**02-Linux操作**

1. **Linux系统**

1.1 **镜像制作**

1.1.1 **x86镜像**

1.1.1.1 **相关链接**

github链接： https://github.com/windy-purple/make\_linux\_iso

<https://www.jb51.net/LINUXjishu/798229.html>

1.1.1.2 **前言**

对于一个极简化的linux系统而言，只需要三个部分就能组成，它们分别是一个linux内核、一个根文件系统和引导。以下是本文制作linux iso镜像所用到的系统和软件：

OS: ubuntu 20

软件: xorriso

1.1.1.3 **制作内核**

1、首先需要去官网选择一个需要的版本下载下来，官网下载地址：

https://mirrors.edge.kernel.org/pub/linux/kernel/

2、利用tar将其解压，然后进入其目录中，然后配置内核，常见的配置有以下几种：

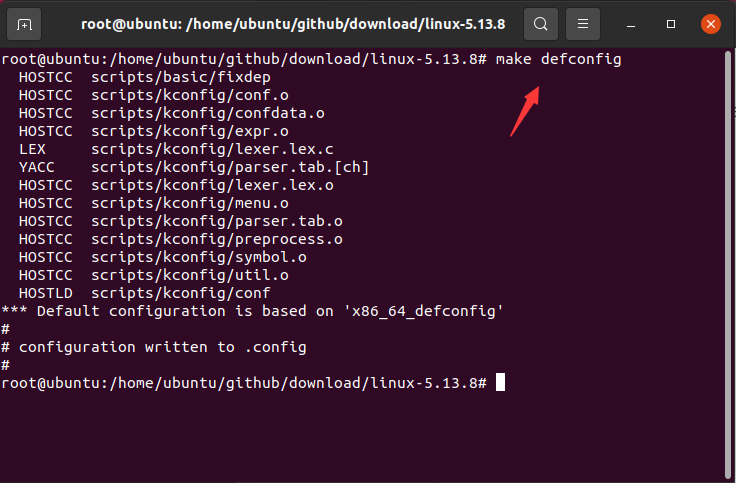
a、make defconfig - 默认配置

b、make allyesconfig - 创建能选yes就选yes的配置

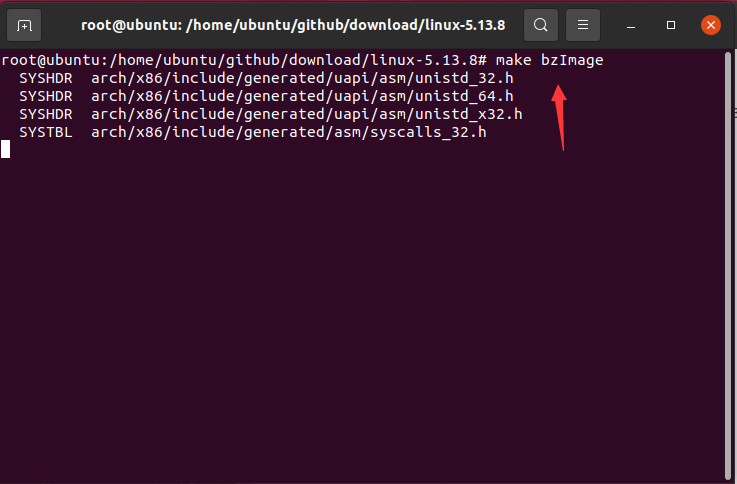
c、make allnoconfig - 创建能选no就选no的配置

