查看自己安装的所有内核版本：

sudo dpkg --get-selections | grep linux就可以看到已经安装的了。

首先，安装，然后使用。

先用虚拟磁盘使用。

<https://blog.csdn.net/p569354158/article/details/45482769>

**这个链接作为入门挂载使用。**

1、首先在自己的工作空间，还是正常地做一个镜像。dd if=/dev/zero  of=./disk.img bs=512M  count=2

2、然后自己想安装 mkfs.nilfs2  ./disk.img   但是提示还没有安装nilfs模块呢，因此，此时自己使用了sudo apt-get install nilfs-tools

多写一个， 卸载为：

sudo apt-get autoremove xxx.

只要执行了一次格式化mkfs.nilfs2  由于就是实际的块内容里面写入superblock等信息，所以即使关机，这个.img里面的内容是不会丢失的，只是挂载丢失了而已。

3、然后重新给虚拟镜像格式化成 nilfs2文件格式， 然后再在第4步骤挂载。 此时默认安装的是nilfs-tools amd64 2.6.7-1版本。

4、对于虚拟的首先  losetup /dev/loop0 ./disk.img   然后 mount -t nilfs2 /dev/loop0 /mnt/nilfs  后面这个目录是自己建立的目录。

5、只是唯一的问题是，在进行umount的时候会出现占用，无法umount而已。

6、当上述都完成了，那么由于lscp是专门用于nilfs的命令，因此，是可以在完全lscp后面没有指令的时候，会进入/proc/mounts中查找nilfs文件系统的。

上述，cp就是checkpoint。lscp默认输出NBLKINC 而非BLKCNT

lscp -b

其中lscp的输出中，ss是snapshot，cp是checkpoint。

**2018年10月15日**

新的疑问以及需要做的操作。

首先这个文件系统，自身是驱动类型，还是说仅仅utilization tools是驱动类型？？

还有由于文件系统不是说是前面自己看的块设备驱动，因此，原来块设备驱动的module格式已经不太适合看文件系统了，所以文件系统怎么看？？

自己下载的源码，仅仅源码就可以了吗？？因为在github上明显看到是，redhat类型的，是否可以在ubuntu这种debain系统类型下看。

需要用到的命令有，

上述自己直接从库中下载，而不是用源代码编辑，因此，下载的.deb安装包是在/var/cache/apt/archives/里面，可以看到，自己确实下载的是 nilfs-tools包。给人感觉，应该nilfs2文件系统本身是已经在操作系统中了？？那怎么查看是否已经在操作系统中了？？

而dpkg -c -L 这两个选项，分别是显示.deb包里面会有什么东西。

现在最大的问题，就是怎么阅读源码，就像应用程序一样，怎么运行以及调试。这个怎么做。

假如说我们从mkfs，入手，开始往后追踪，但是怎么追踪？？是由于已经安装的.ko文件？？

框架上面：

我们知道，vfs是提供给用户的接口，而系统调用要识别，用户态和内核态，因此，文件系统层是为了识别vfs层的接口的，但是在文件系统层，怎么设计，以及怎么挂接上vfs。 这个挂接上怎么调用？？就是想像学习驱动一样学习的。都不行。

好像好多人说的，文件目录的这种文件系统，与真正自己想看的实际nilfs这种文件系统，好像不同。。

**而文件系统又与设备驱动的关系是**，设备驱动自己定义自己设备的xxx\_open(), xxx\_read()， xxx\_write(), xxx\_close()函数，被vfs调用，因为确实是，每一个部分都是注册好，然后当用户要访问磁盘数据，那么磁盘怎么被打开，是磁盘驱动程序的事情，而文件系统只是按照自己的构造，最终告诉磁盘，你读哪些内容，或者写哪些内容即可。

怎么查看已经有的驱动？？

就是说，nilfs都已经被包含在linux的内核中了，而目前缺少的是nilfs-utiities这种tools,就是用户命令包，mkfs.nilfs2 xxx 这个命令，但是对于f2fs-tool有，但是仍有模块存在，有<https://blog.csdn.net/a993096281/article/details/79654208>这个的模块化编译和使用，其实两者结合才可以调试，因为tools是用户的通用指令，而实际源码中，才确实是文件系统自身的代码。可以理解tools是以驱动模块形式的用户命令。

目前自己知道了，正常的/usr/src中是含有源码的，但是由于安装的时候，不需要源码，因为自己不需要编辑的，因此目录里面确实没有源码。

而想要源码，可以从其他地方下载，也可以使用命令：

sudo apt-get install linux-source.

