**实验2.4 系统调用 实验报告**

1. **实验目的**

向现有Linux内核加入一个新的系统调用从而在内核空间中实现对用户空间的读写。例如，设计并实现一个新的内核函数sys\_shiwenhan\_time( )，此函数通过一个引用参数的调用返回当前系统时间，功能上基本与gettimeofday( )相同。也可以实现具有其它功能的系统调用。

1. **实验内容**

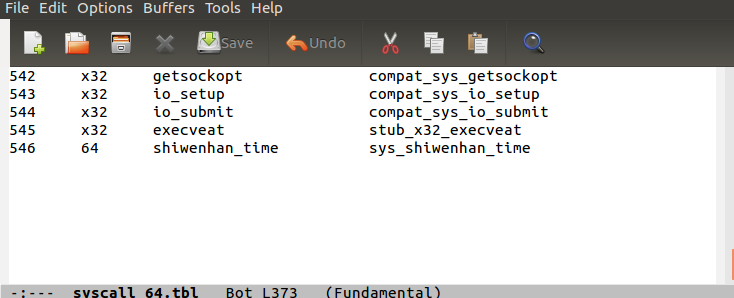
设计并构建一个系统调用，这个系统调用可以返回当前的系统时间。

1. **实验设计原理**

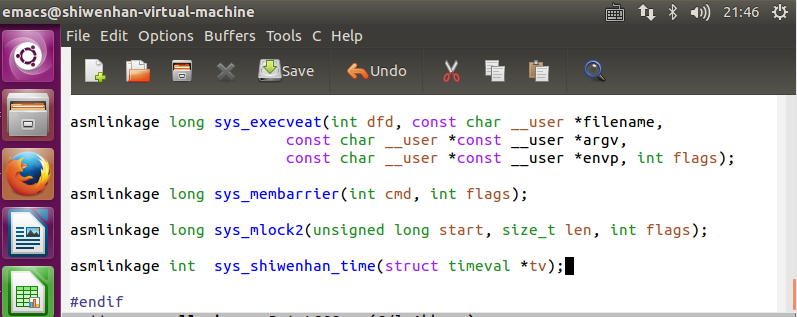
通过向内核文件中添加相应的调用代码，并通过修改系统调用表来创建自己的系统调用的调用号。重新编译新的Linux内核后，重建引导信息，最后编写测试程序测试读取系统时间的正确性。

1. **实验步骤**
2. **添加系统调用号**

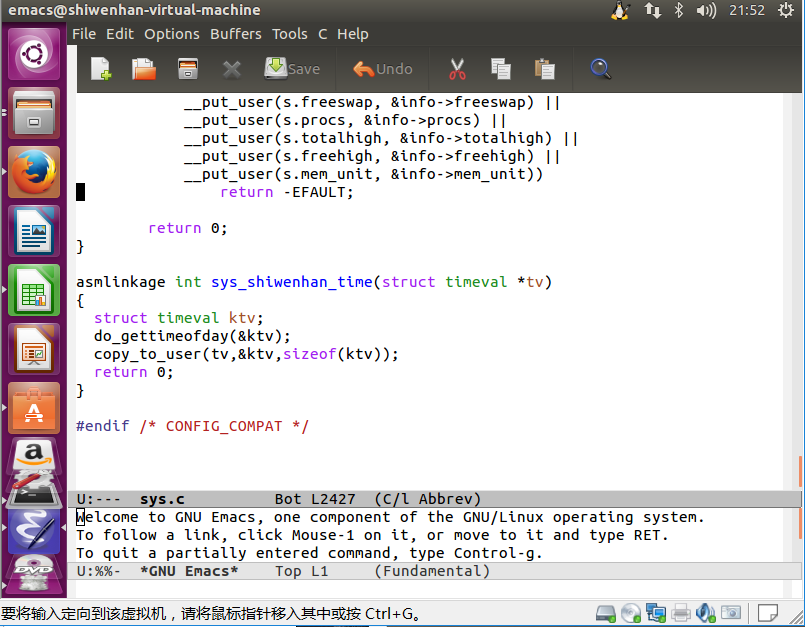
因为接下来的操作需要修改系统文件，先用sudo -a命令进入root权限。

我们用emacs ./arch/x86/entry/syscalls/syscall\_64.tbl打开文件。在最后一行添加shiwenhan\_time这个系统调用，调用名称为sys\_shiwenhan\_time。

