# 操作系统课程设计报告

1. 实验1：
   1. 实验题目：linux内核编译及添加系统调用

学号及姓名：15058317 李秀峰

2、 项目设计方案：

（1）本实验总体设计思路；

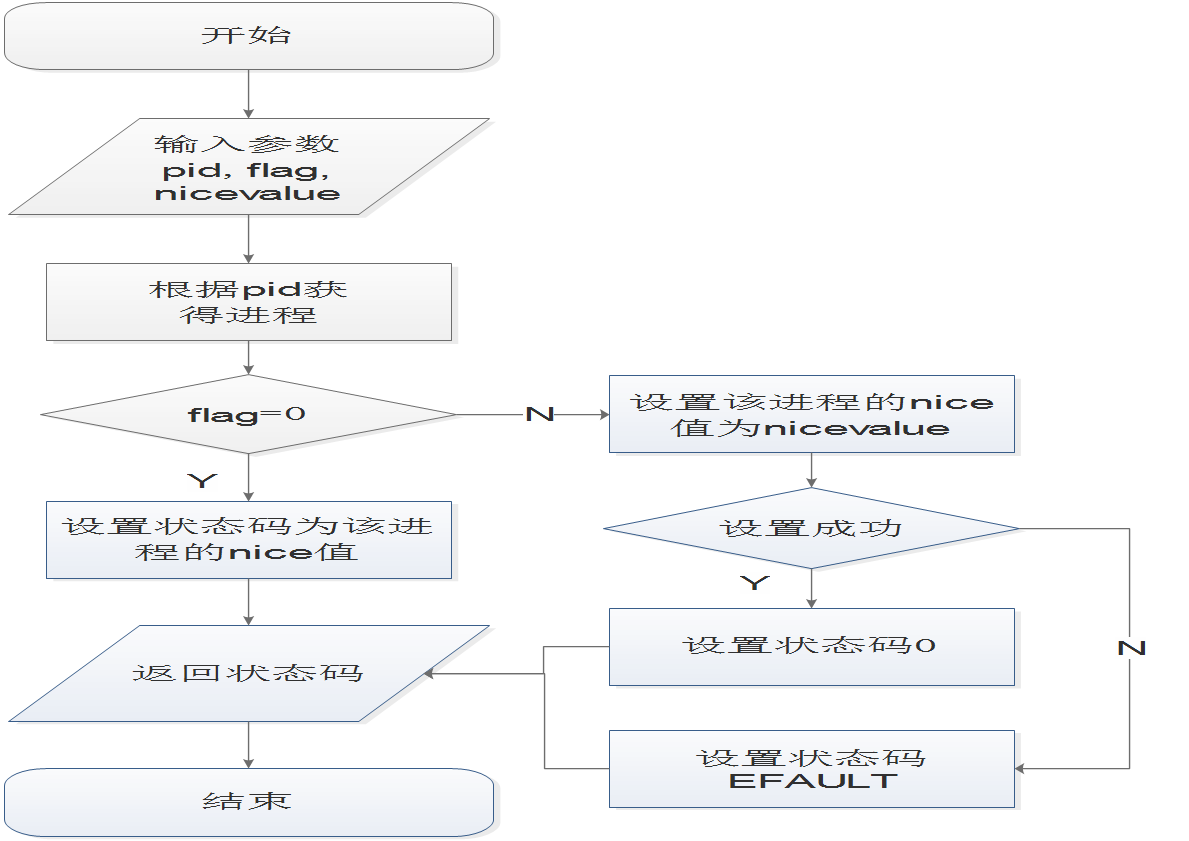
因为编译内核需要很长的时间，所以我将内核下过来解压之后，直接增加系统调用，并将函数写进去，之后再编译内核，编译成功之后，再写个测试函数看看效果。

