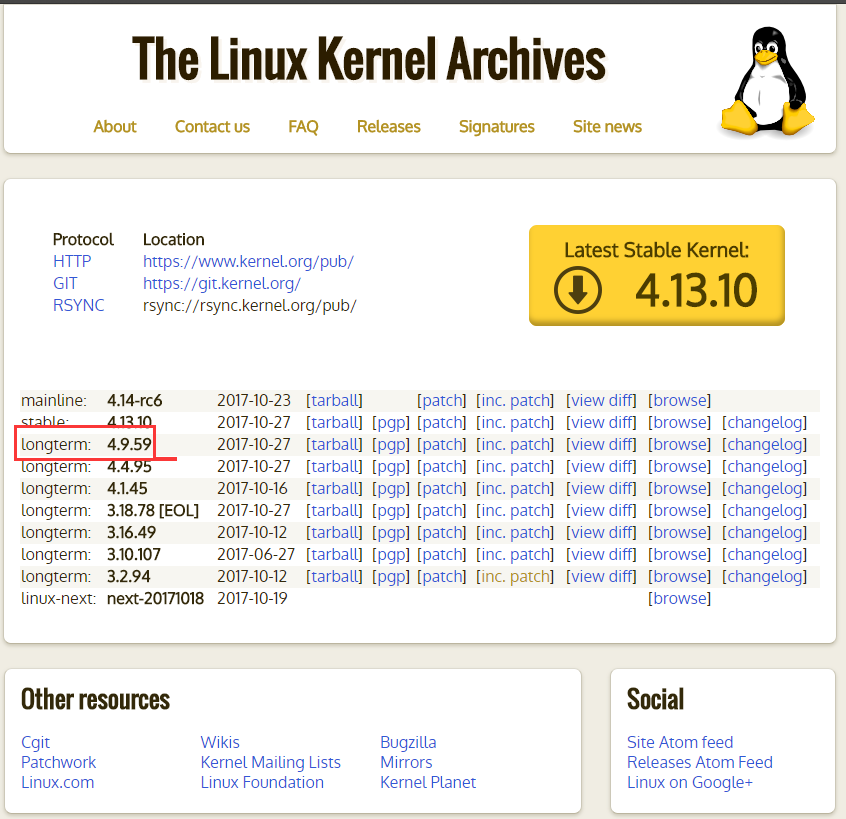
最近想要学习下设备树的知识，鉴于对2440比较熟悉，决定那它下手，有两个目标：

1. 内核如何支持设备树？。
2. 如何编写驱动？

话不多说，先去kernel.org下载了，我下载了一个long term里边最新的4.9.59版本。



解压并修改Makefile,修改ARCH以及CROSS\_COMPILE变量。

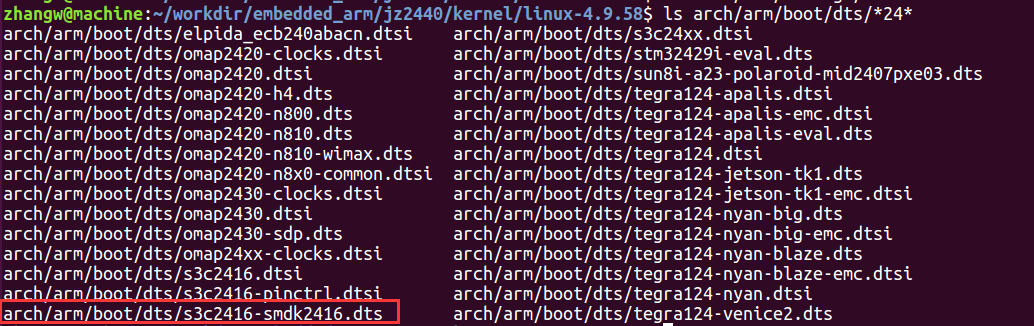
接下来开始看下新内核有没有2440相关的配置。

看来mini2440已经支持的很好了。



应该是不支持设备树的，为何这么说呢？

在arm的dts目录下只有2416的设备树源码，看来2416是支持的。



先下载到板子看能否起来。

直接配置并编译：

make mini2440\_defconfig && make -j8 uImage

接下来就是tftp烧写测试了，简述下我的环境以及文件系统挂载方式。

Uboot：U-Boot 2012.04.01 （韦东山老师视频中移植的）

文件系统：nfs

uImage需要tftp下载至内存然后启动。

首先是为了tftp，uboot中的ip配置

Gatewayip ipaddr serverip netmask

其次是配置machid，mini2440的是1999(7cf)

