【Linux 学习系列四：Kernel 编译下载】

2019-04-19

目录

1.引言 3
2.**Linux Kernel** 编译 3
3.烧写 **kernel**.....11
4.运行 12
5.结束语 14

1. 引言

本篇接上一篇，给大家介绍 Linux Kernel 的编译和下载运行，达到的预期效果是都下载到板子后，可以正常通过串口登录到 Linux 系统里。

2. Linux Kernel 编译

NUC972 Linux Kernel 下载地址为：

https://github.com/OpenNuvoton/NUC970_Linux_Kernel

下载完成后，放到 Ubuntu 虚拟机的~/nuc972/kernel 文件夹下，解压缩。

```
topsemic@topsemic-virtual-machine:~/nuc972$ mkdir kernel
topsemic@topsemic-virtual-machine:~/nuc972$
```

```
topsemic@topsemic-virtual-machine:~/nuc972/kernel$ ls
NUC970_Linux_Kernel-master.zip
topsemic@topsemic-virtual-machine:~/nuc972/kernel$ unzip NUC970_Linux_Kernel-mas
ter.zip
```

编译前还记得前两篇文章说的吧，我们先 source /home/topsemic/nuc972/toolchain/environment.sh

进入到内核目录，第一步先执行 make nuc972_defconfig 生成.config 文件，这一步是不是很眼熟，和上一讲 uboot 编译是类似的。

```
topsemic@topsemic-virtual-machine:~/nuc972/kernel/NUC970_Linux_Kernel-master$ ma
ke nuc972_defconfig
  HOSTCC  scripts/basic/fixdep
  HOSTCC  scripts/kconfig/conf.o
  SHIPPED scripts/kconfig/zconf.tab.c
  SHIPPED scripts/kconfig/zconf.lex.c
  SHIPPED scripts/kconfig/zconf.hash.c
  HOSTCC  scripts/kconfig/zconf.tab.o
  HOSTLD  scripts/kconfig/conf
#
# configuration written to .config
#
```

第二步执行：make uImage,编译内核，报错了，错误是不是和编译 uboot 也类似啊。

```
topsemic@topsemic-virtual-machine:~/nuc972/kernel/NUC970_Linux_Kernel-master$ ma
ke uImage
make: arm-linux-gcc: 命令未找到
scripts/kconfig/conf --silentoldconfig Kconfig
make: arm-linux-gcc: 命令未找到
  WRAP    arch/arm/include/generated/asm/auxvec.h
  WRAP    arch/arm/include/generated/asm/bitsperlong.h
  WRAP    arch/arm/include/generated/asm/cputime.h
  WRAP    arch/arm/include/generated/asm/current.h
```

同样的我们也是要改 Makefile

这次我们用 vim 来改一次，

```
topsemic@topsemic-virtual-machine:~/nuc972/kernel/NUC970_Linux_Kernel-master$ vi
m Makefile
程序 'vim' 已包含在下列软件包中：
 * vim
 * vim-gnome
 * vim-tiny
 * vim-athena
 * vim-athena-py2
 * vim-gnome-py2
 * vim-gtk
 * vim-gtk-py2
 * vim-gtk3
 * vim-gtk3-py2
 * vim-nox
 * vim-nox-py2
请尝试：sudo apt install <选定的软件包>
```

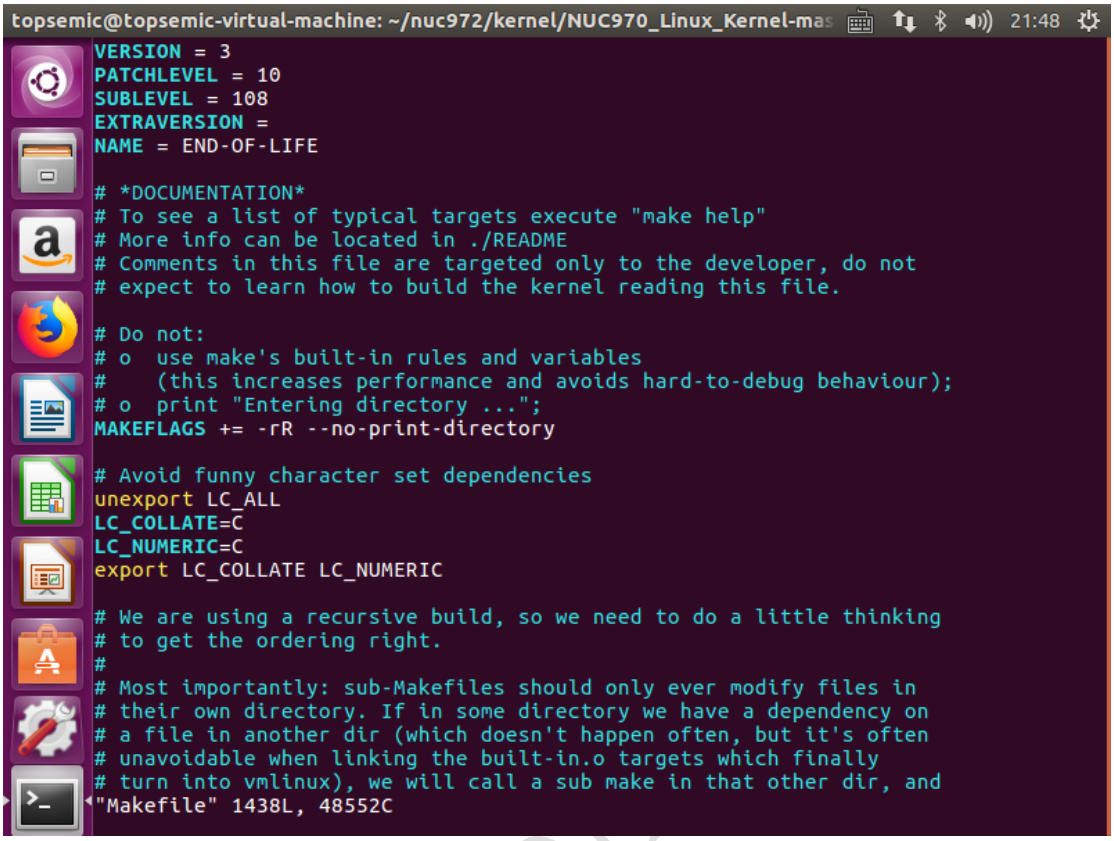
提示我们不支持 vim 这个指令，那我们安装一下吧，人家都给提示了，输入 sudo apt install vim