1. **声明系统调用函数原型**

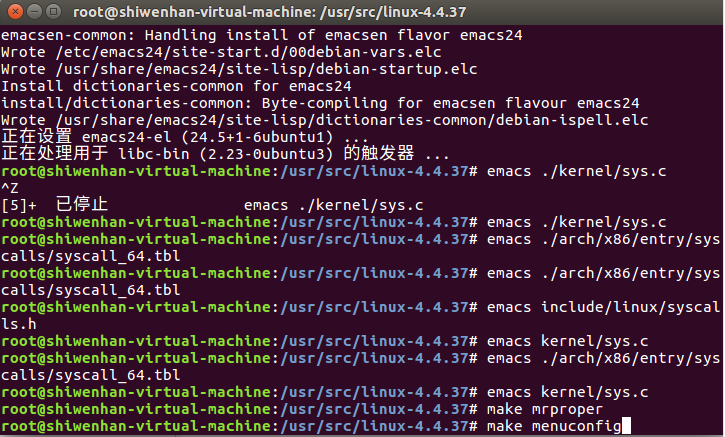
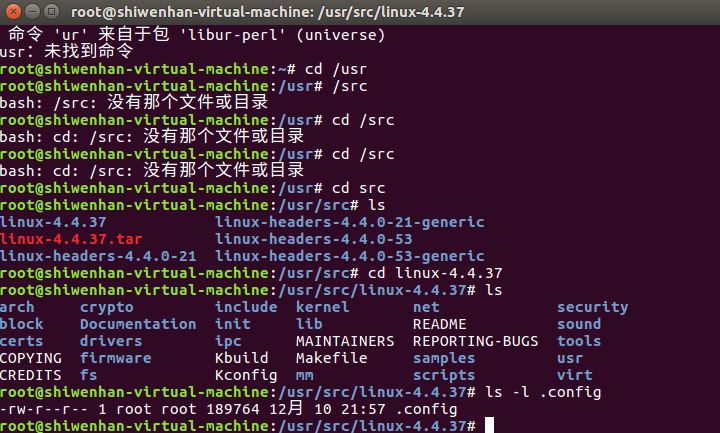
键入命令 emacs include/linux/syscalls.h，添加sys\_shiwenhan\_time的函数原型。

1. **添加系统调用函数的定义**

对实验指导书中的函数做了一些简化，使其成为新的系统调用函数的内容。

至此，我们已经添加了系统调用号、添加了系统调用原型和定义，在重新编译内核之前，使用make mrproper命令清理残留的config和.o文件。

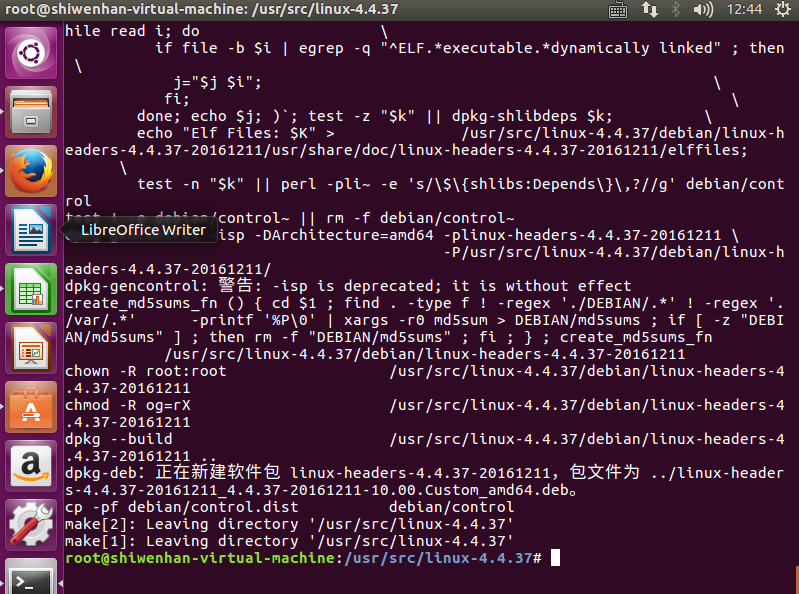
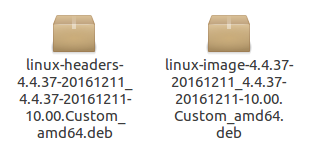
1. **配置config**

