**实验2.4 系统调用**

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mycall()，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。

2、实验内容与步骤

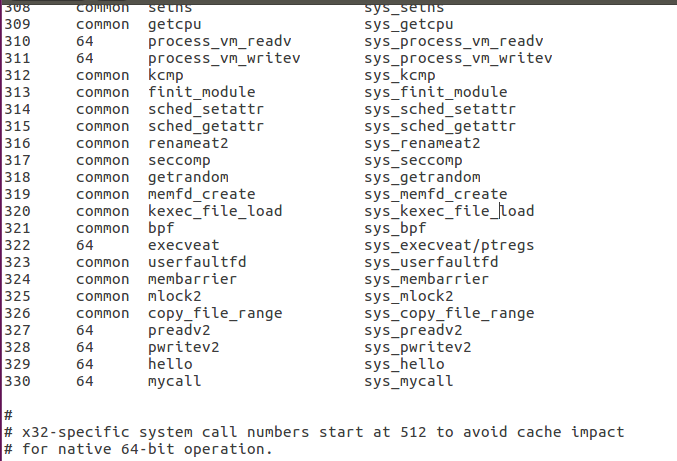
（1）**下载Linux内核源代码**

在<https://www.kernel.org/>上下载了linux-4.8.15.tar.xz放在了home文件夹中，并解压

tar -xvf linux-4.8.15.tar.xz

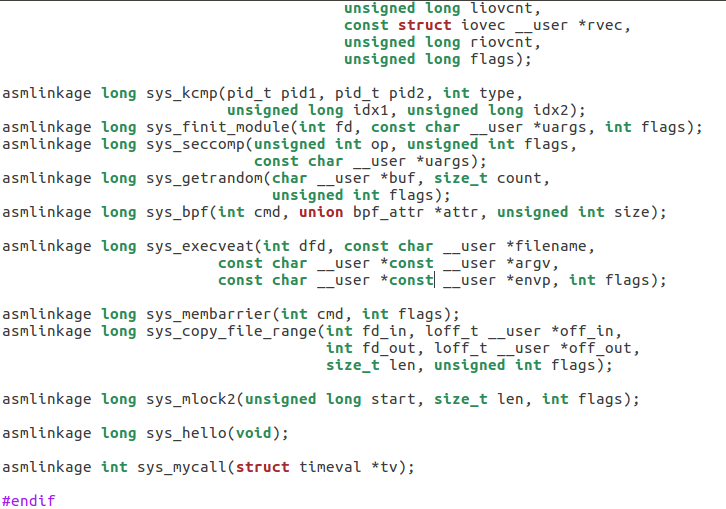
（2）**添加系统调用号**

在/home/linux-4.8.15/arch/x86/entry/syscalls/syscall\_64.tbl添加新的系统调用号



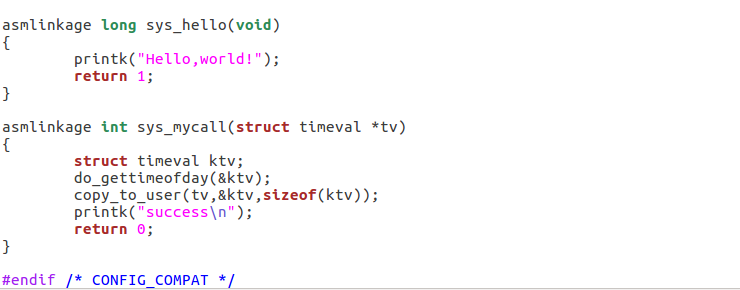
**（3）声明系统调用函数模型**

在/home/linux-4.8.15/include/linux/syscalls.h添加函数声明



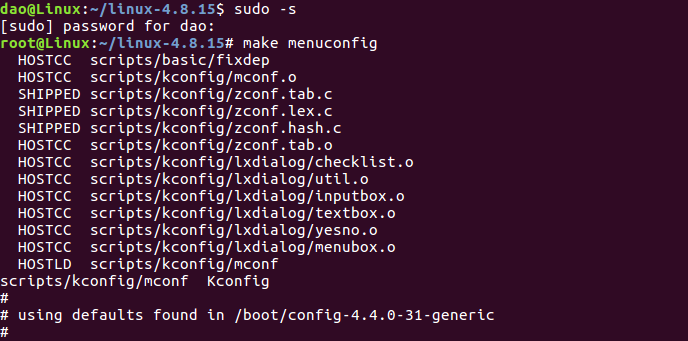
**（4）添加系统调用函数的定义**

在/home/linux-4.8.15/kernel/sys.c中添加相应的调用代码

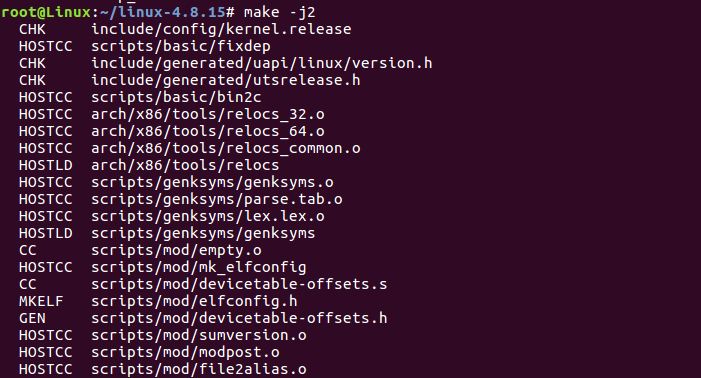


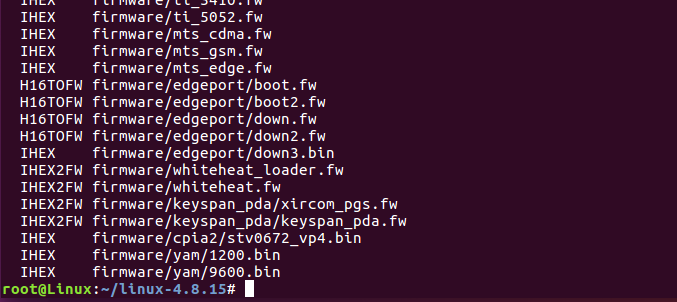
（5）**编译内核**

a、输入命令sudo -s转为root用户，再输入命令make menuconfig 直接选择save，然后退出即可，生成.config



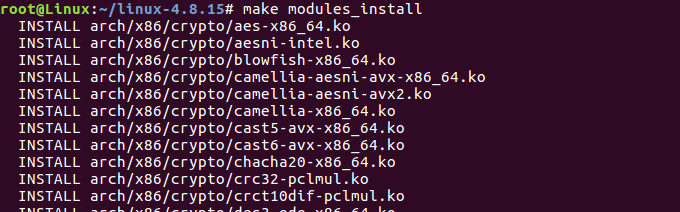
b、配置完毕后，输入命令make -j2使用双线程编译内核

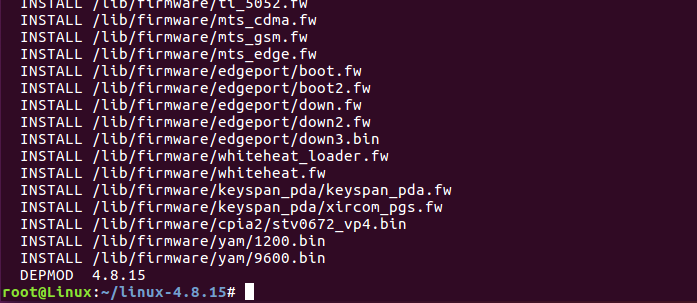




内核编译完毕

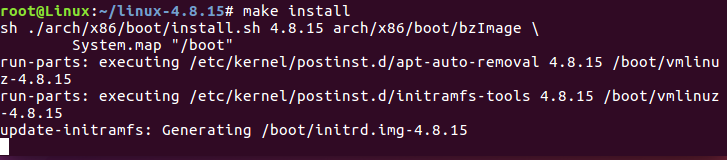
c、输入指令make modules\_install 安装模块

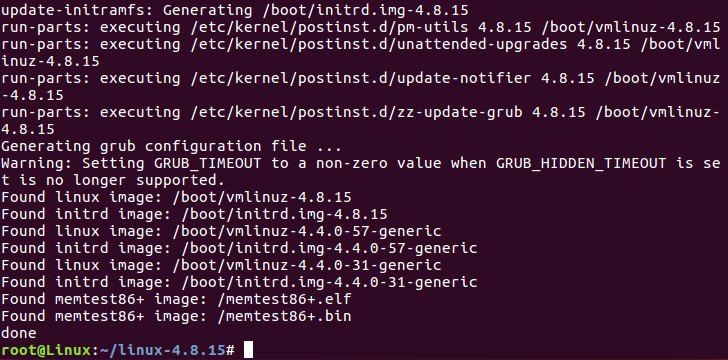




模块安装完毕

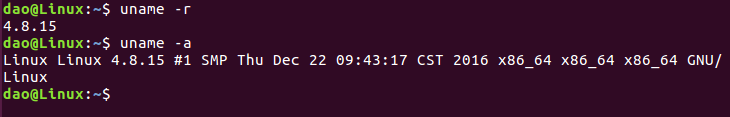
