编译Linux内核

## 实验目的

学习重新编译Linux内核，理解、掌握Linux内核和发行版本的区别。

## 实验内容

在Ubuntu18.04的Linux系统里，下载并重新编译其内核源代码，并手动安装

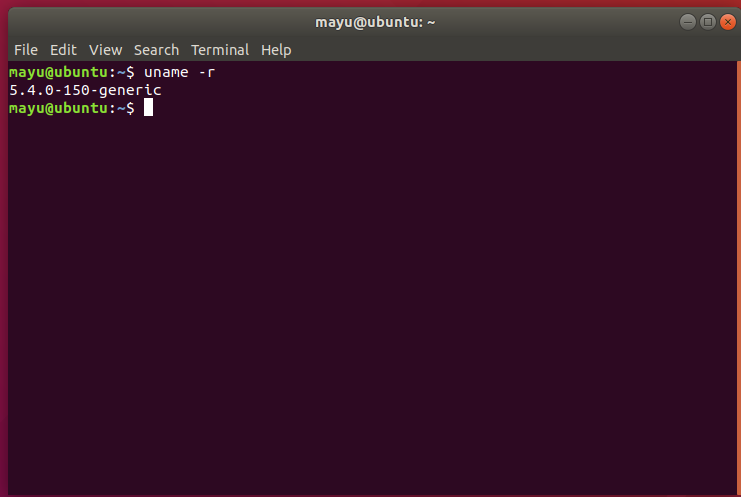
然后，配置GNU的启动引导工具grub，成功运行重新编译成功的Linux内

核。

## 实验过程

### 1.1下载部署内核源代码

#uname -r查看当前内核版本



在kernel.org里下载5.4.0版本（对应你的原本内核版本，防止一些奇怪的适配错误）内核源码，放在桌面（某个位置），解压

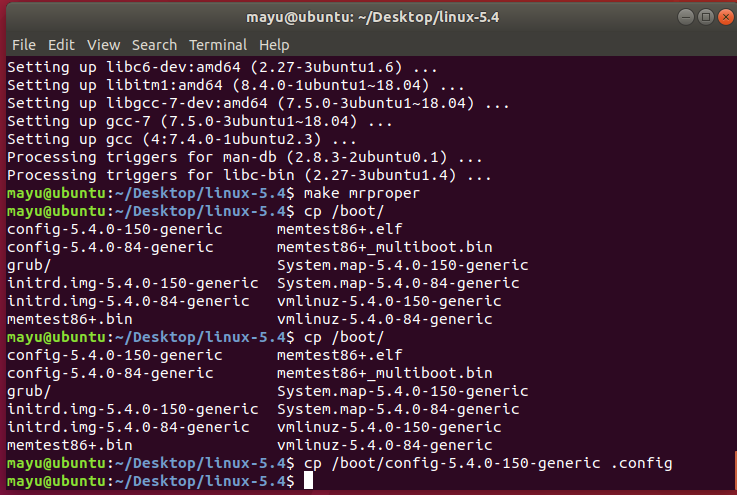
### 1.2 配置内核

切换到内核源代码代码所在目录，执行：

#make mrproper //这条命令清理编译的配置和.o文件

#cp /boot/config-5.4.0-150-generic .config//将旧的内核的配置文件拷贝

#make oldconfig//在旧的配置基础上生成配置文件



* 在这一步遇到了找不到flex的错误，只需要安装

#sudo apt-get install bison  
 #sudo apt-get install flex

参考链接：<https://blog.csdn.net/weixin_43568960/article/details/104565781>

* 遇到scripts/sign-file.c:25:30: fatal error: openssl/opensslv.h: No such file or directory

只需要安装

#sudo apt-get install libssl-dev

参考链接：<https://stackoverflow.com/questions/46008624/how-to-fix-fatal-error-openssl-opensslv-h-no-such-file-or-directory-in-redhat>

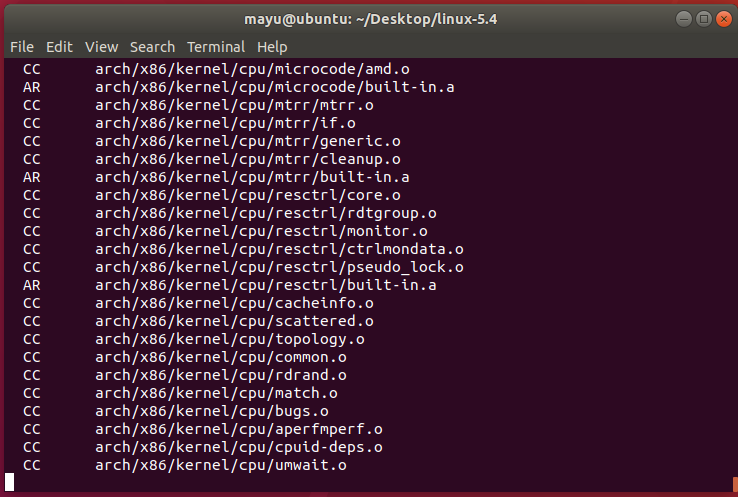
### 1.3编译内核和模块

#make bzImage

bzImage是Linux内核的一种格式，它是一种压缩的内核镜像文件，可以用于引导Linux操作系统。这个文件包含了操作系统内核的代码和必要的数据结构，以及引导操作系统所需的启动代码。

