【Linux 学习系列五：Nand Flash 根文件系统制作】

2019-05-17

版 本 历 史

版本	作者	参与者	日期	备注
V1.0	Topsemic		2019/05/17	创建

目录

1.引言

之前系列的文章介绍了如何编译 Uboot、Kernel 以及使用默认的 ramdisk 根文件系统来构建一个完整的嵌入式 Linux 系统，本篇文章介绍如何从头制作一个放在 NAND Flash 上的根文件系统。经过我这段时间的总结，rootfs 相关的编译、配置等工作还是比较麻烦的。所以你可能会看到一般做核心板的第三方厂家会建议初学者直接使用现成提供的文件系统，比如一个做 NUC972 核心板的厂家，其文档里这么描述：

4.3 使用 busybox 制作文件系统

本节内容介绍如何使用 busybox 制作文件系统中的一些工具软件，如果没有特殊的要求，使用酷客出厂的文件系统就可以，不需要进行本节操作。建议初学者跳过此节。

再比如另外一个做 NXP I.MX6 核心板的厂家，其文档里这么描述：

3.3.1 Yocto构建Linux系统

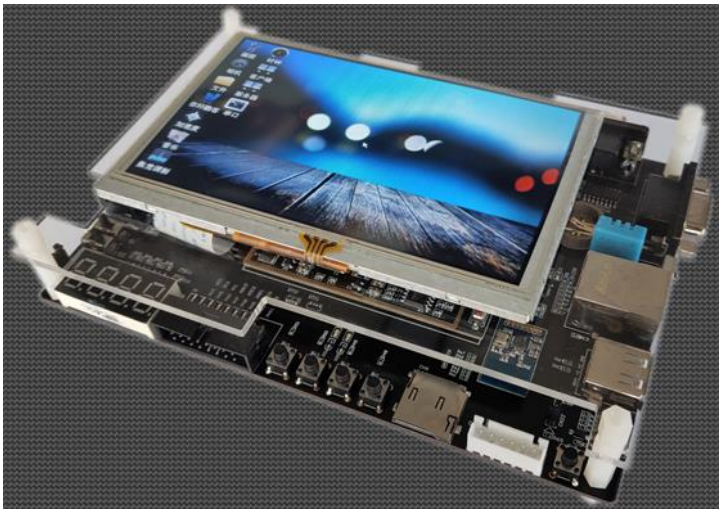
本节适合需要对文件系统进行深度定制的开发者，希望从Yocto构建出符合MYD-Y6ULX系列开发板的文件系统，同时基于它的定制需求。初次体验使用或无特殊需要的开发者可以直接使用MYD-Y6ULX已经提供的文件系统。

他们都建议初学者直接使用他们提供好的文件系统。但是我还是强烈建议大家自己一步一步的去操作制作一次，因为一是这有助于你理解根文件系统是怎么来的，最小的能用的根文件系统包含了哪些东西，二是将来你很有可能需要对文件系统做一些定制化的裁剪和修改工作，只有自己做了一遍你对会明白这个东西到底是怎么回事，比如我原来就搞不清楚 Busybox、Buildroot、Yocto 这些名词在 Linux 中的作用，现在随着实践增多，就大概知道他们的用处了。

2.环境介绍

2.1.硬件

因为本系列博客都会以 NUC972 为平台来介绍，为了大家学习起来方便，之后所有的实验都会在网上可以买到的一个 NUC972 开发板上来做，通过我的仔细甄选，感觉下面这家的开发板性价比最高。



有兴趣购买的朋友，大家去下方他们的淘宝店购买即可：

https://s.click.taobao.com/t?e=m%3D2%26s%3DVqeqPgNPc7IcQipKwQzePOeEDrYVVa64LKpWJ%2Bin0XJRAdhuF14FMYcqQp9jM3JMRitN3%2FurF3weYrs2z1V%2BWARwPYjsuD9IQ67nx0X4I%2FwbK4NckI8ZycEnxKDasWxncCadkoABCrnmL8IX2r%2Bngfx81NZbGhxUxiXvDf8DaRs%3D

