操作系统课程设计实验报告

实验题目：实验一 Linux 内核编译及添加系统调用

姓　　名：张孜远

学　　号：20151521

组　　号：04

专　　业：卓越学院 智能计算与数据科学

班　　级：20186211

老师姓名：任彧老师

日　　期：2022 年 12 月 20 日

目　　录

一 题目介绍

实验目的：

Linux 是开源操作系统，用户可以根据自身系统需要裁剪、修改内核，定制出功能更加合适、运行效率更高的系统，因此编译 Linux 内核是进行内核开发的必要基本功。在系统中根据需要添加新的系统调用是修改内核的一种常用手段，通过本次实验，我应理解 Linux 系统处理系统调用的流程以及增加系统调用的方法。

二 实验内容与思路

实验内容：

1、内核修改时有自己标签，并用 dmesg 验证；

2、Linux 内核标签（系统启动显示一次）；

3、显示当前系统名称和版本的系统调用（内核、用户都有显示）；

4、修改 nice 和 prio 值的系统调用功能（内核、用户都有显示）；

5、改变主机名称为自定义字符串的系统调用（内核、用户都有显示）；

三 遇到问题及解决方法

遇到的问题：

1、编译内核的速度过慢：

在虚拟机软件 VMware Workstation Pro 中为子系统分配更多的硬件资源，其实本质是分配更多的 CPU 内核数，所以可以在编译时通过添加-j 参数并行编译，例如：-j6 就代表 6 核 CPU 内核数，-j8 就代表 8 核 CPU 内核数。（后来了解到：这里对几核 CPU 的设置与一开始设置的虚拟机内核数量多少无关，直接-j8 拉满即可，只要电脑不崩溃，都是可行的。）

2、编译内核时报错：

1）查看了 Linux 内核源代码后，了解到：SYSCALL_DEFINE 这一系列宏函数，在接受参数时需要将“参数类型”与“参数名称”分开来传递。、

2）由于内核版本、自己编写的 C 语言程序等诸多原因，通过网上查阅资料后得知需要在源码中更改部分设置。

实验方法：

从清华镜像网站中下载 Linux 源代码并对其进行编辑，编辑完自己的内核后，重启 Linux 虚拟机选择自己编辑的内核进行启动，使用相关指令测试相关功能。

四 核心代码及实验结果展示

实验过程和结果：

0、首先创建虚拟机，分配 60G 内存与 8 核 CPU；将从清华镜像网站中下载好的 Linux 内核源代码放入 usr/src/文件夹中；

1、分配系统调用号，修改系统调用表：

查看系统调用表（arch/x86/entry/syscalls/syscalls_64.tbl），可得：每个系统调用在表中占一个表项，其格式为：<系统调用号><commom/64/x32><系统调用名><服务例程入口地址>，其中第二项也称为应用二进制接口。

应用二进制接口分为三种：64、x32 和 common，即三种不同的调用约定。

选择一个未使用的系统调用号进行分配，则新添加的系统调用号可使用 335 号。确定调用号后，应在系统调用表中关联新调用的调用号与服务例程入口，即在 syscall_64.tbl 文件中为新调用添加一条记录，修改结果如下图所示：

```
332     common  statx                   __x64_sys_statx
333     common  io_pgetevents           __x64_sys_io_pgetevents
334     common  rseq                    __x64_sys_rseq
335     64      modify_nice             __x64_sys_modify_nice
336     64      show_message            __x64_sys_show_message
337     64      modify_hostname         __x64_sys_modify_hostname
```

2、申明系统调用服务例程原型：

Linux 系统调用服务例程的原型声明在文件 Linux-4.12/include/linux/syscalls.h 中，可在文件末尾添加下图内容：

```
asmlinkage long sys_modify_nice(pid_t pid, int nicevalue, void __user *prio, void __user *nice);
asmlinkage long sys_show_message(char __user *sysname, char __user *version, char __user *release);
asmlinkage long sys_modify_hostname(char __user *hostname, int len);
#endif
```