下面是安装过程

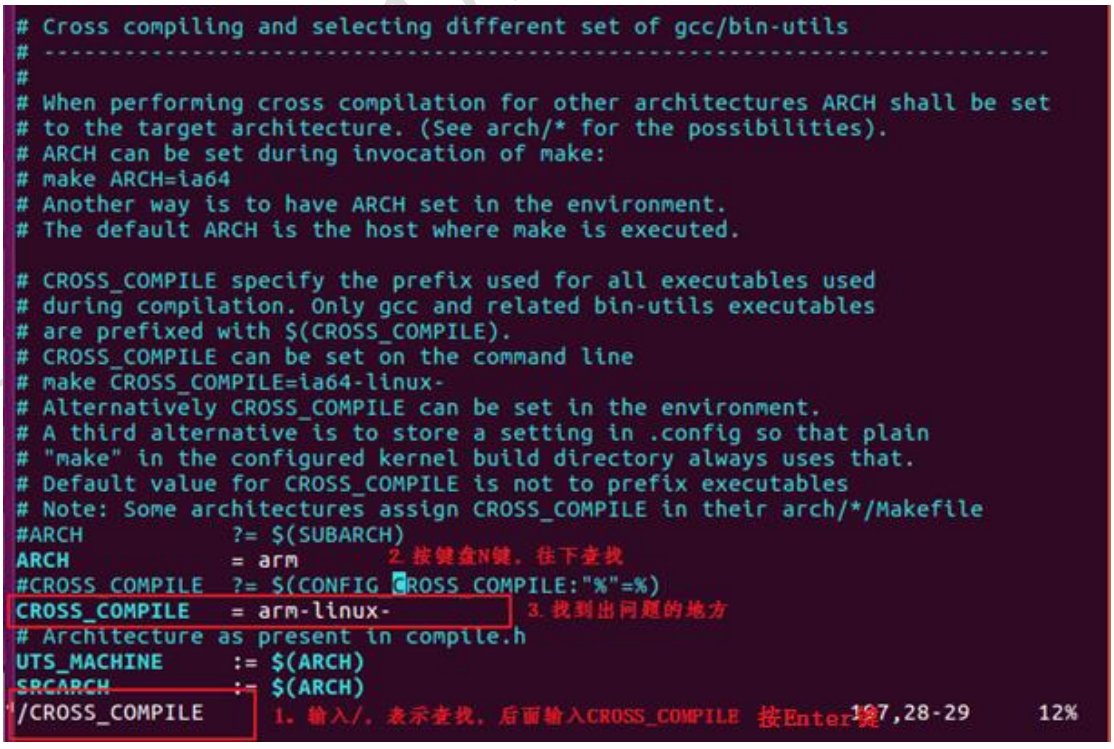
```
 all 2:7.4.1689-3ubuntu1.2 [5,164 kB]
获取:2 http://cn.archive.ubuntu.com/ubuntu xenial-updates/main amd64 vim amd64 2
:7.4.1689-3ubuntu1.2 [1,036 kB]
已下载 6,172 kB，耗时 4分 44秒 (21.7 kB/s)
正在选中未选择的软件包 vim-runtime。
(正在读取数据库 ... 系统当前共安装有 181979 个文件和目录。)
正准备解包 .../vim-runtime_2%3a7.4.1689-3ubuntu1.2_all.deb  ...
正在添加 vim-runtime 导致 /usr/share/vim/vim74/doc/help.txt 转移到 /usr/share/vi
m/vim74/doc/help.txt.vim-tiny
正在添加 vim-runtime 导致 /usr/share/vim/vim74/doc/tags 转移到 /usr/share/vim/vi
m74/doc/tags.vim-tiny
正在解包 vim-runtime (2:7.4.1689-3ubuntu1.2) ...
正在选中未选择的软件包 vim。
正准备解包 .../vim_2%3a7.4.1689-3ubuntu1.2_amd64.deb  ...
正在解包 vim (2:7.4.1689-3ubuntu1.2) ...
正在处理用于 man-db (2.7.5-1) 的触发器 ...
正在设置 vim-runtime (2:7.4.1689-3ubuntu1.2) ...
正在设置 vim (2:7.4.1689-3ubuntu1.2) ...
update-alternatives: 使用 /usr/bin/vim.basic 来在自动模式中提供 /usr/bin/vim (vi
m)
update-alternatives: 使用 /usr/bin/vim.basic 来在自动模式中提供 /usr/bin/vimdiff
 (vimdiff)
update-alternatives: 使用 /usr/bin/vim.basic 来在自动模式中提供 /usr/bin/rvim (r
vim)
update-alternatives: 使用 /usr/bin/vim.basic 来在自动模式中提供 /usr/bin/rview (
rview)
update-alternatives: 使用 /usr/bin/vim.basic 来在自动模式中提供 /usr/bin/vi (vi)
update-alternatives: 使用 /usr/bin/vim.basic 来在自动模式中提供 /usr/bin/view (v
iew)
update-alternatives: 使用 /usr/bin/vim.basic 来在自动模式中提供 /usr/bin/ex (ex)
topsemic@topsemic-virtual-machine:~/nuc972/kernel/NUC970_Linux_Kernel-master$
```

