linux内核编译

> 系统环境如下：
> 发行版：Ubuntu 18.04LTS 64位
> 桌面环境：Deepin 15 桌面版 Release

1. 获取内核源码

1.1 查看源内的内核源码列表

```
1 sudo apt-cache search linux-source
```

```
(base) luzhan@luzhan-GF63-8RC:~$ sudo apt-cache search linux-source
[sudo] luzhan 的密码：
linux-source - Linux kernel source with Ubuntu patches
linux-source-4.15.0 - Linux kernel source for version 4.15.0 with Ubuntu patches
linux-source-4.18.0 - Linux kernel source for version 4.18.0 with Ubuntu patches
linux-source-5.0.0 - Linux kernel source for version 5.0.0 with Ubuntu patches
linux-source-5.3.0 - Linux kernel source for version 5.3.0 with Ubuntu patches
linux-source-4.10.0 - Linux kernel source for version 4.10.0 with Ubuntu patches
linux-source-4.11.0 - Linux kernel source for version 4.11.0 with Ubuntu patches
linux-source-4.13.0 - Linux kernel source for version 4.13.0 with Ubuntu patches
linux-source-4.4.0 - Linux kernel source for version 4.4.0 with Ubuntu patches
linux-source-4.8.0 - Linux kernel source for version 4.8.0 with Ubuntu patches
```

1.2 下载源代码

选择一个版本进行下载。

```
1 sudo apt-get install linux-source-4.18.0
```

```
(base) luzhan@luzhan-GF63-8RC:~$ sudo apt-get install linux-source-4.18.0
正在读取软件包列表... 完成
正在分析软件包的依赖关系树
正在读取状态信息... 完成
建议安装：
  libncurses-dev | ncurses-dev kernel-package libqt3-dev
下列【新】软件包将被安装：
  linux-source-4.18.0
升级了 0 个软件包，新安装了 1 个软件包，要卸载 0 个软件包，有 25 个软件包未被升级。
需要下载 130 MB 的归档。
解压缩后会消耗 147 MB 的额外空间。
获取:1 http://mirrors.tuna.tsinghua.edu.cn/ubuntu bionic-updates/main amd64 linux-source-4.18.0 all 4.18.0-25.26~18.04.1
 [130 MB]
已下载 130 MB，耗时 26秒 (4,900 kB/s)
正在选中未选择的软件包 linux-source-4.18.0。
(正在读取数据库 ... 系统当前共安装有 329382 个文件和目录。)
正准备解包 .../linux-source-4.18.0_4.18.0-25.26~18.04.1_all.deb  ...
正在解包 linux-source-4.18.0 (4.18.0-25.26~18.04.1) ...
正在设置 linux-source-4.18.0 (4.18.0-25.26~18.04.1) ...
```

下载完成后进入/usr/src目录查看

```
(base) luzhan@luzhan-GF63-8RC:/usr/src$ ls
bcmwl-6.30.223.271+bdcom  fortran_thunking.c              linux-headers-4.15.0-91          nvidia-440.64
fortran.c                 fortran_thunking.h              linux-headers-4.15.0-91-generic
fortran_common.h          linux-headers-4.15.0-88         linux-source-4.18.0
fortran.h                 linux-headers-4.15.0-88-generic linux-source-4.18.0.tar.bz2
```

1.3 解压

关于tar命令

参数	说明
-c	建立压缩档案
-x	解压
-t	查看内容
-r	向压缩归档文件末尾追加文件
-u	更新原压缩包中的文件

-z	有gzip属性的
-j	有bz2属性的
-f	创建新的文件名
-v	显示所有过程
-C	将文件解压到特定文件夹

```
sudo mkdir os-test
sudo tar -jxv -f linux-source-4.18.0.tar.bz2 -C ./os-test
```

```
(base) luzhan@luzhan-GF63-8RC:/usr/src$ sudo mkdir os-test
(base) luzhan@luzhan-GF63-8RC:/usr/src$ sudo tar -jxv -f linux-source-4.18.0.tar.bz2 -C ./os-test/
linux-source-4.18.0/
linux-source-4.18.0/samples/
linux-source-4.18.0/samples/uhid/
linux-source-4.18.0/samples/uhid/Makefile
```

