
# Linux | 手摸手教你修改编译安装linux内核

Jackson wang
you are not alone





## 前言

这篇文章写于两年前，当时联创夏令营时做的一个任务，大概就是需要我们修改linux内核的网络协议栈，然后达到发送特定的网络包可以知道这台电脑是否被后门（安装了我们修改的linux内核）或者在这台电脑上面执行shell命令等。之后为了避免答辩的时候卡壳，所以写了一份提纲，也就是这篇文章啦。

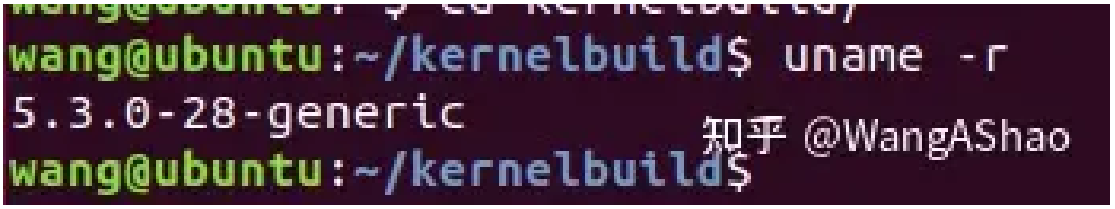
首发于我的csdn



这一篇就作为操作系统专栏的开篇了

### 查看当前的内核版本

```
uname -r
```



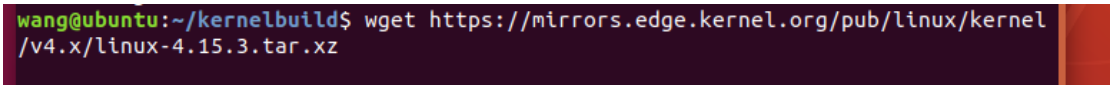

**然后到官网去找一个内核版本进行下载**

传送门

我这是第二次安装，选了一个稍微低一点的版本，第一次选的高版本默认配置编译出来的东西19个G， SSD顶不住了

mirrors.edge.kernel.org...

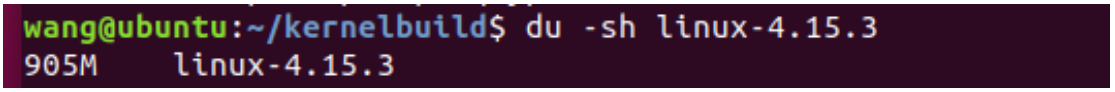
下载的话，wget就可以了




解压的话一条命令就够了：

```
sudo tar -xvJf linux-xxxxx
```

源码还挺大，害怕




然后编译的时候你会遇到一系列的错误，但是概括而言其实错误有两类，一个是缺少必要的包或者工具，再一个是这个版本的内核打了patch，小心处理就好。记得换源，下载快一点。

**然后就需要准备文本配置文件 .config**

其实文本配置文件就是指定启用哪一些模块，以及哪些模块打入内核里面，哪些放在modules文件夹里面

启用的方法可以是直接复制你自己内核里面的配置，也可以用make defconfig默认配置，或者make menuconfig然后save，还有很多，不一一列举了。

我们用默认的就好

```
wang@ubuntu:~/kernelbuild/linux-4.15.3$ make defconfig
  HOSTCC  scripts/basic/fixdep
  HOSTCC  scripts/kconfig/conf.o
  SHIPPED scripts/kconfig/zconf.tab.c
  SHIPPED scripts/kconfig/zconf.lex.c
  HOSTCC  scripts/kconfig/zconf.tab.o
  HOSTLD  scripts/kconfig/conf
*** Default configuration is based on 'x86_64_defconfig'
#
# configuration written to .config
#
wang@ubuntu:~/kernelbuild/linux-4.15.3$
```



```
wang@ubuntu:~/kernelbuild/linux-4.15.3$ wc -l .config
4353 .config
wang@ubuntu:~/kernelbuild/linux-4.15.3$
```

这个默认配置才四千多行，比5.x的一万多行小太多了，估计编译出来的也要小不少

然后make mrproper清除编译中产生的中间文件，当你编译失败然后解决问题重新编译后可以运行一下

```
wang@ubuntu:~/kernelbuild/linux-4.15.3$ make mrproper

  CLEAN   scripts/basic
  CLEAN   scripts/kconfig
  CLEAN   include/config include/generated
  CLEAN   .config
```

啊，把配置文件也删了，太狠了

接下来很多命令要root权限，我们直接进入root模式吧

```
wang@ubuntu:~/kernelbuild/linux-4.15.3$ sudo su -
[sudo] password for wang:
root@ubuntu:~#
```