OK，安装完毕，Ubuntu 系统安装个东西真是方便啊，一条指令就搞定了。

输入 vim Makefile 显示如下：



花花绿绿的看着不错啊，先找到设置交叉编译链的地方，具体方法如下图 1、2、3 步骤描述



修改方法第一篇博客介绍的大家还记得吗？先输入 I，进入到插入模式，然

后修改即可

```
# Note: Some architectures assign CROSS_COMPILE in the
#ARCH           ?= $(SUBARCH)
ARCH            = arm
#CROSS_COMPILE  ?= $(CONFIG_CROSS_COMPILE:"%"=%)
CROSS_COMPILE   = arm-none-linux-gnueabi-
# Architecture as present in compile.h
UTS_MACHINE     := $(ARCH)
SRCARCH         := $(ARCH)
-- 插入 --
```

然后 ESC 退出插入模式，最后输入:wq 保存退出即可。

```
#ARCH           ?= $(SUBARCH)
ARCH            = arm
#CROSS_COMPILE  ?= $(CONFIG_CROSS_COMPILE:"%"=%)
CROSS_COMPILE   = arm-none-linux-gnueabi-
# Architecture as present in compile.h
UTS_MACHINE     := $(ARCH)
SRCARCH         := $(ARCH)
:wq
```

以后大家多使用 vim 吧。

修改完 Makefile 后，再一次 make uImage，

```
    ^
scripts/sortextable.h:104:6: note: ‘relocs_size’ was declared here
  int relocs_size;
      ^
In file included from scripts/sortextable.c:160:0:
scripts/sortextable.h:158:3: warning: ‘relocs_size’ may be used uninitialized in
 this function [-Wmaybe-uninitialized]
   memset(relocs, 0, relocs_size);
   ^
scripts/sortextable.h:104:6: note: ‘relocs_size’ was declared here
  int relocs_size;
      ^
  CC      init/main.o
  CHK     include/generated/compile.h
  UPD     include/generated/compile.h
  CC      init/version.o
  CC      init/do_mounts.o
  CC      init/do_mounts_rd.o
  CC      init/do_mounts_initrd.o
  LD      init/mounts.o
  CC      init/initramfs.o
  CC      init/calibrate.o
  CC      init/init_task.o
  LD      init/built-in.o
  HOSTCC  usr/gen_init_cpio
  /home/topsemic/nuc972/kernel/NUC970_Linux_Kernel-master/scripts/gen_initramfs_
list.sh: Cannot open '../rootfs'
/home/topsemic/nuc972/kernel/NUC970_Linux_Kernel-master/usr/Makefile:67: recipe
for target 'usr/initramfs_data.cpio.gz' failed
make[1]: *** [usr/initramfs_data.cpio.gz] Error 1
Makefile:800: recipe for target 'usr' failed
make: *** [usr] Error 2
```

靠，什么情况，又报错了，我记得之前没遇到这个错误啊。

网上百度查了下，说是和文件系统有关,是权限的问题，使用 sudo 编译就行，于是我激动的 sudo make uImage，咦

```
topsemic@topsemic-virtual-machine:~/nuc972/kernel/NUC970_Linux_Kernel-master$ su
do make
make: arm-none-linux-gnueabi-gcc：命令未找到
  CHK     include/generated/uapi/linux/version.h
  CHK     include/generated/utsrelease.h
make[1]: 'include/generated/mach-types.h' is up to date.
  CC      kernel/bounds.s
/bin/sh: 1: arm-none-linux-gnueabi-gcc: not found
/home/topsemic/nuc972/kernel/NUC970_Linux_Kernel-master/./Kbuild:35: recipe for
target 'kernel/bounds.s' failed
make[1]: *** [kernel/bounds.s] Error 127
Makefile:840: recipe for target 'prepare0' failed
make: *** [prepare0] Error 2
```

这个命令找不动，不应该啊，我已经 source 了，而且输入 arm-none- Table 键也能找到啊，

```
topsemic@topsemic-virtual-machine:~/nuc972/kernel/NUC970_Linux_Kernel-master$ ar
m-none-linux-gnueabi-
arm-none-linux-gnueabi-addr2line    arm-none-linux-gnueabi-gcc-ranlib
arm-none-linux-gnueabi-ar           arm-none-linux-gnueabi-gcov
arm-none-linux-gnueabi-as           arm-none-linux-gnueabi-gdb
arm-none-linux-gnueabi-c++          arm-none-linux-gnueabi-gprof
arm-none-linux-gnueabi-c++filt      arm-none-linux-gnueabi-ld
arm-none-linux-gnueabi-cpp          arm-none-linux-gnueabi-nm
arm-none-linux-gnueabi-cs           arm-none-linux-gnueabi-objcopy
arm-none-linux-gnueabi-cs-daemon    arm-none-linux-gnueabi-objdump
arm-none-linux-gnueabi-elfedit      arm-none-linux-gnueabi-ranlib
arm-none-linux-gnueabi-g++          arm-none-linux-gnueabi-readelf
arm-none-linux-gnueabi-gcc          arm-none-linux-gnueabi-size
arm-none-linux-gnueabi-gcc-4.8.3    arm-none-linux-gnueabi-strings
arm-none-linux-gnueabi-gcc-ar       arm-none-linux-gnueabi-strip
arm-none-linux-gnueabi-gcc-nm
topsemic@topsemic-virtual-machine:~/nuc972/kernel/NUC970_Linux_Kernel-master$ ar
```