2. 修改内核源码

2.1 进入目标文件夹

```
(base) luzhan@luzhan-GF63-8RC:/usr/src$ cd os-test/
(base) luzhan@luzhan-GF63-8RC:/usr/src/os-test$ ls
linux-source-4.18.0
(base) luzhan@luzhan-GF63-8RC:/usr/src/os-test$ cd linux-source-4.18.0/
(base) luzhan@luzhan-GF63-8RC:/usr/src/os-test/linux-source-4.18.0$ ls
arch     CREDITS        drivers      include  Kconfig   MAINTAINERS  README    snapcraft.yaml  ubuntu
block    crypto         dropped.txt  init     kernel    Makefile     samples   sound           usr
certs    debian.master  firmware     ipc      lib       mm           scripts   spl             virt
COPYING  Documentation  fs           Kbuild   LICENSES  net          security  tools           zfs
(base) luzhan@luzhan-GF63-8RC:/usr/src/os-test/linux-source-4.18.0$ cd kernel/
(base) luzhan@luzhan-GF63-8RC:/usr/src/os-test/linux-source-4.18.0/kernel$ ls
acct.c              crash_core.c     iomem.c           livepatch          reboot.c         time
async.c             crash_dump.c     irq               locking            relay.c          torture.c
audit.c             cred.c           irq_work.c        Makefile           resource.c       trace
auditfilter.c       debug            jump_label.c      memremap.c         rseq.c           tracepoint.c
audit_fsnotify.c    delayacct.c      kallsyms.c        module.c           sched            tsacct.c
audit.h             dma              kcmp.c            module-internal.h  seccomp.c        ucount.c
auditsc.c           dma.c            Kconfig.freezer   module_signing.c   signal.c         uid16.c
audit_tree.c        elfcore.c        Kconfig.hz        notifier.c         smpboot.c        uid16.h
audit_watch.c       events           Kconfig.locks     nsproxy.c          smpboot.h        umh.c
backtracetest.c     exec_domain.c    Kconfig.preempt   padata.c           smp.c            up.c
bounds.c            exit.c           kcov.c            panic.c            softirq.c        user.c
bpf                 extable.c        kexec.c           params.c           stacktrace.c     user_namespace.c
capability.c        fail_function.c  kexec_core.c      pid.c              stop_machine.c   user-return-notifier.c
cgroup              fork.c           kexec_file.c      pid_namespace.c    sys.c            utsname.c
compat.c            freezer.c        kexec_internal.h  power              sysctl_binary.c  utsname_sysctl.c
configs             futex.c          kmod.c            printk             sysctl.c         watchdog.c
configs.c           futex_compat.c   kprobes.c         profile.c          sys_ni.c         watchdog_hld.c
context_tracking.c  gcov             ksysfs.c          ptrace.c           taskstats.c      workqueue.c
cpu.c               groups.c         kthread.c         range.c            task_work.c      workqueue_internal.h
cpu_pm.c            hung_task.c      latencytop.c      rcu                test_kprobes.c
```

2.2 修改sys.c

```
sudo vim sys.c
```

```
(base) luzhan@luzhan-GF63-8RC:/usr/src/os-test/linux-source-4.18.0/kernel$ sudo vim sys.c
```

添加头文件 `#include <linux/linkage.h>`

```
/*
 *  linux/kernel/sys.c
 *
 *  Copyright (C) 1991, 1992  Linus Torvalds
 */
#include <linux/linkage.h>

#include <linux/export.h>
```

这个程序中有很多`#ifdef`之类的预编译命令，为了防止混入其中，`shift+g`跳至文件尾，在最后添加我们自定义的函数。

内容如下，具体打印的内容可以自定义。

```
asmlinkage int sys_helloworld(void){
            printk(KERN_EMERG “hello luzhan!”)
            return 1;
}
```

```
}
#endif /* CONFIG_COMPAT */

// my own system call implement
// author: luzhan

asmlinkage int sys_helloworld(void){
        printk(KERN_EMERG "hello luzhan!");
        return 1;
}
~
```

2.3 添加函数声明

进入头文件所在的目录

```
cd /usr/src/os-test/linux-source-4.18.0/arch/x86/include/asm/
```

修改 syscalls.h，添加函数声明。

注意这里的文件名，不是syscall.h。同目录下这两个文件都有，我一开始用 `Tab` 命令补全后直接进入了syscall.h这个文件进行修改，导致第一次编译失败。

```
(base) luzhan@luzhan-GF63-8RC:/usr/src/os-test/linux-source-4.18.0/arch/x86/include/asm$ vim syscalls.h
```