d、输入指令make install 安装内核





安装完毕

e、重启后检查内核版本，变为刚编译的内核了！



（6）测试系统调用

sys\_hello和sys\_mycall

源代码如下：

**//sys\_hello.c**

#include <stdio.h>

#include <linux/kernel.h>

#include <sys/syscall.h>

#include <linux/unistd.h>

int main()

{

long int amma = syscall(329);

printf("System call sys\_hello returned %ld\n",amma);

return 0;

}

**//sys\_mycall.c**

#include <stdio.h>

#include <linux/kernel.h>

#include <sys/syscall.h>

#include <linux/time.h>

#include <linux/unistd.h>

int main()

{

struct timeval gettime;

struct timeval mycalltime;

gettimeofday(&gettime,NULL);

long int amma = syscall(330,&mycalltime);

printf("gettimeofday: %d %d\n",gettime.tv\_sec,gettime.tv\_usec);

printf("mycalltime: %d %d\n",mycalltime.tv\_sec,mycalltime.tv\_usec);

printf("System call sys\_mycall returned %ld\n",amma);

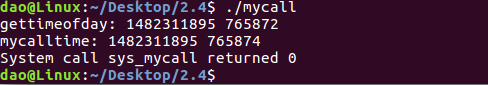
return 0;

}

测试程序运行截图：

a、sys\_mycall

syscall的参数为系统调用号。通过系统调用打印出了系统时间，功能上基本与gettimeofday( )相同。



b、sys\_hello

syscall的参数为系统调用号。该语句会将此输出作为系统的日志，通过dmesg命令查看。由于日志并没有立即输出，我又执行了一次测试程序，日志便将两次的Hello,world!都输出了，并且发现还输出了前面sys\_call系统调用中的打印语句，打印出了success。