**作用**：make bzImage命令的主要作用是编译和构建Linux内核的启动镜像。这个启动镜像可以用来引导计算机并加载Linux操作系统。该命令会执行以下操作：

* 编译内核源代码：将内核源代码文件编译成可执行的二进制文件。
* 创建bzImage文件：将编译后的内核文件打包成一个压缩的bzImage文件。
* 生成配置文件：根据之前的内核配置文件生成一个新的配置文件，以确保内核编译时使用了正确的选项和参数。



内核正在编译(可以ctrl+c停止，下次会接着编译)

* 如果遇到报错：make[1]: \*\*\* No rule to make target 'debian/canonical-certs.pem', needed by 'certs/x509\_certificate\_list'. Stop.

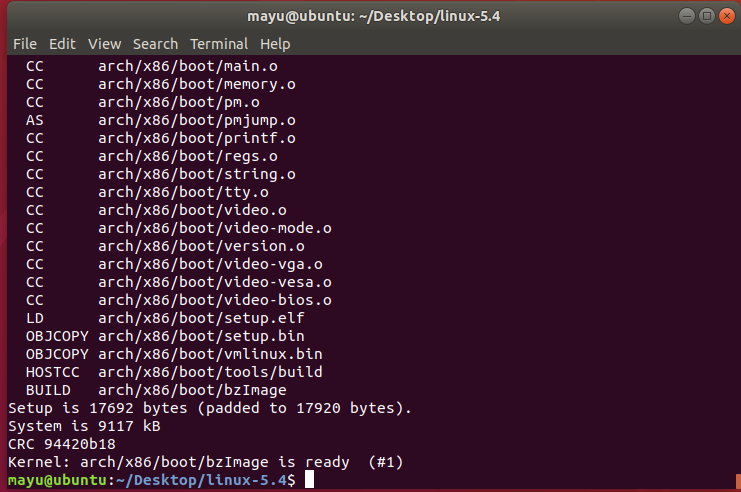
Makefile:1652: recipe for target 'certs' failed

make: \*\*\* [certs] Error 2

将.config文件中的CONFIG\_SYSTEM\_TRUSTED\_KEYS项设置成y，即可：

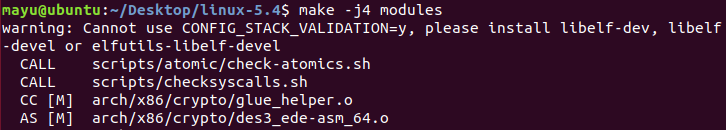
CONFIG\_SYSTEM\_TRUSTED\_KEYS=y

参考：<https://askubuntu.com/questions/1329538/compiling-the-kernel-5-11-11>

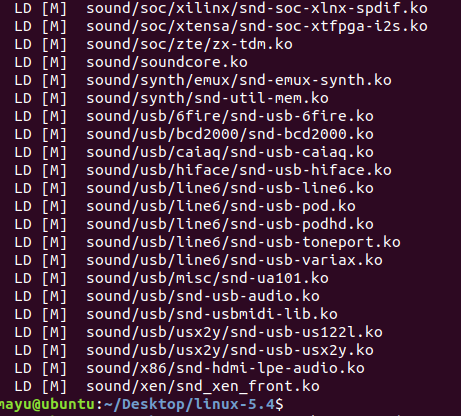


内核编译完成，耗时约半小时（如果在命令后加入 -j 4，四线程会快一点）

在编译模块前，考虑到了线程数的多少和虚拟机的cpu数的关系，如果线程数大于cpu，线程会竞争，参考：<https://blog.51cto.com/u_2982693/3354878>