于是又寻求度娘，有人说 Makefile 路径问题，于是我再改一次，把绝对路径加上

```
#ARCH           ?= $(SUBARCH)
ARCH            = arm
#CROSS_COMPILE  ?= $(CONFIG_CROSS_COMPILE:"%"=%)
CROSS_COMPILE   = /home/topsemic/nuc972/toolchain/arm-2014.05/bin/arm-none-linux
-gnueabi-
# Architecture as present in compile.h
```

绝对路径

再来一次，

这次倒是没有刚才 arm-none-linux-gnueabi-gcc 找不到的问题，不过还是有一开始提示的错误：

```
  CC      init/init_task.o
  LD      init/built-in.o
  HOSTCC  usr/gen_init_cpio
  /home/topsemic/nuc972/kernel/NUC970_Linux_Kernel-master/scripts/gen_initramfs_
list.sh: Cannot open '../rootfs'
/home/topsemic/nuc972/kernel/NUC970_Linux_Kernel-master/usr/Makefile:67: recipe
for target 'usr/initramfs_data.cpio.gz' failed
make[1]: *** [usr/initramfs_data.cpio.gz] Error 1
Makefile:800: recipe for target 'usr' failed
make: *** [usr] Error 2
```

这是怎么回事？ 我灵机一动，上面提示没有../rootfs

```
  HOSTCC  usr/gen_init_cpio
  /home/topsemic/nuc972/kernel/NUC970_Linux_Kernel-master/scripts/gen_initramfs_
list.sh: Cannot open '../rootfs'
/home/topsemic/nuc972/kernel/NUC970_Linux_Kernel-master/usr/Makefile:67: recipe
for target 'usr/initramfs_data.cpio.gz' failed
make[1]: *** [usr/initramfs_data.cpio.gz] Error 1
Makefile:800: recipe for target 'usr' failed
make: *** [usr] Error 2
```

于是，我 cd .. ,回到上级目录，mkdir rootfs 建了一个 rootfs 文件夹，再次进入到内核目录，输入 sudo make uImage

这一次看起来不太一样，好像要成功的样子，

```
  CHK     include/generated/compile.h
  SKIPPED include/generated/compile.h
  KSYM    .tmp_kallsyms1.o
  KSYM    .tmp_kallsyms2.o
  LD      vmlinux
  SORTEX  vmlinux
  SYSMAP  System.map
  OBJCOPY arch/arm/boot/Image
  Kernel: arch/arm/boot/Image is ready
cp arch/arm/boot/Image    ../image/970image
cp: 无法创建普通文件'../image/970image': 没有那个文件或目录
/home/topsemic/nuc972/kernel/NUC970_Linux_Kernel-master/arch/arm/boot/Makefile:4
8: recipe for target 'arch/arm/boot/Image' failed
make[1]: *** [arch/arm/boot/Image] Error 1
/home/topsemic/nuc972/kernel/NUC970_Linux_Kernel-master/arch/arm/Makefile:296: r
ecipe for target 'uImage' failed
make: *** [uImage] Error 2
```

不过最后还是有点小问题，还是目录没有的问题，我们再回到上级目录建个文件夹 image

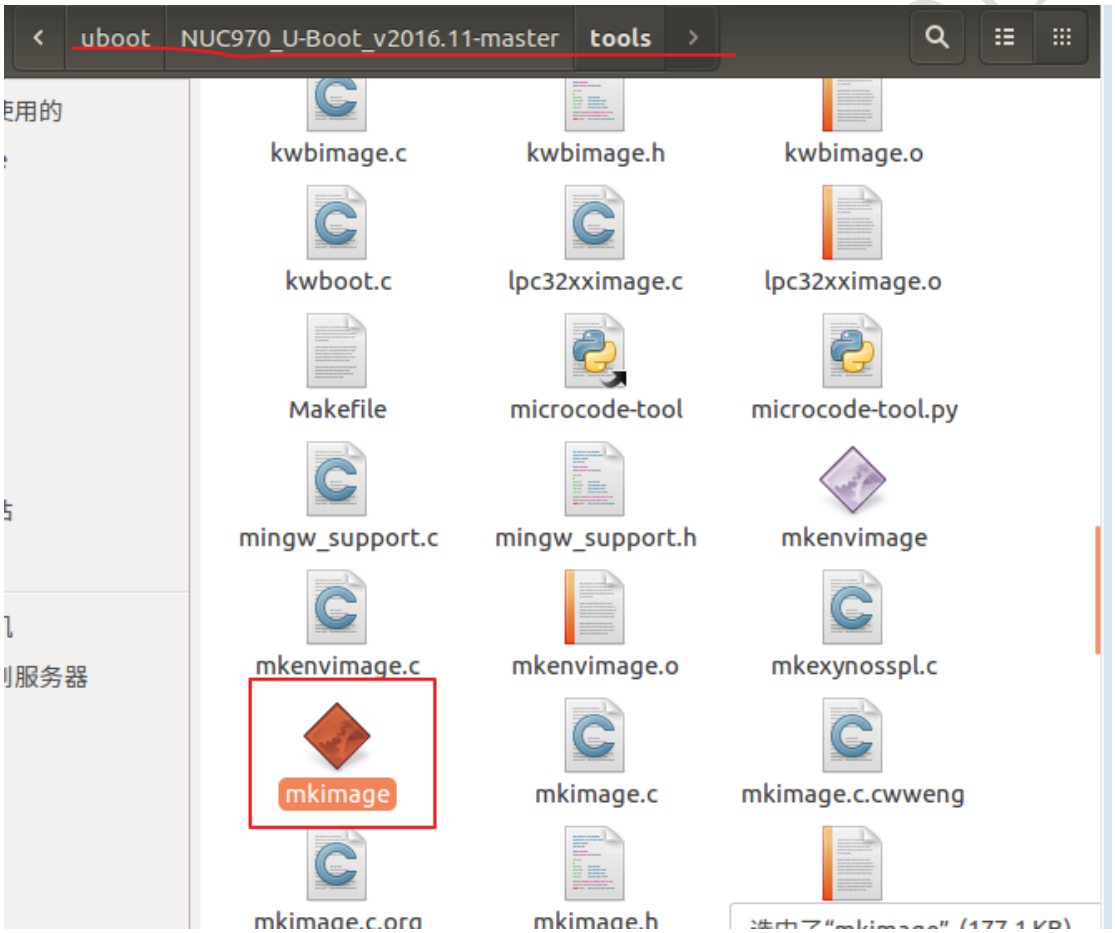
```
topsemic@topsemic-virtual-machine:~/nuc972/kernel$ mkdir image
topsemic@topsemic-virtual-machine:~/nuc972/kernel$ cd
image/                          rootfs/
NUC970_Linux_Kernel-master/
```