当成一个软件安装的即可。

其他路径需要知道的是：

usr一般是系统文件，里面基本上最大。而源文件，就是usr/src  以及用户的库文件为/usr/lib中。而自己以前编译的时候，使用的启动文件路径在哪？

那也很简单，就是在/boot路径下而已。

在编辑新的内核之前，自己先使用，github上传。但是不再使用git clone的那种，那么怎么对于一个完全新的文件夹进行处理？？

步骤如下：

至少有个git的配置文件，有正确的路径。

1、git init   2、git remote add origin https://github.com/lxclxclxc/ssdsim\_version1 如果已经有了，那么git remote rm origin 然后再设置即可。当然可以直接进入.git中去删除配置的。

3、然后本地添加，git add \*     git commit -m " upload"

但是由于自己上传的时候，出现还没有配置用户名和密码。因此：**git config --global user.email  以及git config --global user.name**

4、不管配置什么，自始至终，都没有能够仅仅使用命令来创建新的repository.真是够可以的，使用的git remote add xxx仅仅是做了链接。

5、git push 使用的基本命令是：git push origin 本地分支名称

6、创建本地分支的方法是：直接git branch newfenzhi  即可。

7、自己里面的文件夹怎么都不能上传，使用git add .r然后再使用git commit -m ""  这个总是报错。

changes not staged for commit.

其实还是外国人给出的答案靠谱，因为基本上自己修改的都是从其他git上面下载的，然后，这些子文件夹里面其实都含有.git文件夹，这时候自己在最外层使用git add . 以及 git commit -m ""就会出错。。。。**要将原来旧的.git给删除了，不然会去跟踪的**。

8、但是对于文件夹仍然不能上传，最后发现，因为原来的错误，导致git add 的文件夹名字相同，以为是commit了，其实需要使用git rm -r --cached 想删除掉在版本管理中的缓存，一定要按照这个，不然git rm就是文件都删除了。

git rm -r --cached \* 就是重新开始使用git add -A了。  **但是在git rm -r --cached之后，要重新做个git commit -m "" 这样才是真正更新了**，然后，再重新git add \*tianjiann 才可以的。

9、对于出现的git unable to create ...这种需要删除.git/index.lock文件才可以。     git push origin : xxx分支名字

就是删除，如果省略git push就是默认推送到master. 只有明确指定git push origin xxx分支名字     这样才可以。

10、删除本地分支的 git branch -D bendifenzhimingcheng

11、搞半天，这种文件大的情况需要自己主动将commitid这种错误删除了，真的是烦人的很！！！

12、git branch -m 新的名字    这个是将分支名字更换的-m

13、最后还是重新做了一遍，才行，原来的.git全部删除，重新做完全不复杂，因为原来出现的问题太多了，因此很乱，将文件处理好，大的删除，那么重新开始，就两步骤，git init， git remote add origin xxxx, git add -A, git commit -m "" git push origin master

**第二、**

以上将软件保存以后，就敢于更改内核了。更改方法是，

sudo apt-get install linux-source

然后在/usr/src内部看看源代码，

然后修改下 fs内部的nilfs，然后看看有没有效果。

但是假如自己的版本不是sudo apt-get install linux-source版本的，因为按照此下载的仅仅是linux-source-3.13.0.tar.bz2版本，

