**osLab10\_2013747\_张怡桢**

**《操作系统》课第10次实验报告**

|  |  |
| --- | --- |
| 学院: | 软件学院 |
| 姓名: | 张怡桢 |
| 学号: | 2013747 |
| 邮箱: | 2662765987@qq.com |
| 时间: | 11/24/2022 |

1. **开篇感言**

老师教程X86，奈何吾之arm，摸石头过河成长。

2. **实验题目**

一个新系统调用实现列举进程和文件内核拷贝等多项功能，并返回结果到用户程序。

3. **实验目标**

1. Add a new system call with arguments into the linux kernel
2. The new system call will return all processes information to user mode
3. 实现内核中文件拷贝 (见pdf文档)

4. **原理方法**

3.1 Linux 内核概述：

Linux 内核是 Linux 操作系统（OS）的主要组件，也是计算机硬件与其进程之间的核心接口。它负责两者之间的通信，还要尽可能高效地管理资源。

之所以称为内核，是因为它在操作系统中就像果实硬壳中的种子一样，并且控制着硬件（无论是电话、笔记本电脑、服务器，还是任何其他类型的计算机）的所有主要功能。

|  |  |
| --- | --- |
|  |  |

5. **具体步骤**

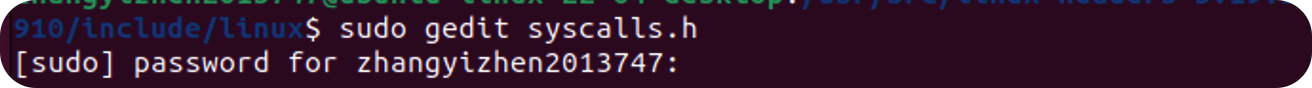
**Step1 (Linux kernel 5.19)**

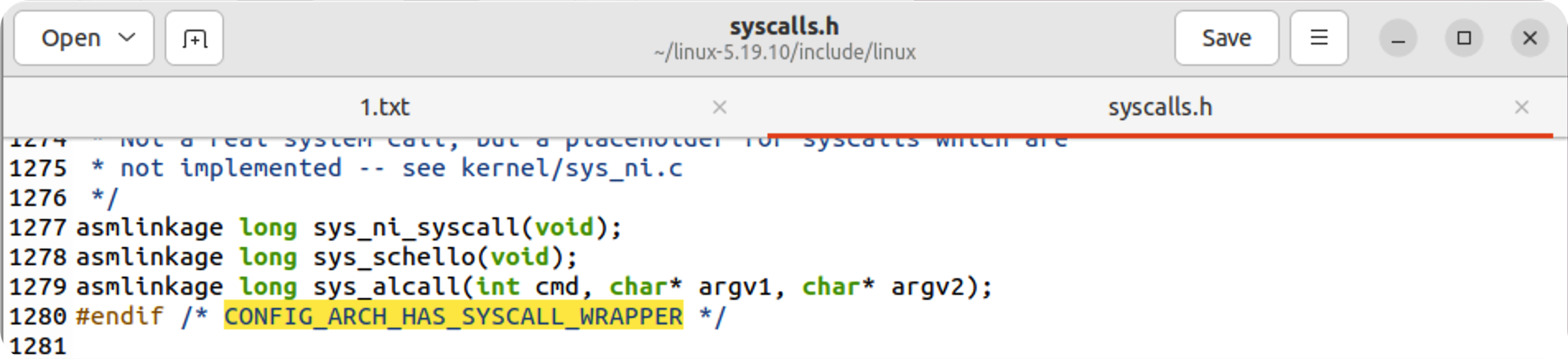
include/linux/syscalls.h

在文件(No. 1279)

#endif /\* CONFIG\_ARCH\_HAS\_SYSCALL\_WRAPPER \*/之前，添加一行:

asmlinkage long sys\_alcall(int cmd, char\* argv1, char\* argv2);





**Step2 (Linux kernel 5.19)**

kernel/sys.c

在文件SYSCALL\_DEFINE0(gettid)函数之后（No. 959），添加如下行:



|  |
| --- |
| SYSCALL\_DEFINE3(alcall,int,cmd,char\*,argv1 ,char\*,argv2){  if(cmd==1){  struct task\_struct \*p;  char buf[2048];    int num = 0;  p = &init\_task;  while (((p = next\_task(p)) != &init\_task) && (num++ < 20)){  char temp[100];  snprintf(temp,2048,"pid:%d\tcomm:%s\tstate:%ld\n",p->pid,p->comm,p->stats);  strcat(buf,temp);  }  copy\_to\_user(argv1,buf,2048);   }  else if(cmd ==2){  char buf[2048];  loff\_t offset = 0;  size\_t len = 0;  loff\_t ret = 0;  char sour[100];  char targ[100];  copy\_from\_user(sour,argv1,100);  copy\_from\_user(targ,argv2,100);  struct file \*file1,\*file2;  file1 = filp\_open(sour,O\_RDONLY,0644);  file2 = filp\_open(targ,O\_WRONLY|O\_CREAT,0644);  while (1)  {  ret = offset;  len = kernel\_read(file1,buf,(size\_t)2048,&offset);  if(len == 0){  break;  }  kernel\_write(file2,buf,len,&ret);  strcpy(buf,"");  }  filp\_close(file1,NULL);  filp\_close(file2,NULL);  }  return 0; } |

**Step3 (Linux kernel 5.19)**

由于我的电脑不是X86架构的，所以我按着老师给的教程的思路修改了arm/tools下的syscall.tbl，但是编译之后并不能生效，在最后的输出依然失败，询问同学，查询网站，找到解决办法【<https://blog.csdn.net/m0_51683653/article/details/124133370>】

include/uapi/asm-generic/unistd.h

修改及添加系统调用号（注意系统调用号不能随意加，只能一次加1）

在892行加入

#define \_\_NR\_alcall 452

\_\_SYSCALL(\_\_NR\_alcall, sys\_alcall)