直接在头文件下方添加函数声明。

```
#include <linux/compiler.h>
#include <linux/linkage.h>
#include <linux/signal.h>
#include <linux/types.h>

// my own system call
// author: luzhan
asmlinkage int sys_helloworld(void);
```

2.4 添加系统调用

我用的是64位的Ubuntu，实验指导书用的是32位的，目标目录有所不同。

```
cd /usr/src/os-test/linux-source-4.18.0/arch/x86/entry/syscalls/
sudo vim syscall_64.tbl
```

```
c(base) luzhan@luzhan-GF63-8RC:/usr/src/os-test/linux-source-4.18.0/arch/x86/include/asm$ cd ..
(base) luzhan@luzhan-GF63-8RC:/usr/src/os-test/linux-source-4.18.0/arch/x86/include$ cd ..
(base) luzhan@luzhan-GF63-8RC:/usr/src/os-test/linux-source-4.18.0/arch/x86$ cd ..
(base) luzhan@luzhan-GF63-8RC:/usr/src/os-test/linux-source-4.18.0/arch$ ls
alpha  arm    c6x    hexagon  Kconfig  microblaze  nds32  openrisc  powerpc  s390  sparc  unicore32  xtensa
arc    arm64  h8300  ia64     m68k     mips        nios2  parisc    riscv    sh    um     x86
(base) luzhan@luzhan-GF63-8RC:/usr/src/os-test/linux-source-4.18.0/arch$ cd x86/syscalls
bash: cd: x86/syscalls: 没有那个文件或目录
(base) luzhan@luzhan-GF63-8RC:/usr/src/os-test/linux-source-4.18.0/arch$ cd x86
(base) luzhan@luzhan-GF63-8RC:/usr/src/os-test/linux-source-4.18.0/arch/x86$ ls
boot     entry   ia32     Kconfig        kernel  Makefile         math-emu  oprofile  power      realmode  video
configs  events  include  Kconfig.cpu    kvm     Makefile_32.cpu  mm        pci       purgatory  tools     xen
crypto   hyperv  Kbuild   Kconfig.debug  lib     Makefile.um      net       platform  ras        um
(base) luzhan@luzhan-GF63-8RC:/usr/src/os-test/linux-source-4.18.0/arch/x86$ cd entry/syscalls/
(base) luzhan@luzhan-GF63-8RC:/usr/src/os-test/linux-source-4.18.0/arch/x86/entry/syscalls$ ls
Makefile  syscall_32.tbl  syscall_64.tbl  syscallhdr.sh  syscalltbl.sh
(base) luzhan@luzhan-GF63-8RC:/usr/src/os-test/linux-source-4.18.0/arch/x86/entry/syscalls$ vim syscall_64.tbl
(base) luzhan@luzhan-GF63-8RC:/usr/src/os-test/linux-source-4.18.0/arch/x86/entry/syscalls$ sudo vim syscall_64.tbl
```

这就是系统调用的番号表。

添加我们自己的系统调用，这里番号是335，`sys_helloworld` 就是之前自定义的函数名。

番号要记一下，后面会用到。

```
327     64      preadv2                 __x64_sys_preadv2
328     64      pwritev2                __x64_sys_pwritev2
329     common  pkey_mprotect           __x64_sys_pkey_mprotect
330     common  pkey_alloc              __x64_sys_pkey_alloc
331     common  pkey_free               __x64_sys_pkey_free
332     common  statx                   __x64_sys_statx
333     common  io_pgetevents           __x64_sys_io_pgetevents
334     common  rseq                    __x64_sys_rseq
335     64      helloworld              sys_helloworld
```

3. 编译

3.1 安装依赖项

```
sudo apt-get update
sudo apt-get install libncurses5-dev
sudo apt-get install libssl-dev
sudo apt-get install libelf-dev
sudo apt-get install bison -y
sudo apt-get install flex
```