怎么下载指定的版本，下载完了以后，怎么编译重新安装？？？

而在此之前，还有解压文件，那么.tar.bz2  和单独的.bz2解压是不一样的，

就像tar.gz   和单纯的.gz是不一样。

分别是：

<https://www.jb51.net/LINUXjishu/43356.html>

tar -xvf 对应单纯的tar

tar -xjvf 对应.tar.bz2

tar -xzvf 对应.tar.gz

上述两个为什么还是用tar呢？因为tar分别用j z来调用bzip2 和gzip

单独的bz2   bzip2 -d 或者bunzip2 解压

单独的gz用：gzip -d  或者gunzip 解压

好像不太对，就是单独的gzip其实带上-d就是说decompress的意思，而带un的不用带-d就是解压。。

unzip解压 .zip

unrar解压.rar

2018年10月16日 （昨天整个都在理顺 文件系统模块是需要剥离出来的，以及文件系统操作工具）（今天就是实现这个剥离，然后后面按照驱动来调试）

第三、就是怎么很随意地更换内核处理？？

首先基本的清理自己电脑空间时候，需要使用abc=$(ls | grep -v filename1 |grep filename2)这种可以继续后面加|grep这种，将挨个排除好多名字，但是自己如果排除文本中每个字符串的关键词，那么grep -v hao 这个就会将整个nihao这个字符串给排除了。

在给abc使用$()赋值完成记得，使用的时候直接$abc不用再加括号了。显示的时候直接echo $abc即可。

自己首先要找到，soruces.list文件位置，在/etc/apt/sources.list

有的太旧了，更换后要执行apt-get update 来更新源头。

2、很神奇的一点，为了看内核自己的module在/lib/modules/kernelversionxxx/kernel/fs中竟然有编译好的module。那就是说确实如果想要这个文件系统，那么只用insmod就行了，不想要rmmod，也不用自己找原始代码进行处理了。但是为了自己读代码，那还是只能使用这个。重新编辑才对。虽然文件系统里面的makefile已经是按照module类型添加了，使用的是make menu中的旧的配置文件，然后再加上专门找到的fs中m的配置即可。<https://blog.csdn.net/qwerrfxgj/article/details/78094598>

3、按照正常来说，自己肯定想下载的是ubuntu工作方修改好的版本源码，那么，问题就来了，怎么找到已经安装上来的版本源码，

看着好像目前ubuntu官网中的 14.04.5LTS（Trusty Tahr）比较成熟？选用该源代码，从ubuntu官网找到，然后下载，既然仓库里面自己也对sources.list修改了，还是没有其他的linux-source版本，因此直接去官网下载了，下载完成移动到/usr/src文件夹里面。

进入下载页面，自己需要找到一个，PC桌面使用版本。**但是自己并不是要下载一个用来安装的iso.torrent这种直接安装镜像，而是要这个版本对应的 linux内核的，要重新换内核，并不是要换ubuntu系统版本。而是仅仅用内核代码，换内核**。所以官网上面的iso.torrent并不是自己的目标。而是原始内核代码才是。

4、自己需要分清楚，此处总结为：

**安装ubuntu操作系统，是包含官方的驱动和内核的。 而目前自己仅仅是想更换内核，（因为fs文件系统已经被内核包含了，而不是像一些板卡，更新完全在驱动）， 而更换内核都有两种方法**，

**一种是：** 对于已经编译好的内核版本直接使用。就是ubuntu给出了他们已经做好了

**二种是：**真正的从原始代码开始编辑，因为自己是为了读源码，所以只能使用第二种方法。

上述两种方法，其实最终的文件要求就是在/boot里面放置内核文件

内核是装在**/boot/**的   vmlinxxx  initridxxx Systemxxx configxxx四部分文件。

1、此处对第一种方法也给出来。（其实无非是第一种方法是，将第二种产生的结果下载下来，然后替换即可）

首先查看已经安装的内容有：

sudo  dpkg  --get-selections | grep linux

这种就显示的linux-header-xxx-generic,  linux-header-xxx, linux-image-xxx， 这种，已经被编译好的，使用过程中是：

sudo apt-get install linux-header-xx ， linux-header-xx-generic , linux-image-xxx

其实其他的linux-xxx-xxx都能直接安装的，只是安装完以后位置在哪而已。

2、对于第二种，其实下载了源码开始编译的是，

1）下载源码

2）cp /boot/config xxxx     当前要编译的源码最外层路径下;

3）想配置的，配置完成，一般带界面的make menuconfig   而普通的是  make config

完全可以修改config文件的内容，也可以使用

4)然后 make modules/ make bzimage

5) 然后分别剩下两个安装： make modules\_install/   make install

上述是完整的操作。本来只阅读，nilfs驱动即可，但是自己为了，将整个关系理顺，因此需要知道整个源代码。

而由于原来的文件系统已经被配置的是module模式，因此没有问题的。自己只用下载下来完整的内核版本，开始阅读即可。。。

这种。

linux-source-3.13.0这种，

4、

 第四、

假如按照apt-cache search linux-source这种显示出来，哪种版本，但是由于source.list太老了，因此，此处只有3.13的可以显示。