d、make menuconfig - 基于ncurser的图形化界面配置

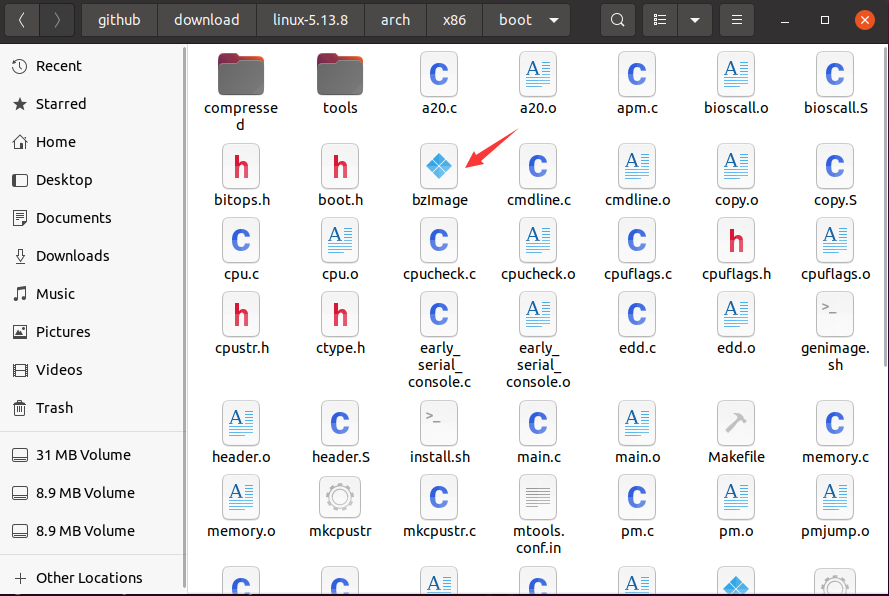
这里采用命令make defconfig使用默认的即可，如下图所示：



然后使用make bzImage命令编译出内核即可，如下图所示：



编译好的内核文件在arch文件夹相应的架构文件夹下面，如下图所示：

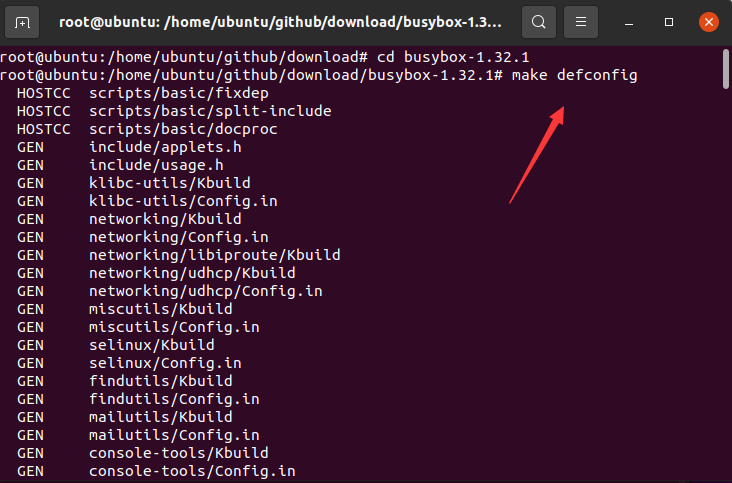


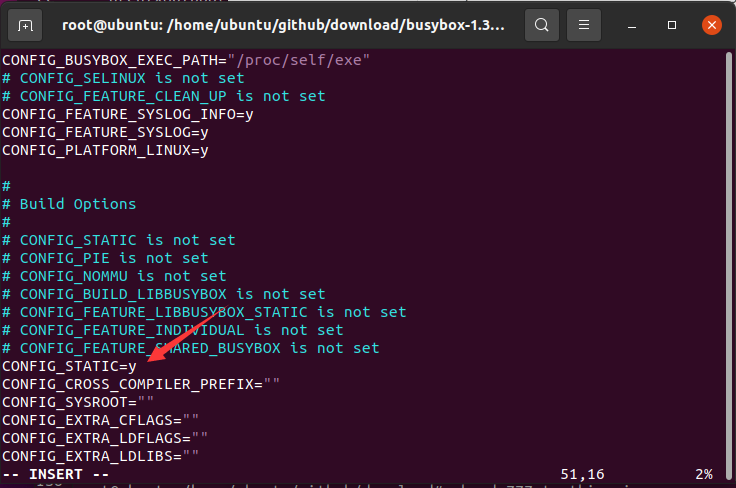
1.1.1.4 **制作根文件系统**

1、我们这里利用busybox来制作一个根文件系统，busybox可以简单理解为一个linux工具的集合。首先还是下载busybox，官网下载地址：

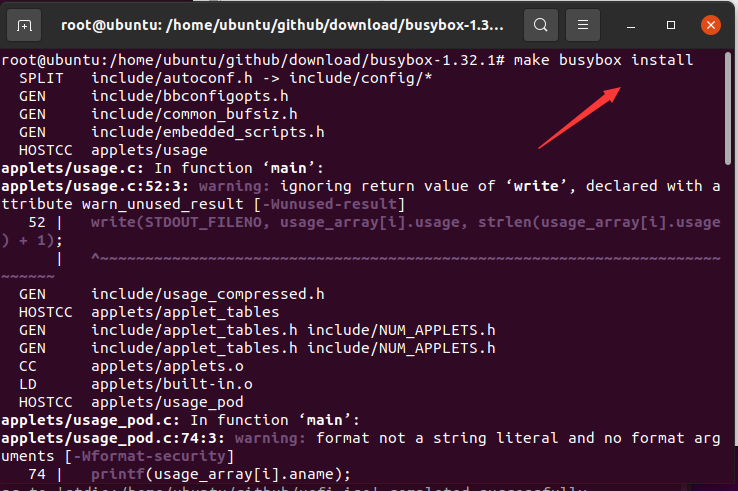
https://busybox.net/downloads/

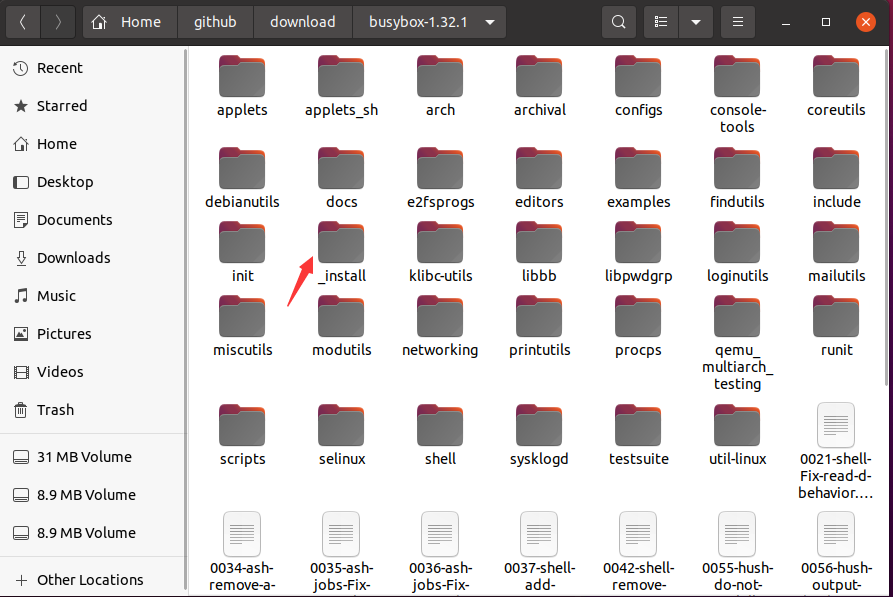
2、编译busybox与编译内核步骤基本一致，将下载好的压缩包进行解压，然后进入文件夹中，使用make defconfig配置默认编译选项，这里需要注意的是，在生成的.config配置文件中，需要设置CONFIG\_STATIC=y，如果没有，添加即可，如下图所示：