### make开始编译

这里由于我给虚拟机分了两个cpu，所以我就make -j 2了，快一点

然后就开始编译了，耐心等待即可，可以去看一部电影或者喝一杯卡布奇诺

```
  CC      /home/wang/kernelbuild/linux-4.15.3/tools/objtool/subcmd-config.o
  LD      /home/wang/kernelbuild/linux-4.15.3/tools/objtool/libsubcmd-in.o
  AR      /home/wang/kernelbuild/linux-4.15.3/tools/objtool/libsubcmd.a
  GEN     /home/wang/kernelbuild/linux-4.15.3/tools/objtool/arch/x86/lib/inat-t
ables.c
  CC      /home/wang/kernelbuild/linux-4.15.3/tools/objtool/arch/x86/decode.o
  LD      /home/wang/kernelbuild/linux-4.15.3/tools/objtool/arch/x86/objtool-in
.o
  CC      /home/wang/kernelbuild/linux-4.15.3/tools/objtool/builtin-check.o
  SYSHDR  arch/x86/include/generated/uapi/asm/unistd_32.h
  SYSHDR  arch/x86/include/generated/uapi/asm/unistd_64.h
  CC      /home/wang/kernelbuild/linux-4.15.3/tools/objtool/builtin-orc.o
  SYSHDR  arch/x86/include/generated/uapi/asm/unistd_x32.h
  CC      /home/wang/kernelbuild/linux-4.15.3/tools/objtool/check.o
  CC      /home/wang/kernelbuild/linux-4.15.3/tools/objtool/orc_gen.o
  CC      /home/wang/kernelbuild/linux-4.15.3/tools/objtool/orc_dump.o
  CC      /home/wang/kernelbuild/linux-4.15.3/tools/objtool/elf.o
  CC      /home/wang/kernelbuild/linux-4.15.3/tools/objtool/special.o
  CC      /home/wang/kernelbuild/linux-4.15.3/tools/objtool/objtool.o
  CC      /home/wang/kernelbuild/linux-4.15.3/tools/objtool/libstring.o
  CC      /home/wang/kernelbuild/linux-4.15.3/tools/objtool/str_error_r.o
  LD      /home/wang/kernelbuild/linux-4.15.3/tools/objtool/objtool-in.o
  LINK    /home/wang/kernelbuild/linux-4.15.3/tools/objtool/objtool
```



神奇，怎么4.x版本安装得这么快，十分钟吧就安装好了，而且，为什么这么小，不管了，先试试吧

```
  LD      arch/x86/boot/compressed/vmlinux
  OBJCOPY arch/x86/boot/vmlinux.bin
  ZOFFSET arch/x86/boot/zoffset.h
  AS      arch/x86/boot/header.o
  LD      arch/x86/boot/setup.elf
  OBJCOPY arch/x86/boot/setup.bin
  BUILD   arch/x86/boot/bzImage
Setup is 15548 bytes (padded to 15872 bytes).
System is 7789 kB
CRC cd86856e
Kernel: arch/x86/boot/bzImage is ready  (#1)
wang@ubuntu:~/kernelbuild/linux-4.15.3$ du -sh .
1.3G    .
wang@ubuntu:~/kernelbuild/linux-4.15.3$
```



### 编译完成后就make modules_install

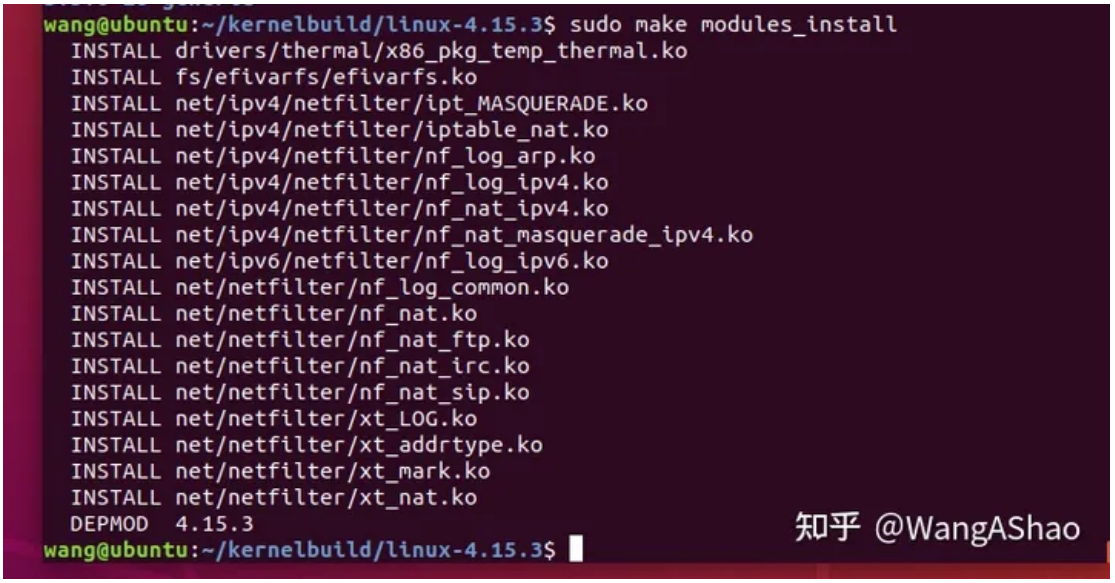
这一步是安装内核模块，他会把这些模块安装到 /lib/modules这个目录里面

