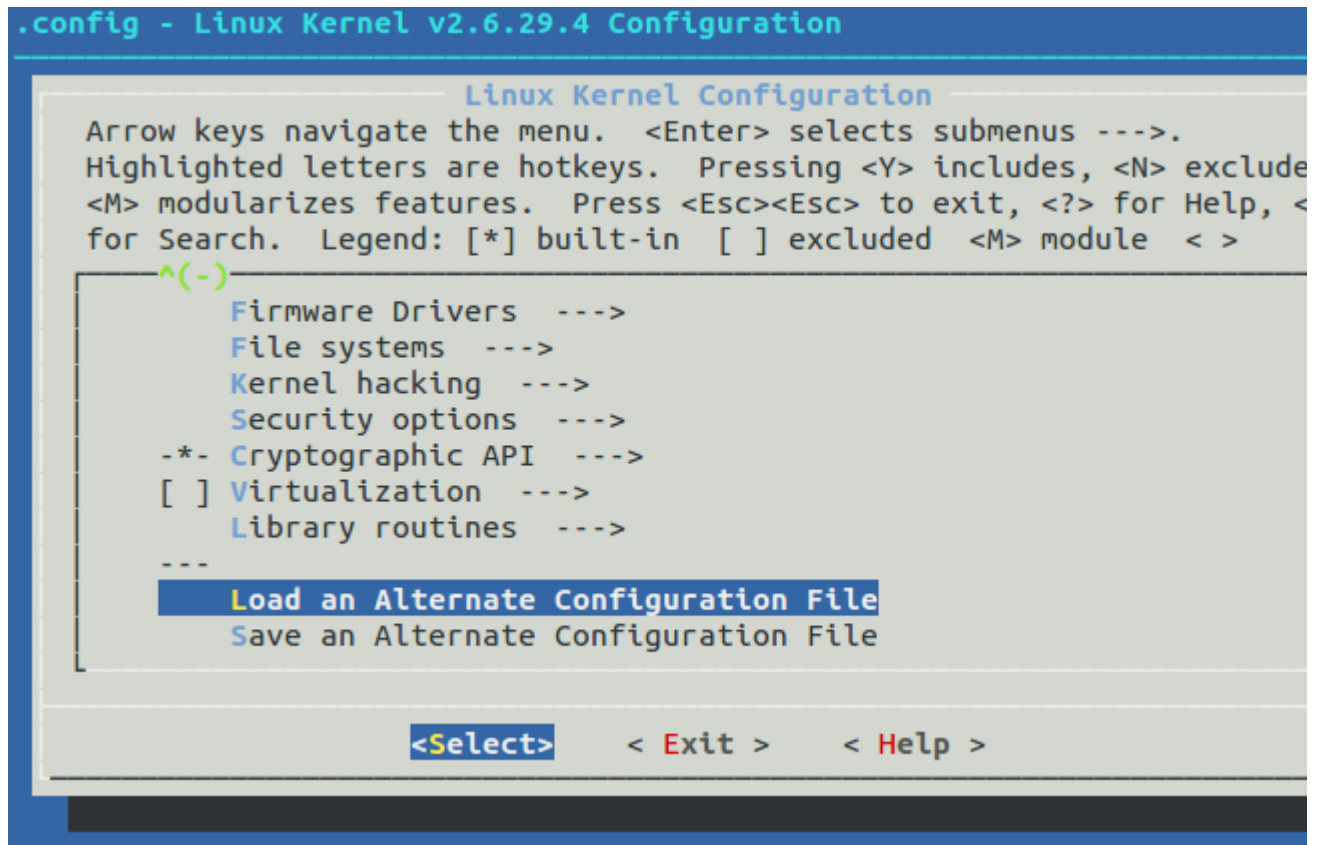
错误 2:

`include/linux/kvm.h:241:9: 错误: 重复的成员 'padding'`

`arch/x86/kvm/svm.c`: 在函数 `'io_interception'` 中:

`arch/x86/kvm/svm.c:1099:30: 警告: 变量 'rep' 被设定但未被使用 [-Wunused-but-set-variable]`

arch/x86/kvm/svm.c:1099:12: 警告： 变量 'down' 被设定但未被使用
 [-Wunused-but-set-variable]
 make[2]: *** [arch/x86/kvm/svm.o] 错误 1
 make[1]: *** [arch/x86/kvm] 错误 2
 make[1]:正在离开目录 `/home/cody/kernel/linux-2.6.29.4'
 make: *** [debian/stamp/build/kernel] 错误 2
 找到 Virtualization 选择,把勾选去掉.如下例如所示.