而安装的方法是：

目前还是只考虑从仓库下载，那么一般从usr/src也能看到自己的现有的版本内核，但是从命令中怎么得到，为：uname -a一样，使用cat /proc/version

然后就是仍然使用 上述的搜索，apt-cache search linux-source 同样是没有要的内核版本。

**我们首先做的是，先将其从内核中剥离出来，参考，f2fs，就是 专门用insmod nilfs.ko这种安装，然后再 apt-get install f2fs-tools这种工具，就真的可以使用mkfs.f2fs这个命令了。 就可以调试了。就是这回事**。

参考链接， <https://blog.csdn.net/qwerrfxgj/article/details/78094598>

以上所述，确实是核心。

在实现上面的，那么就，需要修改，/etc/apt/sources.list位置在这。

第五、

剩下的，就是看怎么实现的vfs的接口内容，以及nilfs-utilies这种tool的怎么调用也要看。

而第一个问题就是，sudo apt-get install nilfs-tools 这种，是否自动安装insmod nilfs.ko

可以测试下，工具安装后，看是否存在lsmod nilfs

在自己安装nilfs-tools工具之后，并没有安装nilfs2 的模块被安装，但是自己mount -t nilfs2 /dev/loop0  /mnt/nilfs之后，却出现了nilfs2模块，就是在挂载的过程中，系统自动调用的。 mount命令。

而mount完，自己用df -h 并不能看出来每个设备的文件系统内容，使用df -h -T 就是print-type 就是file system type类型。看到是nilfs2类型，所以自己需要做的是，编辑源代码的fs文件，然后替换lib中的nilfs2.ko即可。

以上，已经将整个设计的框架，可以调试的条件说清楚了。

第六、

确实遇到了一个bug，就是使用sudo apt-get install nilfs2-tools

这种可能版本不行，出现了umount device busy。

<https://bugs.launchpad.net/ubuntu/+source/nilfs-tools/+bug/1252063>

这个链接是国外用户提出的，自己umount失败以后，使用了lsof  /mount filename    lsof命令是用于list open files

这个命令显示了进程编号，然后kill -9 进程id  仍然不行，

应该nilfs2.ko和nilfs-tools都使用最新的版本。

下载git clone nilfs-tools的时候，自己由于没有选择分支，因此，默认下载的是master分支。而由于自己并没有设置本地分支，因此git checkout 远程分支A

这个默认还就是下载下来了，当本地已经有 远程分支A  这个名字的话，这个才是本地的切换

Chenl 看的是ocfs2文件系统。

下载下来以后，使用文中的，./config   然后make

但是最严重的的问题出现了，就是：里面根本没有正常的configure和makefile

还算好的一点是，readme里面还给出了需要使用的基本package， 按照他指明的package将其安装。

编译通过的方法。

因此采用如下链接的操作，来处理新的makefile文件是makefile.am

<https://blog.csdn.net/zhbpd/article/details/78179639>

1. aclocal

2. autoconf

3. autoheader

4. automake --add-missing

5. ./configure

6. make

7. make install

项目 https://github.com/kamalmostafa/minimodem 就是这样的一个工程。

./configure 时会出错，提示下面两个模块没有：

No package 'fftw3f' found  
No package 'sndfile' found

这是输入 sudo apt-get install fftw3-dev libsndfile1-dev 即可安装缺失的模块，然后重新 configure 即可。  
其实，出什么错误，都是报出来，缺少什么了，自己还是按照他的要求，所有安装包也不大，全部安装一遍。

**接下来，就安装成功了，**但是安装到哪个位置了？？其实就是问，自己安装的一些命令在哪？应该是在/usr/bin中间的命令，但是后面需要理解的是 这些二进制命令的工作原理，就是调用方法。（一大部分都是从源码读出来的，毕竟是函数操作）

下面是新安装的，步骤

首先安装，automake软件，注意不要选用automake1.11这个可能语法不兼容。。。真是烦人，直接安装用automake即可。

然后提示UUID 缺失。然后，后面将提示的package全部安装了。

并且由于在automake --add-missing报错，因此，执行了，libtoolize  --automake --copy --debug --force 这个命令。