最后再来一次 sudo make uImage

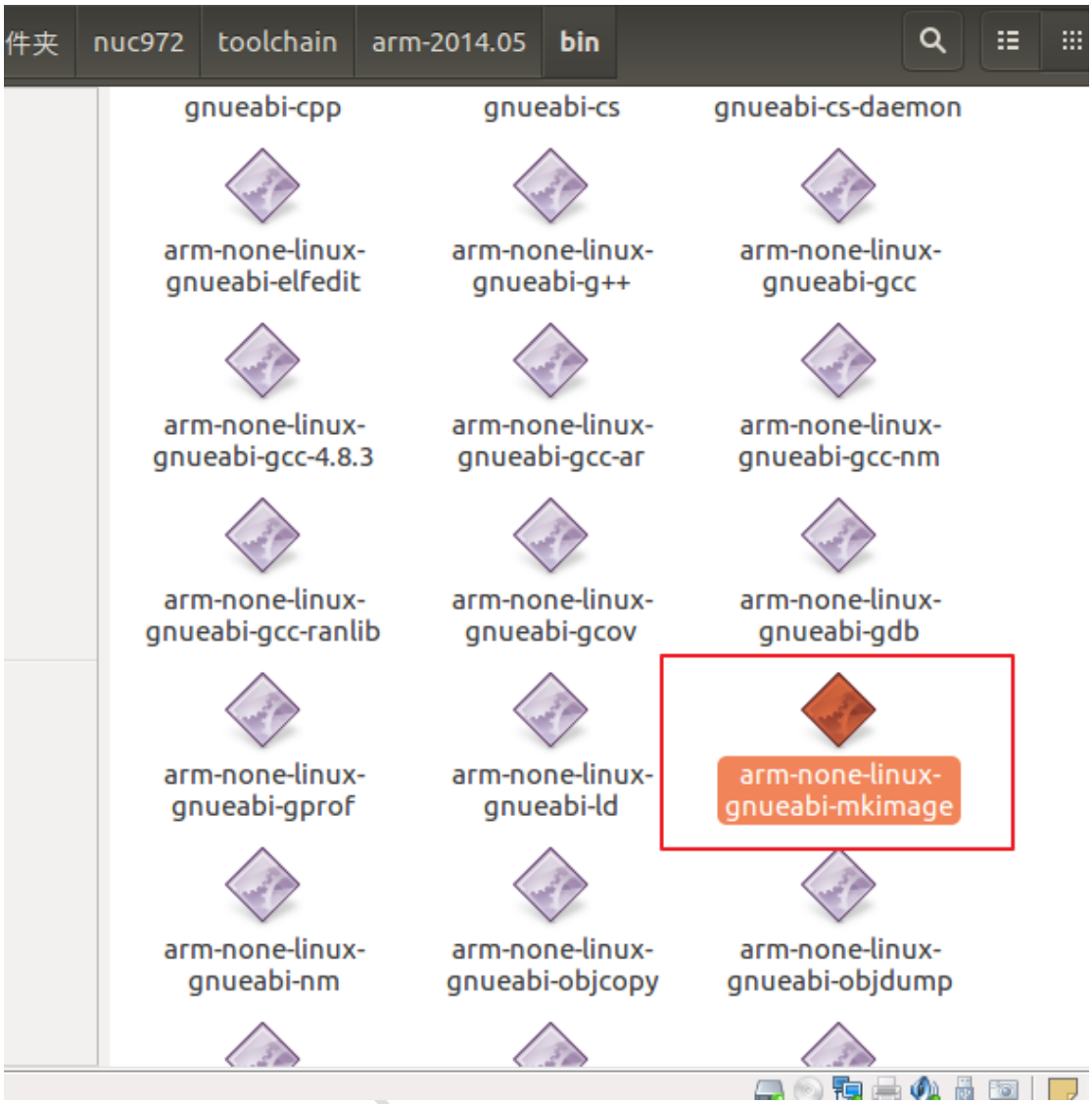
```
  LD      arch/arm/boot/compressed/vmlinux
  OBJCOPY arch/arm/boot/zImage
  Kernel: arch/arm/boot/zImage is ready
  UIMAGE  arch/arm/boot/uImage
"mkimage" command not found - U-Boot images will not be built
/home/topsemic/nuc972/kernel/NUC970_Linux_Kernel-master/arch/arm/boot/Makefil
4: recipe for target 'arch/arm/boot/uImage' failed
make[1]: *** [arch/arm/boot/uImage] Error 1
/home/topsemic/nuc972/kernel/NUC970_Linux_Kernel-master/arch/arm/Makefile:296
ecipe for target 'uImage' failed
make: *** [uImage] Error 2
```

