Linux内核编译及添加系统调用

目录

[一、前言 1](#_Toc28390856)

[二、实验准备 1](#_Toc28390857)

[三、主要参考资料 1](#_Toc28390858)

[四、 实验过程 1](#_Toc28390859)

### 一、前言

操作系统实验一:编译内核&增加nice系统调用.

### 二、实验准备

* 虚拟机: VirtualBox 6.0.12 [官网](https://www.virtualbox.org/)
* Ubuntu: 18.04.3(desktop-amd64) [阿里云镜像](https://opsx.alibaba.com/mirror)
* 内核版本:5.3.7 [官网](https://www.kernel.org/)

### 三、主要参考资料

* 资料1:[Ubuntu 18.04环境下编译Linux内核以及添加系统调用](https://www.okcode.net/article/746#_61)
* 资料2:[添加一个系统调用，实现对指定进程的nice值的修改或读取功能，并返回系统最新的nice值即优先级prio](https://blog.csdn.net/babybabyup/article/details/79839734)

### 四、 实验过程

#### 4.1编译内核

* 编译内核严格按照[Ubuntu 18.04环境下编译Linux内核以及添加系统调用](https://www.okcode.net/article/746#_61)这篇参考资料进行,直至重启系统后使用命令uname -a 显示最新内核
* 耗时:晚上睡觉前开始sudo make 一觉醒来就搞定了

#### 4.2增加系统调用

主要参考资料:[添加一个系统调用，实现对指定进程的nice值的修改或读取功能，并返回系统最新的nice值即优先级prio](https://blog.csdn.net/babybabyup/article/details/79839734)

**以下命令在/usr/src/linux-5.3.6中执行**

##### 4.2.1 题目描述

1. 添加一个系统调用，实现对指定进程的nice值的修改或读取功能，并返回系统最新的nice值即优先级prio。 建议调用原型为：int mysetnice(pid\_t pid, int flag, int nicevalue,void\_user\*prio,void\_user\*nice)

**参数含义**：

* pid：进程ID
* flag：若值为0，表示读取nice值；若值为1表示修改nice值。
* prio，nice：指向进程当前优先级及nice值。
* 返回值：系统调用成功时返回0，失败时返回错误码EFAULT。

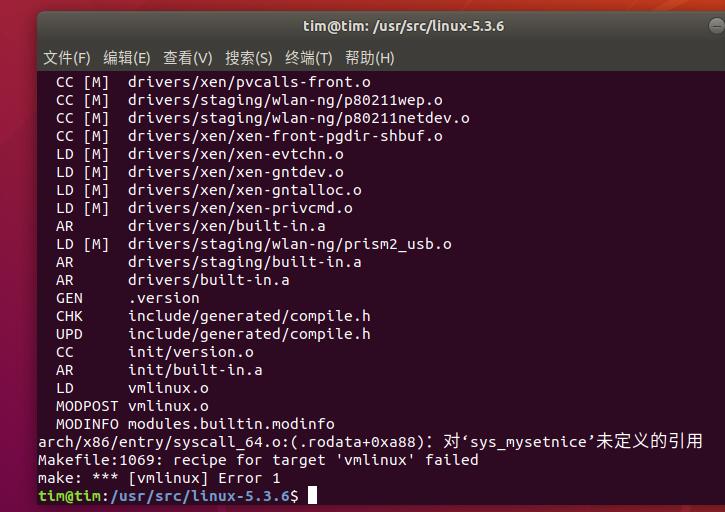
1. 写一个简单的应用程序测试

##### 4.2.2 步骤描述

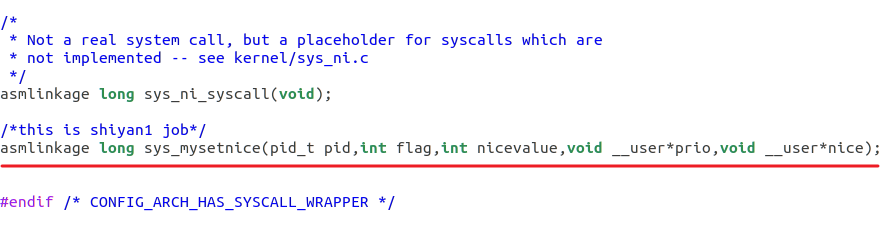
* 修改系统调用表
* 申明系统调用服务例程原型
* 实现系统调用服务例程
* 重新编译
* 编写测试函数进行测试

