**浙江大学实验报告**

课程名称： 操作系统 实验类型： 综合型

实验项目名称： 添加系统调用

学生姓名： 张佳瑶 学号： 3170103240

电子邮件地址： 1531077171@qq.com

实验日期： 2019 年 12 月 13 日

1. 实验环境

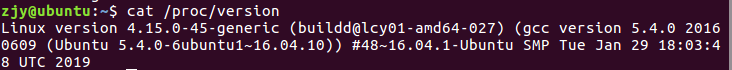
处理器：Intel® Core™ i7-6700HQ CPU @ 2.60GHz

Windows10

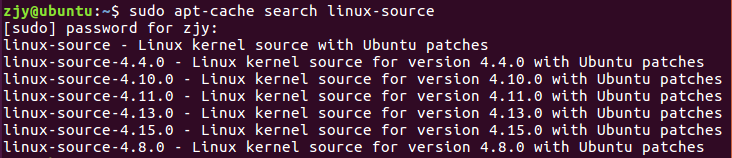
Linux version 5.0.0-29-generic (buildd@lgw01-amd64-039) (gcc version 7.4.0 (Ubuntu 7.4.0-1ubuntu1~18.04.1))

1. 实验内容和结果及分析

查看当前Ubuntu版本：



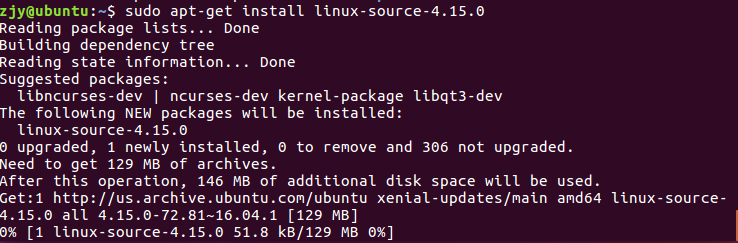
查看内核源代码：



当前Ubuntu版本为Ubuntu16.04.10，内核版本为4.15.0。

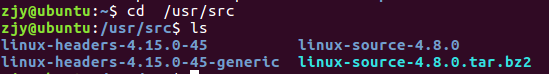
准备编译内核，下载4.15.0版本的内核源码：

sudo apt-get install linux-source-4.15.0



下载好的内核源代码默认放置在/usr/src：

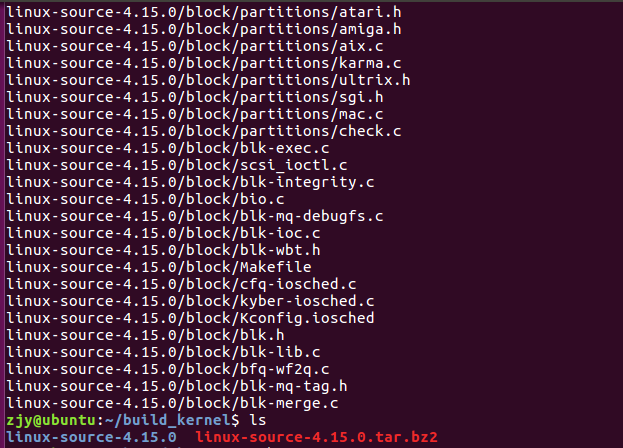
cd /usr/src  
ls



将其拷贝到新建的文件夹build\_kernel下，并解压：

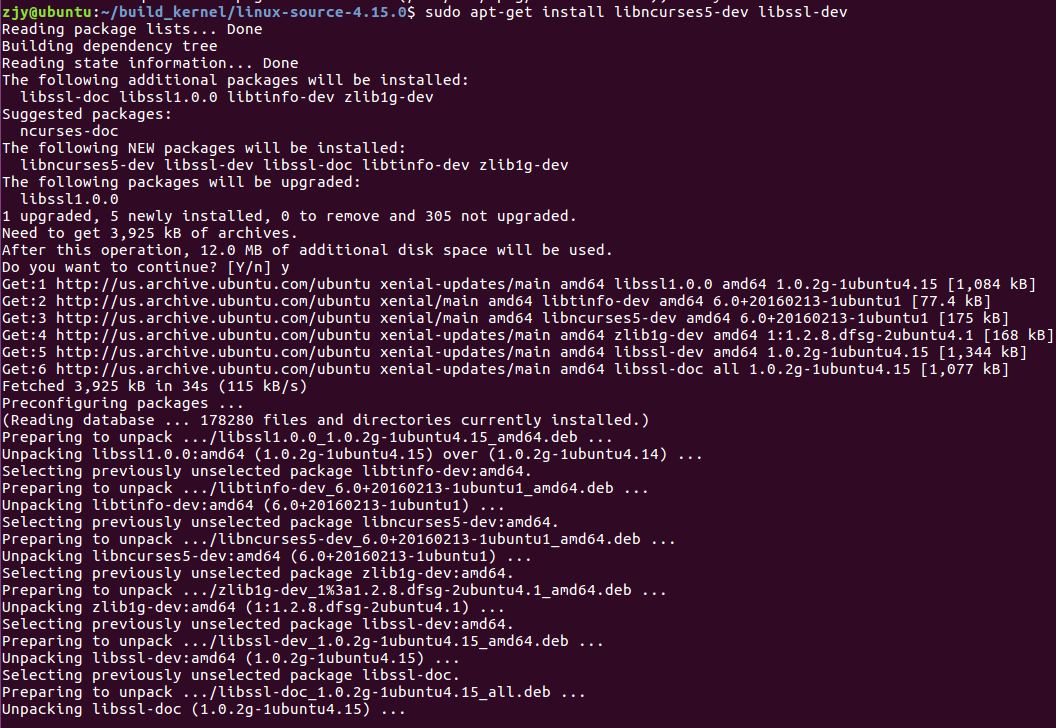
cd  
mkdir build\_kernel  
cd build\_kernel  
cp /usr/src/linux-source-4.15.0.tar.bz2 .  
tar jxvf linux-source-4.15.0.tar.bz2

