

实验一

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，读者应理解 Linux 系统处理系统调用的流程以及增加系统调用的方法。

实验内容

内核修改时有自己标签，用dmesg验证

- Linux内核标签（系统启动显示一次）
- 显示当前系统名称和版本的系统调用（内核、用户都有显示）
- 修改nice和prio值的系统调用功能（内核、用户都有显示）
- 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）

实验方法

下载 linux 源代码并对其进行编辑，编译并选择内核，测试相关功能。

实验过程和结果

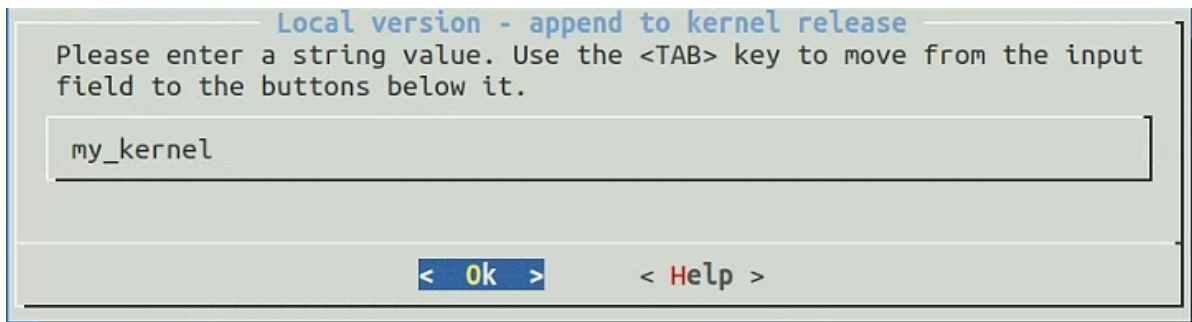
内核编译的过程

```
make menuconfig # 生成.config配置文件
make localmodconfig # 精简内核，使其只编译目前系统下加载的内核
make -j8 # 使用cpu所有8核进行编译，速度更快
make modules # 编译模块
make modules_install # 安装模块
make install # 安装内核
```

修改内核标签

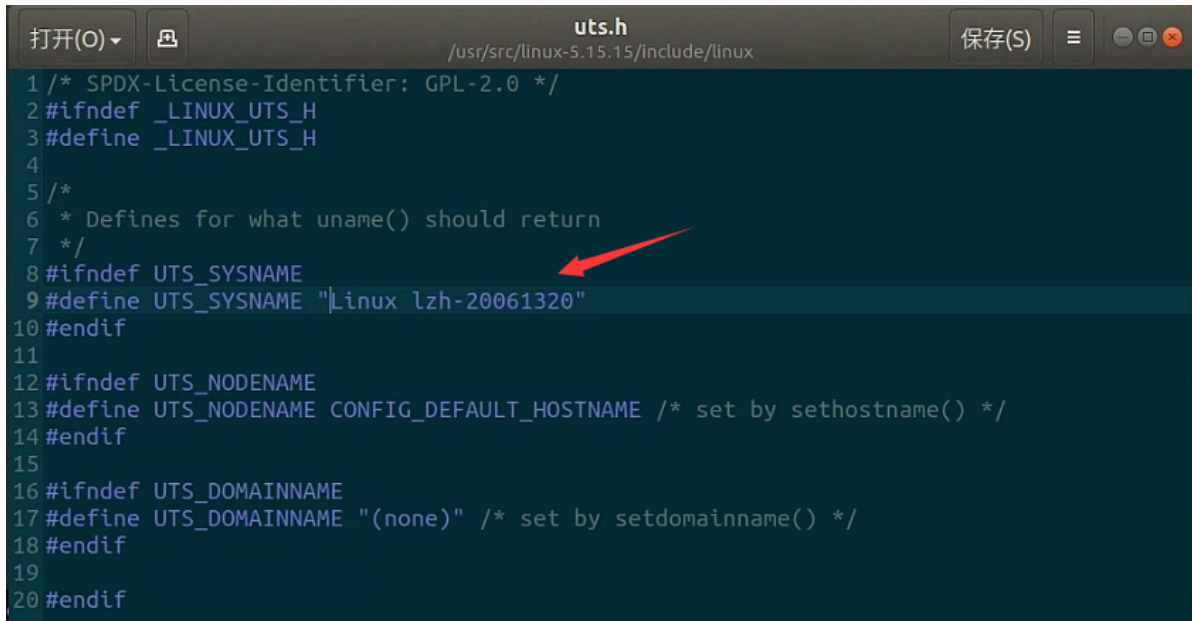
方案一

在make menuconfig后的菜单栏选择General setup，再选择Local version - append to kernel release，在弹出的窗口完成修改，如下图：