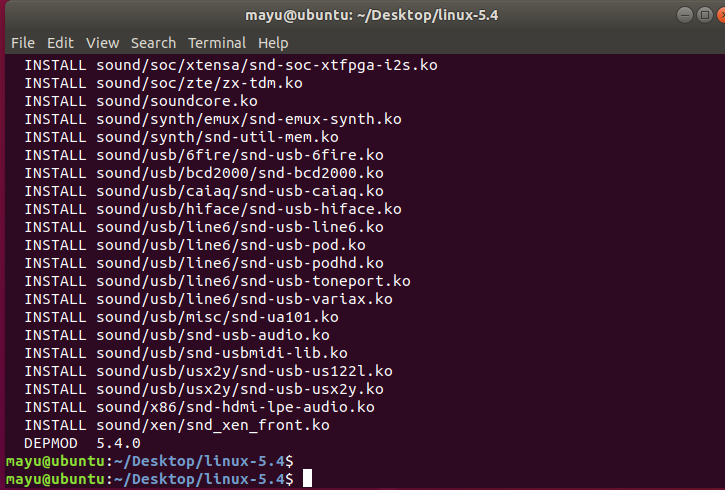
编译模块



编译模块结束，在过程中能看见模块里面包括设备驱动程序、文件系统支持、网络协议栈等。

# make modules\_install

将生成的ko库文件拷贝到指定目录lib\modules。



编译模块结束

**1.4 准备启动文件**

#sudo cp arch/x86/boot/bzImage /boot/vmlinuz-5.4.0-mayu

//Linux 内核在 /boot 目录下以类似 vmlinuz-<kernel-version> 的方式命名。

//目的是将编译好的内核启动镜像复制到 /boot 目录下，以供引导时使用

#sudo cp System.map /boot/System.map-5.4.0-mayu

// System.map：这是 Linux 内核的符号映射文件，它包含了内核中各个符号的地址和偏移量, 以System.map -<kernel-version>命名

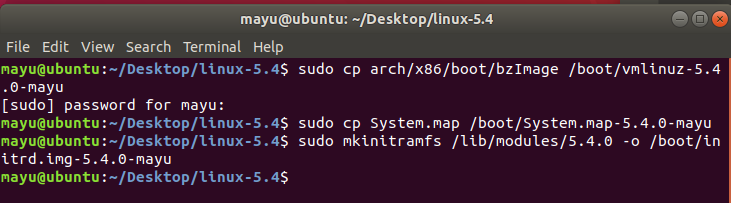
//命令的目的是将符号映射文件复制到 /boot 目录下，以供内核调试和符号查找使用

#sudo mkinitramfs /lib/modules/5.4.0 -o /boot/initrd.img-5.4.0-mayu

//mkinitramfs：这是一个用于创建 initramfs（初始 RAM 文件系统）的命令。

/// initramfs 是一个小型的文件系统，用于在引导时加载必要的模块和文件，以便能够挂载根文件系统并启动系统。

//以initrd.img -<kernel-version>命名



#sudo vim /etc/default/grub

注释掉其中的GRUB\_TIMEOUT\_STYLE=hidden

将TIME从0改到3

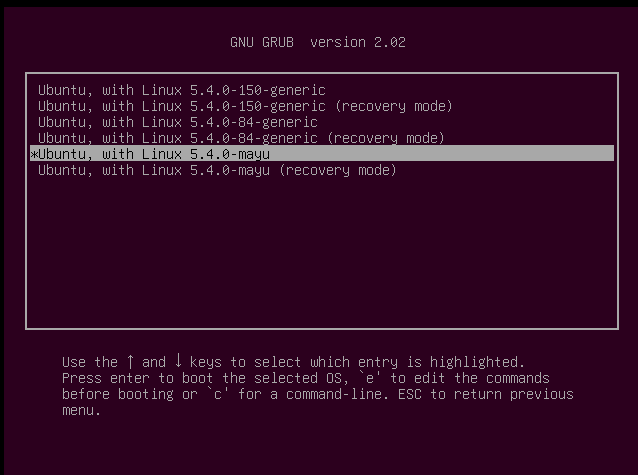
/etc/default/grub 文件是用于配置 GRUB (GRand Unified Bootloader) 引导加载器的默认设置的配置文件。GRUB是一个常用的引导加载器，用于在多个操作系统之间选择引导，并启动特定的操作系统。以下是 /etc/default/grub 文件的主要作用：

1. **引导菜单配置**：/etc/default/grub 文件包含了用于配置 GRUB 引导菜单的各种选项。这些选项定义了引导菜单的外观、默认引导项以及引导时的行为。
2. **默认内核选择**：你可以在该文件中指定默认要引导的内核版本。这是在没有手动选择引导项时，系统将引导的内核版本。
3. **引导菜单外观**：你可以配置引导菜单的颜色、字体、背景图像等外观属性，以满足用户界面的需求。
4. **内核引导参数**：GRUB 支持传递参数给引导的内核。这些参数可以在 /etc/default/grub 中配置，以便在引导时传递给内核。这对于配置系统启动时的各种选项非常有用，例如启用调试模式或禁用图形界面。
5. **引导超时时间**：你可以设置引导菜单的超时时间，即等待用户选择引导项的时间。如果用户没有进行选择，系统将自动引导默认的内核。
6. **其他引导选项**：除了上述内容外，/etc/default/grub 文件还包含其他 GRUB 引导加载器的配置选项，例如启用安全引导、设置引导分辨率等。

修改了 /etc/default/grub 文件后，运行

#sudo update-grub

重启之后选择advanced options for Ubuntu，里面就会有你编译的内核



重启之后也可以看到当前内核版本确实和之前不一样了

#sudo update-grub