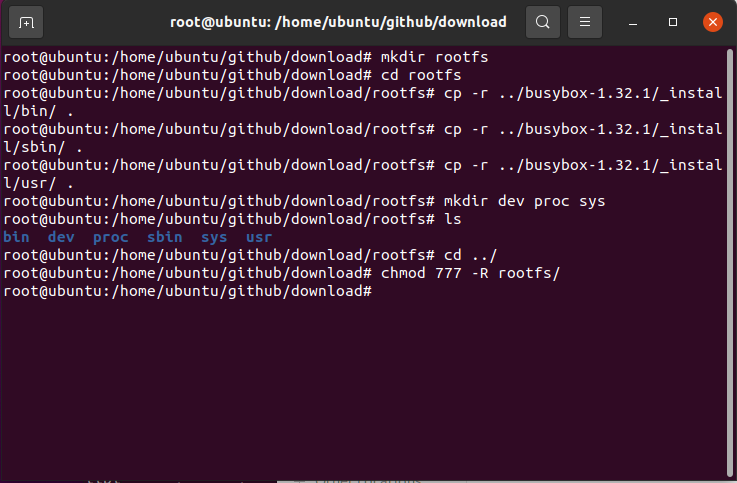


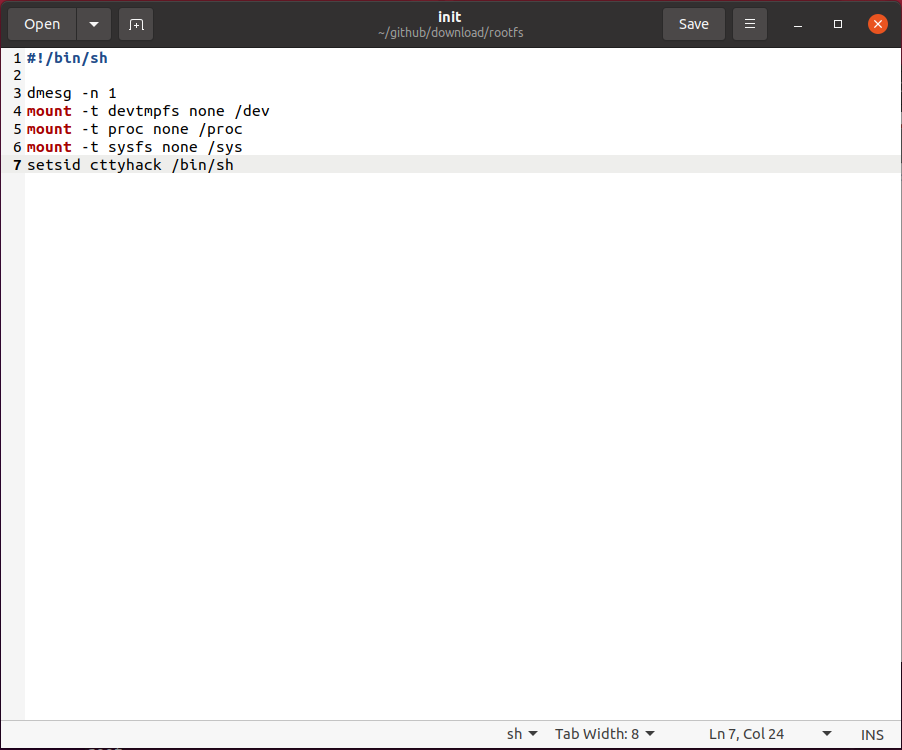
3、然后使用make busybox install命令编译busybox，编译好后会在当前目录下面生产一个\_install文件夹，如下图所示：



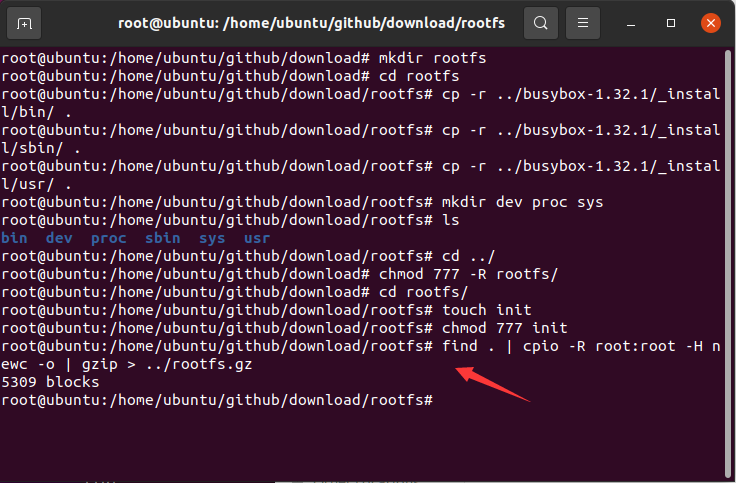


4、然后创建一个rootfs文件夹，并将\_install文件夹下面除linuxxrc以外的所有文件及文件夹都拷贝到rootfs文件夹下面，最后创建dev等文件夹，最后在根目录下面创建init文件即可，文件内容如下图所示：