接上图，使用命令 make menuconfig调出配置选项，将虚拟机中原Linux系统中的config拷贝至新内核的根目录下。

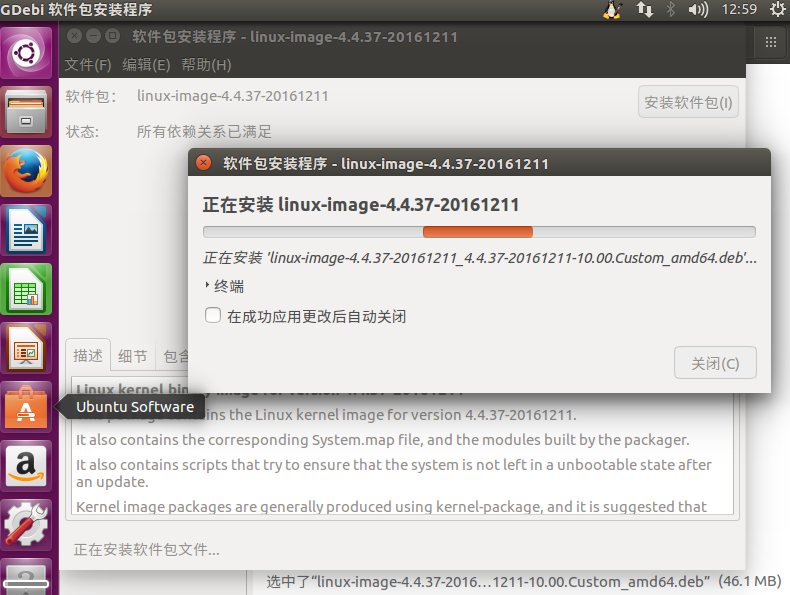
1. **重新编译内核**

由于直接make的编译时间过长，因此采用debian的方法来简化内核的编译。

# make-kpkg --initrd --append-to-version -20161211 kernel-image kernel-headers得到新的内核deb文件和内核头。

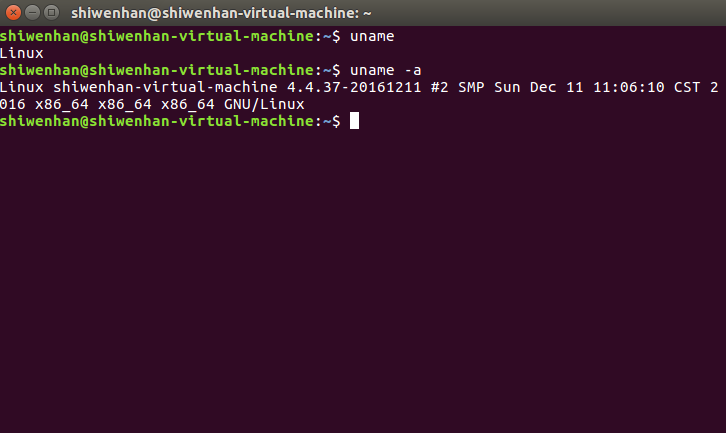
 最后得到了如下两个关键文件，分别是内核image和headers。

安装这两个文件即可完成这次内核的编译。



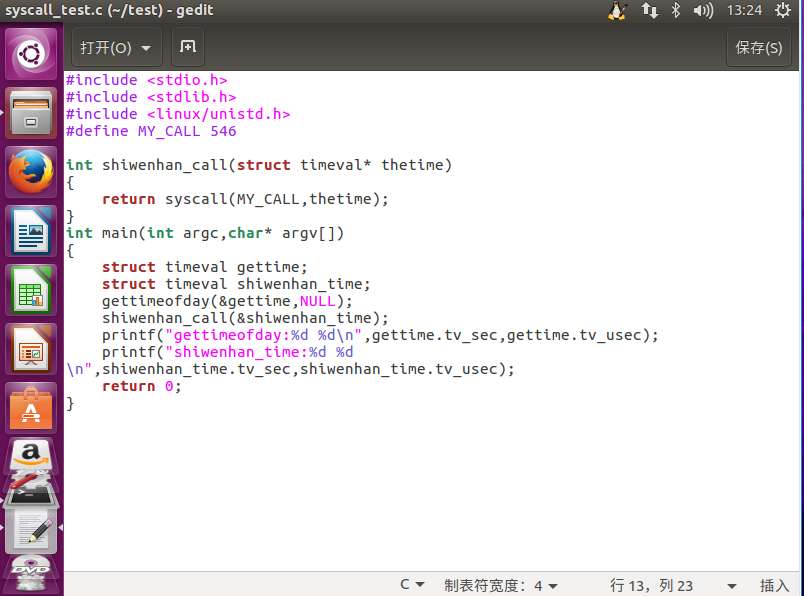
1. **引导到grub**

由于使用了gdebi工具，因此grub被自动更新。

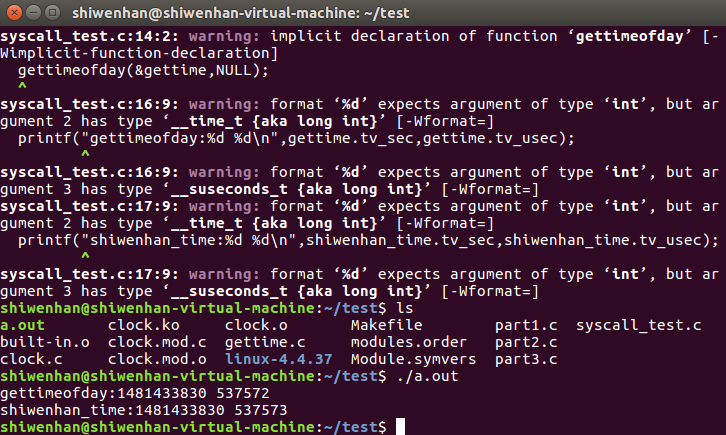
直接reboot进入新的内核。

1. **实验结果及分析**

也即测试这个系统调用在新内核的可行性。

在新内核中输入测试程序如下：

我们在上面载入的系统调用编号是546，显然我们在这里应该定义一个宏，这个宏与我的调用编号相对应。这个程序打印两个时间，第一个为gettimeofday的标准时间，第二个为我们自己设置的，与其功能相同的系统调用函数，它们打印出来的时间应该是相差无几的。

用gcc编译并运行得到如下结果。

最后两行显示了，原生函数的时间与新添加系统调用的时间相差无几，这与我们之间的分析是吻合的。表明添加系统调用的整个过程是正确的。

1. **人员任务分配**

本次实验由小组成员

史文翰

No.2014211218

Cla.2014211304

独自完成。