如果后续运行的时候还有错误可以根据错误提示继续装，比如下方

```
(base) luzhan@luzhan-GF63-8RC:/usr/src/os-test/linux-source-4.18.0$ sudo make mrproper
(base) luzhan@luzhan-GF63-8RC:/usr/src/os-test/linux-source-4.18.0$ sudo make clean
(base) luzhan@luzhan-GF63-8RC:/usr/src/os-test/linux-source-4.18.0$ sudo make menuconfig
  HOSTCC  scripts/basic/fixdep
  UPD     scripts/kconfig/.mconf-cfg
  HOSTCC  scripts/kconfig/mconf.o
  YACC    scripts/kconfig/zconf.tab.c
/bin/sh: 1: bison: not found
scripts/Makefile.lib:196: recipe for target 'scripts/kconfig/zconf.tab.c' failed
make[1]: *** [scripts/kconfig/zconf.tab.c] Error 127
Makefile:562: recipe for target 'menuconfig' failed
make: *** [menuconfig] Error 2
```

```
(base) luzhan@luzhan-GF63-8RC:/usr/src/os-test/linux-source-4.18.0$ sudo make menuconfig
  YACC    scripts/kconfig/zconf.tab.c
  LEX     scripts/kconfig/zconf.lex.c
/bin/sh: 1: flex: not found
scripts/Makefile.lib:188: recipe for target 'scripts/kconfig/zconf.lex.c' failed
make[1]: *** [scripts/kconfig/zconf.lex.c] Error 127
Makefile:562: recipe for target 'menuconfig' failed
make: *** [menuconfig] Error 2
```