3、实现系统调用服务例程：

下面为新调用的 modify_nice、show_message 和 modify_hostname 编写服务例程，通常添加在 linux-4.19.25/kernel/sys.c 文件中，如下图所示：

```
SYSCALL_DEFINE4(modify_nice, pid_t, pid, int, nicevalue, void __user*, prio, void __user*, nice)
{
        int old_prio, old_nice;
        int cur_prio, cur_nice;
        struct pid *ppid;
        struct task_struct *pcb;

        ppid = find_get_pid(pid);
        pcb = pid_task(ppid, PIDTYPE_PID);
        old_prio = task_prio(pcb);
        old_nice = task_nice(pcb);
        printk("OLD NICE: %d, OLD PRIO: %d\n",old_nice,old_prio);
        set_user_nice(pcb, nicevalue);
        cur_prio = task_prio(pcb);
        cur_nice = task_nice(pcb);
        printk("NEW NICE: %d, NEW PRIO: %d\n",cur_nice,cur_prio);
        if(copy_to_user(prio, &cur_prio, sizeof(cur_prio)))
                return -EFAULT;
        if(copy_to_user(nice, &cur_nice, sizeof(cur_nice)))
                return -EFAULT;
        return 0;
}
```

```c
SYSCALL_DEFINE3(show_message, char __user*, sysname, char __user*, version, char __user*, release)
{
        int i;
        int j;
        int t;
        struct new_utsname *u;
        char tmp1[__NEW_UTS_LEN + 1];
        char tmp2[__NEW_UTS_LEN + 1];
        char tmp3[__NEW_UTS_LEN + 1];

        down_read(&uts_sem);

        u = utsname();
        i = 1 + strlen(u->sysname);
        j = 1 + strlen(u->release);
        t = 1 + strlen(u->version);
        memcpy(tmp1, u->sysname, i);
        memcpy(tmp2, u->release, j);
        memcpy(tmp3, u->version, t);

        up_read(&uts_sem);
        printk("Original Sysname: Linux,\nNew Sysname: %s \n",tmp1);
        printk("Original Version: 5.4.0-131-generic,\nNew Version: %s \n",tmp3);
        printk("Original Release: #147~18.04.1-Ubuntu SMP Sat Oct 15 13:10:18 UTC 2022,\nNew Release: %s \n",tmp2);
        if(copy_to_user(sysname,tmp1,i))
                return -EFAULT;
        if(copy_to_user(version,tmp3,t))
                return -EFAULT;
        if(copy_to_user(release,tmp2,j))
                return -EFAULT;
        return 0;
}
```

```c
SYSCALL_DEFINE2(modify_hostname, char __user *, hostname, int , len)
{
        int errno;
        char tmp[__NEW_UTS_LEN];

        errno = -EFAULT;
        if(!copy_from_user(tmp,hostname,len))
        {
                struct new_utsname *u;
                down_write(&uts_sem);
                u=utsname();
                memcpy(u->nodename, tmp, len);
                memset(u->nodename + len, 0, sizeof(u->nodename) - len);
                errno = 0;
                uts_proc_notify(UTS_PROC_HOSTNAME);
                up_write(&uts_sem);
                printk("Original hostname: Linux,\nNew hostname: %s \n",tmp);
        }
        return errno;
}
```

函数说明：

第三步与第二步的关系，类比为 C 语言中头文件与函数实现的关系，即：第二步对函数进行了声明，第三步给函数一个具体的实现。

以 modify_nice 为例：该系统调用需要具备对指定进程的 nice 值的修改及读取的功能，同时返回进程最新的 nice 值及优先级 prio，实现步骤分解如下：