建议大家选择 4.3 寸电阻屏的，因为价格最便宜，另外再配个摄像头和 WIFI 模块供后面实验使用。

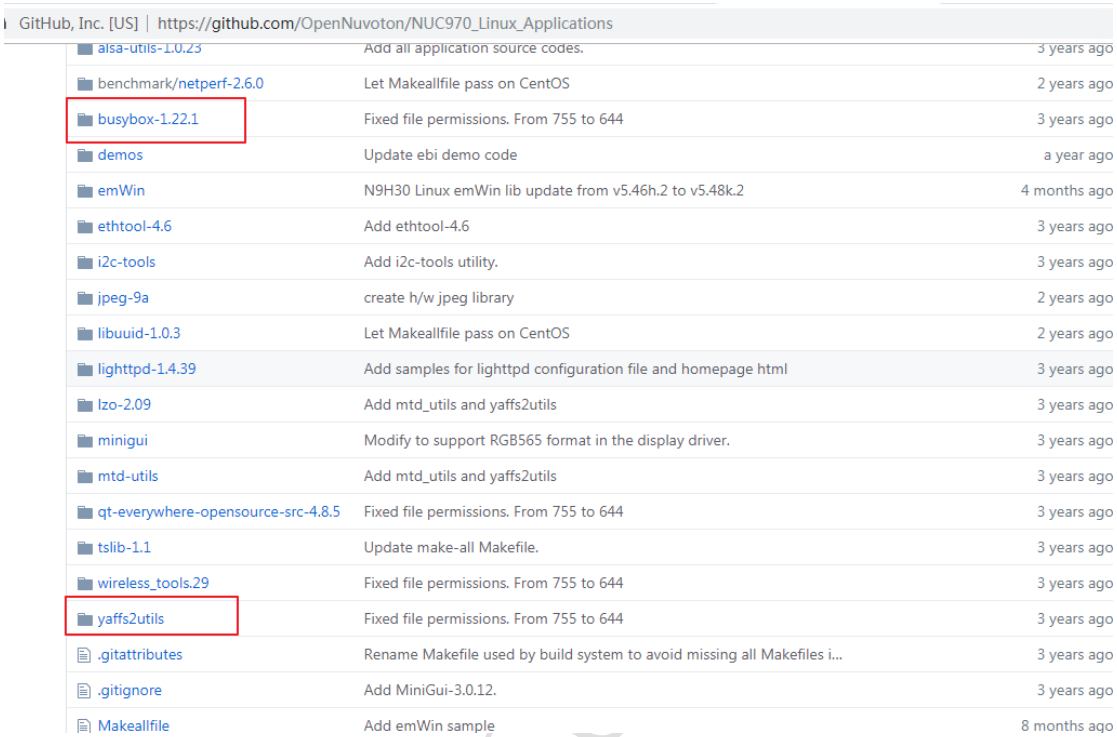
希望大家不要让自己的开发板吃灰，真正的用起来，相信大家跟着我的整个系列文章，自己亲自动手去实践，最后一定有所收获。

2.2.软件

本篇新用到的软件工具一个是 busybox，它用来生成文件系统里的几个最基

本的文件夹，二是打包工具 mkyaffs2，它用来生成能直接烧录到板子里的镜像文件。下载地址在：

https://github.com/OpenNuvoton/NUC970_Linux_Applications



大家有没有注意，我为什么每次都把这个官方的链接放过来，让大家自己去下载，原因是我们要知道我们用的东西的源头在哪里，最官方的资料渠道在哪里，这很重要。

我们这次要实现的目标是：自己亲手制作的文件系统能够跑起来，同时能够运行我们第二篇文章介绍的 Helloworld 程序。

3.使用 Busybox 制作根文件系统

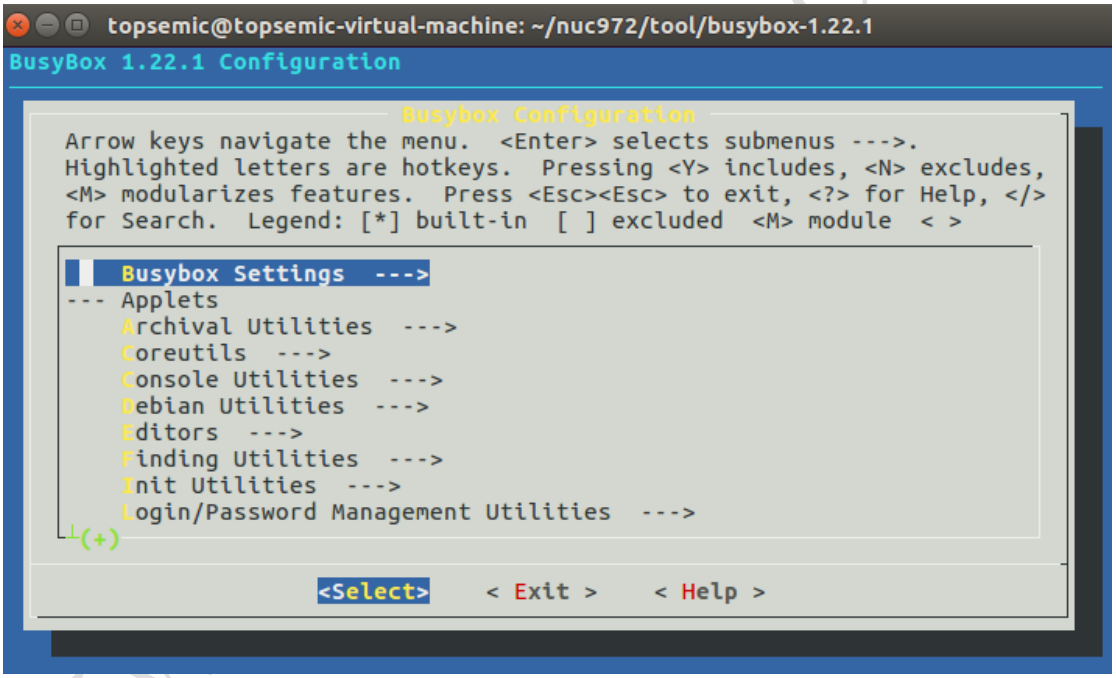
1）进入到 nuc972 文件夹，新建一个 tool 文件夹，用来存放 busybox、mkyaffs2 等工具，把上面下载的 NUC970_Linux_Applications-master.zip 文件解压到 tool 文件夹里，因为解压后含有 NUC970_Linux_Applications-master，我们把里面的内容移出来，然后删除掉空的 NUC970_Linux_Applications-master 文件夹和 NUC970_Linux_Applications-master.zip 源文件。然后进入到 busybox 目录里。同样的，在编译前要设置交叉编译的环境变量。

```
cd ~/nuc972
mkdir tool
```

```
unzip NUC970_Linux_Applications-master.zip
cd NUC970_Linux_Applications-master
mv ./* ../
cd ..
rm NUC970_Linux_Applications-master -fr
rm NUC970_Linux_Applications-master.zip
cd busybox-1.22.1/
source ~/nuc972//toolchain/environment.sh
```