并顺次把下一个syscalls序号加1改成453即可。

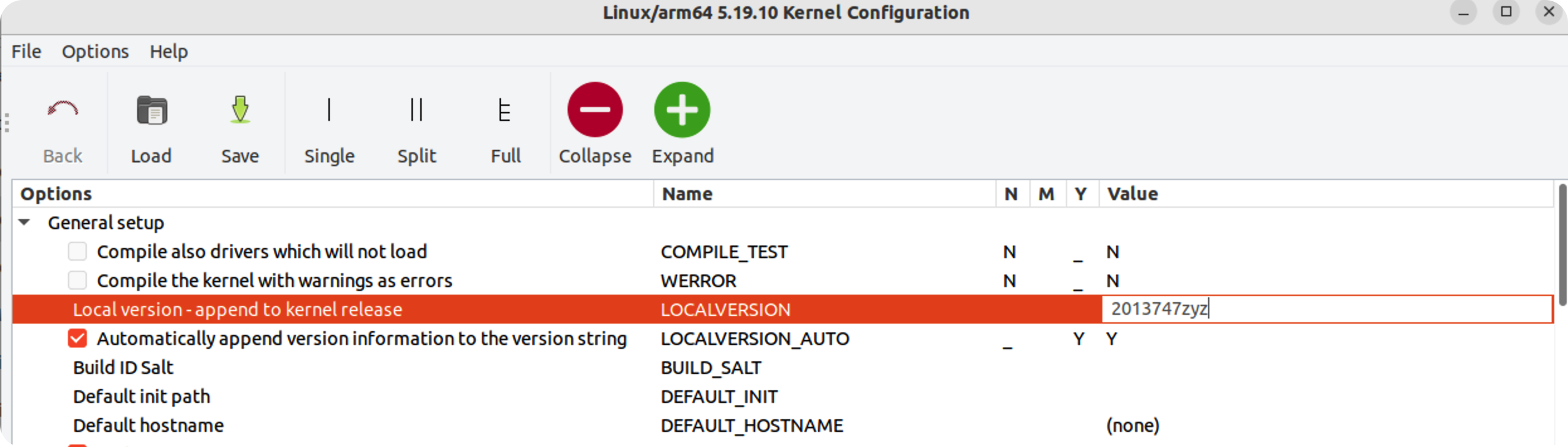


**Step4**

make oldconfig

make gconfig

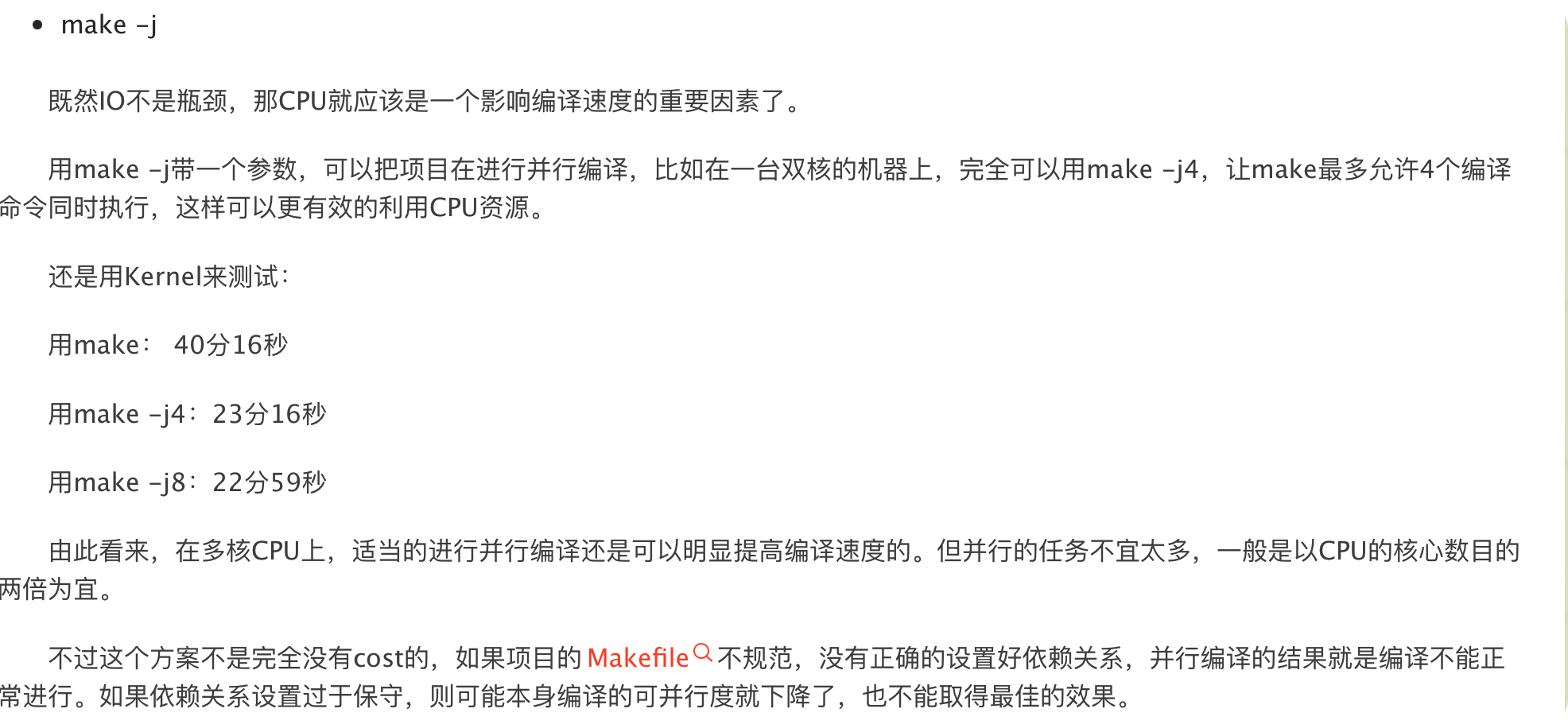
在这一步我在版本名上把原来的zyz2013747改成了2013747zyz以区别更改之后的新版本