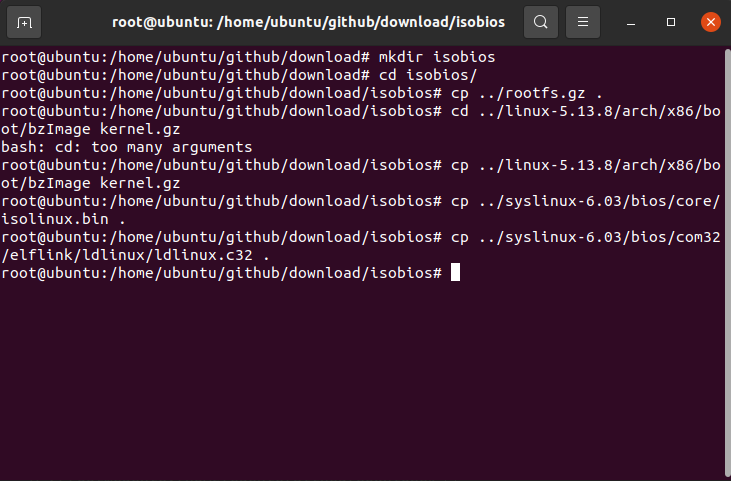
5、最后利用命令find . | cpio -R root:root -H newc -o | gzip > ../rootfs.gz将文件系统打包，至此，一个文件系统就创建完成了，如下图所示：



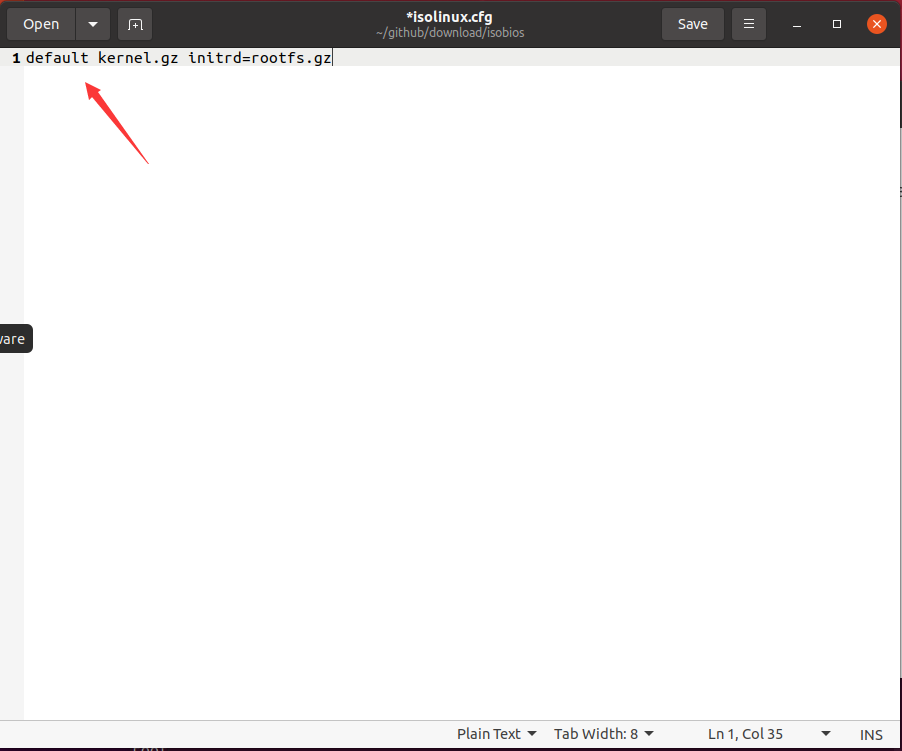
**BIOS**

1、这里我们使用syslinux来创建bios引导的一个linux iso镜像，syslinux官方下载地址如下： https://mirrors.edge.kernel.org/pub/linux/utils/boot/syslinux/

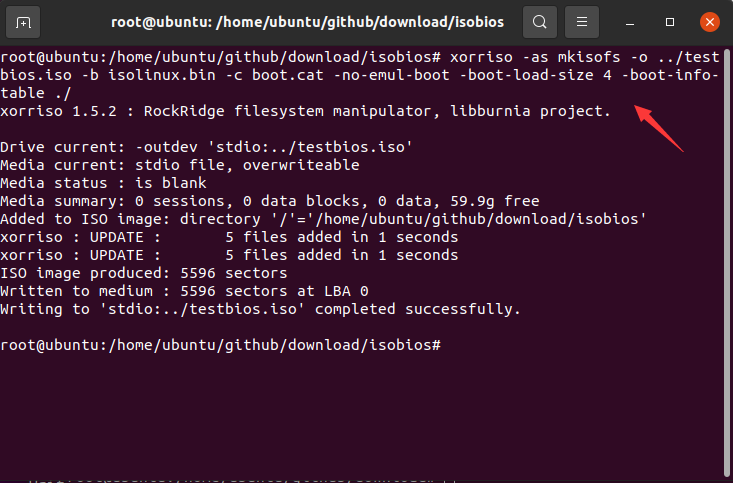
2、将下载好的syslinux解压，然后创建文件夹isobios，将解压后的syslinux文件夹下面的bios/core/isolinux.bin、bios/com32/elflink/ldlinux/ldlinux.c32复制到isobios文件夹下面，如下图所示：



