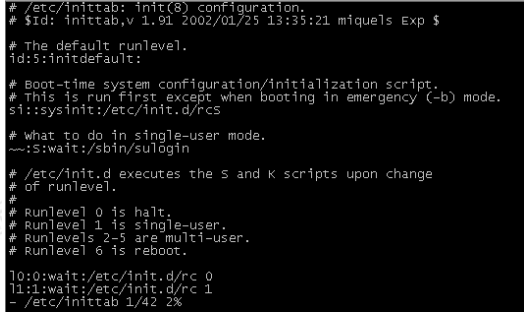
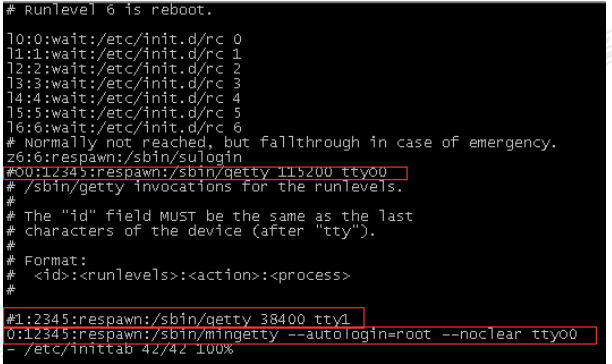
**一．设置root用户自动登录 开机无法进入文件系统**

1. **统解决方法：**

首先修改开发板 linux 系统的 inittab 配置文件，执行如下命令：  
Target# vi /etc/inittab



找到如下两行代码，并在其前面添加#，将这两行代码注释掉，代码如下：  
O0:12345:respawn:/sbin/getty 115200 ttyO0  
1:2345:respawn:/sbin/getty 38400 tty1  
在文件最后添加如下代码：  
0:12345:respawn:/sbin/mingetty --autologin=root --noclear ttyO0



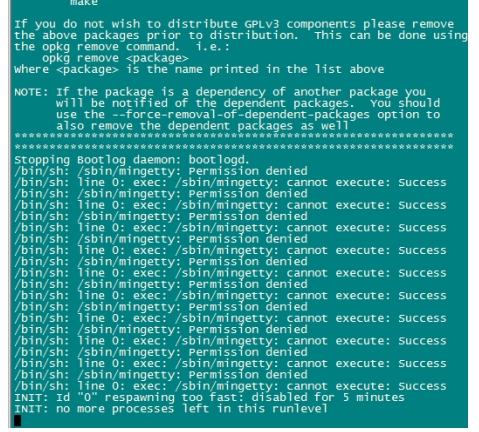
保存并退出 vi。  
将"AM437x 开发板光盘资料/demo/app/root 用户自动登录/bin"目录下的 mingetty 程  
序拷贝到开发板根文件系统的/sbin 目录下，以下是在 Ubuntu 下使用 U 盘实现拷贝，用户也可使用网络进行拷贝。



重启开发板系统，即可直接进入 root 用户。

**问题**：：：

完成以上步骤 开机重启出现



**解决方法：**

mingetty倒进/sbin文件下 要使用 chmod +x /sbin/mingetty 修改权限

**二．执行make menucofig发生错误“Unable to find the ncurses libraries or the required header files”的解决方法**

起始  
执行 make menuconfig 有错误发生  
错误:  
\*\*\* Unable to find the ncurses libraries or the  
\*\*\* required header files.  
\*\*\* 'make menuconfig' requires the ncurses libraries.  
\*\*\*  
\*\*\* Install ncurses (ncurses-devel) and try again.  
\*\*\*  
make[1]: \*\*\* [scripts/kconfig/dochecklxdialog] 错误 1  
make: \*\*\* [menuconfig] 错误2