方案二

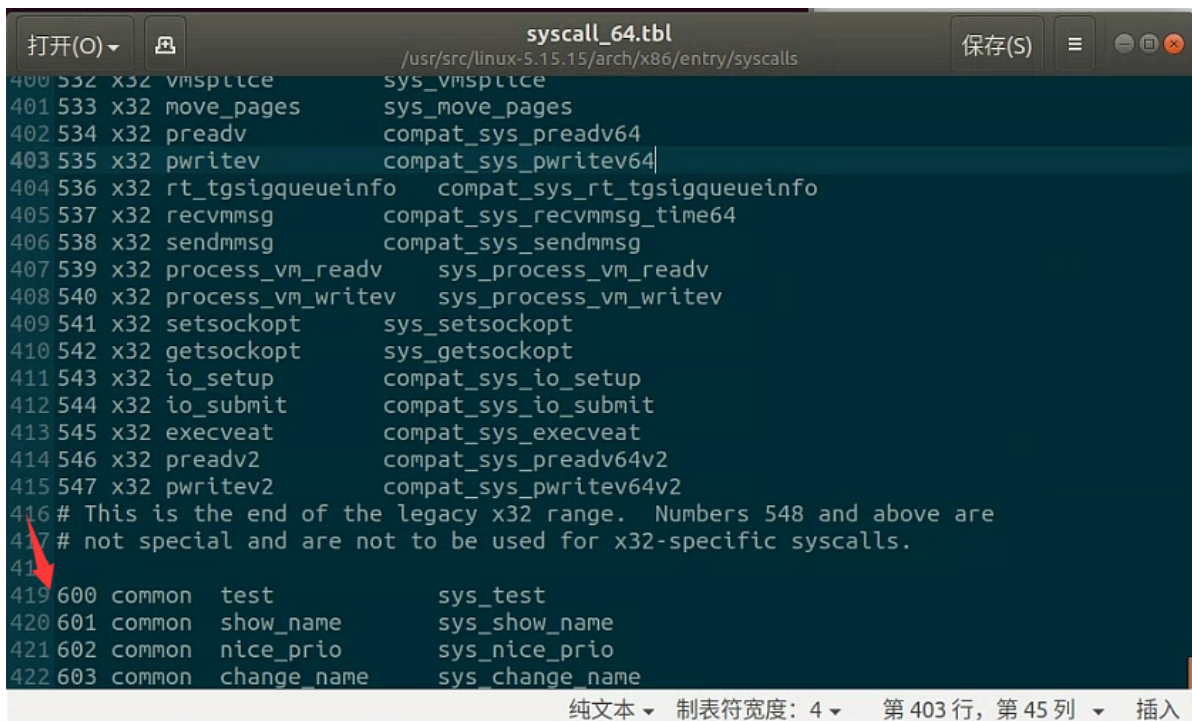
在 linux 源代码 uts.h 中修改，如下图。



该方法只能更改 `uname -a` 命令显示的版本字符串（即启动时无效）。

添加系统调用

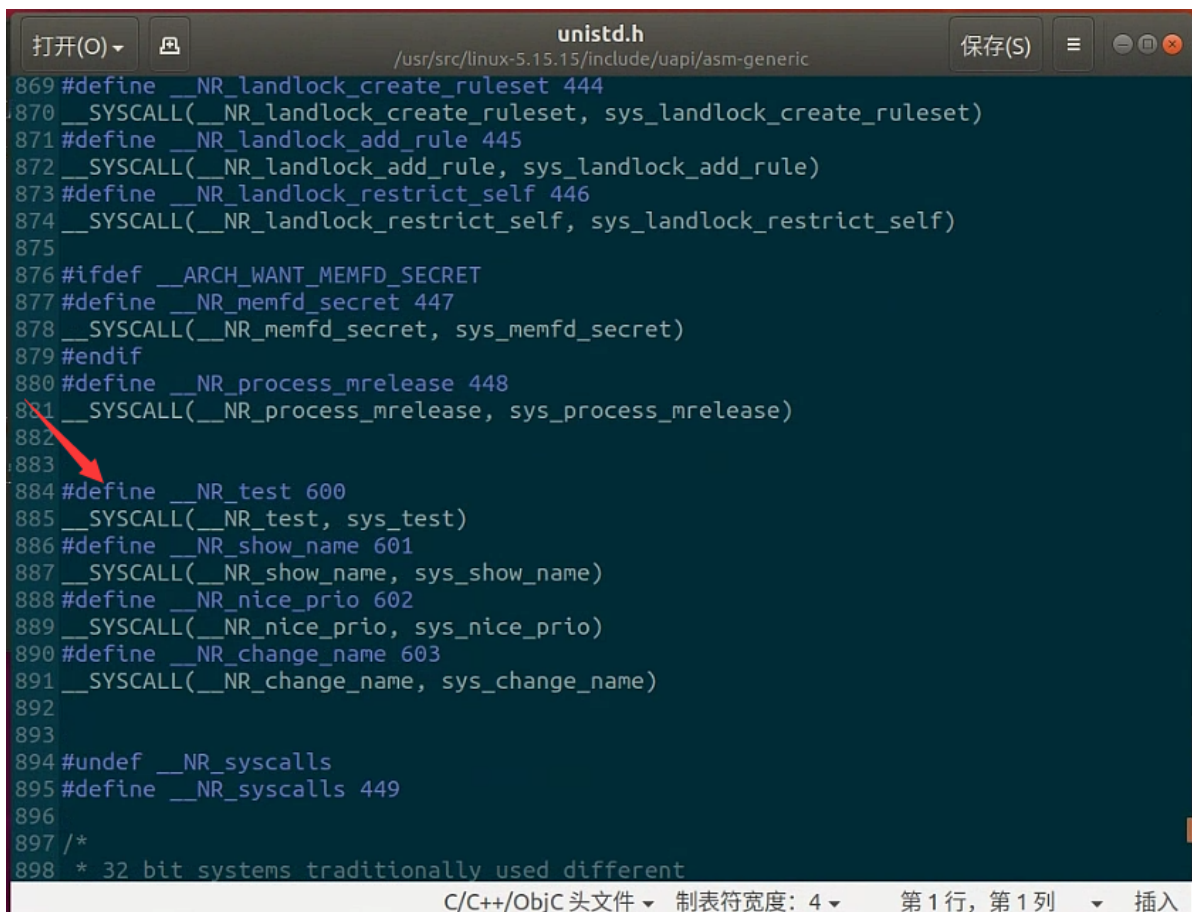
- 在 `arch/x86/entry/syscalls/syscall_64.tbl` 中增加系统调用，一般从600开始添加，如下图所示：



```
syscall_64.tbl
/usr/src/linux-5.15.15/arch/x86/entry/syscalls

400 532 x32 vmspltee sys_vmspltee
401 533 x32 move_pages sys_move_pages
402 534 x32 preadv compat_sys_preadv64
403 535 x32 pwritev compat_sys_pwritev64
404 536 x32 rt_tgsigqueueinfo compat_sys_rt_tgsigqueueinfo
405 537 x32 recvmmsg compat_sys_recvmmsg_time64
406 538 x32 sendmmsg compat_sys_sendmmsg
407 539 x32 process_vm_readv sys_process_vm_readv
408 540 x32 process_vm_writev sys_process_vm_writev
409 541 x32 setsockopt sys_setsockopt
410 542 x32 getsockopt sys_getsockopt
411 543 x32 io_setup compat_sys_io_setup
412 544 x32 io_submit compat_sys_io_submit
413 545 x32 execveat compat_sys_execveat
414 546 x32 preadv2 compat_sys_preadv64v2
415 547 x32 pwritev2 compat_sys_pwritev64v2
416 # This is the end of the legacy x32 range. Numbers 548 and above are
417 # not special and are not to be used for x32-specific syscalls.
418
419 600 common test sys_test
420 601 common show_name sys_show_name