结果为：



下载库：

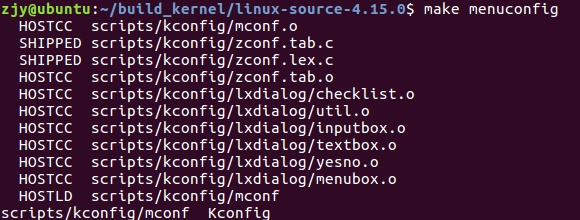
sudo apt-get install libncurses5-dev libssl-dev



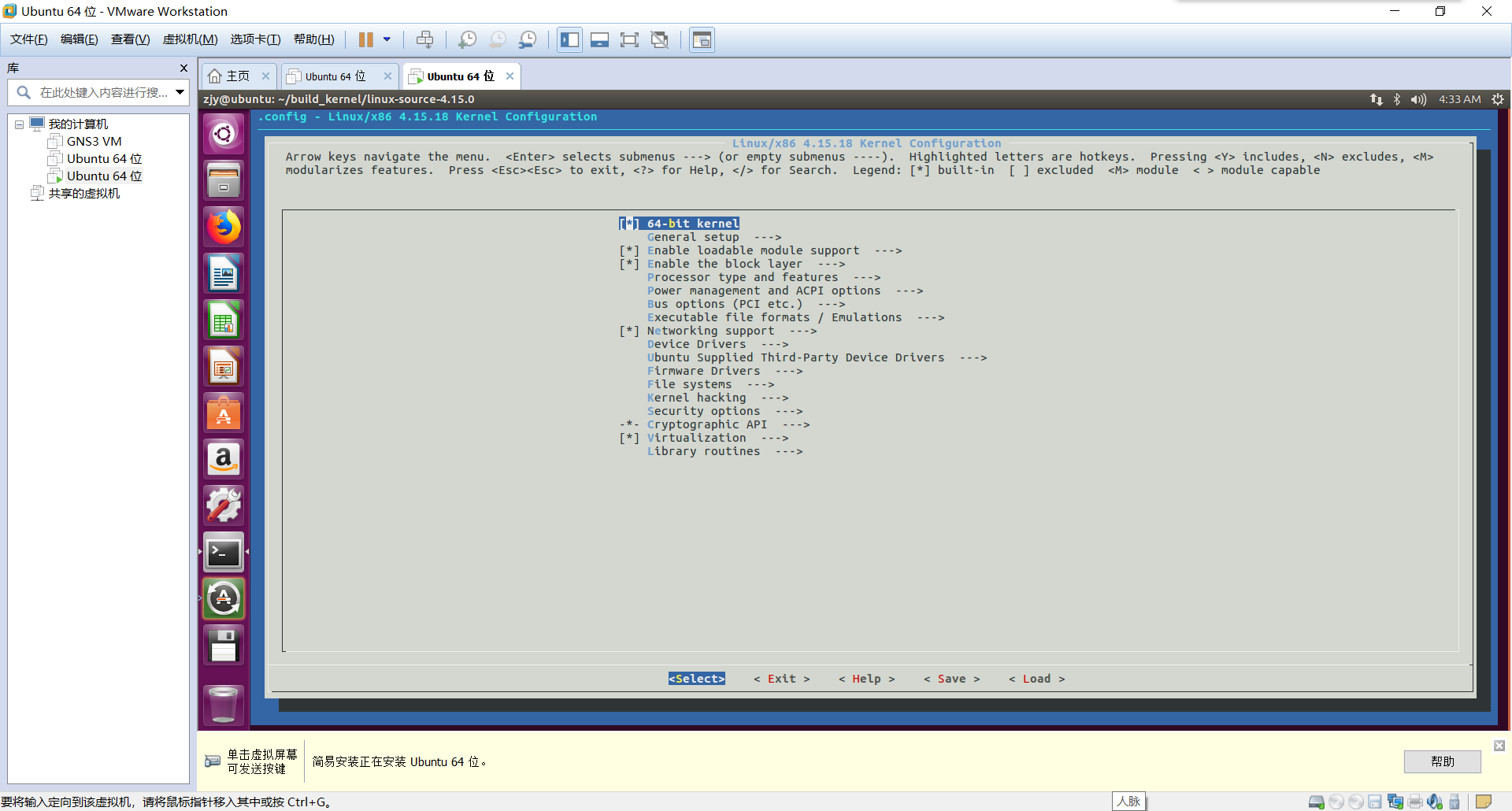
配置内核：

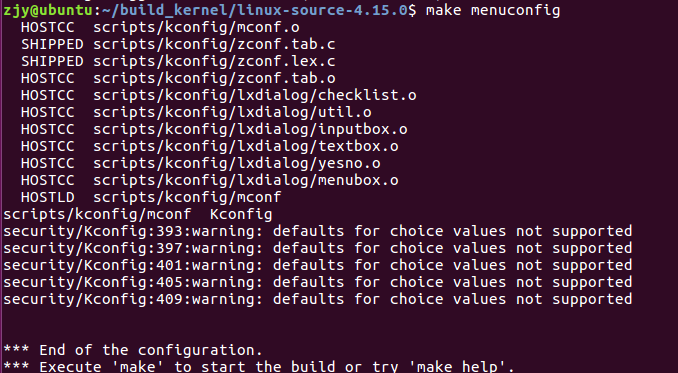
zjy@ubuntu:~/build\_kernel$ cp /usr/src/linux-headers-4.15.0-45-generic/.config ./linux-source-4.15.0  
zjy@ubuntu:~/build\_kernel$ cd linux-source-4.15.0  
zjy@ubuntu:~/build\_kernel/linux-source-4.15.0$ make menuconfig





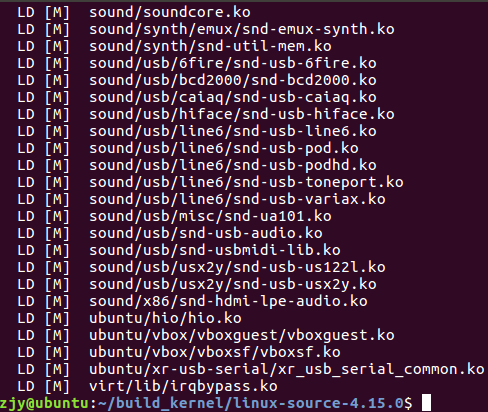
make menuconfig菜单出现后依次选择load，OK，save，OK，exit，exit：