怎么还有错误，我之前编译根本没有这些问题的啊，郁闷了。。。

再次百度，说这个命令是 uboot 编译出来的，因为没有找到它的原因，我看了下我的 uboot 里 tools 目录，果然是有这个命令的，



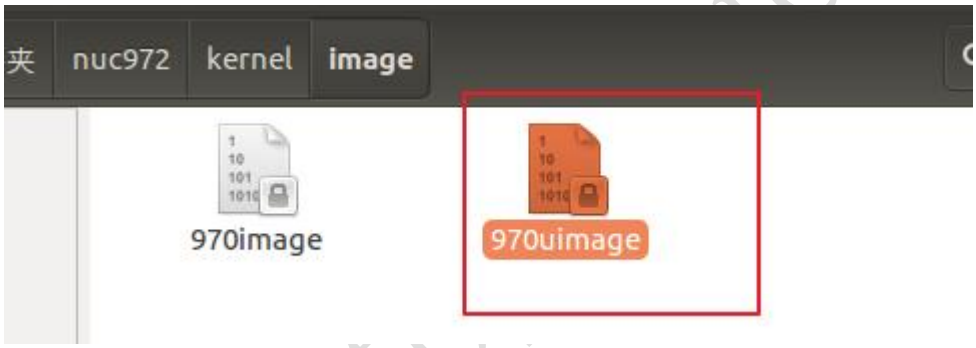
有网友说把它复制到交叉编译链的 bin 目录里就行了，于是我放进去尝试，还是有问题，最后又百度，有网友说 (https://blog.csdn.net/wangliang888888/article/details/54906729)还得改个名字，前面得加上交叉编译的名字，于是加上 arm-none-linux-gnueabi-，



最后再次编译，终于大功告成，不容易啊！

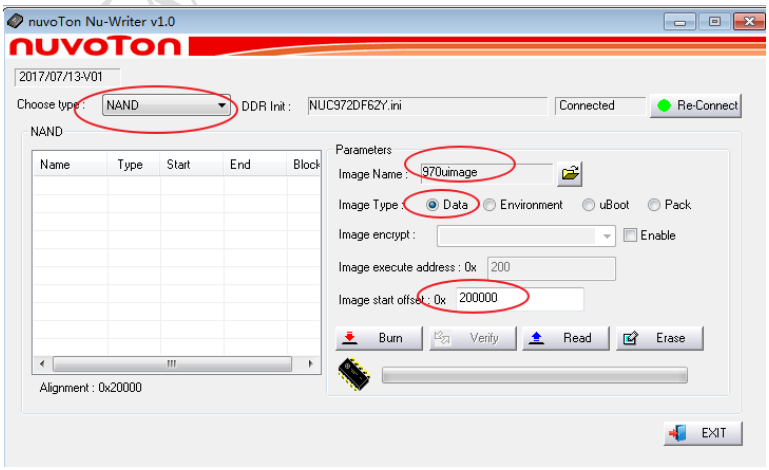
```
topsemic@topsemic-virtual-machine:~/nuc972/kernel/NUC970_Linux_Kernel-master$ su
do make uImage
  CHK     include/generated/uapi/linux/version.h
  CHK     include/generated/utsrelease.h
make[1]: 'include/generated/mach-types.h' is up to date.
  CALL    scripts/checksyscalls.sh
  CC      scripts/mod/devicetable-offsets.s
  GEN     scripts/mod/devicetable-offsets.h
  HOSTCC  scripts/mod/file2alias.o
  HOSTLD  scripts/mod/modpost
  CHK     include/generated/compile.h
  SKIPPED include/generated/compile.h
  Kernel: arch/arm/boot/Image is ready
cp arch/arm/boot/Image   ../image/970image
  Kernel: arch/arm/boot/zImage is ready
  UIMAGE  arch/arm/boot/uImage
Image Name:   Linux-3.10.108
Created:      Fri Apr 19 22:45:24 2019
Image Type:   ARM Linux Kernel Image (uncompressed)
Data Size:    1836048 Bytes = 1793.02 kB = 1.75 MB
Load Address: 00008000
Entry Point:  00008000
  Image arch/arm/boot/uImage is ready
cp arch/arm/boot/uImage   ../image/970uimage
topsemic@topsemic-virtual-machine:~/nuc972/kernel/NUC970_Linux_Kernel-master$
```