421 602 common nice_prio sys_nice_prio
422 603 common change_name sys_change_name
```

- 在 include/uapi/unistd.h 中添加常量 (该步骤可选) :



```
unistd.h
/usr/src/linux-5.15.15/include/uapi/asm-generic

869 #define __NR_landlock_create_ruleset 444
870 __SYSCALL(__NR_landlock_create_ruleset, sys_landlock_create_ruleset)
871 #define __NR_landlock_add_rule 445
872 __SYSCALL(__NR_landlock_add_rule, sys_landlock_add_rule)
873 #define __NR_landlock_restrict_self 446
874 __SYSCALL(__NR_landlock_restrict_self, sys_landlock_restrict_self)
875
876 #ifdef __ARCH_WANT_MEMFD_SECRET
877 #define __NR_memfd_secret 447
878 __SYSCALL(__NR_memfd_secret, sys_memfd_secret)
879 #endif
880 #define __NR_process_mrelease 448
881 __SYSCALL(__NR_process_mrelease, sys_process_mrelease)
882
883
884 #define __NR_test 600
885 __SYSCALL(__NR_test, sys_test)
886 #define __NR_show_name 601
887 __SYSCALL(__NR_show_name, sys_show_name)
888 #define __NR_nice_prio 602
889 __SYSCALL(__NR_nice_prio, sys_nice_prio)
890 #define __NR_change_name 603
891 __SYSCALL(__NR_change_name, sys_change_name)
892
893
894 #undef __NR_syscalls
895 #define __NR_syscalls 449
896
897 /*
898 * 32 bit systems traditionally used different
```