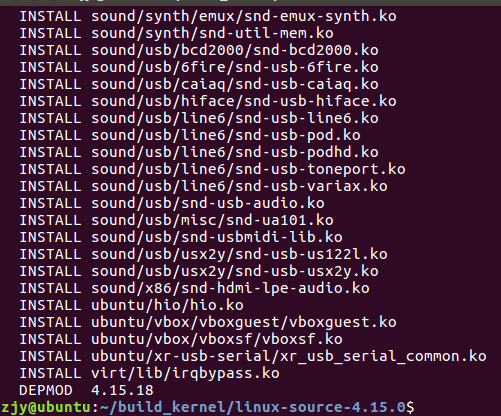


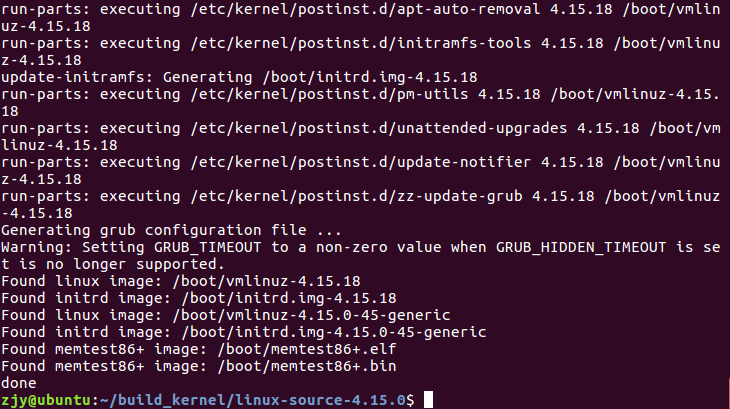


内核编译准备结束，开始第一次编译内核：

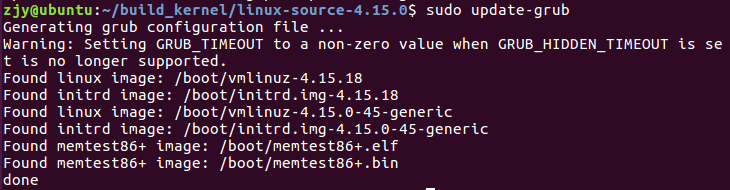
make -j8

  
sudo make modules\_install -j 8

  
sudo make install -j 8

  
sudo update-initramfs -c -k 4.15.18

  
sudo update-grub