而后面的源码下载，为：

git clone -b v4.18  [https://github.com/torvalds/linux](https://github.com/torvalds/linux/tree/v4.18)  自己选择了4.18版本，而基本在一个版本固定之前会有好多rc版本，这的意思是release candidate. 就是指定了将来可能的候选版本。

当内核源代码下载出来以后，那么就是：参考如下的操作.类似于下面的f2fs模块处理，因为makefile文件如果拷贝出去的话，需要处理一些makefile的配置。

<https://blog.csdn.net/qwerrfxgj/article/details/78094598>

里面的这个配置很有意思：

**KERNELDIR:=/usr/src/linux-4.10.1**

**PWD:=/home/tsy/f2fs**

**default:**

**make -C $(KERNELDIR) M=$(PWD) modules**

clean:

        rm -rf \*.o \*.mod.c \*.ko \*.symvers

上面Makefile中，红色部分是新加的。其中，PWD是在第1步中，把F2FS拷贝出来的路径；KERNELDIR是第1步中编译的源代码的路径（Makefile尽量手打，拷贝可能有错）  
而最新的 内核，由于配置是 nilfs2.o 这种 以及nilfs2-y : 这种是y是编译到内核，而m是编译成模块。    每一步都会遇到以前的问题，就是makefile文件在vim设置的时候，对于tab空格会显示红色警告，需要配置vim为：

**我最大的疑问就是怎么知道这个内核包含了f2fs，因为自己目前只能看module内容，看不到编译到内核中的内容。完整的正确的提问是：**

内核里面有内置模块，是静态的，与可加载内核是不同的，可加载内核可以使用insmod modprobe rmmod modinfo lsmod等命令，

<https://linux.cn/article-6511-1.html>这个链接完美地解决了这个问题

也确实，得能做到这个，因为毕竟，这些需求都是开发人员的基本需求。

命令是：核心好像是检查/lib/modules/xxxx内核版本/modules.builtin这个文件中。而关于内置模块的详细配置，同样可以找到

说是在/sys/module文件夹下面，找到内置的和可加载的都在这里面，每个模块都有parameters这个目录的。

2018年10月17日

**另一个新的问题**，就是下载好了linux的内核代码，怎么做到只指定某一个模块的编辑？？？怎么处理。自己也没有时间追踪顶层 makefile的路径，只指定即可。

<https://www.cnblogs.com/guanguangreat/p/7920378.html>

这个链接说的很清楚。编译模块.ko类型的时候，分为两种，1、在内核源码目录内编译，和 2、内核源码目录外编译。两种。

一般内核驱动的makefile有自己的标准，需要重新温习下，因为也很难说将整个内核从最顶层的，makefile往下追踪，也难。

在源代码的各级子目录中，见到好多obj-$(CONFIG\_XXX) += xxx.o这种就是要利用，make menuconfig之后生成的config文件中的CONFIG\_xxx变量来决定是obj- M 或者obj- y

前者是动态模块，后者是直接在内核内builtin模块。

而实际的生成xxx.ko最终的模式是：obj-$(CONFIG\_xxx) = xxx.o  这种才是最终的，最厉害的。

而给xxx.o进行配置的有三种：

xxx-y := 1.o 2.o 3.o

或者xxx-objs := 4.o 5.o

xxx-$(CONFIG\_ZZZ) += ZZZ.o

第一种和第二种是一定会加入到最后的xxx.ko，而第三种要不要加入仍然要看make menuconfig生成的config文件中的CONFIG\_ZZZ

2、从上述可以看到，第二种验证内核中是否是将某个模块动态加载还是，除了前面的查看 /lib/modules/xxx内核版本    中的moudles.builtin文件，还可以看目前内核版本中/boot  中的config-xxx-generic文件，里面定义的CONFIG\_XXX的内容是m还是y。这样就能知道这个内核在make 的时候是否将这个模块当成动态的，还是内置的了。