##### 4.2.3 具体操作

1. 修改系统调用表

* sudo vim arch/x86/entry/syscalls/syscall\_64.tbl
* 1571416426714
* **一定要使用\_\_x64 否则报如下错误**
* 

1. 申明系统调用服务例程原型

* sudo vim include/linux/syscalls.h
* **添加以下代码**
* asmlinkage long sys\_mysetnice(pid\_t pid,int flag,int nicevalue,void \_\_user\*prio,void \_\_user\*nice);
* 

1. 实现系统调用服务例程

* vim kernel/sys.c
* SYSCALL\_DEFINE5(mysetnice,pid\_t,pid,int,flag,int,nicevalue,void \_\_user\*,prio,void \_\_user\*,nice){  
   struct pid \* kpid;  
   struct task\_struct \* task;  
   kpid = find\_get\_pid(pid);/\* 返回pid \*/  
   task = pid\_task(kpid, PIDTYPE\_PID);/\* 返回task\_struct \*/  
   int n;  
   n = task\_nice(task);/\* 返回进程当前nice值 \*/  
   int p;  
   p = task\_prio(task);/\*返回进程当前prio值\*/  
   if(flag == 1)  
   {  
   set\_user\_nice(task, nicevalue);/\* 修改进程nice值 \*/  
   n = task\_nice(task);/\*重新取得进程nice值\*/  
   p = task\_prio(task);/\*重新获取进程prio值 这里和参考资料不一样!!! \*/  
   copy\_to\_user(nice,&n,sizeof(n));/\*将nice值拷贝到用户空间\*/  
   copy\_to\_user(prio,&p,sizeof(p));/\*将prio值拷贝到用户空间\*/  
   return 0;   
   }  
   else if(flag == 0)  
   {  
   copy\_to\_user(nice,&n,sizeof(n));/\*将nice值拷贝到用户空间\*/  
   copy\_to\_user(prio,&p,sizeof(p));/\*将prio值拷贝到用户空间\*/  
   return 0;  
   }  
   return EFAULT;  
  }

1. 编译

* 使用ccache**加快编译**
* 参考文档:[ubuntu 使用ccache加快linux内核编译速度](https://blog.csdn.net/fuyuande/article/details/90485882) **注意将**usr**改为自己的用户名,6G改为10G**
* sudo make clean  
  #仍采用原内核配置文件  
  sudo make localmodconfig #使用localmodconfig加速编译  
  sudo make -j8 #使用多线程编译  
  sudo make modules\_install  
  sudo make install
* **整个过程大概只用了30分钟**

1. 编写测试函数

* vim test.c
* #define \_GNU\_SOURCE  
  #include <unistd.h>  
  #include<sys/syscall.h>  
  #include<stdio.h>  
  #include<stdlib.h>  
  int main(){  
   pid\_t pid;  
   int nicevalue;  
   int flag;  
   int p = 0;  
   int n = 0;  
   int \*prio;  
   int \*nice;  
   prio = &p;  
   nice = &n;  
   /\*  
   \* 获取pid  
   \* \*/  
   printf("请输入pid：\n");  
   scanf("%d",&pid);  
   /\*  
   \* 获取nice；  
   \* \*/  
   printf("pid赋值成功\n请输入nice：\n");  
   scanf("%d",&nicevalue);  
   /\*  
   \* 获取flag；  
   \* \*/  
   printf("nice赋值成功\n请输入flag：\n");  
   scanf("%d",&flag);  
   /\*  
   \* 调用添加的系统调用；  
   \* \*/  
   syscall(334,pid,flag,nicevalue,prio,nice);  
   /\*  
   \* 输出最新的prio以及nice；  
   \* \*/  
   printf("现在的nice为%d\n,prio为%d\n",n,p);  
   return 0;  
  }
* top  
  gcc test.c  
  ./a.out