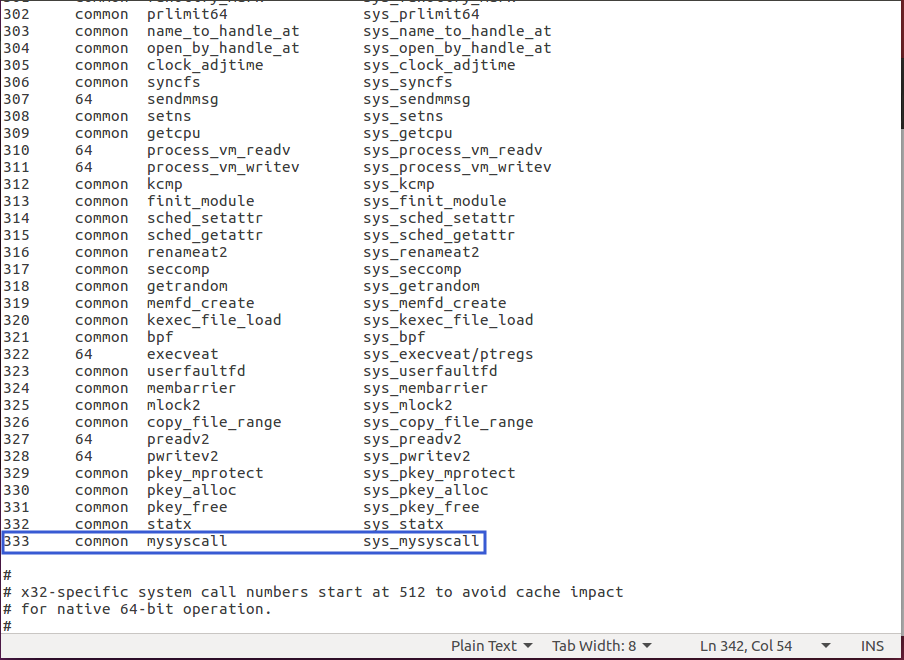
重启，验证内核是否编译成功



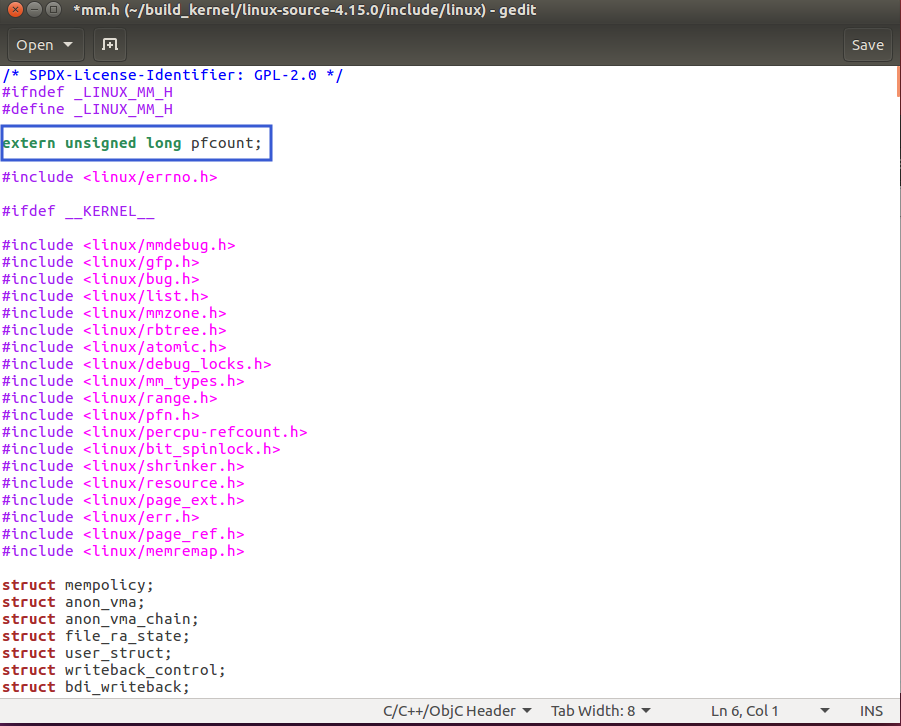
内核版本已经更新为4.15.18，证明内核编译成功。

开始准备添加系统调用。

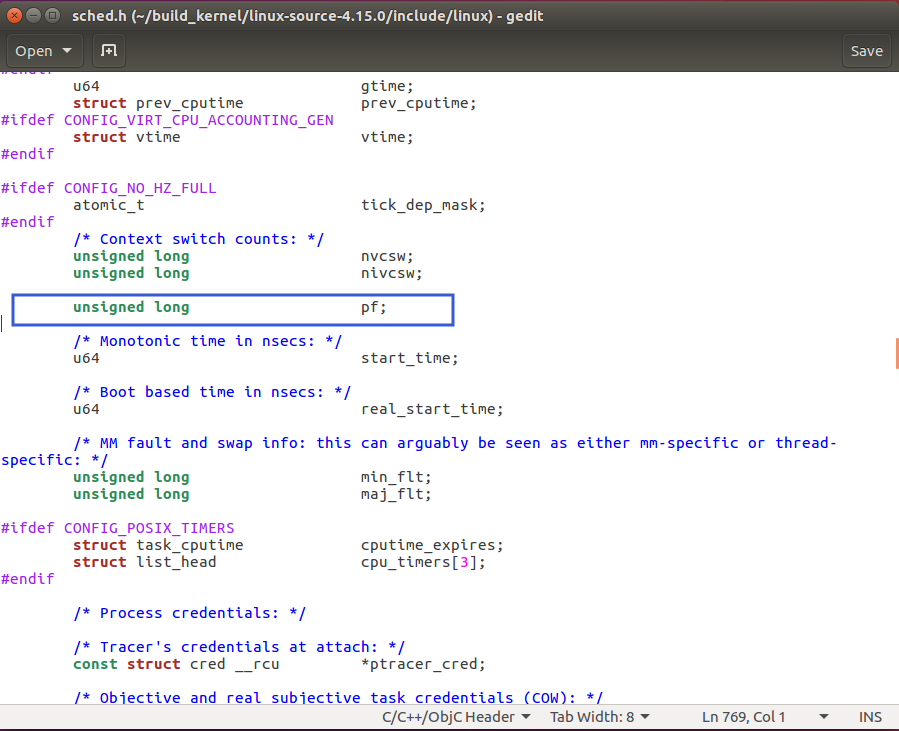
以下统一在下载的内核源码~/build*kernel/linux-source-4.15.0文件夹下，在arch/x86/entry/syscalls/syscall*64.tbl添加333 common mysyscall sys\_mysyscall。