2）make menuconfig 进入配置界面

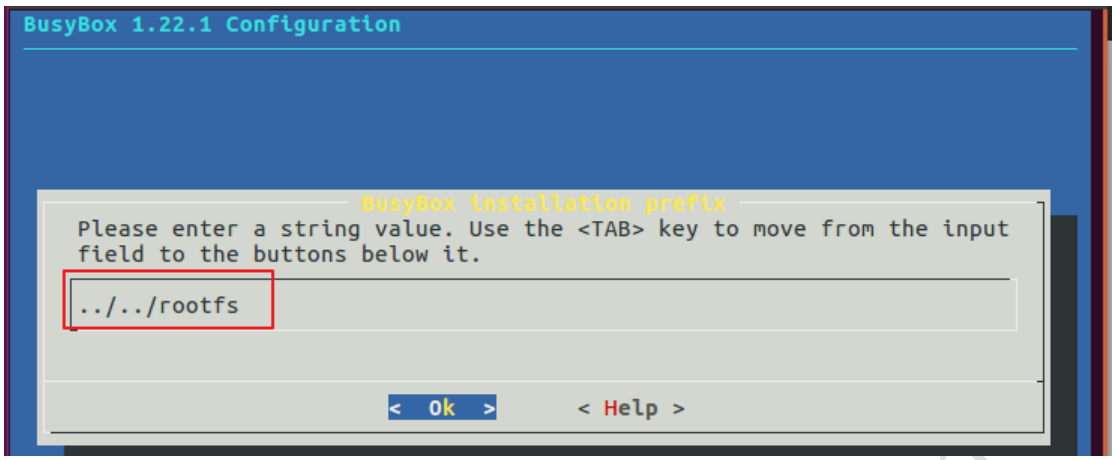
```
make menuconfig
```



我们这里就设置一个地方，其他的全部使用默认配置。

修改 make install 生成的文件夹路径，我们把默认的./_install 改成 nuc972 目录下的 rootfs。

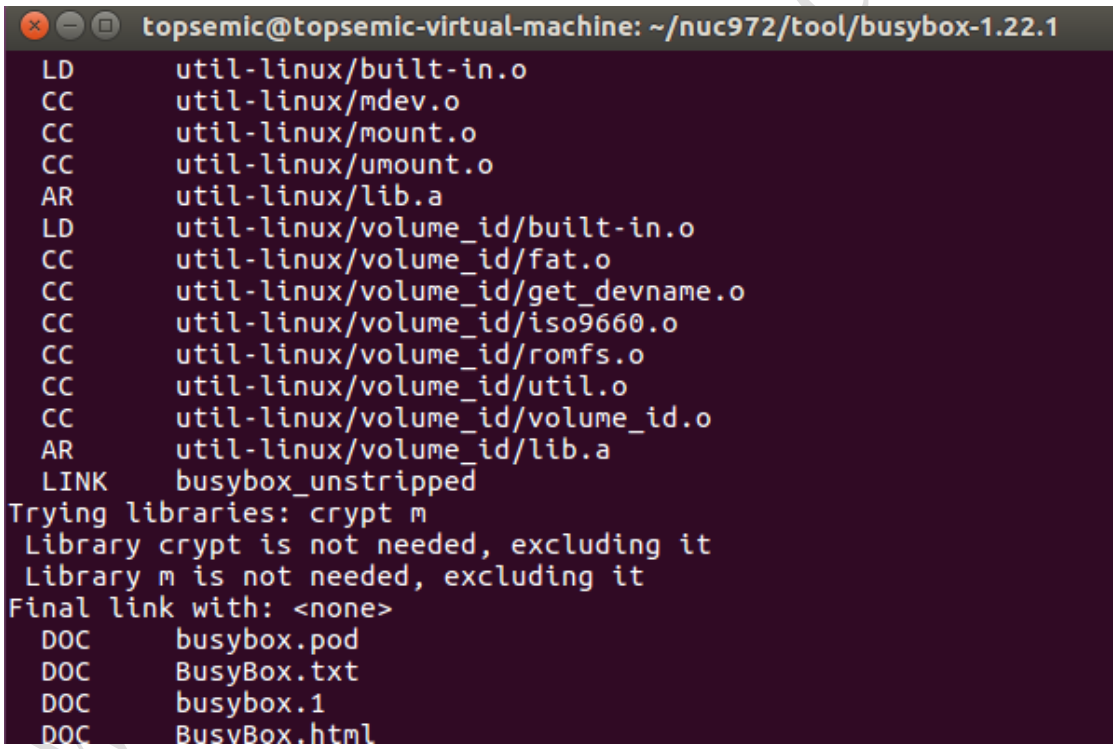
```
Busybox Settings    --->
   Installation Options ("make install" behavior)    --->
      (../../rootfs) BusyBox installation prefix
```



