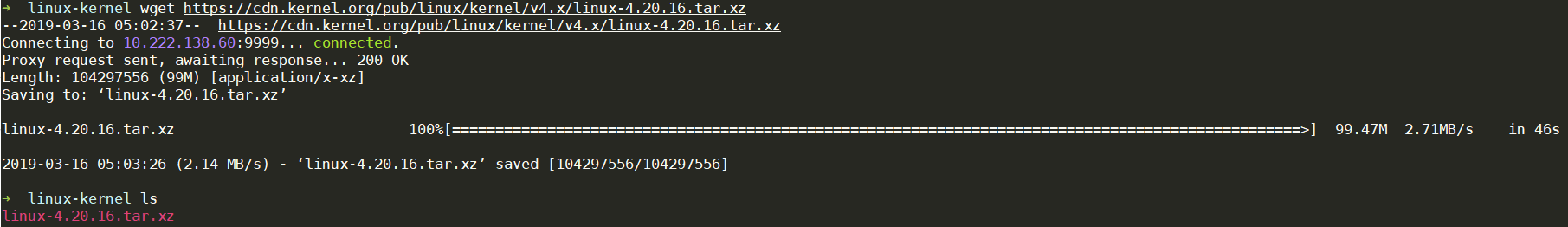
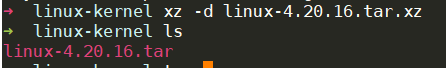


下载的是4.20.16这个稳定版的tarball



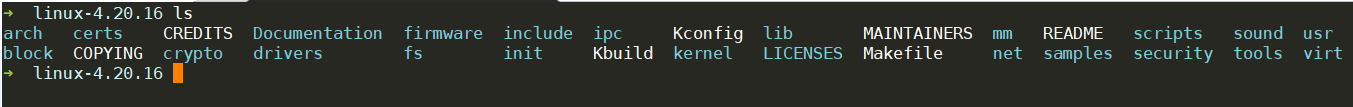
然后进行了解压



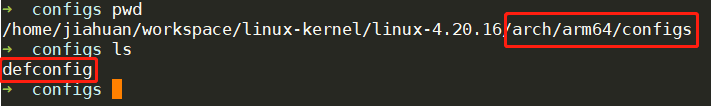
解压生成linux-4.20.16.tar文件，接着解压，



就解压完成了



依据上文中下载和解压的linux内核文件，现在我们来构建linux内核，



上文中解压完了内核之后，现在我们需要编译内核，

我们编译的目标内核是在arm64上面运行的，所以这里我们

进入arch/arm64/configs目录下面可以看到一个配置文件，这配置文件

defconfig，是default config的缩写。

我们现在要在linux-4.20.16目录下面开始构建内核，首先我们需要生成一个配置文件，

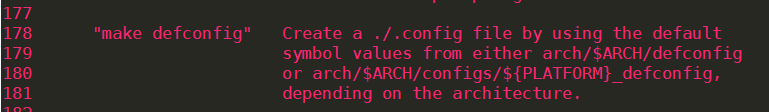
这个配置文件告诉make命令我们要构建的是什么kernel，我们

在linux-4.20.16目录下面执行

-> ARCH=arm64 CROSS\_COMPILE=aarch64-linux-gnu- make defconfig

这里有个疑问，make defconfig是干什么的，

在Documentation/admin-guide/README.rst中可以看到，



这个命令是用于在linux-4.20.16目录下生成一个.config文件用的。

这个文件是从某个arch目录下面的defconfig中拷贝出来的。

所以，这个ARCH=arm64应该就是用于替代上面的$ARCH这个变量的，

也就是说，是从arch/arm64/defconfig目录下面拷贝出来的。

CROSS\_COMPILE指定跨平台编译使用的工具链



哇，一坨翔，没有aarch64-linux-gnu-gcc

那么，也就是说，当前的编译环境中缺乏跨平台编译工具链，我门在编译面向arm64平台的内核的时候，没有编译工具链，需要安装，找到了这个：

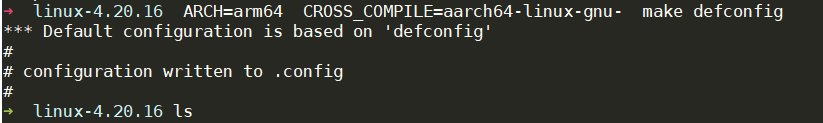
apt-get install binutils-aarch64-linux-gnu gcc-aarch64-linux-gnu