（2）本实验中主要函数的接口设计；

就一个编写的服务例程，修改获取nice值

（3）主要函数的程序设计思路，并画出程序设计流程图。

首选有个pid参数，保存进程的pid，然后flag值，nicevalue值，如果flag是0，则获取pid的nice值并返回；如果flag是1，则将为该pid的进程的nice值设置为nicevalue，成功返回0，失败返回EFAULT



（4）对设计方案的创新性分析

个性化输出，增加提示

3、项目实现过程：

（1）详细记录项目实现过程中遇到的问题、原因及解决方法（可用截图加文字说明的方法进行编写）

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 问题 | 原因 | 解决方案 |
| 1 | 移动文件进/usr/src时提示  Permission denied | 没有权限 | sudo获取权限，  或者su切换用户 |
| 2 | vim操作文件的时候上下左右键变成ABCD，且vim及其不好操作 | Vim版本有问题 | sudo apt-get remove vim-common  sudo apt-get install vim |
| 3 | make menuconfig时不能改变状态M不能变成\* | 有哪里设置不对，影响了它状态的改变 | 不用管，不影响使用 |
| 4 | 本地测试时用getpriority和setpriority，但是放在sys里面时却不能这么用，提示没找到 | 不能这么使用系统调用函数 | 参看函数源码，自己  实现该功能 |

（2）分析程序运行结果

当随便输入一个pid时，返回-1，正确，没有找到相应进程，返回EFAULT；

当输入正在运行进程的pid，输入flag为0，返回当前进程的nice值，正确，返回nice值；

当输入正在运行进程的pid，输入flag为1，设置当前进程的nice值为nicevalue，正确，返回状态码0；

（3）分析程序实现中的创新性；

创新性，因为程序比较简单，就没做什么改进，就加了点人性化的提示

4、对项目的进一步思考：

（1）分析你所实现的程序的性能，包括优缺点；

程序遍历获得进程，熟识了相关知识，但效率可能偏低

（2）考虑的改进思路；

如果输入的nice值不是有效的nice值，应该提供改进

（3）你认为教师对本实验的设计是否合理？难易程度如何？有什么改进建议？

合理，易

5、参考文献（含阅读书籍、论文、网络资源等）

<http://blog.csdn.net/u014630623/article/details/48480689>

<http://blog.csdn.net/member82/article/details/48444435>

<http://www.cnblogs.com/zero1665/archive/2010/05/05/1728347.html>

6、程序完整代码

/usr/src/linux-4.10.10/arch/x86/entry/syscalls/syscall\_64.tbl

系统调用表，每个表项记录了某个系统调用的服务例程的入口地址

332 64 xfsetnice sys\_xfsetnice

/usr/include/asm-gerneric/unistd.h

惟一的标识每一个系统调用

#define \_\_NR\_xfsetnice 332

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

/usr/src/linux-4.10.10/kernel/sys.c

编写的服务例程保存在此

/\*

\*add by xf

\*if flag equals 0 then getneicevalue

\*if flag equals 1 then setnicevalue and return 0

\*/

SYSCALL\_DEFINE3(xfsetnice, pid\_t, pid, int, flag, int, nicevalue)

{

int num=-EFAULT;

struct task\_struct \*p;

for(p = &init\_task;(p = next\_task(p)) != &init\_task;){

if(p->pid == pid){

if(flag == 0){

printk("xf's get nice,the process's nice = %d",task\_nice(p));

num=task\_nice(p);

}

if(flag == 1){

printk("xf's set nice,the process changed to %d",nicevalue);

set\_user\_nice(p,nicevalue);

num=0;

}

return num;

}

}

return num;

}

测试函数：

#include<stdio.h>

#include<sys/syscall.h>

#include<unistd.h>

#define xfsetnice 332

int main()

{

printf("之前：%ld\n", syscall(xfsetnice,2603,0,1));

printf("中间：%ld\n", syscall(xfsetnice,2603,1,2));

printf("之后：%ld\n", syscall(xfsetnice,2603,0,1));

}