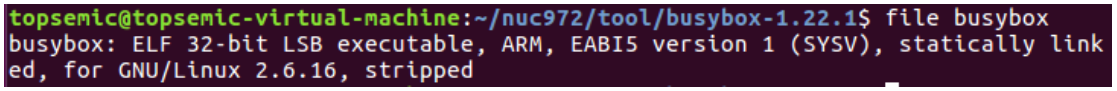
3）make 编译

```
make
```

编译成功后显示如下：



编译完成之后我们可以查看一下 busybox 的文件属性，可以看到如下信息，说明我们交叉编译的没问题。



4）make install 安装

```
make install
```

这样就在前面我们设置的地方生成一个 rootfs 的文件夹，进到里面看一下

```
cd ~/nuc972/rootfs
ls
```

```
topsemic@topsemic-virtual-machine:~/nuc972/tool/busybox-1.22.1$ cd ~/nuc972/root
fs
topsemic@topsemic-virtual-machine:~/nuc972/rootfs$ ls
bin  linuxrc  sbin  usr
```

bin、sbin、usr ： 存放一些命令

linuxrc ： 挂载根文件系统时，第一个执行的文件

4）再创建几个文件夹

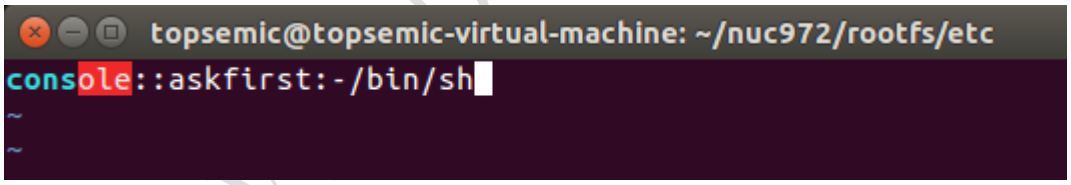
```
mkdir etc lib dev
```

5）在 rootfs 目录的 dev 目录下创建设备节点

```
cd dev
sudo mknod console c 5 1
sudo mknod null c 1 3
```

6）在根文件系统 rootfs/etc 目录下创建一个 inittab 文件，在文件里面填写：.console::askfirst:-/bin/sh

```
cd ../etc
vim inittab
```



4.文件系统打包

1）进入到 tool 文件夹的 yaffs2utils 目录里

```
cd yaffs2utils
```

2）编译，生成 mkyaffs2

```
make clean
make
```

注：编译这个工具，不需要修改 Makefile，用 file 看下属性，它是在 x86-64 运行的，这是没问题的，因为它就是在 Ubuntu 下使用的，不是在嵌入式环境下去执行使用的。

```
topsemic@topsemic-virtual-machine:~/nuc972/tool/yaffs2utils$ file mkyaffs2
mkyaffs2: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linke
d, interpreter /lib64/l, for GNU/Linux 2.6.32, BuildID[sha1]=bd3c7de7717faf6d8a6
b688916a38790d3d74c07, not stripped
```

将它复制到/usr/bin 下以后用起来方便了，可以直接使用这个指令了，你可一在终端敲 mkya，然后 TAB 建看不能出来。

```
sudo cp mkyaffs2 /usr/bin/
```

4) 生成文件系统

```
sudo mkyaffs2 --inband-tags -p 2048 rootfs rootfs_yaffs2.img
```

```
topsemic@topsemic-virtual-machine:~/nuc972$ sudo mkyaffs2 --inband-tags -p 2048
rootfs rootfs_yaffs2.img
mkyaffs2 0.2.9_20120815: image building tool for YAFFS2.

stage 1: scanning directory 'rootfs'... [done]
scanning complete, total objects: 96.

stage 2: creating image 'rootfs_yaffs2.img'
[================================================================] 96/96 100%

operation complete,
96 objects in 650 NAND pages.
topsemic@topsemic-virtual-machine:~/nuc972$
```