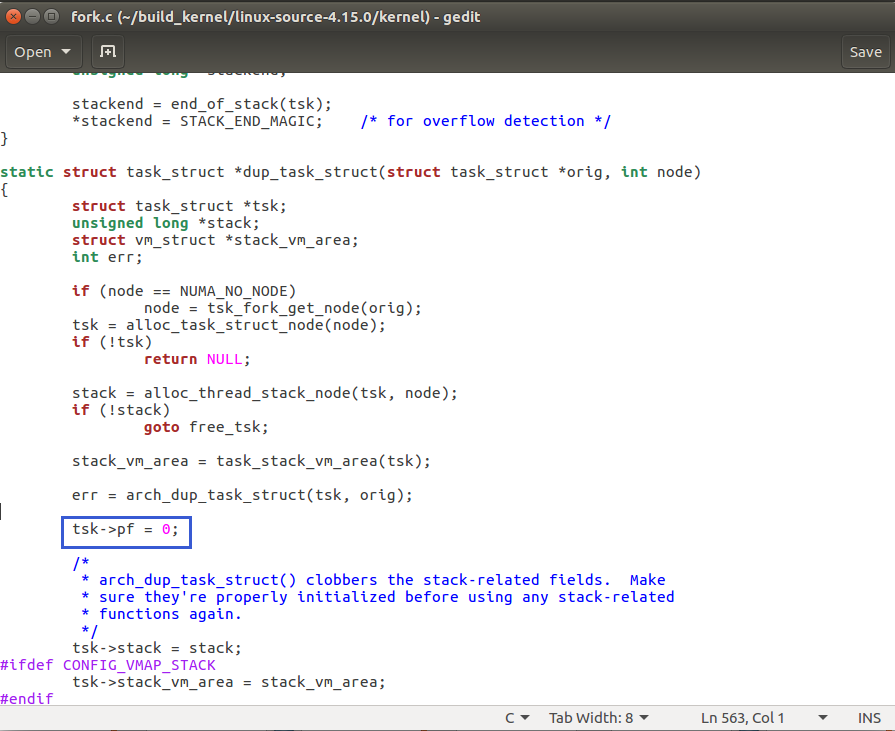
在include/linux/mm.h中添加extern unsigned pfcount long pfcount;。



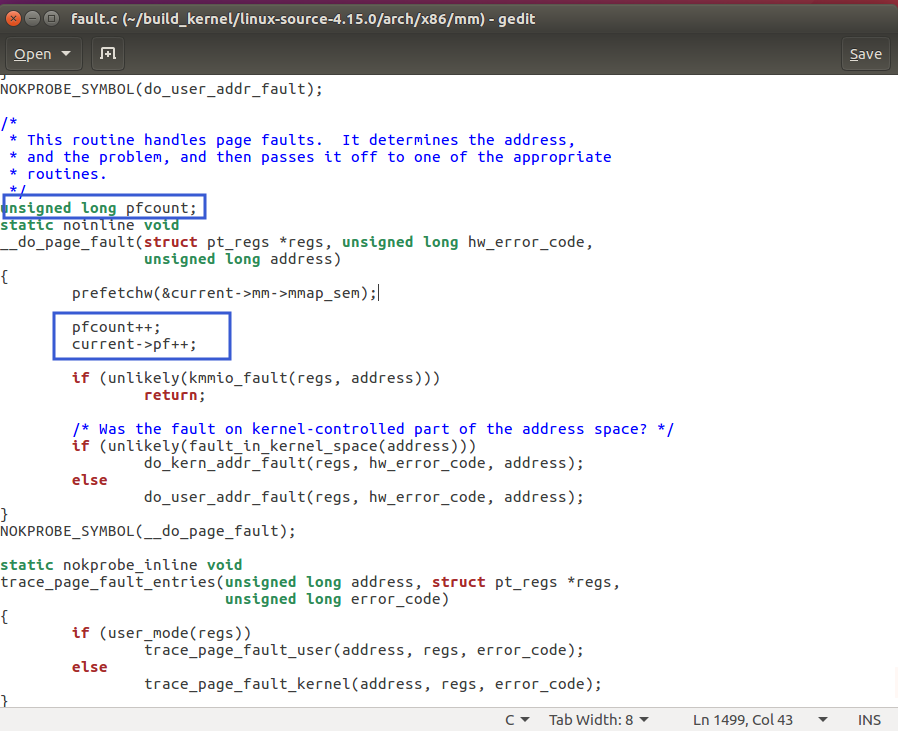
在include/linux/sched.h中添加unsigned long pf;



在kernel/fork.c中添加tsk->pf = 0;