错误 3:

重复的定义'codec',解决办法是找到 include/sound/soc-dai.h,可以看到里面定义了一个 struct 和一个 union 类型的 codec, 注释掉 struct 的那个。

错误 4:

[GCC-4.6.3 编译 linux2.6.32.12 内核出现“重复的成员‘page’”错误的解决方法](#)

使用 gcc4.6.3 编译 linux2.6.32.12 内核出现错误如下:

```
In file included from drivers/net/igbvf/ethtool.c:36:0:
drivers/net/igbvf/igbvf.h: 在文件作用域:
drivers/net/igbvf/igbvf.h:128:15: 错误: 重复的成员'page'
make[4]: *** [drivers/net/igbvf/ethtool.o] 错误 1
make[3]: *** [drivers/net/igbvf] 错误 2
make[2]: *** [drivers/net] 错误 2
make[1]: *** [drivers] 错误 2
make[1]:正在离开目录 `/usr/src/linux-2.6.32.2'
make: *** [debian/stamp/build/kernel] 错误 2
```


解决方法:

1. 根据 linux 社区的建议, 此错误是由于 gcc 版本与内核版本的冲突导致的。他们的建议是更换新版本的内核, 但是某些特殊条件下, 我们不能更换内核版本, 于是我们修改内核代码适应当前的编译器。
2. 按照错误的指示, 错误的代码是在 drivers/net/igbvf/igbvf.h 文件的第 128 行。
3. 打开文件, 看 128 行, 代码为: struct page *page;再往上看, 第 123 行, 也有 struct page *page 这行代码, 这个结构定义在内部的一个结构体中。就是他的名字与 128 行的重复了, 而 4.6.3 的编译器对不支持这种方式的定义, 我们修改 128 行的代码为 struct page *pagep; 保存退出;

错误 4:

```
arch/x86/include/asm/ptrace.h:146:13: note: previous declaration of 'syscal
make[2]: *** [arch/x86/kernel/ptrace.o] Error 1
make[1]: *** [arch/x86/kernel] Error 2
make: *** [arch/x86] Error 2
```

解决方法为:

```
#vi /usr/src/linux-2.6.33.3/arch/x86/include/asm/ptrace.h
```

在程序的 130 行 和 142 行分别进行增加、删除操作, 具体操作如图: 绿色为要加进入的代码, 紫色为要减掉的代码。

```
--- linux-2.6.32.59/arch/x86/include/asm/ptrace.h
+++ fix_ptrace.o_compile_error/arch/x86/include/asm/ptrace.h
@@ -130,6 +130,7 @@
#ifdef __KERNEL__

#include <linux/init.h>
+#include <linux/linkage.h>

struct cpuinfo_x86;
struct task_struct;
@@ -142,8 +143,8 @@
        int error_code, int si_code);
void signal_fault(struct pt_regs *regs, void __user *frame, char *where);

-extern long syscall_trace_enter(struct pt_regs *);
-extern void syscall_trace_leave(struct pt_regs *);
+extern asmregparm long syscall_trace_enter(struct pt_regs *);
+extern asmregparm void syscall_trace_leave(struct pt_regs *);

static inline unsigned long regs_return_value(struct pt_regs *regs)
{
```

错误 5: 错误 gcc 不识别 -m 参数

```
gcc: error: unrecognized command line option '-m'
gcc: error: elf_i386: No such file or directory
make[1]: *** [arch/x86/vdso/vdso32-int80.so.dbg] Error 1
```

原因在于可能 gcc 版本过高, 所以我改成了 gcc-4.6, 因为 4.5 装不成, 之后继续 make。

```
sudo apt-get install gcc-4.5
cd /usr/bin
sudo ln -s -f gcc-4.5 gcc
```

错误 6:错误图忘了保存,所以没贴上来,如果你的错误不是上述,可能之前你在增加系统调用的时候,就是增加函数、增加头文件等步骤时出现了代码编写的错误,请务必检查仔细。

以上就是我在增加系统调用以及帮别人改时,7、8 个小时遇到的所有问题以及个人的解决方法,如有错误或者其他问题,欢迎小伙伴们和我交流。

对 make menuconfig 和内核配置感兴趣的同学,我贴两个链接,里边有详细的关于 config 的解释和关于如何精简配置内核的步骤。

<http://blog.csdn.net/xuyuefei1988/article/details/8635539> make menuconfig 详解

<http://www.dangkai.com/ArticlePage/Article66492.htm> Linux 内核裁剪步骤之 make menuconfig

软件 1302 王玲娜