5.内核修改

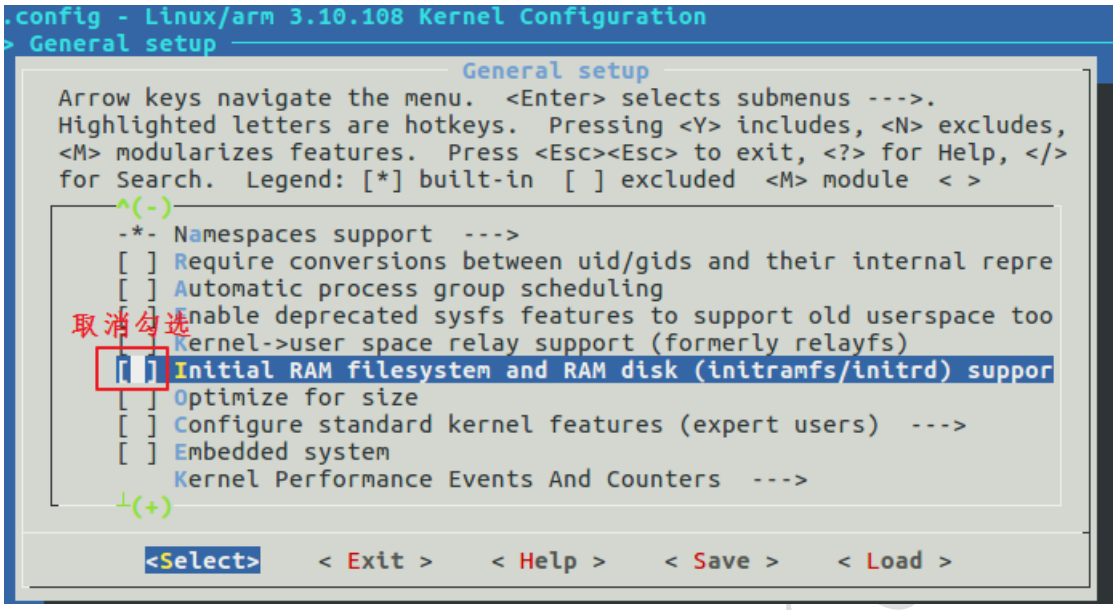
内核也要做一定的修改，具体如下：

1）make menuconfig 进入到内核配置界面

```
make menuconfig
```

2）默认用的 RAM filesytem 需要去掉

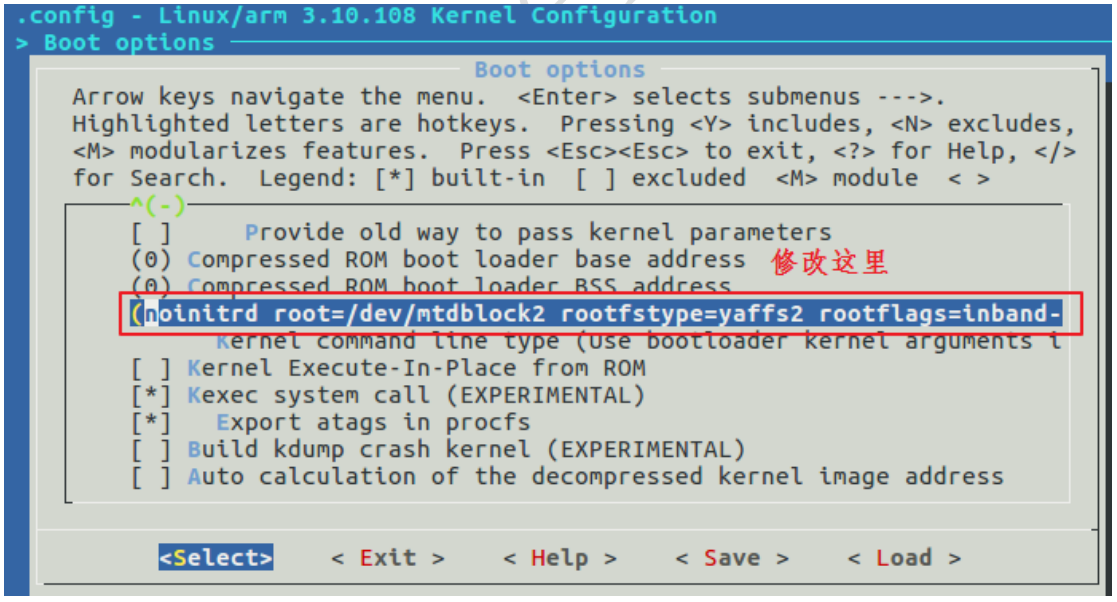
```
General setup —>

[ ] Initial RAM filesystem and RAM disk (initramfs/initrd) support
```



3）默认的 Boot option 需要修改成如下：

Boot options --->

(noinitrd root=/dev/mtdblock2 rootfstype=yaffs2 rootflags=inband-tags console=ttyS0,115200n8 rdinit=/sbin/init mem=64M)