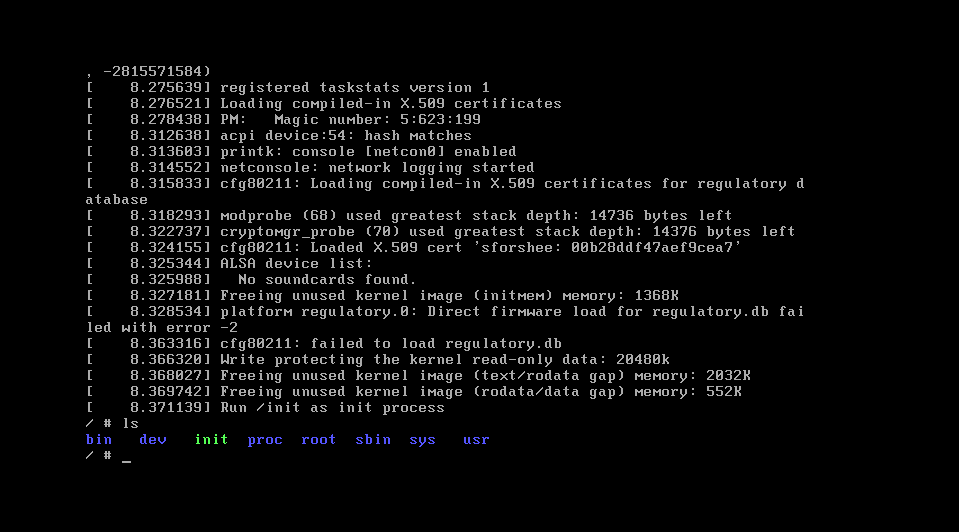
3、在isobios文件夹下面创建配置文件isolinux.cfg，文件内容如下所示：



4、最后，在isobios文件夹下面使用命令xorriso -as mkisofs -o ../testbios.iso -b isolinux.bin -c boot.cat -no-emul-boot -boot-load-size 4 -boot-info-table ./生成iso镜像文件，如下图所示：