在 image 目录里的 970uimage 就是我们需要的文件了。



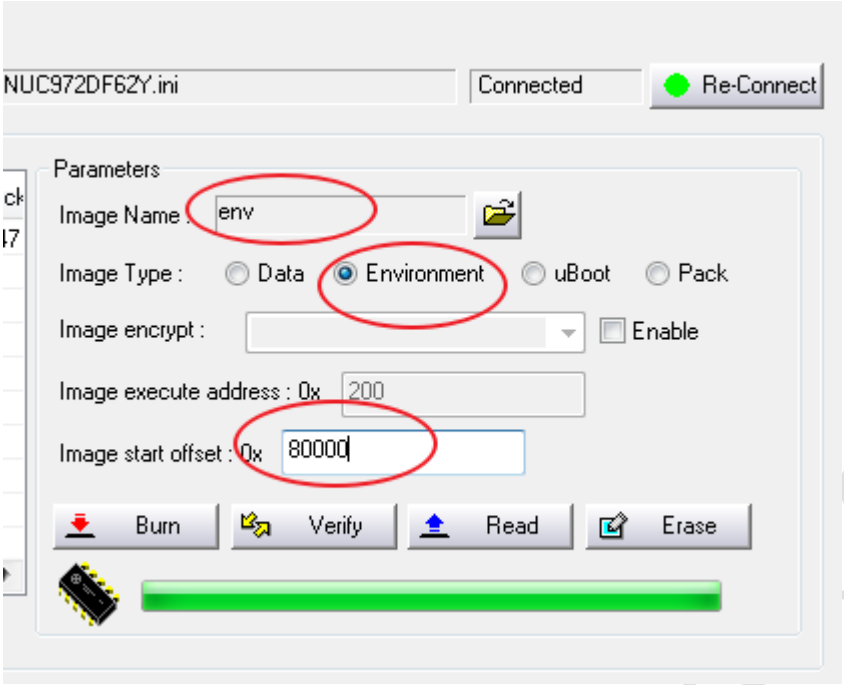
3. 烧写 kernel

1)具体怎么烧录，上一篇已经描述过了，这里要注意烧写地址：



2)另外上一节已经把 uboot 烧写进去了,这里不再重复了

3)这里还要再烧个 env.txt



板子里有三个东西就可以了。

4. 运行

1）使用 debug 串口连接上后，会进入如下界面

```
U-Boot 2016.11 (Apr 14 2019 - 20:45:16 +0800)

CPU: NUC972
Board: NUC972
DRAM:  64 MiB
NAND:  128 MiB
In:    serial
Out:   serial
Err:   serial
Net:   emac
=>
```

2）这时参考官方文档输入 set bootcmd nboot 0x7fc0 0 0x200000\; bootm 0x7fc0

J. 設定開機自動運行腳本,完成後重新啟動EV-Board

```
U-Boot> set bootcmd nboot 0x7fc0 0 0x200000\; bootm 0x7fc0

U-Boot> saveenv

Saving Environment to NAND...

Erasing Nand...

Erasing at 0xe0000 -- 100% complete.

Boot>
```

提示如下：

```
U-Boot 2016.11 (Apr 14 2019 - 20:45:16 +0800)

CPU: NUC972
Board: NUC972
DRAM:  64 MiB
NAND:  128 MiB
In:    serial
Out:   serial
Err:   serial
Net:   emac
=> set bootcmd nboot 0x7fc0 0 0x200000\; bootm 0x7fc0
Unknown command 'set' - try 'help'
=>
```

这什么情况，官方文档逗我玩呢？？

后来发现把 set 改为 setenv 就好了，写文档的人得靠点谱啊，不然真是坑人啊。

然后再输入 saveenv

```
=> setenv bootcmd nboot 0x7fc0 0 0x200000\; bootm 0x7fc0
=> saveenv
Saving Environment to NAND...
Erasing NAND...
Erasing at 0xe0000 -- 100% complete.
Writing to NAND... OK
=>
```

3）按下复位键，这时就呼呼的跑起来了

```
Serial-COM18
nuc970-ehci nuc970-ehci: Nuvoton NUC970/N9H30 EHCI Host Controller
nuc970-ehci nuc970-ehci: new USB bus registered, assigned bus number 1
nuc970-ehci nuc970-ehci: irq 23, io mem 0xb0005000
nuc970-ehci nuc970-ehci: USB 2.0 started, EHCI 0.95
hub 1-0:1.0: USB hub found
hub 1-0:1.0: 2 ports detected
ohci_hcd: USB 1.1 'Open' Host Controller (OHCI) Driver
nuc970-ohci nuc970-ohci: Nuvoton NUC970/N9H30 OHCI Host Controller
nuc970-ohci nuc970-ohci: new USB bus registered, assigned bus number 2
nuc970-ohci nuc970-ohci: irq 24, io mem 0xb0007000
hub 2-0:1.0: USB hub found
hub 2-0:1.0: 2 ports detected
usbcore: registered new interface driver usb-storage
TCP: cubic registered
NET: Registered protocol family 17
Warning: unable to open an initial console.
VFS: Cannot open root device "ram0" or unknown-block(1,0): error -2
Please append a correct "root=" boot option; here are the available partitions:
Kernel panic - not syncing: VFS: Unable to mount root fs on unknown-block(1,0)
CPU: 0 PID: 1 Comm: swapper Not tainted 3.10.108 #2
Backtrace:
[<c0012090>] (dump_backtrace+0x0/0x10c) from [<c00122a4>] (show_stack+0x18/0x1c)
 r6:c030c804 r5:00008000 r4:c030c79c r3:00000000
[<c001228c>] (show_stack+0x0/0x1c) from [<c02a022c>] (dump_stack+0x20/0x28)
[<c02a020c>] (dump_stack+0x0/0x28) from [<c029dd78>] (panic+0xa4/0x1fc)
[<c029dcd4>] (panic+0x0/0x1fc) from [<c034e0b0>] (mount_block_root+0x230/0x2d8)
 r3:00000030 r2:c3843dc8 r1:c3843ec0 r0:c030c79c
 r7:c0369c30
[<c034de80>] (mount_block_root+0x0/0x2d8) from [<c034e2ac>] (mount_root+0x58/0x70)
[<c034e254>] (mount_root+0x0/0x70) from [<c034e42c>] (prepare_namespace+0x168/0x1c8)
 r5:c0369c30 r4:c03a1b00
[<c034e2c4>] (prepare_namespace+0x0/0x1c8) from [<c034dccc>] (kernel_init_freeable+0x208/0x250)
 r6:c03a1aa0 r5:00000007 r4:c036d974
[<c034dac4>] (kernel_init_freeable+0x0/0x250) from [<c029d6d4>] (kernel_init+0x10/0x158)
[<c029d6c4>] (kernel_init+0x0/0x158) from [<c000ee18>] (ret_from_fork+0x14/0x3c)
```