set machid 7cf

下载至内存：

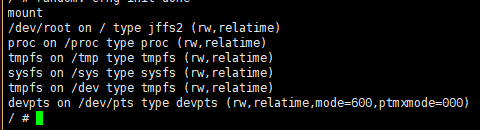
tftp 30800000 uImage

启动内核：

bootm 30800000

直接启动没有问题，

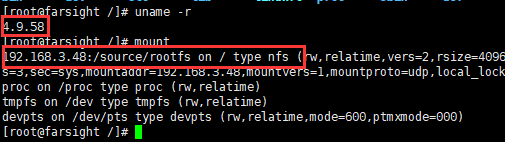
我是用2期的jffs2文件系统，直接启动成功。



uname -r

4.9.58

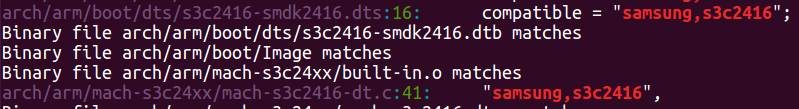
测试下nfs挂载，也是OK的。和busybox（1.20）和gcc（4.3.2）的版本有关。



当然mini2440没有用到设备树，我们需要看下内核下有没有24xx用设备树的。

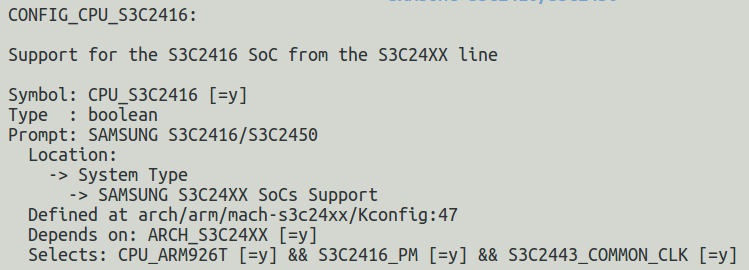


看来2416是使用了设备树，我们知道使用设备树的时候machid将不会使用，也就是machine\_desc中必须有一个compatible来和设备树中的兼容。我们搜索一下。