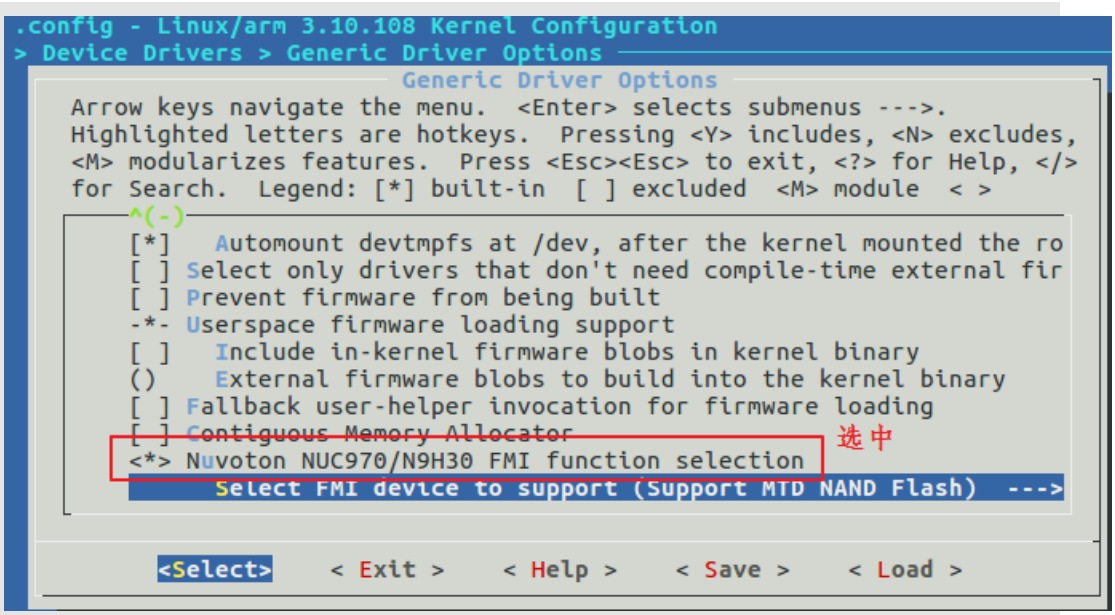
4）NAND Flash 驱动相关配置

Device Drivers --->

Generic Driver Options --->

<*> Nuvoton NUC970 FMI function selection

Select FMI device to support (Support MTD NAND Flash) --->



5）选中 MTD 的支持

Device Drivers --->

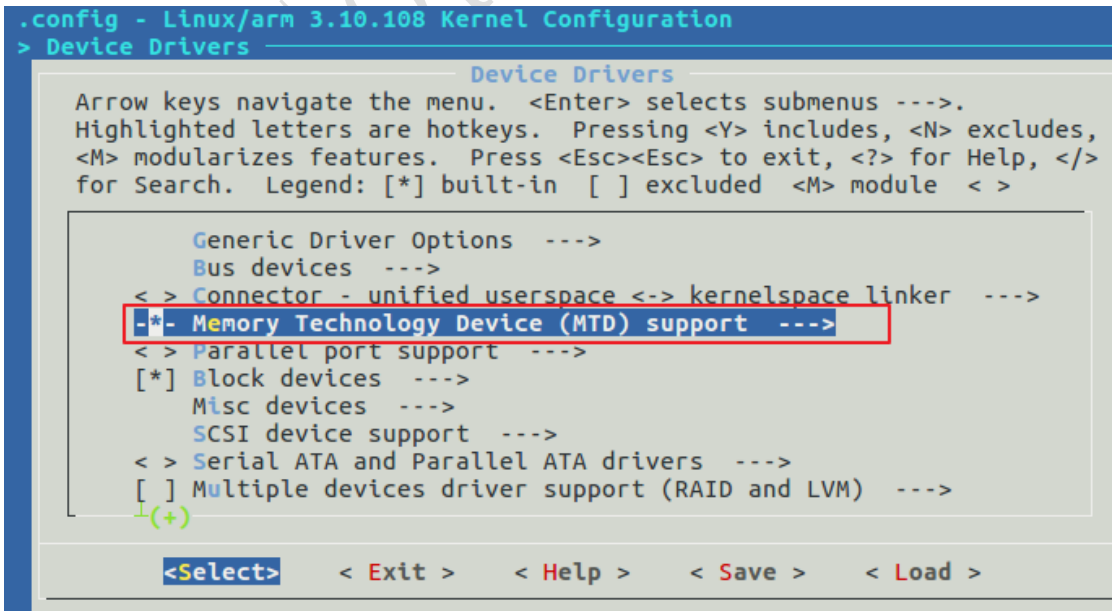
<*> Memory Technology Device (MTD) support --->