靠，又出现错误了，这个足足卡了我 1 个多小时今天晚上。

最后也是灵光一现，定位到是上面编译内核时提示的 rootfs 文件夹有问题那里，难道我自己随便建个空的 rootfs 就可以吗？

另外回忆之前我用别人提供好的 bsp 包没出现这个问题，那里默认是有 rootfs 文

件目录的，于是把那里的 rootfs 搞过来，解压后，再重新编译下（先 sudo make clean，再 sudo make uImage）并重新下载一次就 ok 了。我猜测内核里用到 rootfs 文件夹里的东西。 原来都没有考虑过这个问题啊我。

最后成功的现象是：

```
Block layer SCSI generic (bsg) driver version 0.4 loaded (major 254)
io scheduler noop registered
io scheduler deadline registered
io scheduler cfq registered (default)
nuc970-uart.0: ttyS0 at I/O 0x0 (irq = 36) is a NUC970
brd: module loaded
libphy: nuc970_rmii0: probed
ehci_hcd: USB 2.0 'Enhanced' Host Controller (EHCI) Driver
ehci_nuc970_probe() - name: nuc970-ehci
nuc970-ehci nuc970-ehci: Nuvoton NUC970 EHCI Host Controller
nuc970-ehci nuc970-ehci: new USB bus registered, assigned bus number 1
nuc970-ehci nuc970-ehci: irq 23, io mem 0xb0005000
nuc970-ehci nuc970-ehci: USB 2.0 started, EHCI 0.95
hub 1-0:1.0: USB hub found
hub 1-0:1.0: 2 ports detected
ohci_hcd: USB 1.1 'Open' Host Controller (OHCI) Driver
nuc970-ohci nuc970-ohci: Nuvoton NUC970 OHCI Host Controller
nuc970-ohci nuc970-ohci: new USB bus registered, assigned bus number 2
nuc970-ohci nuc970-ohci: irq 24, io mem 0xb0007000
hub 2-0:1.0: USB hub found
hub 2-0:1.0: 2 ports detected
usbcore: registered new interface driver usb-storage
TCP: cubic registered
NET: Registered protocol family 17
Freeing unused kernel memory: 6932K (c0345000 - c0a0a000)


BusyBox v1.22.1 (2016-02-03 14:11:04 CST) built-in shell (ash)
Enter 'help' for a list of built-in commands.

-/bin/sh: /mnt/mmcblk0p1/scan_zbar: not found
~ # ls
bin      etc      linuxrc  proc     sys      usr
dev      lib      mnt      sbin     tmp      var
~ #
```

可以成功登陆进去，输入指令了！

至于这个 rootfs 文件夹里东西是怎么来的，我们下篇文章再说。

5. 结束语

本篇文章写下来，自己收获了很多，本以为之前调通的东西，很容易就操作出来了，没想到遇到这么多新的问题，不过遇到问题是好事，会让你思考一些原来没有注意到的地方。

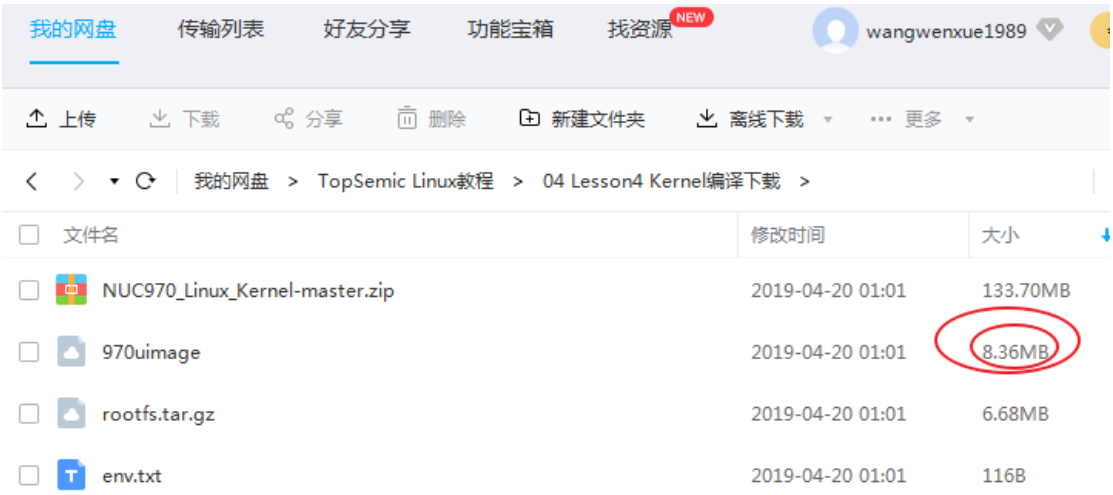
随便提下，这里内核默认用的 ramdisk 根文件系统，这个文件系统是存在 RAM 里的，所以你在里面写入东西，一断电就没有了。这和常规在 NAND 上的 Yaffs2 文件系统不一样。下一篇介绍如果换成 NAND Yaffs2 文件系统。

经过这一篇的讲解，你如果实际动手实现后，应该对嵌入式 Linux 更加感兴

趣了吧，相当于我们会重装系统了。虽然困难重重，但是不要放弃。

本期相关的资料还是在百度网盘，链接：https://pan.baidu.com/s/1pN5Jc8L2FbrD8YgX6yHWhQ 提取码：lke3；（04 Lesson4 kernel 编译下载）里面。

如果你最终编译的 970uimage 文件很小，只有不到 2MB，那是有问题的，我这边最后编译的是 8.36MB，大家注意下。



有任何问题欢迎大家留言交流讨论。

下期会给大家介绍如果改成 NAND Yaffs2 文件系统。