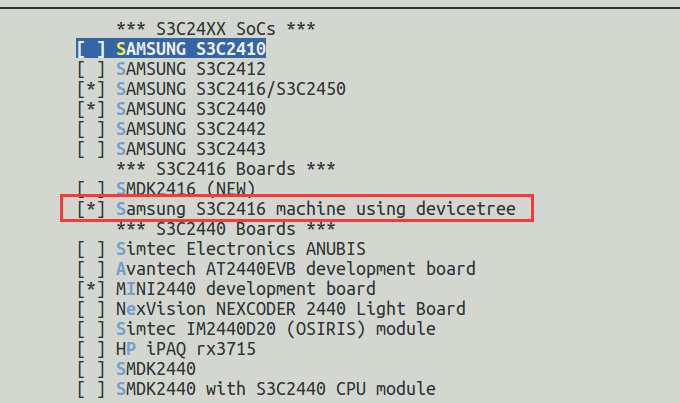


果真是有的，我们能否直接使用2416的uImage和设备树呢？暂时还不知道，我们先让内核支持2416吧。

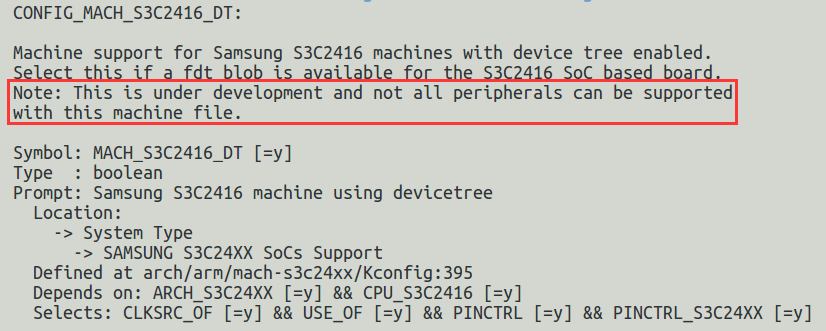
先支持2416这颗SOC



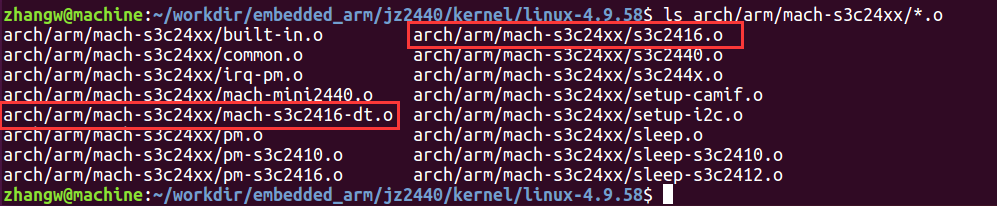
开启后果真发现了2416有设备树的相关选项



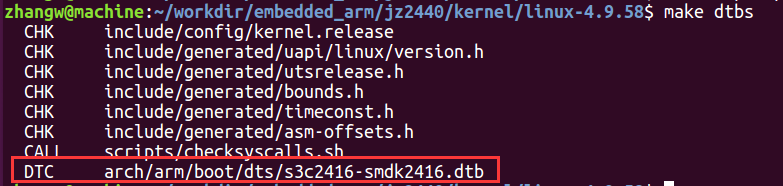
红框内的文字我们可知，设备树对外设的支持还不完全。



编译后查看可执行文件发现内核已经支持2416了。我们烧写试试看。



编译设备树，果真出来了dtb格式的设备树：



Tftp下载并启动吧：

tftp 30000000 uImage;

tftp 33000000 s3c2416-smdk2416.dtb;

bootm 30000000 – 33000000

说明下bootm命令，-代表没有ramdisk，另外需要uboot支持设备树传递，具体设备树对uboot和内核之间的传参变化参考内核的文档：Documentation/devicetree/booting-without-of.txt

你会发现竟然可以启动，并打印一些信息，但会有很多的backtrace

Booting Linux on physical CPU 0x0

Linux version 4.9.58 (zhangw@machine) (gcc version 4.8.3 20140320 (prerelease) (Sourcery CodeBench Lite 2014.05-29) ) #5 Sun Oct 29 01:22:06 CST 2017

CPU: ARM920T [41129200] revision 0 (ARMv4T), cr=c000717f

CPU: VIVT data cache, VIVT instruction cache

OF: fdt:Machine model: SMDK2416

但至少在内核启动后读到了设备树的信息。



使用2416来直接启动2440可能会有些crazy，但是至少我们有了一些参考。

接下来，我们在不改动内核任何代码的情况下，使用我提供的设备树来启动内核，听起来有一些不可思议吧。

跟着我做。

cp arch/arm/mach-s3c24xx/mach-s3c2416-dt.c arch/arm/mach-s3c24xx/mach-jz2440-dt.c

使用字符替换将s3c2416 全部替换为 jz2440

：%s/s3c2416/ jz2440/g

修改arch/arm/mach-s3c24xx/Makefile文件，加入69行代码，这里我偷了一个懒，即没有写Kconfig，直接使用2416的config。



接下来编译内核，并将我提供的设备树 tftp下载运行测试。

设备可以启动，并且可以nfs挂载。

test\_cmd=tftp 30000000 uImage;tftp 33000000 s3c2440-jz2440.dtb;bootm 30000000 – 33000000

bootargs=root=/dev/nfs nfsroot=192.168.3.48:/source/rootfs ip=192.168.3.222 console=ttySAC0,115200

我们将mini2440-machine代码不编译进入内核，发现依旧可以起来，这是为什么呢？

也就是在使用设备树之后，我只是将原先的machine中描述资源的代码放入了设备树中而已，如果想知道他是如何工作的，那我们只能去看驱动了，因为驱动才是使用这些资源的。

Mini2440 machine init简要分析：

MACHINE\_START(MINI2440, "MINI2440")

/\* Maintainer: Michel Pollet <buserror@gmail.com> \*/

.atag\_offset = 0x100, //uboot传递的tag在内存的偏移大小

.map\_io = mini2440\_map\_io,

.init\_machine = mini2440\_init,

.init\_irq = s3c2440\_init\_irq,

.init\_time = mini2440\_init\_time,

MACHINE\_END

这个结构体很重要，我们需要先知道我们实现的4个callback何时调用。

map\_io init\_machine init\_irq init\_time

arch\_initcall(customize\_machine);

**setup.c (arch\arm\kernel) :** customize\_machine() **-> machine\_desc->init\_machine();**

**mmu.c (arch\arm\mm) : devicemaps\_init ()->mdesc->map\_io();**

**irq.c (arch\arm\kernel) : init\_IRQ()-> machine\_desc->init\_irq();**

**time.c (arch\arm\kernel) : time\_init()-> machine\_desc->init\_time();**

**我提供的设备树将是一个最简单的设备树。**