1）根据进程号 pid 找到相应的进程控制块进程控制块 PCB（因为进程控制块 PCB 中记录了用于描述进程情况及控制进程运行所需要的全部信息）；

2）根据进程控制块 PCB 读取它的 nice 值和优先级 prio；

3）根据进程控制块 PCB 修改相应进程的 nice 值；

4）将得到的 nice 值和优先级 prio 进行返回。

4、重新编译内核：

上面三个步骤已经完成添加一个新系统调用的所有工作，但是要让这个系统调用真正在内核中运行起来，还需要重新编译内核。

Linux 内核编译步骤：

1、 下载实验环境 Linux 虚拟机：VMware Workstation Pro；

2、 下载内核源码：来自清华镜像；

3、 解压缩内核源码文件；

4、 使用 make mrproper 指令清除残留的.config 和.o 文件；

5、 使用 make menuconfig 指令配置内核；

执行 make menuconfig 指令配置选项，均采取默认的模式，生成.config 配置文件；

6、 使用 make -j8 指令编译内核（速度更快），生成启动映像文件；

7、 使用 make modules 指令编译模块；

8、 使用 make modules_install 指令和 make install 指令安装内核；

10、使用 update-grub2 指令配置 grub 引导程序；

11、使用 reboot 指令重启系统，选择新编译的内核；

实验结果如下图所示：

1、重新启动，选择编译过的内核版本，查看新的系统版本：

```
zhangziyuan@ubuntu:~$ uname -a
Linux ubuntu 4.19.25 #1 SMP Sun Nov 6 17:17:51 PST 2022 x86_64 x86_64 x86_64 GNU
/Linux
```

2、使用 dmesg 指令查看启动标签：（写入内核中）

```
zhangziyuan@ubuntu:~$ dmesg
[    0.000000] 20151521 ZhangZiYuan
[    0.000000] Linux version 4.19.25 (root@ubuntu) (gcc version 7.5.0 (Ubuntu 7.
5.0-3ubuntu1~18.04)) #1 SMP Sun Nov 6 17:17:51 PST 2022
[    0.000000] Command line: BOOT_IMAGE=/boot/vmlinuz-4.19.25 root=UUID=5cb819a5
-a4b0-47c1-a8eb-7dc3a5d31b09 ro find_preseed=/preseed.cfg auto noprompt priority
=critical locale=en_US quiet
[    0.000000] KERNEL supported cpus:
[    0.000000]   Intel GenuineIntel
[    0.000000]   AMD AuthenticAMD
[    0.000000]   Centaur CentaurHauls
```

3.1、查看系统版本号信息：

```
zhangziyuan@ubuntu:~/exp1$ gcc show.c
zhangziyuan@ubuntu:~/exp1$ ./show
Original Sysname: Linux
Original Version: 5.4.0-131-generic
Original Release: #147~18.04.1-Ubuntu SMP Sat Oct 15 13:10:18 UTC 2022
New Sysname:Linux
New Release:#1 SMP Sun Nov 6 17:17:51 PST 2022
New Version:4.19.25
```

3.2、使用 dmesg 指令查看内核中输出内容：

```
                New Release: 4.19.25
[  881.543712] Original Sysname: Linux,
                New Sysname: Linux
[  881.543713] Original Version: 5.4.0-131-generic,
                New Version: #1 SMP Sun Nov 6 17:17:51 PST 2022
[  881.543713] Original Release: #147~18.04.1-Ubuntu SMP Sat Oct 15 13:10:18 UTC
 2022,
                New Release: 4.19.25
zhangziyuan@ubuntu:~/exp1$
```

4.1、修改 Linux 中 hostname 名称：

```
zhangziyuan@ubuntu:~/exp1$ gcc hostname.c
zhangziyuan@ubuntu:~/exp1$ ./hostname
请输入修改的hostname：ZZY20151521
New hostname:ZZY20151521
```

4.2、使用 dmesg 指令查看内核中输出内容：

```
[ 1078.167674] Original hostname: Linux,
               New hostname: ZZY20151521
               \xff\xff\xff\xff:\xc0\xa1c
zhangziyuan@ubuntu:~/exp1$
```

5.1、查看进程 pid=1 的 nice 和 prio 值：

```
zhangziyuan@ubuntu:~/exp1$ gcc setnice.c
zhangziyuan@ubuntu:~/exp1$ ./setnice
Please input variable(pid, flag, nicevalue): 1 0 0
Current priority is : [20], current nice is [0]
```

5.2、修改并查看进程 pid=1 的 nice 和 prio 值：

```
zhangziyuan@ubuntu:~/exp1$ gcc setnice.c
zhangziyuan@ubuntu:~/exp1$ ./setnice
Please input variable(pid, flag, nicevalue): 1 0 0
Current priority is : [20], current nice is [0]
zhangziyuan@ubuntu:~/exp1$ ./setnice
Please input variable(pid, flag, nicevalue): 1 1 1
Original priority is: [20], original nice is [0]
Current priority is : [21], current nice is [1]
```

