实验七 系统调用实验

## 实验目的

学习Linux内核的系统调用，理解、掌握Linux系统调用的实现框架、用户界面、参数传递、进入、返回过程。

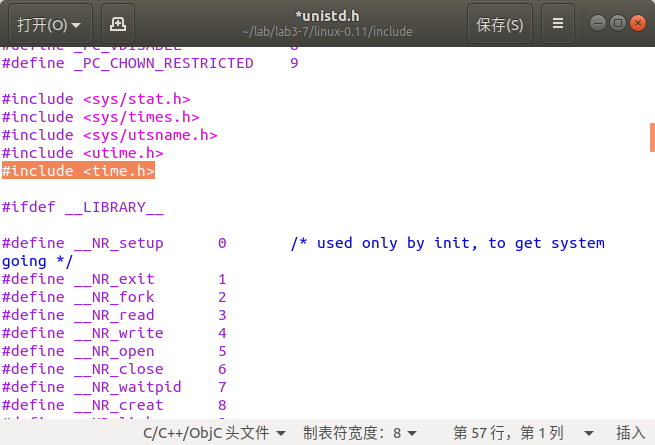
## 实验内容

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。

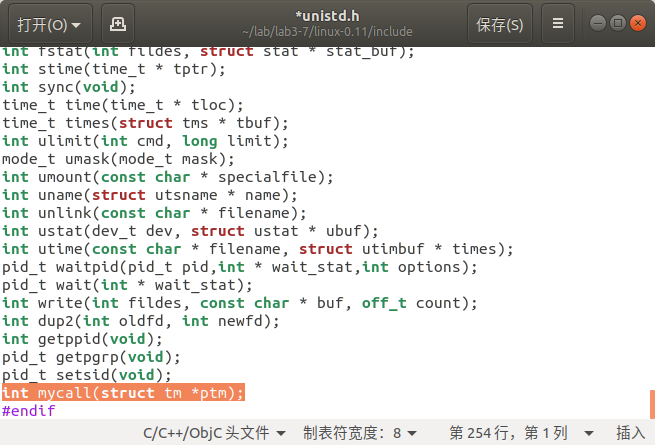
## 实验步骤

1. 在include/unistd.h中添加用户系统调用接口和系统调用号

添加头文件







修改

1. 在kernel/system\_call.s中修改调用总数

|  |
| --- |
| 2020-06-08 18-49-27 的屏幕截图 |

1. 修改 include/linux/sys.h的函数调用表

|  |
| --- |
| 2020-06-08 18-50-09 的屏幕截图 |
| 2020-06-08 18-50-18 的屏幕截图 |

1. 编写mycall.c实现系统调用

时钟读取部分仿照内核main.c中初始化设置时钟的方式

|  |
| --- |
| #include <time.h>  #include <linux/sched.h>  #include <linux/kernel.h>  #include <asm/segment.h>  #include <asm/io.h>  #define CMOS\_READ(addr) ({ \  outb\_p(0x80|addr,0x70); \  inb\_p(0x71); \  })  #define BCD\_TO\_BIN(val) ((val)=((val)&15) + ((val)>>4)\*10)  int sys\_mycall(struct tm \*ptm)  {  struct tm time;  unsigned char \*from=(void\*)&time,\*to=(void\*)ptm;  int i=0;  do {  time.tm\_sec = CMOS\_READ(0);  time.tm\_min = CMOS\_READ(2);  time.tm\_hour = CMOS\_READ(4);  time.tm\_mday = CMOS\_READ(7);  time.tm\_mon = CMOS\_READ(8);  time.tm\_year = CMOS\_READ(9);  } while (time.tm\_sec != CMOS\_READ(0));  BCD\_TO\_BIN(time.tm\_sec);  BCD\_TO\_BIN(time.tm\_min);  BCD\_TO\_BIN(time.tm\_hour);  BCD\_TO\_BIN(time.tm\_mday);  BCD\_TO\_BIN(time.tm\_mon);  BCD\_TO\_BIN(time.tm\_year);  for(i=0;i<sizeof(struct tm);i++)  put\_fs\_byte(from[i],&to[i]);  return 0;  } |

1. 修改makefile

|  |
| --- |
| 2020-06-08 18-51-34 的屏幕截图  2020-06-08 18-51-54 的屏幕截图 |

1. 编写exp3-7.c

|  |
| --- |
| #define \_\_LIBRARY\_\_  #include "unistd.h"  #include "time.h"  #include "stdio.h"  \_syscall1(int, mycall,struct tm\*,ptm);  int main(int argc, char\*\* argv)  {  struct tm time;  mycall(&time);  printf("2%03d-%02d-%02d %02d:%02d:%02d\n",  time.tm\_year,  time.tm\_mon,  time.tm\_mday,  time.tm\_hour,  time.tm\_min,  time.tm\_sec);  return 0;  } |

1. 挂载硬盘，替换头文件

|  |
| --- |
| 2020-06-08 18-53-15 的屏幕截图 |

1. 编译运行测试

好像模拟器的模拟的硬件时钟有问题，比物理机时间快