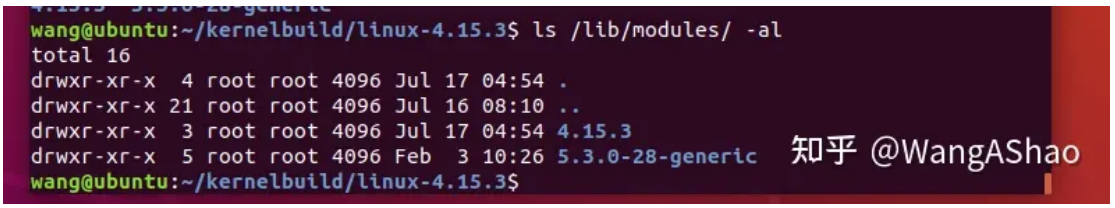
可以看到现在这个目录下面一开始只有我当前内核得模块得文件

```
wang@ubuntu:~/kernelbuild/linux-4.15.3$ ls /lib/modules/
5.3.0-28-generic
wang@ubuntu:~/kernelbuild/linux-4.15.3$
```

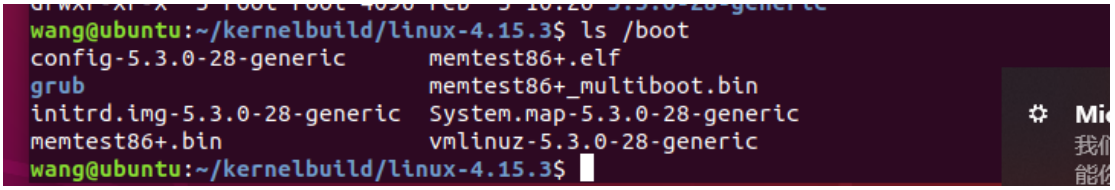
然后咱们执行以下这个模块安装命令试试

这什么情况？怎么一秒就安装好了。。。




但是现在内核文件还没有生成

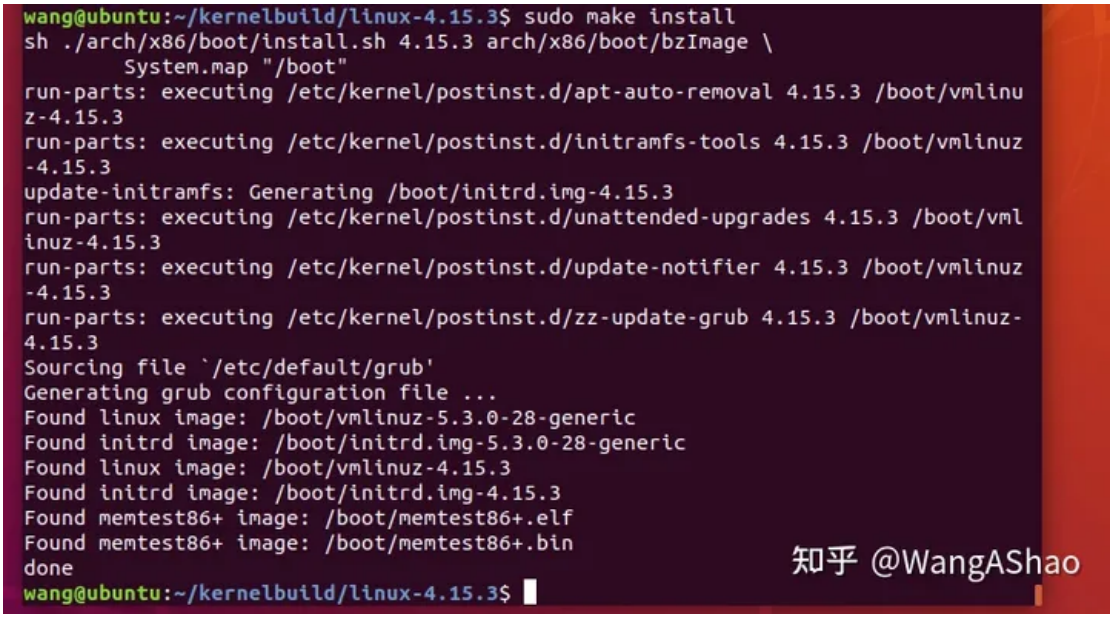



我们接下来生成内核文件

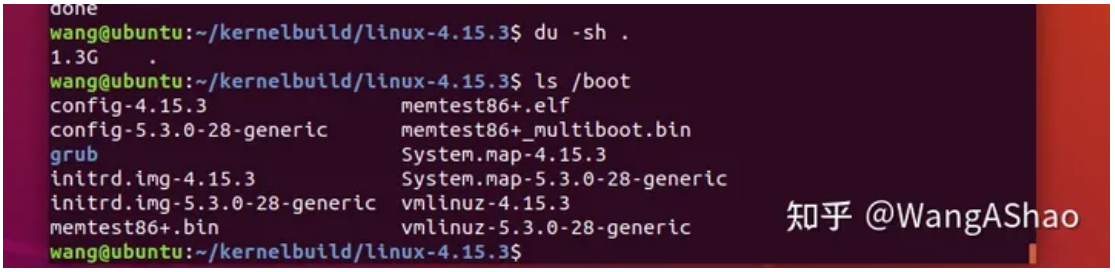
### 再然后就make install

这一步就是安装内核了，这一步具体干了什么事情呢?

安装了内核相关的模块，安装bzImage, 生成initramfs文件以及会修改grub的配置文件，但是你还得设置成开机选择内核版本， 编辑 `/etc/default/grub` 这个文件，不过不同版本的不太一样，我之前的5.x的是修改两个条目，其中一个改成menu，还有一个是等待时间，我改成了10




我傻了，怎么安装得这么快，算了，先试试吧




```
GRUB_DEFAULT=0
GRUB_TIMEOUT_STYLE=menu
GRUB_TIMEOUT=10
GRUB_DISTRIBUTOR=`lsb_release -i -s 2> /dev/null || echo Debian`
GRUB_CMDLINE_LINUX_DEFAULT="quiet"
GRUB_CMDLINE_LINUX="find_preseed=/preseed.cfg auto noprompt priority=critical locale=en_US"
```

然后`sudo update-grub`

然后重启一下试试


可以启动，但是是一个最小的linux，图形界面都没有，我就说为啥编译这么快，我人傻了


然后查看一下版本，发现确实没毛病


**暂且不管这个busybox是否支持网络功能，我先用它来做一下实验，看能否打印我的printk**

这个是在init/main.c下面的start_kernel函数下面

```
    boot_cpu_init();
    page_address_init();
    pr_notice("%s", linux_banner);
    printk("******************By WJP***************\n");
    printk("\nfuck linux!\n\n");
    printk("******************By WJP***************\n");
    setup_arch(&command_line);
    /*
     * Set up the the initial canary and entropy af
     * and after adding latent and command line entropy.
```



然后重新make，增量编译很快就结束了

```
  AS      arch/x86/boot/header.o
  LD      arch/x86/boot/setup.elf
  OBJCOPY arch/x86/boot/setup.bin
  BUILD   arch/x86/boot/bzImage
Setup is 15548 bytes (padded to 15872 bytes).
System is 7789 kB
CRC 93176a24
Kernel: arch/x86/boot/bzImage is ready  (#2)
wang@ubuntu:~/kernelbuild/linux-4.15.3$
```