3.2 编译

```
sudo make mrproper
sudo make clean
```

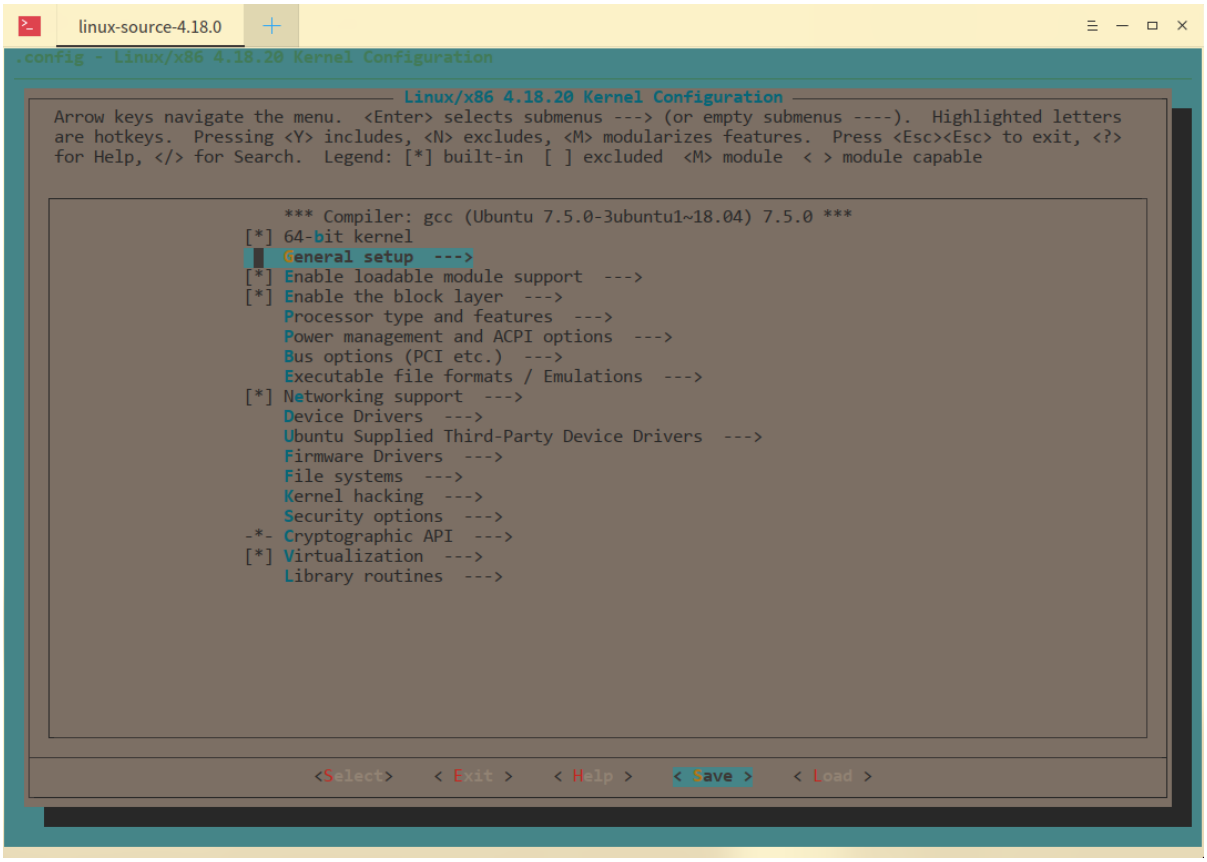
```
3 sudo make menuconfig
```

mrproper清除编译过程中产生的所有中间文件。

clean清除上一次产生的编译中间文件。

执行最后一条命令后会出现图形化配置窗口，方向键向下选中General setup这一项，然后向右选中Save，

回车后选择 ok，exit，最后回到该界面选中Exit退出。



编译前查看一下cpu的属性，来确定编译的时候要用几个线程。毕竟编译时间很长，多大作业数还是很有必要的。

```
1 lscpu
```

```
(base) luzhan@luzhan-GF63-8RC:/usr/src/os-test/linux-source-4.18.0$ lscpu
架构：              x86_64
CPU 运行模式：      32-bit, 64-bit
字节序：            Little Endian
CPU:                12
在线 CPU 列表：     0-11
每个核的线程数：    2
每个座的核数：      6
座：                1
NUMA 节点：         1
厂商 ID：           GenuineIntel
CPU 系列：          6
型号：              158
型号名称：          Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz
步进：              10
CPU MHz：           847.291
CPU 最大 MHz：      4100.0000
CPU 最小 MHz：      800.0000
BogoMIPS：          4416.00
虚拟化：            VT-x
L1d 缓存：          32K
L1i 缓存：          32K
L2 缓存：           256K
L3 缓存：           9216K
NUMA 节点0 CPU：    0-11
标记：              fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse
se2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc art arch_perfmon pebs bts rep_good nopl xtopology nonstop_ts
 cpuid aperfmperf tsc_known_freq pni pclmulqdq dtes64 monitor ds_cpl vmx est tm2 ssse3 sdbg fma cx16 xtpr pdcm pcid sse
_1 sse4_2 x2apic movbe popcnt tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm abm 3dnowprefetch cpuid_fault epb in
pcid_single pti ssbd ibrs ibpb stibp tpr_shadow vnmi flexpriority ept vpid fsgsbase tsc_adjust bmi1 avx2 smep bmi2 erms
invpcid mpx rdseed adx smap clflushopt intel_pt xsaveopt xsavec xgetbv1 xsaves dtherm ida arat pln pts hwp hwp_notify h
p_act_window hwp_epp md_clear flush_l1d
(base) luzhan@luzhan-GF63-8RC:/usr/src/os-test/linux-source-4.18.0$
```

可以看到我这里CPU有12个核，每个核线程数是2，所以最多可以用24个线程。

按如下命令开始编译，作业数根据自己实际设备来。

```
sudo make bzImage -j24
sudo make modules -j24
sudo make modules_install -j24
sudo make install -j24
```

我总共编译了差不多30分钟。我是双系统，硬件资源可以都用上。装虚拟机的话应该不会分配这么多资源，时间可能会更久，所以尽量不要犯错。

```
arch/x86/entry/syscall_64.o:(.rodata+0xa78)：对‘__x64_sys_helloworld’未定义的引用
Makefile:1040: recipe for target 'vmlinux' failed
make: *** [vmlinux] Error 1
```

第一次函数定义写错文件后的报错信息

3.3 修改启动项

上述步骤完成后，可以在/boot目录下看到我们编译完的文件，如下。

vmlinuz-4.18.20

```
1 vim /etc/default/grub
```

```
# Written by com.deepin.daemon.Grub2
GRUB_CMDLINE_LINUX=""
GRUB_CMDLINE_LINUX_DEFAULT="quiet splash"
GRUB_DEFAULT=1
GRUB_DISTRIBUTOR=`lsb_release -i -s 2> /dev/null || echo Debian`
GRUB_TIMEOUT="-1"
GRUB_TIMEOUT_STYLE=hidden
~
~
```