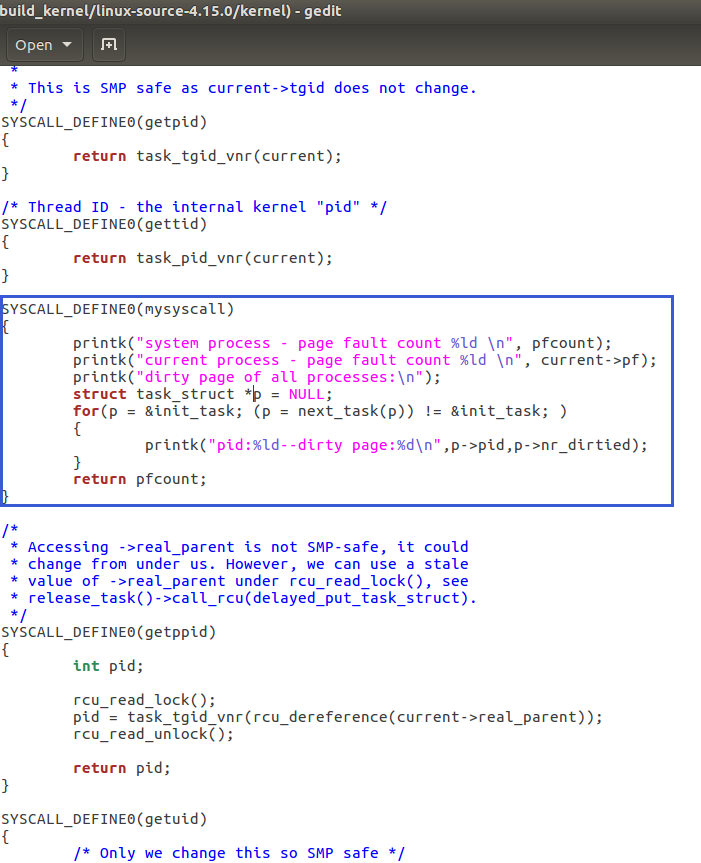


在arch/x86/mm/fault.c中添加unsigned long pfcount;和pfcount++;current->pf++;



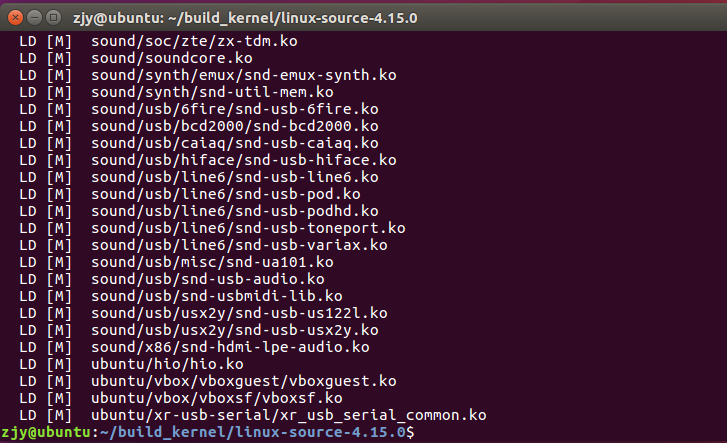
在kernel/sys.c中添加

SYSCALL\_DEFINE0(mysyscall)



然后重新编译内核：

make -j8  
sudo make modules\_install -j 8  
sudo make install -j 8  
sudo update-initramfs -c -k 4.8.15  
sudo update-grub



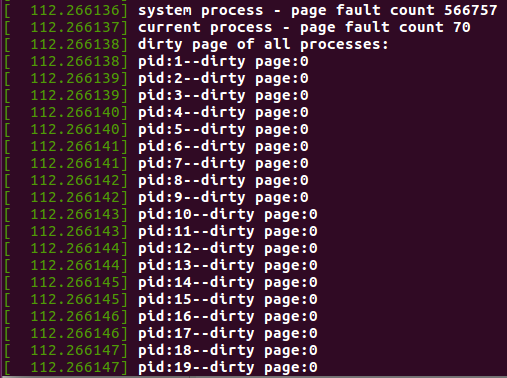
重启。

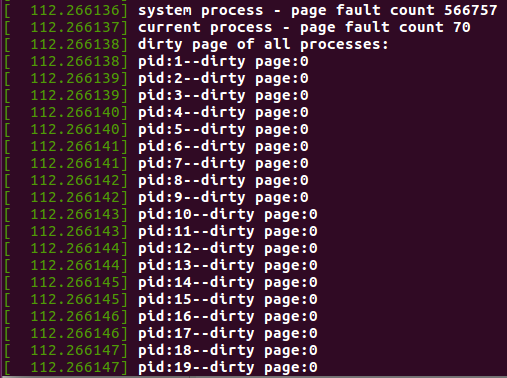
编写用户态程序并编译。

#include <linux/unistd.h>   
#include <sys/syscall.h>   
#define \_\_NR\_mysyscall 333  
  
int main()  
{  
 pfcount = syscall(\_\_NR\_mysyscall);