5.3、使用 dmesg 指令查看内核中输出内容：

```
[ 1058.987381] OLD NICE: 0, OLD PRIO: 20
[ 1058.987381] NEW NICE: 1, NEW PRIO: 21
zhangziyuan@ubuntu:~/exp1$
```

五 个人实验改进与总结

5.1 个人实验小结

项目实现创新点说明	1、 查看 Linux 内核源代码，了解 Linux 下的文件目录结构，以及根目录下的每个目录一般会存放什么形式、什么功能的文件： /：根目录； bin：一般用户可用，启动时会用到的命令，即在文件系统还没有被挂载时，也能够使用的命令； boot：1、grub：开机设置相关文件； 2、内核文件（vmlinuz） dev：存放设备文件 etc：1、rc.d：用于存放不同运行等级的启动脚本的链接文件； 2、X11 etc 文件夹包含系统特有的可编辑配置文件，即用于控制程序运行的本地文件； home：用户目录； lib：存放程序的动态库和模块文件； media：挂载本地磁盘或其他存储设备； mnt：挂载其他临时文件系统； opt：发行版附加的一些软件包的安装目录； root：root 用户的家； sbin：存放很多只有 root 用户才能执行的命令，以及一些系统的更新、备份、还原和开关机用到的命令； srv：存放服务进程所需的数据文件（如 www 网络服务器和 ftp 服务）和一些服务的执行脚本； tmp：存放各种临时文件； usr：包含诸多文件内容： bin 文件夹中存放非必要可执行文件（在单用户模式中不需要），面向所有用户； include 文件夹中存放了标准头文件； lib 文件夹中存放了/usr/bin/和/usr/sbin/中二进制文件； local 文件夹中存放了本地数据的第三层次，具体到本台主机； share 文件夹中存放了体系结构无关(共享)数据。 sbin 文件夹中存放了非必要的系统二进制文件。例如：大量网络

	服务的守护进程； src 文件夹中存放了源代码，例如：内核源代码及其头文件。不过一般的发行版是不会保留内核源码，需要用户自己下载安装； 整个 usr 文件夹用于存储只读用户数据的第二层次；包含绝大多数的(多)用户工具和应用程序； var：变量文件，在正常运行的系统中其内容不断变化的文件，如日志、脱机文件和临时电子邮件文件；有时是一个单独的分区；

5.2 个人实验总结

实验体会：

这是操作系统的第一个实验，对于从来没有接触过 Linux 系统的我来说，颇具挑战性。首先，在编译内核的时候，需要先对 Linux 内核源码有了一定的了解（即代码的逻辑结构），我先花了 3 天时间阅读核心的 Linux 内核源码；同时在添加系统调用的时候，更需要对内核中的各个源函数的功能通过查阅资料进行了解，达到能够自主修改的程度。

在做完第一个操作系统实验之后，有种苦难过后的重生感。一方面是感叹一开始接触新知识就要掌握并应用它确实很难，但在刻苦努力之后的成功，特别是自己添加的系统调用函数能够显示正确结果，这更具成就感。

六 参考文献

[1] kernel 环境配置.veristas501.
https://veritas501.space/2018/06/03/kernel%E7%8E%AF%E5%A2%83%E9%85%8D%E7%BD%AE/

[2] set_user_nice 源码.
https://code.woboq.org/linux/linux/kernel/sched/core.c.html#set_user_nice

[3] Completely Fair Scheduler.wikipedia.
https://en.wikipedia.org/wiki/Completely_Fair_Scheduler

[4] CFS Scheduler.The Linux Kernel documentation.
https://www.kernel.org/doc/html/latest/scheduler/sched-design-CFS.html

[5] Linux 内核源码
https://elixir.bootlin.com/linux/v5.10.109/source

[6] Linux 内核编译错误：make[1]:“debian/canonical certs.pem”
https://blog.csdn.net/m0_51203305/article/details/120805372

[7] sudo make modules_install 报错
https://blog.csdn.net/waterwhoami/article/details/110621237

[8] 如何在 Ubuntu20.04 编译内核边添加一个系统调用
https://blog.csdn.net/bar_workshop/article/details/111647568