来自 <[*https://gts3.org/2017/cross-kernel.html*](https://gts3.org/2017/cross-kernel.html)>

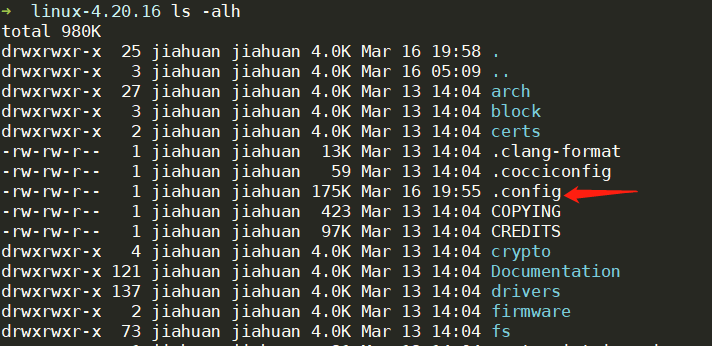
执行完了之后，aarch64-linux-gnu-gcc等跨平台编译的工具就有了。

接着，我们再次执行ARCH=arm64 CROSS\_COMPILE=aarch64-linux-gnu- make defconfig

成功了，如下:

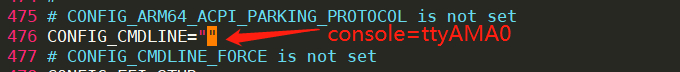


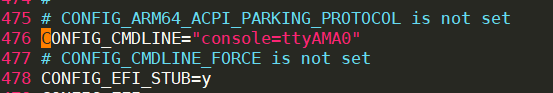
这个就生成了



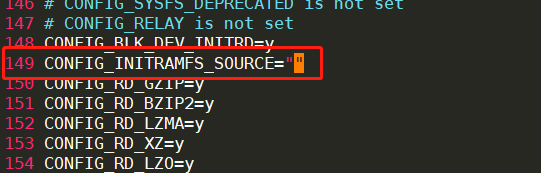
那么我们随后在linux-4.20.16目录下进行make的时候，就会用到这个配置文件，

根据[Euka Buka](https://www.youtube.com/channel/UCQraBsguF59oYKCdalrbZqA)的介绍，这里需要手动修改两个选项，vim .config，然后找到CONFIG\_CMDLINE这个选项，将其值修改为console=ttyAMA0

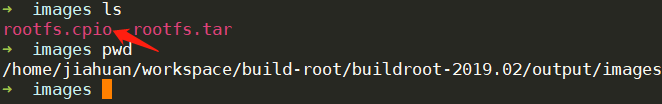




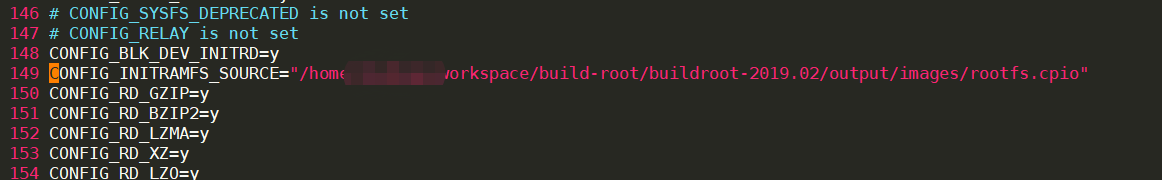
接着需要修改CONFIG\_INITRAMFS\_SOURCE，



将其值修改成我们在build\_root的时候构建的rootfs.cpio文件的路径

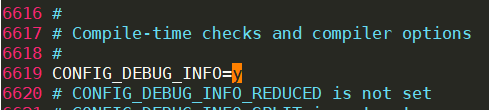


也就是修成这个样子：



上面这个就是告诉这个make，内核中包含的rootfs采用一个已经编译好的版本。

另外一个就是要注意



将debug选项赋值y。

接着，我们就可以开始编译内核了，

ARCH=arm64 CROSS\_COMPILE=aarch64-linux-gnu- make -j8



编译就完成了，生成了一些文件，那么这些文件有什么区别呢，可以参见如下博文https://blog.csdn.net/panzhenjie/article/details/48173265

这篇博客讨论了Image vmlinux等文件的区别

<https://www.cyberciti.biz/tips/compiling-linux-kernel-26.html>

<https://gts3.org/2017/cross-kernel.html>