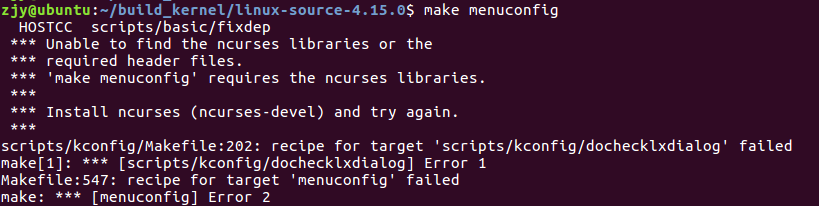
}

编译用户态程序并运行，然后输入dmesg验证：

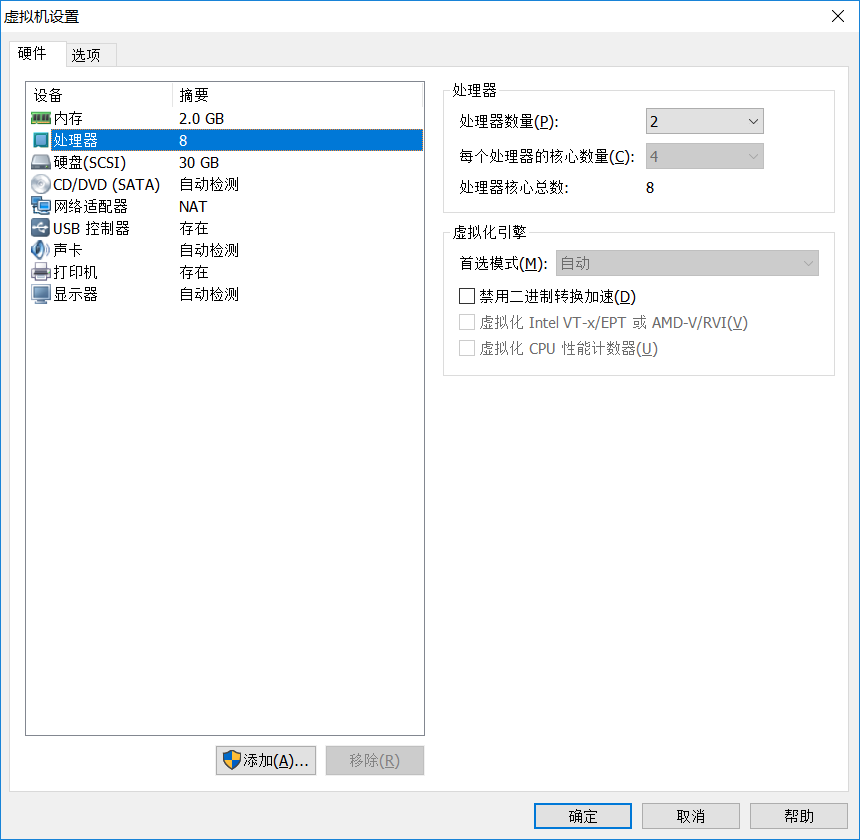




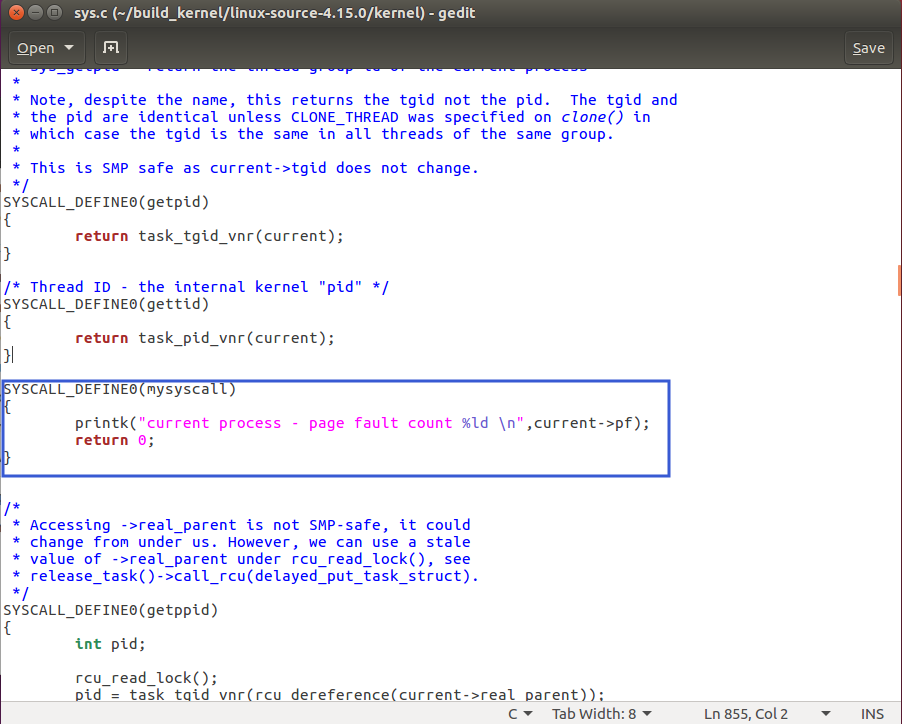
1. 讨论、心得（20分）
2. 一开始没有下载库，果然出现了问题：



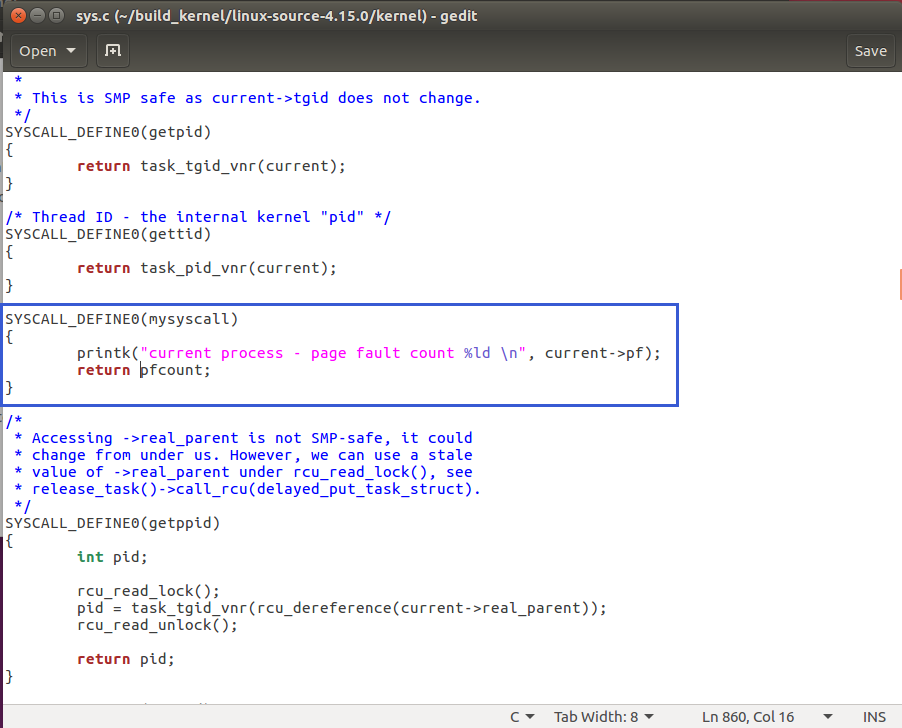
根据实验指示下载了ncurses-devel后可以正常编译。

1. 一开始虚拟机的配置是1核1线程，编译地极慢，请教同学后发现可以根据自身电脑配置更改虚拟机配置，加快编译速度，于是就更改为2核4线程，以j8来编译。  
   
2. 第一次完成二次编译内核后，在命令行输入sudo reboot重启，重启后虚拟机就提示禁用CPU无法开机，找不到原因，只好重新做了一遍实验。使用VMware重启，一切正常。
3. 发现问题重要统计系统缺页数，因此sys.c应该把return 0改为return pfcount.

原先是：

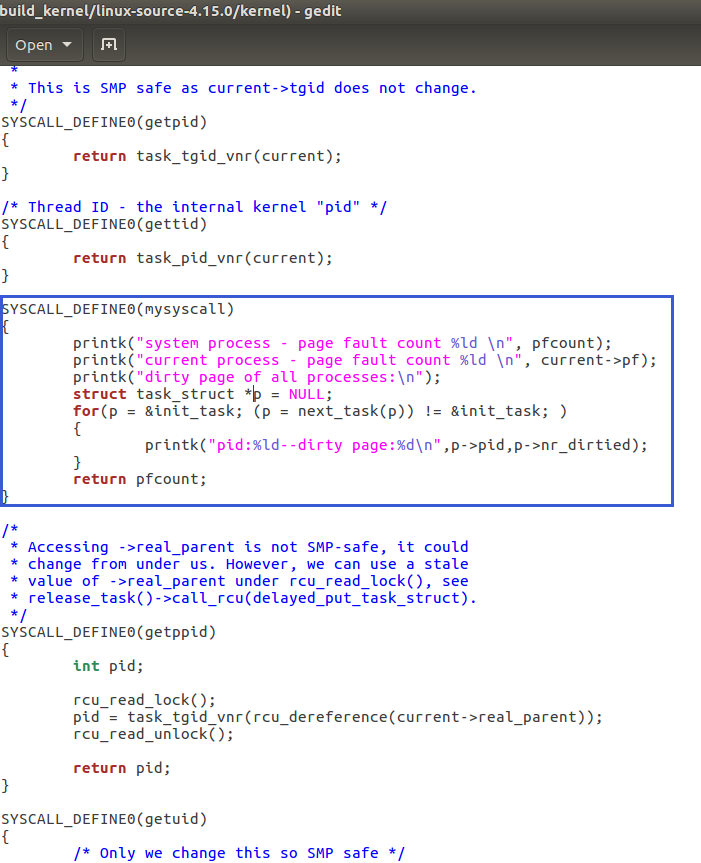


应该是：



而且应该修改一下用户态程序。

发现少看了一个要求，实际上应该是：



1. 因为中间做错了几次，就重新编译了好几次，发现后面编译的很快。
2. 问题
3. 多次运行test程序，每次运行test后记录下系统缺页次数和当前进程缺页次数，给出这些数据。test程序打印的缺页次数是否就是操作系统原理上的缺页次数？有什么区别？