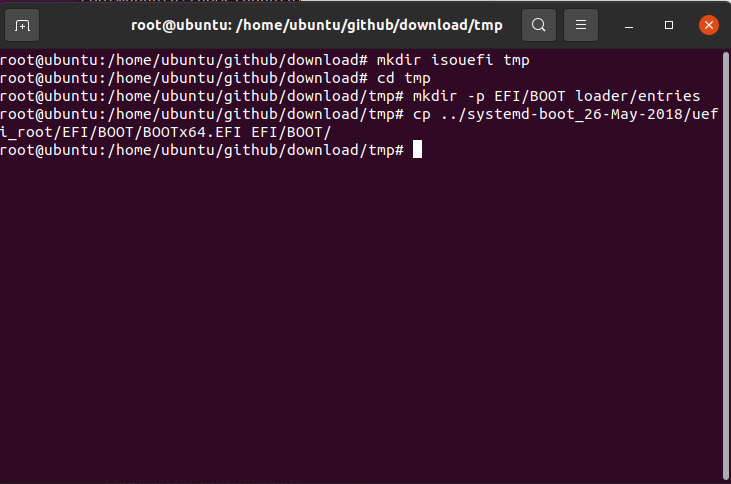


5、使用虚拟机vmware创建一个虚拟机，如下图所示，便是我们创建的一个linux iso镜像跑起来的样子。

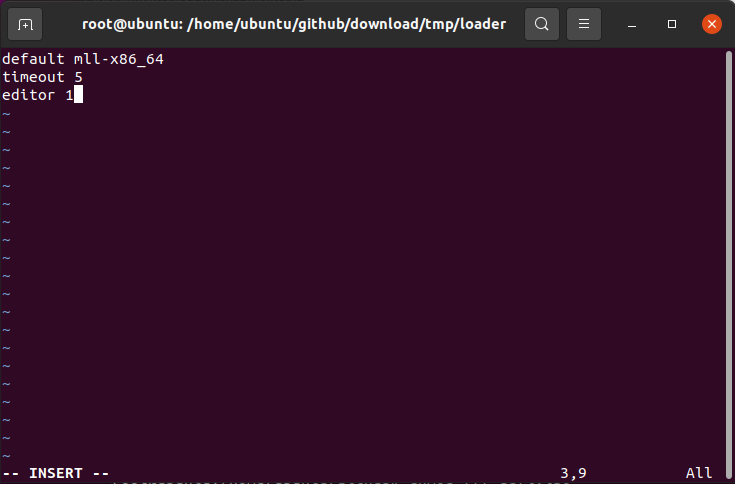


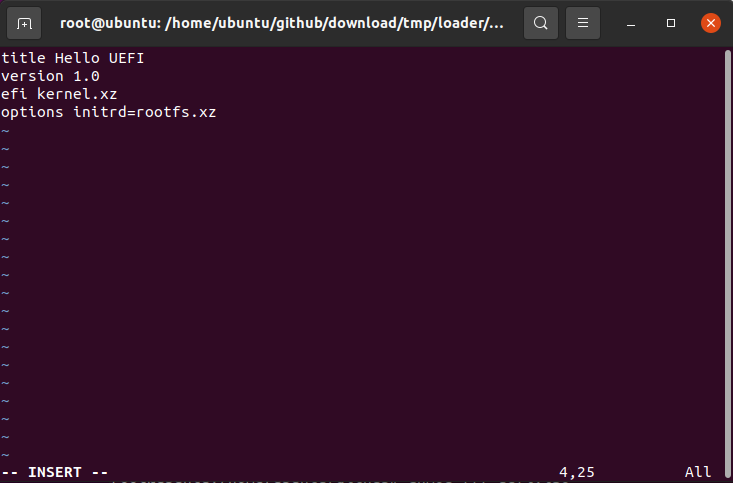
**UEFI**

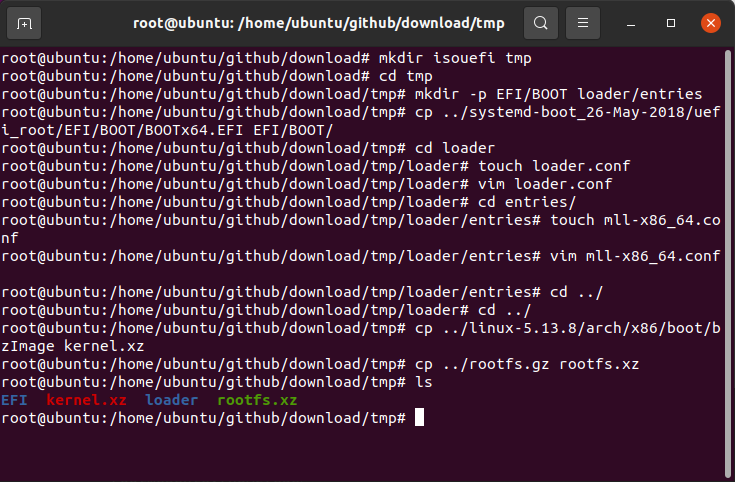
1、uefi这里采用system-boot和syslinux配合来制作，首先，创建两个文件夹isouefi和tmp，其中，isouefi用来挂载设备，tmp文件夹用来临时存放文件以计算大小，然后在tmp文件夹下面创建EFI/BOOT和loader/entries目录，接着，将解压后的systemboot下面的uefi\_boot/EFI/BOOT/BOOTx64.EFI文件拷贝到tmp/EFI/BOOT目录下面，如下图所示：



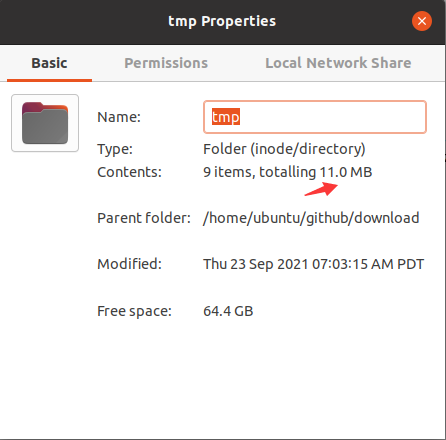
2、接着，在tmp/loader目录下面，创建文件loader.conf配置文件，第一行表示默认配置是entries目录下那个文件，第二行设置默认超时时间；然后在entries文件夹下面创建相应的配置文件，这里是mll-x86\_64.conf，文件内容和bios的差不多，不在单独细说，最后再将前面准备好的内核和文件系统拷贝到tmp目录下面，如下图所示：

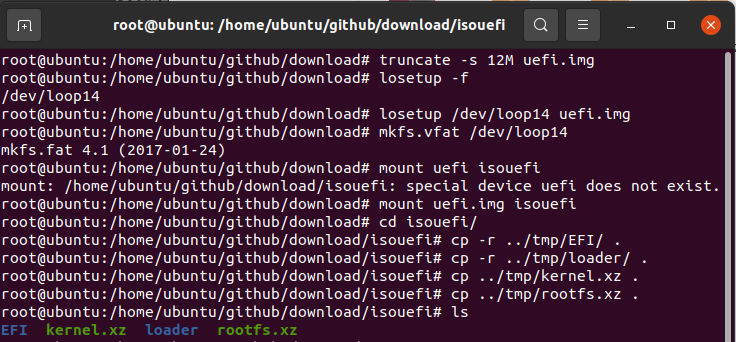




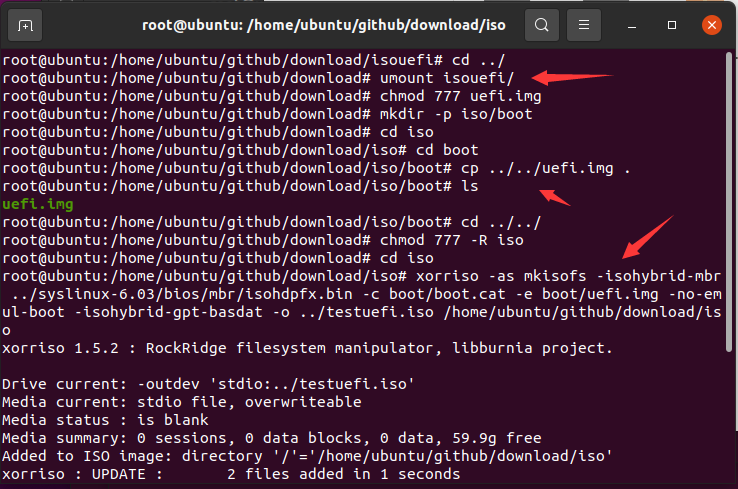


3、此时就可以根据tmp文件夹的总大小创建一个相同大小的img文件了，这里的tmp是11M，为了稳妥起见，这里创建一个12M的img文件，命令为truncate -s 12M uefi.img，然后使用losetup -f命令寻找一个当前未使用的逻辑设备，然后使用losetup命令将我们前面创建的img文件虚拟成改逻辑设备，接着利用mkfs.vfat将该设备格式化成vfat系统，接着使用mount命令将其挂载到isouefi文件夹下面，最后将tmp文件