make clean

make -j5

// make -jx（x越大，编译的时候用的线程越多，就越快！！！！！）



sudo make

sudo make modules

sudo make modules\_install

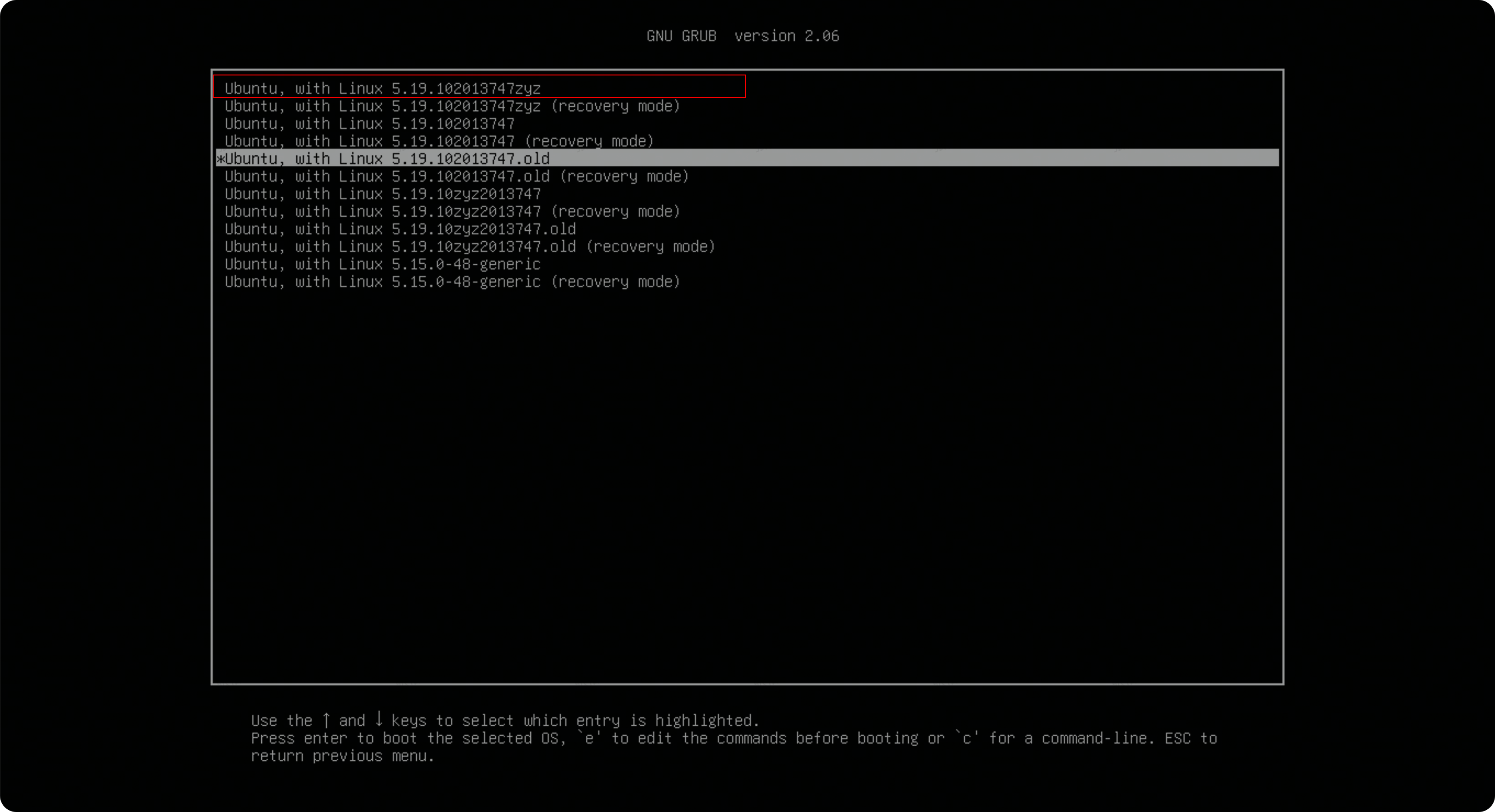
sudo make install

**Step 5**

重新启动:

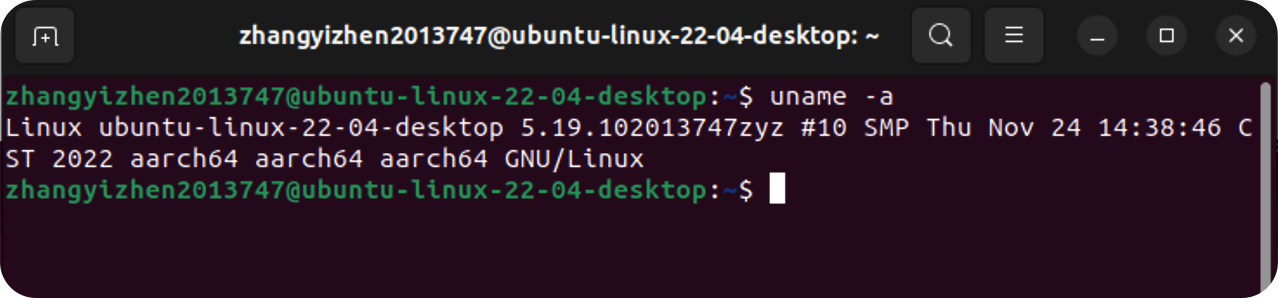
sudo reboot

可以看到有2013747zyz后缀的新版本，选择进入即可



确认新内核是否成功运行：

uname -a



**Step 6**

编写用户态测试程序testschello.c

|  |
| --- |
| C #include <unistd.h> #include <sys/syscall.h> #include <sys/types.h> #include <stdio.h> #include <stdlib.h> #include <string.h> #define \_NR\_alcall 452 int main(int argc, char \*\*argv) {   if(strcmp(argv[1],"1")==0){  char\* temp = malloc(1024);  char\* temp2 = malloc(1024);  syscall(\_NR\_alcall, 1,temp,temp2);  printf("%s\n",temp);  }  else if(strcmp(argv[1],"2")==0){  printf("%s\n",argv[2]);  printf("%s\n",argv[3]);  syscall(\_NR\_alcall, 2,argv[2],argv[3]);  }    return 0; } |