在仅打印系统缺页数和当前进程缺页数上查看多次运行的结果：

第一次运行：





第二次运行：





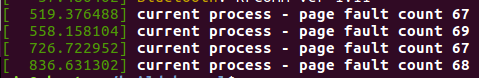
第三次运行：





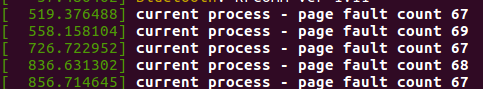
第四次运行：





第五次运行：





|  |  |  |
| --- | --- | --- |
| 运行次数 | 系统缺页数 | 当前进程缺页数 |
| 1 | 539764 | 67 |
| 2 | 583003 | 69 |
| 3 | 584373 | 67 |
| 4 | 586191 | 68 |
| 5 | 586634 | 67 |

可以看出每次运行，系统缺页次数都会明显增长，而当前进程的缺页次数则基本不变。

1. 除了通过修改内核来添加一个系统调用外，还有其他的添加或修改一个系统调用的方法吗？如果有，请论述。

除了内核编译法来添加系统调用之外，也可以通过module进行内核添加来添加系统调用。这种方法是 采用系统调用拦截的一种方式，改变某一个系统调用号对应的服务程序，变为自己编写的程序，从而相 当于添加了系统调用。

1. 对于一个操作系统而言，你认为修改系统调用的方法安全吗？请发表你的观点。

我认为对于一个操作系统而言，修改系统调用并不够安全。因为当修改系统调用时，有可能会对原来系 统调用表中的其他调用进行修改，从而使得其名称发生变化或者是缺少对应编号的系统调用，可能会给 调用正常系统调用接口的准确性和安全性造成威胁。