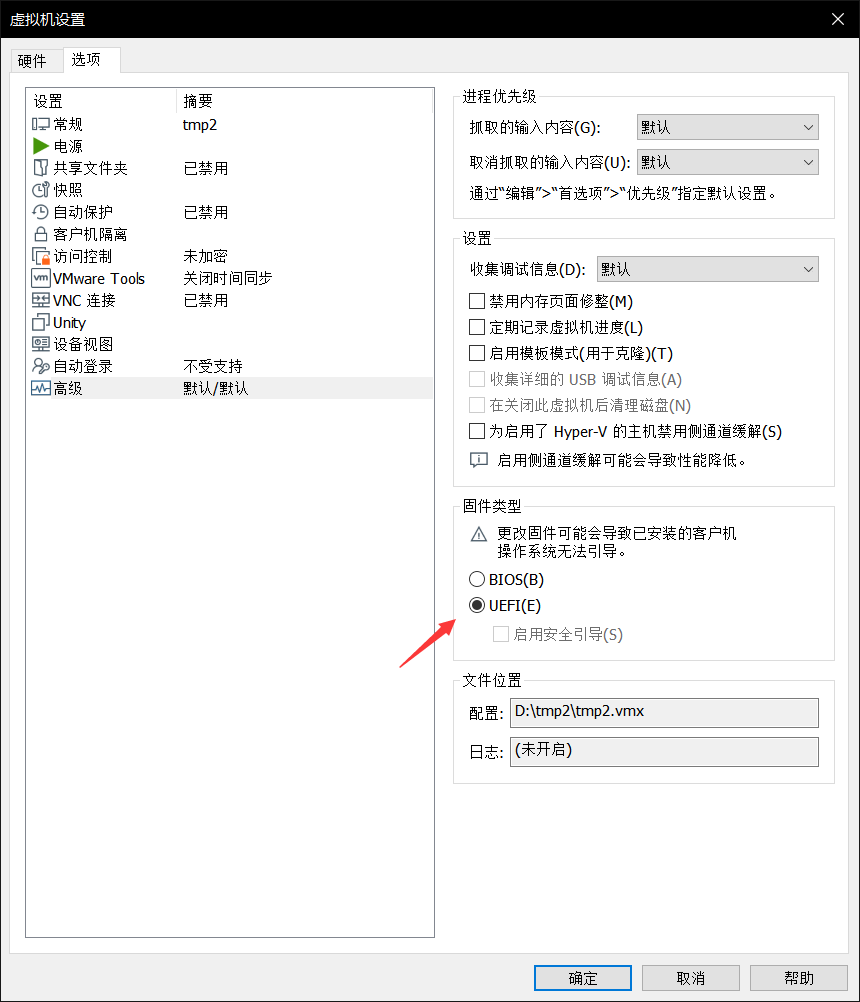




4、接着利用umount命令取消挂载，这样我们就得到一个包含内核、文件系统等的img文件，接着创建一个iso文件夹，并且在该文件夹下面将创建一个boot文件夹，然后将img复制到iso/boot下面，最后利用xorriso工具生成iso文件即可，如下图所示：



5、最后，新建一个虚拟机，引导选择uefi，启动即可，如下图所示：





1.1.2 **arm镜像**

boot编译

官方 uboot GitHub：

https://github.com/uboot/uboot

NXP 提供的 uboot 下载链接:

https://source.codeaurora.org/external/imx/uboot-imx/

野火提供的 uboot 下载链接：

https://gitee.com/Embedfire/ebf\_linux\_uboot

https://github.com/Embedfire/ebf\_linux\_uboot

安装，缺什么自己补

|  |
| --- |
| Bash sudo apt install make git gcc-arm-none-eabi gcc bison flex libssl-dev dpkg-dev lzop libncurses5-dev   sudo make distclean sudo make ARCH=arm CROSS\_COMPILE=arm-none-eabi- mx6ull\_fire\_mmc\_defconfig  #编译 uboot sudo make ARCH=arm CROSS\_COMPILE=arm-none-eabi    编译kernal |

1.1.3 **树莓派镜像**

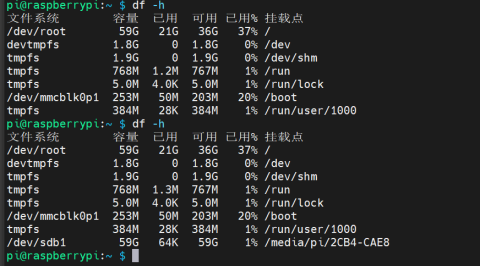
1.1.3.1 **相关链接**

https://blog.csdn.net/meihualing/article/details/110268054#t11

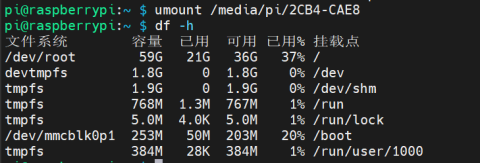
1.1.3.2 **树莓派镜像备份**

**1、方法一**

镜像拷贝，插入卡二次对比



将卡卸载



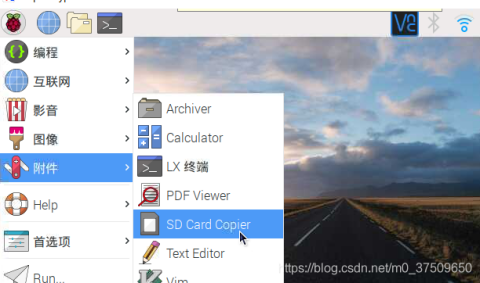
|  |
| --- |
| Bash dd bs=4M if=/dev/mmcblk0 of=/dev/sda |