<*> Caching block device access to MTD devices

-*- NAND Device Support --->

-*- Nuvoton NUC970 MTD NAND --->

NUC970 NAND Flash pin selection (Port C) --->

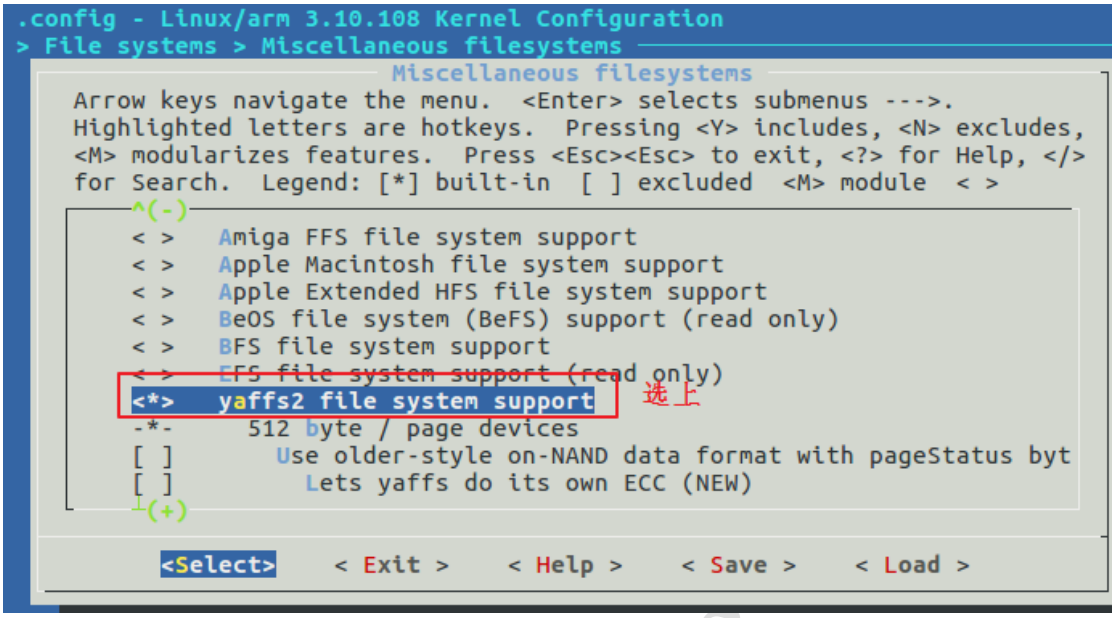


6）选中 yaff2 文件系统的支持

File systems --->

[*] Miscellaneous filesystems --->

```
<*>     yaffs2 file system support
```



7）保存配置，编译

6.结果查看

6.1.验证文件系统是否可用

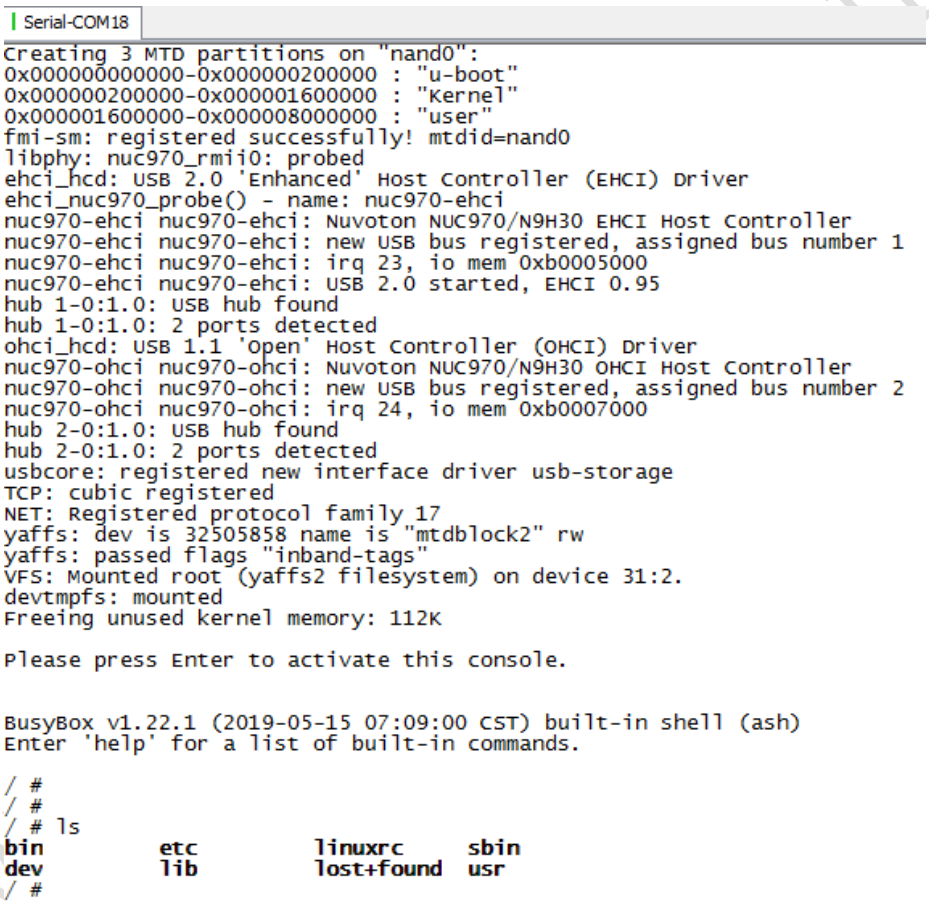
1）将上述生成的 kernel 970uimage、文件系统 rootfs_yaffs2.img、还有之前第三讲生成的 uboot.bin，uboot-spl.bin，以及 env.txt 下载到板子里，env.txt 做了一些改动，主要是加入了 boocmd 那一行，这样下载进去就能直接启动了。

```
baudrate=115200
bootcmd=nboot 0x7fc0 0 0x200000; bootm 0x7fc0
bootdelay=1
ethact=emac
ethaddr=00:00:00:11:66:88
stderr=serial
stdin=serial
stdout=serial
```

烧写前，最好对芯片进行一次 Erase。几个文件的烧写类型及地址大家注意下，别搞错了。

Image Name	Image Type	Image start offset
u-boot-spl.bin	uboot	0x200
u-boot.bin	data	0x100000
env.txt	env	0x80000
970uImage	data	0x200000
rootfs_yaffs2.img	data	0x2000000