目前所有的证明都是：nilfs2都是按照动态模块处理的。因为make menuconfig生成的文件中config是**CONFIG\_NILFS2\_FS=m**

3、在内核中编译，模块，确实是最终的Makefile里面的内容很简单，里面根本没有make -C xx M 这句话，仅仅是模块关系的指定，例如obj-$()+= xx.o

xx-y := 1.o  2.o  3.o等

3、其中vim中红色tab原因，自己解决方法是：<https://zhidao.baidu.com/question/1431903767982172099.html>

是给自己目录中.vimrc文件进行的配置。

set tabstop=4

set expandtab

set smarttab

4、而实际在内核之外编译的时候，需要的完整的Makefile的解释为：

核心句子：make -C xx M=yy modules    //由于-C是选项而 M不是，所以M使用等号M=

结束。其中-C是为了说明，当在内核源码路径之外，那么需要跳到目前内核有效的内部的编译器build，在lib/modules里面，一般xx=

KDIR:= /lib/modules/$(shell uname -r)/build

PWD ：= $（shell pwd）

$(MAKE) -C $(KDIR) M=S(PWD) modules

5、以上搞了半天，原来，假如单独使用这个nilfs2/下面的文件，那么仍然会缺少头文件，可以一个一个拷贝，但是没有意义。因为自己仅仅是想得到，缺少的.ko，那么总不能全部编译，因此有方法单独编译，真正的使用内核架构

<https://blog.csdn.net/beckdon/article/details/40983983?utm_source=blogxgwz0>

虽然不太知道，原理，

这个是实际使用的，<https://blog.csdn.net/xu910121/article/details/53144997>

前面有多余的make oldconfig , make prepare 等因为提示缺少 autoconf.h等kernel configuration is invalid

6、最后实在是配置make oldconfig这种出现新的配置太烦人了，报告出好多缺失的文件，真的好烦人，例如libelf-dev的缺失。

sudo apt-get install  libelf-dev

7、出现的，内核代码中include<linux/xxx.h>这个是哪个位置缺少的？？？

因为自己把它给提取出来了，然后旧的版本与新的版本内核，之间有很大差别。因此提取出来到外面以后，就很麻烦。

8、最后实在没有办法了。。。放弃了，升级内核吧。

分别下载包，然后自动安装。

为：

<https://www.tecmint.com/upgrade-kernel-in-ubuntu/>

里面指出来，

<http://kernel.ubuntu.com/~kernel-ppa/mainline/> 这个网址是主线ubuntu的。

$ wget http://kernel.ubuntu.com/~kernel-ppa/mainline/v4.17/linux-headers-4.17.0-041700\_4.17.0-041700.201806041953\_all.deb

$ wget http://kernel.ubuntu.com/~kernel-ppa/mainline/v4.17/linux-headers-4.17.0-041700-generic\_4.17.0-041700.201806041953\_amd64.deb

$ wget http://kernel.ubuntu.com/~kernel-ppa/mainline/v4.17/linux-image-unsigned-4.17.0-041700-generic\_4.17.0-041700.201806041953\_amd64.deb

但是遇到版本不对的，提示少了libssl1.1版本，但是自己看到的只有。。。小于1.1版本的。所以说ubuntu版本低，真的是问题大堆大堆的，直接换到16.04版本上面吧。虽说不好，但是也是4.13版本内核了，不是3.13版本。真的是什么都下载不下来。

更换到16.04希望可以将nilfs2模块编译通过吧，别再缺少头文件了。。。

$ sudo dpkg -i \*.deb

这样避免使用apt-get 得不到想要的。

最终还是高的版本sudo apt-get install linux-source才可以有新的内容。。。实在是没有办法。

最后，在16.04version上下载4.13版本kernel linux-source-4.13.0

然后拷贝出来，新写上makefile就完成了。没有再报错。

2018年10月18日

1、将所有的源代码权限进行修改，为

chmod -R 777 ./

2、查找普通的gcc的编译路径

echo 'main(){}' | gcc -E -v -

3、按照正常来讲，

gcc的时候，添加的  头文件，是-I  xxx   加链接库文件，使用的是-L xxx

4、但是自己使用的Makefile中的EXTRA\_CFLAGS是不对的，因为单独独立的模块编译，并没有用到这个EXTRA\_CFLAGS当做宏定义，因为这样使用的时候，都是根据顶层使用到这个变量$(EXTRA\_CFLAGS)