- 在 include/linux/syscalls.h 中声明新添加的系统调用。

```
1376 long __do_semtimedop(int semid, struct sembuf *tsems, unsigned int nsops,
1377                      const struct timespec64 *timeout,
1378                      struct ipc_namespace *ns);
1379
1380 int __sys_getsockopt(int fd, int level, int optname, char __user *optval,
1381                    int __user *optlen);
1382 int __sys_setsockopt(int fd, int level, int optname, char __user *optval,
1383                    int optlen);
1384 asmlinkage long sys_test(void);
1385 asmlinkage long sys_show_name(void __user* name);
1386 asmlinkage long sys_nice_prio(pid_t pid, int flag, int nicevalue, void
1387                               __user* prio, void __user* nice);
1387 asmlinkage long sys_change_name(char __user * name, int len);
1388 #endif
```

- 在 kernel/sys.c 中编写相应的系统调用代码

对系统调用功能的测试

显示当前系统名称和版本的系统调用

编译运行 ex1/show_name_test.c, 结果如下:

```
dobychoao@my: ~/proj/ex1
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
dobychoao@my:~/proj/ex1$ ls
1.cpp mychangenname nice_prio_test show_name_test
a.out mychangenname.c nice_prio_test.c show_name_test.c
dobychoao@my:~/proj/ex1$ ./show_name_test
user:
sysname: Linux lzh-20061320
release: 5.15.15
version: #1 SMP Mon Sep 26 11:16:54 CST 2022
dobychoao@my:~/proj/ex1$
```

修改nice和prio值的系统调用功能

编译运行 ex1/nice_prio_test.c, 结果如下:

查看进程号为1731的 nice 和 prio :


```
dobychao@my: ~/proj/ex1
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
dobychao@my:~/proj/ex1$ ./nice_prio_test
输入0查看nice和prio, 输入1修改nice和prio, 输入其他值退出。
0
请输入pid:
1731
现在的nice为0
现在的prio为20
输入0查看nice和prio, 输入1修改nice和prio, 输入其他值退出。
1
dobychao@my:~/proj/ex1$

[ 6.361642] Bluetooth: BNEP filters: protocol multicast
[ 6.361647] Bluetooth: BNEP socket layer initialized
[ 6.730172] Generic FE-GE Realtek PHY r8169-0-200:00: attached PHY driver (mi
t_bus:phy_addr=r8169-0-200:00, irq=MAC)
[ 6.926346] r8169 0000:02:00:00 enp2s0: Link is Down
[ 7.583400] loop26: detected capacity change from 0 to 8
[ 10.211799] wlo1: authenticate with 8c:53:c3:cb:2f:43
[ 10.219821] wlo1: send auth to 8c:53:c3:cb:2f:43 (try 1/3)
[ 10.289965] wlo1: authenticated
[ 10.294201] wlo1: associate with 8c:53:c3:cb:2f:43 (try 1/3)
[ 10.310111] wlo1: RX AssocResp from 8c:53:c3:cb:2f:43 (capab=0x31 status=0 ai
d=2)
[ 10.312222] wlo1: associated
[ 10.694580] IPv6: ADDRCONF(NETDEV_CHANGE): wlo1: link becomes ready
[ 17.534877] Bluetooth: RFCOMM TTY layer initialized
[ 17.534886] Bluetooth: RFCOMM socket layer initialized
[ 17.534895] Bluetooth: RFCOMM ver 1.11
[ 18.288156] rfkll: input handler disabled
[ 19.628424] show_signal: 47 callbacks suppressed
[ 19.628428] traps: vmttoolsd[2119] general protection fault ip:7f8cc0a0908c sp
:7fff88129328 error:0 in libvmttools.so[7f8cc0963000+e4000]
[ 61.817218] 修改成功
[ 204.696821] 用户查询nice
dobychao@my:~/proj/ex1$
```