对启动项配置进行修改。我因为装了双系统所以这里早就改过了。

GRUB_TIMEOUT是设置引导界面等待时间的，单位为秒。我这里设置“-1”就是无限等待。

如果看到一下语句，需要进行注释，注释后就会显示引导菜单。

```
GRUB_HIDDEN_TIMEOUT=0
```

编辑完后更新启动项

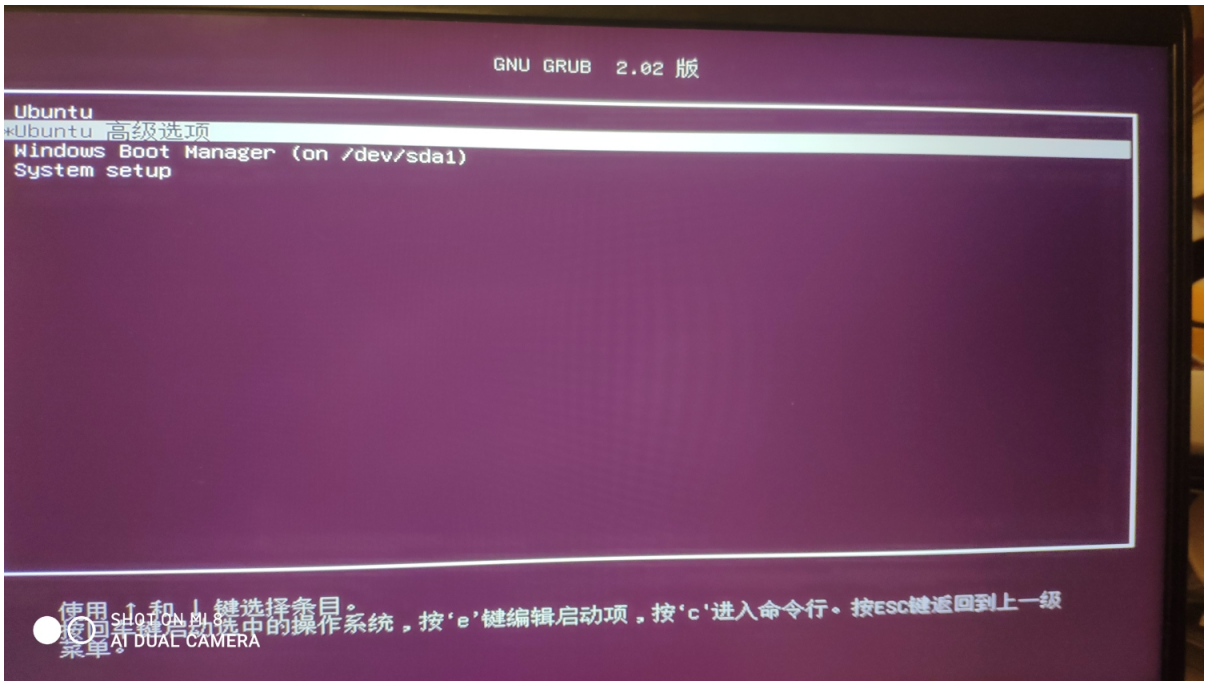
```
sudo update-grub
```

3.4 重启

重启系统

```
reboot
```

选择“Ubuntu 高级选项”



当前选中的Linux 4.18.20 就是我们之前编译后的内核

选择后，如下图

3.5 查看版本号

```
1 uname -r
```

```
(base) luzhan@luzhan-GF63-8RC:~$ uname -r
4.18.20
```

4. 对自定义系统调用进行测试

4.1 编写测试程序

编写我们的测试程序，335就是之前记下的系统调用番号。然后进行编译。

```c
#include <linux/unistd.h>
#define _NR_helloworld 335
int main(void)
{
    syscall(_NR_helloworld);
    return 0;
}
```

```
(base) luzhan@luzhan-GF63-8RC:~/My-Project/OS/EX-4$ vim sys_helloworld.c
(base) luzhan@luzhan-GF63-8RC:~/My-Project/OS/EX-4$ gcc -o sys_helloworld sys_helloworld.c
```

4.2 进行测试

重新打开一个终端， 用dmesg命令打印开机信息。

```
./sys_helloworld
sudo dmesg -c
```

```
(base) luzhan@luzhan-GF63-8RC:~/My-Project/OS/EX-4$ ./sys_helloworld
(base) luzhan@luzhan-GF63-8RC:~/My-Project/OS/EX-4$ sudo dmesg -c
[sudo] luzhan 的密码：
[  672.352360] hello luzhan!
```

可以看到调用成功了。

5. 删除

进入之前那文件夹执行一下命令，把中间文件清理一下，瞬间就空出来16G。

```
sudo make mrproper
sudo make clean
```

内核可以通过如下命令删除

```
dpkg --get-selections|grep linux
sudo apt-get remove linux-image-<具体内核版本>
sudo apt-get remove linux-headers-<具体内核版本>
```

参考链接：

https://blog.csdn.net/qq_33897261/article/details/105029388?fps=1&locationNum=2

https://www.cnblogs.com/tod-reg20130101/articles/9280792.html