所以单独编译还是直接自己添加-I 的好。

5、据自己观察，也就是说，用到的include <linux/xxx.h>就是进入的/usr/src/linux-header-xxxx.generic中的include linux中？？？

6、其实关于module 的编译中，注意，还是很神奇的，对于#include<linux/xxx.h>这种理解，自己查看echo 'main(){}' | gcc -E -v -

这种倒是体现出来了好多路径，感觉应该是builtin里面的，但是应该是module太神奇了？？竟然必须make的时候，主动包含？？ -I  /usr/include/linux   这个路径才行。然后缺少的就需要放置在/usr/include/linux中就行了。

7、当然如果想放置，也可以使用，找到一个好的版本，能运行通过的，然后使用find . -name xxx看哪里缺就补充哪里。

第二种就是，6条说明了，因为，内核模块编译，目前还不知道指定的路劲，而能编译通过的内核中，都是在内核源代码的顶层自己指定编译路径，就是在自己的include/linux/中添加的。而自己将单独的模块拿出来，就只能 将缺少的文件加在系统gcc指定的编译路径中，但是需要自己在makefile中主动添加。

8、虽然配置不难，但是比预想的整整晚了2天。。。。

开始看mkfs.nilfs2这个命令，将linux中用户命令的调用追踪一遍最好了。

注意，用户命令使用的时候，要注意环境变量，而环境变量显示是：

$PATH  前面已经说过了，不用加()而在定义变量value的时候，才用加value = $()这种。而想临时修改方法：

PATH=$PATH:xxxx  这种修改，关闭当前shell窗口就没有了。

而全局临时，为export PATH=$PATH:xxxx

而永久的更改为：在profile文件中修改，在/etc/profile文件中但是   强烈注意的是：ubuntu的PATH设置是在/etc/environment文件中的。是加在其PATH后面的用:加上的。

其实上述两种在文件中的添加方法都对。

但是为了马上生效，除非关机开机，或者source /etc/environment

执行一下。

还可以只改变自己当前用户的环境是~/.bashrc文件在最后加上export PATH=$PATH:xxxxx然后仍然为了修改有效，要对修改后的文件.bashrc执行source命令。而source命令的作用是：

搜索的时候，竟然看到source与sh  bash命令类似，意义为：将source后面的文件读一遍然后执行里面的命令。上面的操作。也算是更新了一下系统设置。

2018年10月19日

今天在虚拟机和台式机之间传输时候，u盘又出问题了，真的是各种小问题都会出来，那就还是使用以前的filezilla软件吧。

所有的方法，行的通的在另一篇文章中

<https://mp.csdn.net/postedit/79533214>

传输好了文件，才开始，真正的看代码，看mkfs.nilfs2 这种指令的工作原理。整个代码的设计方法。

第二部分：

就是实际的vfs理解。

//////////////////////////////////////////\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*////////////////////////////////////////////////////////////////////////////

**终极问题就是，**内核源码编译完以后，其各个编译结果，例如生成的.ko文件都放在哪，怎么从makefile中看出来的。

其中/usr意思是： unix system resource

/usr/lib 与/lib的区别是什么，前者容易迷惑人的原因就是/usr/lib中是应用程序使用到的.so等这种，因此以为是用户的呢。

而/lib中的/lib/modules这个才是自己一直想知道的，.ko以及内核内置模块，而其他在/lib内部的是开机时候用到的函数库。并且/lib中也确实由于是内核模块，是被/bin  /sbin调用的库文件。

而/bin   /sbin就是用户使用的系统命令时候，所调用的函数库，而函数库不可避免要调用/lib中的内容。而/sbin更多的是系统bin命令，

/usr/bin也是用来储存用户命令的，一般是用户定义的命令。而非系统默认命令。例如自己加载的lscp在/usr/bin中。因为用户的应用也是unix system resource一部分，因此是合理的。

我们可以理解/lib是内核级别的。/usr/lib是应用软件级别，以及系统级别的。

/lost+found是用来fsck方式的恢复的，

/initrd 与/boot都是启动 相关的。