mmcblk0就是树莓派上原来的存储卡 sda就是插入的USB存储设备（可能会随着接入的设备多少而改变，如可能为sdb，sdc等，自己根据情况去判断是哪一个）

这种方案失败，只拷贝了boot(FAT) 却无法拷贝rootfs (ext4)，再继续查找方法

**2、方法成功**，树莓派自带工具

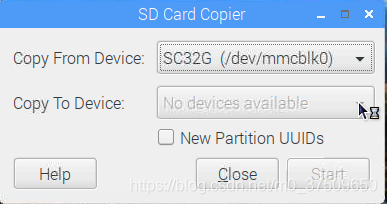
树莓派系统自带sd card cpoier，位置如图：



从上节图上挂在可以看见

源设备：/dev/mmcblk0p1

目的设备：/dev/sdb（/dev/sdb1 ，其中有a b c d 可以选择，从挂载可以看见是/dev/sdb1，所以选择/dev/sdb）

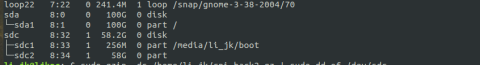


**3、方法三**，速度巨慢

插入到ubuntu上

|  |
| --- |
| Bash sudo dd if=/dev/sdc | gzip>/home/hanson/Documents/rpi-back2.gz |

备份到ubuntu上，其中备份文件要保存的位置、文件名和 SD 卡的路径要根据实际选择



备份的时候终端没有进度条，像卡死一样，等着就行。可以另开一个终端，运行如下命令，可以观察到目标文件大小的变化。

|  |
| --- |
| Bash watch -d -n 5 ls -lh /home/hanson/Documents/rpi-back2.gz |

还原的时候在 Windows 下解压文件需要加上.img 后缀。

Linux 下的还原方法

|  |
| --- |
| Bash  sudo gzip -dc /home/hanson/Documents/rpi-back2.gz | sudo dd of=/dev/sda |

2. **Linux工具**

2.1 **ubuntu界面安装**

|  |
| --- |
| C++ $ apt-get update $ apt-get install x-window-system-core $ apt-get install gnome-core |

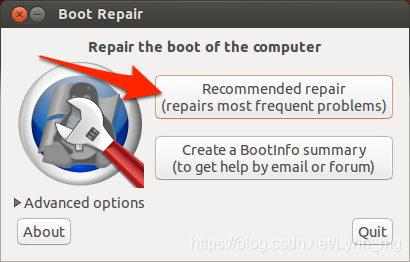
2.2 **ubuntu修复引导**

boot-repair修复

ubuntu启动盘进入

|  |
| --- |
| C++ sudo add-apt-repository ppa:yannubuntu/boot-repair sudo apt-get update  sudo apt-get install -y boot-repair  boot-repair |

点击下图



运行boot-repair点recommand repair之后会自动执行

2.3 **修改启动项**

修改linux引导默认启动选项

\*\*第一步：\*\*开机或者重启，在启动项选择菜单处记住win10对应序号（注意：系统顺序从0开始），比如我的系统处于第三项，那么我的序号就是2，记住后打开ubuntu系统

\*\*第二步：\*\*打开终端，在终端打开/etc/default/grub文件

可用ctrl+alt+t快速打开终端

可用命令sudo nano /etc/default/grub打开该文件

\*\*第三步：\*\*在上述命令回车后，将文本中的GRUB\_DEFAULT=0中的0改成系统序号2，同时可以修改默认的等待时间，GRUB\_TIMEOUT=10（默认等待时间为10秒）。修改完成后保存并关闭。

\*\*第四步：\*\*上述步骤完成后，只是将内容修改并保存了，并没有更新配置，还需在终端输入sudo update-grub更新配置

2.4 **win lin相互引导**

linux可以引导Lin 、Win

windows只可以引导 Win

2.5 **linux下安装win**

https://zhuanlan.zhihu.com/p/625253859

[gnome-boxes](https://link.zhihu.com/?target=https%3A//unbroken.blog.csdn.net/article/details/127274218%3Fspm%3D1001.2101.3001.6650.1%26utm_medium%3Ddistribute.pc_relevant.none-task-blog-2%257Edefault%257ECTRLIST%257ERate-1-127274218-blog-92078057.235%255Ev32%255Epc_relevant_default_base3%26depth_1-utm_source%3Ddistribute.pc_relevant.none-task-blog-2%257Edefault%257ECTRLIST%257ERate-1-127274218-blog-92078057.235%255Ev32%255Epc_relevant_default_base3%26utm_relevant_index%3D2)

3. **树莓派**

3.1 **镜像安装**

3.1.1 **安装工具**

https://blog.csdn.net/toopoo/article/details/122374575

|  |
| --- |
| Plaintext sudo snap install rpi-imager |