修改进程号为1731的 nice 和 prio：

```
dobychao@my: ~/proj/ex1
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
dobychao@my:~/proj/ex1$ ./nice_prio_test
输入0查看nice和prio, 输入1修改nice和prio, 输入其他值退出。
0
请输入pid:
1731
现在的nice为0
现在的prio为20
输入0查看nice和prio, 输入1修改nice和prio, 输入其他值退出。
1
请输入pid:
1731
请输入nice:
5
现在的nice为5
现在的prio为25
输入0查看nice和prio, 输入1修改nice和prio, 输入其他值退出。
0
dobychao@my:~/proj/ex1$

[ 10.211799] wlo1: authenticate with 8c:53:c3:cb:2f:43
[ 10.219821] wlo1: send auth to 8c:53:c3:cb:2f:43 (try 1/3)
[ 10.289965] wlo1: authenticated
[ 10.294201] wlo1: associate with 8c:53:c3:cb:2f:43 (try 1/3)
[ 10.310111] wlo1: RX AssocResp from 8c:53:c3:cb:2f:43 (capab=0x31 status=0 ai
d=2)
[ 10.312222] wlo1: associated
[ 10.694580] IPv6: ADDRCONF(NETDEV_CHANGE): wlo1: link becomes ready
[ 17.534877] Bluetooth: RFCOMM TTY layer initialized
[ 17.534886] Bluetooth: RFCOMM socket layer initialized
[ 17.534895] Bluetooth: RFCOMM ver 1.11
[ 18.288156] rfkll: input handler disabled
[ 19.628424] show_signal: 47 callbacks suppressed
[ 19.628428] traps: vmttoolsd[2119] general protection fault ip:7f8cc0a0908c sp
:7fff88129328 error:0 in libvmttools.so[7f8cc0963000+e4000]
[ 61.817218] 修改成功
[ 204.696821] 用户查询nice
[ 305.431528] 修改nice成功
dobychao@my:~/proj/ex1$

top - 23:20:15 up 5 min, 1 user, load average: 2.97, 1.60, 0.71
任务: 1 total, 0 running, 1 sleeping, 0 stopped, 0 zombie
%Cpu(s): 15.9 us, 9.0 sy, 1.4 ni, 72.7 id, 0.8 wa, 0.0 hi, 0.2 si, 0.0 st
KiB Mem : 8028588 total, 5222344 free, 1357052 used, 1449192 buff/cache
KiB Swap: 8388604 total, 8388604 free, 0 used, 6107060 avail Mem

 进  用户      PR  NI   VIRT   RES   SHR  %CPU  %MEM    TIME+  COMMAND
   1731  dobychao    25    5 1059640 128200  98620  21.7   1.6   0:55.22  Xorg
```

改变主机名称为自定义字符串的系统调用

编译运行 ex1/mychangename.c，结果如下：

```
dobychao@S1-pro: ~/proj/ex1
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
dobychao@S1-pro:~/proj/ex1$ ./mychangename
请输入修改的主机名称: my
dobychao@S1-pro:~/proj/ex1$ hostname
my
```

内核态下：

```
[ 19.628428] traps: vmttoolsd[2119] general protection fault ip:7f8cc0a0908c sp
:7fff88129328 error:0 in libvmttools.so[7f8cc0963000+e4000]
[ 61.817218] 修改成功
dobychao@S1-pro:~/proj/ex1$
```

实验体会

在做这个实验时，我感觉很有挑战性。编译内核需要掌握一些相关的技能，并且需要耐心细致地进行操作。同时，添加系统调用也需要我们对内核源代码有一定的了解，并能够灵活地进行修改。

但是，在做完这个实验后，我感觉很有成就感。编译出来的内核可以用来启动系统，并且我们自己添加的系统调用也可以正常使用，这让我感觉很棒。

总之，做这个实验可能有一定的挑战性，但是完成后会有很大的成就感。

参考文献

Ubuntu20.04编译内核边添加一个系统调用

https://blog.csdn.net/bar_workshop/article/details/111647568

解决Linux"没有规则可制作目标"debian/canonical-revoked-certs.pem"，由"certs/x509_revocation_list"需求。停止。"方法

<https://www.xgboke.com/16586.html>

求助：ubuntu 编译内核后（make -jn），sudo make modules_install报错

<https://blog.csdn.net/waterwhoami/article/details/110621237>

Linux内核编译错误：make[1]: *** 没有规则可制作目标"debian/canonical-certs.pem"，由"certs/x509_certificate_list"需求。停止

https://blog.csdn.net/m0_51203305/article/details/120805372

Kali Linux 2021.3编译安装升级内核

<https://zhuanlan.zhihu.com/p/415906419>

kernel_read和kernel_write实例

<https://blog.csdn.net/wangkai6666/article/details/121312577>

Linux内核源码

<https://elixir.bootlin.com/linux/v5.10.109/source>