2）上电，看看效果，调试串口打印信息如下：



这一次很顺利啊，一次就成功了，这就说明我们上述做的根文件系统是可以正常使用的了。

6.2.验证 helloworld 能否在板子上运行

我们还要看看我们之前编译的 helloworld 程序能不能运行，我们把 helloworld 放到板子里去，怎么把这个文件放到板子上呢？有以下几种方式：

a) 把 Helloworld 在 Ubuntu 下放到 rootfs 文件夹里，然后按照上述的方法打

包、下载进去。

b）通过 scp 或者 NFS 服务，直接通过网口放到板子里

c) 先放到 U 盘里，然后 U 盘插入到板子上，再复制过去

方法 a）自然没问题，不过有些麻烦，方法 b)现阶段我们的文件系统并不支持，暂时先不弄了，下一篇文章会介绍 scp 的移植，我们来试试方法 c)吧

1）插入 U 盘到电脑上，把 helloworld 拷贝过去

2）把 U 盘插入到板子上，这时调试串口会自动输出如下信息，提示我们有 USB Mass Storage 设备接入了，看来内核已经把 USB 这部门驱动已经做进去了。这不是有点像我们给 PC 机重装 Windows 系统的过程，默认鼠标键盘也是可以直接使用的。

```
BusyBox v1.22.1 (2019-05-15 07:09:00 CST) built-in shell (ash)
Enter 'help' for a list of built-in commands.

/ #
/ #
/ # ls
bin         etc         linuxrc     sbin
dev         lib         lost+found  usr
/ #
/ #
/ #
/ #
/ # usb 1-2: new high-speed USB device number 2 using nuc970-ehci
usb-storage 1-2:1.0: USB Mass Storage device detected
scsi0 : usb-storage 1-2:1.0
scsi 0:0:0:0: Direct-Access     FSL UF32 Thumb Drive      R3.a PQ: 0 ANSI: 0
sd 0:0:0:0: [sda] 127946 512-byte logical blocks: (65.5 MB/62.4 MiB)
sd 0:0:0:0: [sda] Write Protect is off
sd 0:0:0:0: [sda] No Caching mode page found
sd 0:0:0:0: [sda] Assuming drive cache: write through
sd 0:0:0:0: [sda] No Caching mode page found
sd 0:0:0:0: [sda] Assuming drive cache: write through
 sda: sda1
sda: p1 size 127945 extends beyond EOD, enabling native capacity
sd 0:0:0:0: [sda] No Caching mode page found
sd 0:0:0:0: [sda] Assuming drive cache: write through
 sda: sda1
sda: p1 size 127945 extends beyond EOD, truncated
sd 0:0:0:0: [sda] No Caching mode page found
sd 0:0:0:0: [sda] Assuming drive cache: write through
sd 0:0:0:0: [sda] Attached SCSI removable disk
```

我们进入到/dev 目录里，可以看到有 sda1 设备，LINUX 所有的存储设备都是映射成“文件”来访问的，包括 U 盘、光驱、硬盘等。这个 sda1 就对应我们刚插入的 U 盘。

```
/ # cd /dev
/dev # ls
bsg                 ram15               tty22               tty48
bus                 ram2                tty23               tty49
console             ram3                tty24               tty5
cpu_dma_latency     ram4                tty25               tty50
full                ram5                tty26               tty51
kmem                ram6                tty27               tty52
kmsg                ram7                tty28               tty53
mem                 ram8                tty29               tty54
mtd0                ram9                tty3                tty55
mtd0ro              random              tty30               tty56
mtd1                sda                 tty31               tty57
mtd1ro              sda1                tty32               tty58
mtd2                tty                 tty33               tty59
mtd2ro              tty0                tty34               tty6
mtdblock0           tty1                tty35               tty60
mtdblock1           tty10               tty36               tty61
mtdblock2           tty11               tty37               tty62
network_latency     tty12               tty38               tty63
network_throughput  tty13               tty39               tty7
null                tty14               tty4                tty8
ptmx                tty15               tty40               tty9
ram0                tty16               tty41               ttyS0
ram1                tty17               tty42               urandom
ram10               tty18               tty43               vcs
ram11               tty19               tty44               vcs1
ram12               tty2                tty45               vcsa
ram13               tty20               tty46               vcsa1
ram14               tty21               tty47               zero
/dev #
```

那么该如何访问 U 盘的东西呢？

我们需要挂载，使用 mount 指令，我们执行以下命令：

```
mkdir mnt
cd mnt
mkdir usb
cd usb
mount -t vfat /dev/sda1 /mnt/usb
cd /
mkdir opt
cp /mnt/usb/helloworld /opt
```

完成的事情是把 sda1 挂载到了/mnt/usb 目录上，这样/mt/usb 目录里就有了 U 盘里的内容，然后再复制到了系统的/opt 目录。

我们在/opt 目录里执行./helloworld，提示如下错误：

-/bin/sh: ./helloworld: not found

这是怎么回事呢？原因很简单，helloworld 的执行是需要依赖一些库的，不可能系统里什么都没有就执行能执行成功了。所以需要我们把交叉编译链

~/nuc972/toolchain/arm-2014.05/arm-none-linux-gnueabi/libc/lib 里的库文件复制到板子的 lib 目录里。

可以先到 arm-2014.05/arm-none-linux-gnueabi/libc/lib 目录中，压缩下

```
tar -cvf lib.bin *
```

然后通过 U 盘放到板子的/lib 目录中解压

```
tar -xvf lib.bin
```

这样操作之后，再次执行 helloword，结果如下：

```
/opt # ./helloworld
Hello World
```

另外可以对板子断电-再上电，看看我们之前操作的内容都还在，因为它是存到 NAND Flash 里的，所以断电是不丢失的。

7.结束语

如果你亲自动手实现了这篇文章介绍的所有内容，相信你对根文件系统一定有了一个非常感官的认识。包括这篇在内，我用了 5 篇文章一直在介绍嵌入式 Linux 基础环境相关的内容，你应该也能感受到这和我们以前接触的单片机开发还是有着较大的差异。从下一篇文章开始，我们就正式进入到具体的应用了。我会陆续介绍 GPIO、UART 使用；网络通信；WIFI 使用；摄像头采集数据；EC20 4G 模块使用；QT 编程；Opencv 使用等。大家有任何想法，欢迎给我留言反馈，发邮件也可以，邮箱：Topsemic@sina.com ，或者关注 Topsemic 微信公众号留言

本期相关的资料在百度网盘，链接:

https://pan.baidu.com/s/1pN5Jc8L2FbrD8YgX6yHWhQ 提取码：lke3；（05

Lesson5 NAND Flash 根文件系统）

里面包含了如下内容：