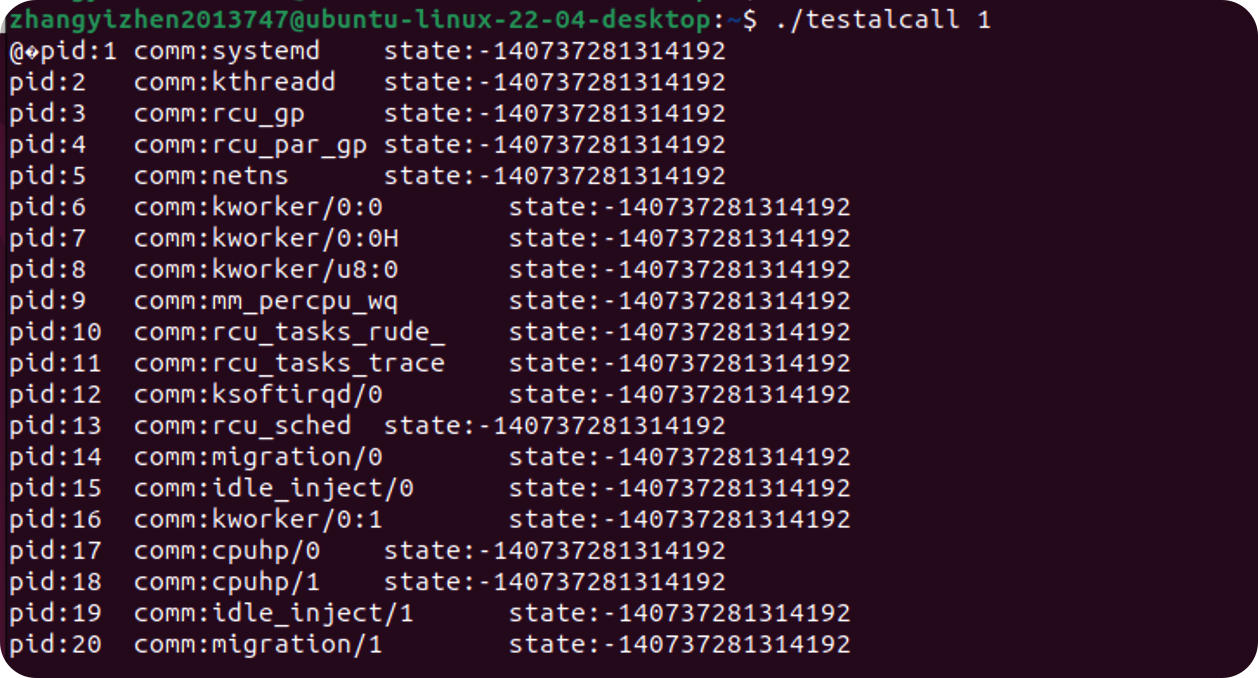
**Step 7**

编译用户态测试程序testalcall.c，并执行

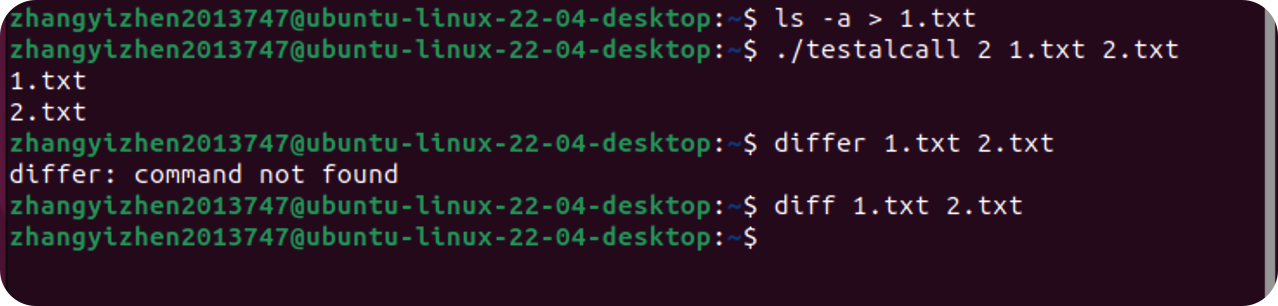
gcc -o testalcall testalcall.c



1. 实现列举进程



1. 文件内核拷贝



6. **总结心得**

更新编译linux内核的过程让我学会了很多东西，遇到了不少问题，查询了很多相关的资料，对于linux的内核有了进一步的了解。

再一次重新编译的过程中，我将原来的版本后缀zyz2013747改成了2013747zyz，可以看到，修改成功，并且得到了最后的新系统调用结果。

7. **参考资料**

1. [ARM架构增加新的系统调用](https://blog.csdn.net/m0_51683653/article/details/124133370)