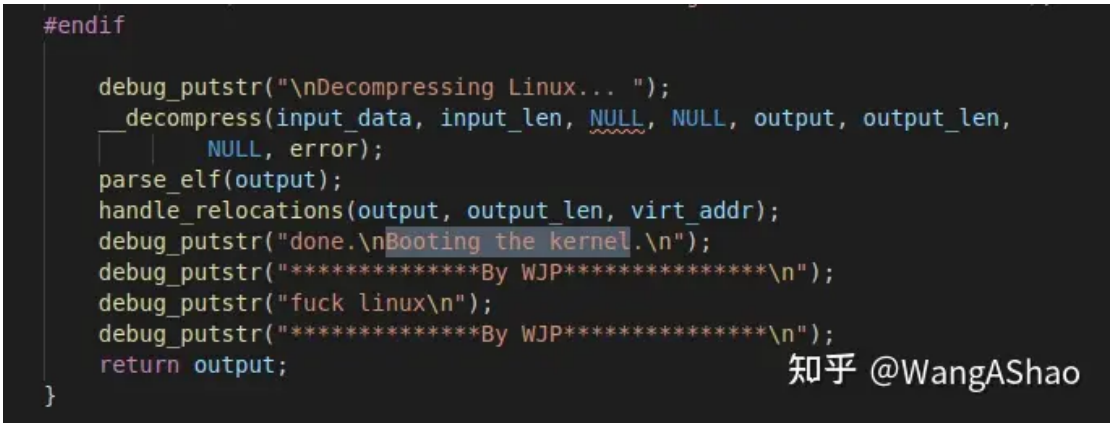


然后重复上述步骤后，发现真的出现了一个新的内核

```
wang@ubuntu:~/kernelbuild/linux-4.15.3$ sudo update-grub
Sourcing file `/etc/default/grub'
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-5.3.0-28-generic
Found initrd image: /boot/initrd.img-5.3.0-28-generic
Found linux image: /boot/vmlinuz-4.15.3
Found initrd image: /boot/initrd.img-4.15.3
Found linux image: /boot/vmlinuz-4.15.3.old
Found initrd image: /boot/initrd.img-4.15.3
Found memtest86+ image: /boot/memtest86+.elf
Found memtest86+ image: /boot/memtest86+.bin
done
wang@ubuntu:~/kernelbuild/linux-4.15.3$
```

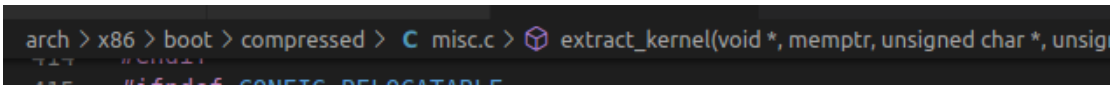


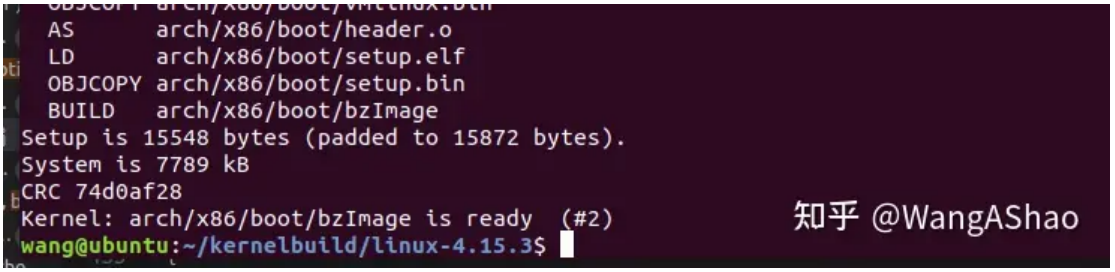
然后我发现我添加的那里貌似不是一个好地方，我按照这个里面的字符串查找了一下，有了新发现



哈哈，就在这里添加了，fuck linux

对了，还要注意我们是x86的架构，所以要选对文件，所以是这个文件




编译成功了，开森！


哈哈，成功了，与内核达成了同步！

接下来做一做计网lab，打算手撸一个TCP协议，不然内核里面的TCP协议根本看不懂嘛！然后就可以撸后门，shell命令了。

发布于 2021-12-18 12:17

Linux 内核 Linux 操作系统内核

写下你的评论...

还没有评论，发表第一个评论吧

**文章被以下专栏收录**



**你管这破玩意叫操作系统**
源码面前没有magic可言

**推荐阅读**

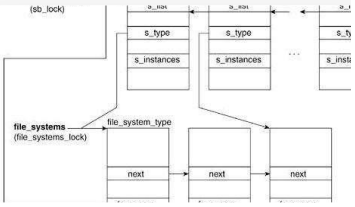
**如何编译 Linux 内核**

Linux...　　发表于Linux...

**Linux内核编译及添加系统调用（详细版）**

实验一：Linux内核编译及添加系统调用（HDU）花了一上午的时间来写这个，良心制作，发现自己刚学的时候没有找到很详细的，就是泛泛的说了下细节地方也没有，于是自己写了这个，有点长，如果…

Linux内核园




**深入详解Linux内核中（内核和伪文件系统）**

Linux嵌入式