可以看出，是因为找不到 ncurses 导致的错误。那么尝试安装 ncurses  
$sudo apt-get install ncurses  
信息：  
正在读取软件包列表... 完成  
正在分析软件包的依赖关系树  
读取状态信息... 完成  
现在没有可用的软件包 ncurses , 但是它被其他的软件包引用了。  
这可能意味着这个缺失的软件包可能已被废弃，  
或者只能在其他发布源中找到  
E： 软件包 ncurses 还没有可供安装的候选者  
根据<http://www.linuxquestions.org/qu> ... nfig-archum-311781/  
再试：  
$sudo apt-get install libncurses\*  
信息：  
正在读取软件包列表... 完成  
正在分析软件包的依赖关系树  
读取状态信息... 完成  
注意，根据正则表达式“libncurses\*” 选中了 libncurses4  
注意，根据正则表达式“libncurses\*” 选中了 libncurses5  
注意，根据正则表达式“libncurses\*” 选中了 libncurses-ruby1.8  
注意，根据正则表达式“libncurses\*” 选中了 libncurses-ryby1.9  
注意，根据正则表达式“libncurses\*” 选中了 libncurses-dev  
注意，根据正则表达式“libncurses\*” 选中了 libncursesw5-dbg  
注意，根据正则表达式“libncurses\*” 选中了 libncursesw5-dev  
注意，根据正则表达式“libncurses\*” 选中了 libncurses-ruby  
注意，根据正则表达式“libncurses\*” 选中了 libncurses5-dbg  
注意，根据正则表达式“libncurses\*” 选中了 libncurses5-dev  
注意，根据正则表达式“libncurses\*” 选中了 libncursesw5  
已经不需要下列自动安装的软件包：  
libchewing3-data ttf-wqy-zenhei scim-chewing thunderbird-locale-zh-tw language-support-translations-zh  
libchewing3 xfonts-wqy openoffice.org-l10n-zh-cn openoffice.org-l10n-zh-tw openoffice.org-help-zh-cn  
openoffice.org-help-zh-tw language-support-fonts-zh  
使用‘apt-get autoremove’ 来删除它们。  
将会安装下列额外的软件包：  
libncurses-ruby libncurses-ruby1.8 libncurses-ruby1.9 libncurses4 libncurses5-dbg libncurses5-dev  
libncursesw5-dbg libncursesw5-dev libruby1.8 libruby1.9 ruby1.8 ruby1.9  
建议安装的软件包：  
rdoc1.8 ri1.8 ruby1.8-examples rdoc1.9 ri1.9 ruby1.9-examples  
下列【新】软件包将被安装：  
libncurses-ruby libncurses-ruby1.8 libncurses-ruby1.9 libncurses4 libncurses5-dbg libncurses5-dev  
libncursesw5-dbg libncursesw5-dev libruby1.8 libruby1.9 ruby1.8 ruby1.9  
共升级了0个软件包，新安装了12个软件包，要卸载0个软件包，有60个软件未被升级。  
需要下载8868kB的软件包。  
操作完成后，会消耗掉33.9MB的额外磁盘空间。  
您希望继续执行吗？[Y/n]

选择Y继续执行，完成后，执行sudo make menuconfig，OK！！

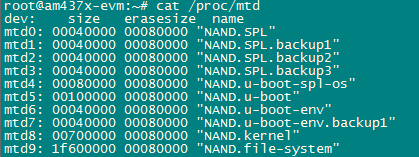
**三．固化linux系统到 NAND FLASH**

**问题：擦除NAND FLASH 下SPI分区和SPI备份分区失败 以及写失败**

1.查看系统是否分区正常，执行如下命令：

Target# cat /proc/mtd

错误



2.MLO 文件已经存放在 SD 启动卡 boot 分区。启动开发板，进入开发板文件系统。 擦

除 NAND FLASH 下 SPL 分区和 SPL 备份分区，执行如下命令：

Target# flash\_eraseall /dev/mtd0

Target# flash\_eraseall /dev/mtd1

Target# flash\_eraseall /dev/mtd2

Target# flash\_eraseall /dev/mtd3

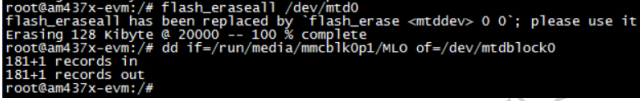
将 MLO 文件烧写到 NAND FLASH 下 SPL 分区和 SPL 备份分区，执行如下命令：

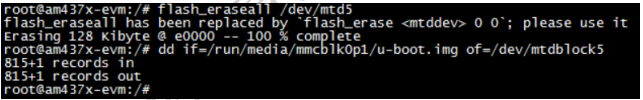
Target# dd if=/run/media/mmcblk0p1/MLO of=/dev/mtdblock0

Target# dd if=/run/media/mmcblk0p1/MLO of=/dev/mtdblock1

Target# dd if=/run/media/mmcblk0p1/MLO of=/dev/mtdblock2

Target# dd if=/run/media/mmcblk0p1/MLO of=/dev/mtdblock3





**出现错误：**

C://Users/Administrator/AppData/Local/YNote/data/1057434283@qq.com/f640e62bcfbd4541a9cc3a21f1756973/clipboard.png

**错误原因：**

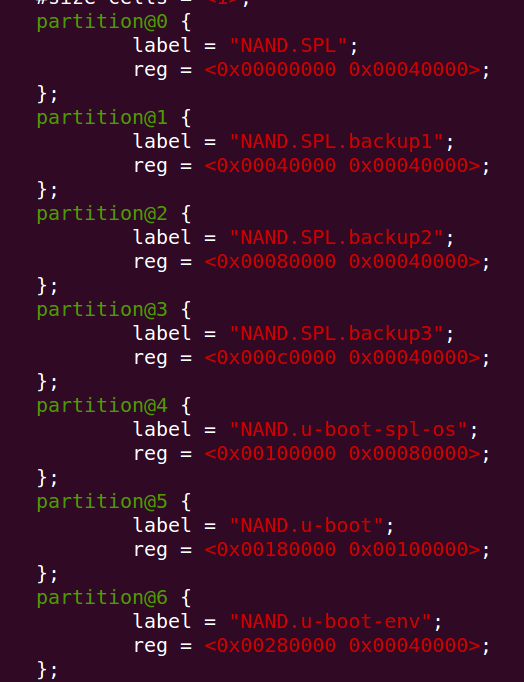
**因为nand的擦除大小大于MTD 分区大小导致MTD无法访问，以及1G nand与512M nand芯片型号的和大小的不同还有片内分区的大小不同。**

**解决方法：**

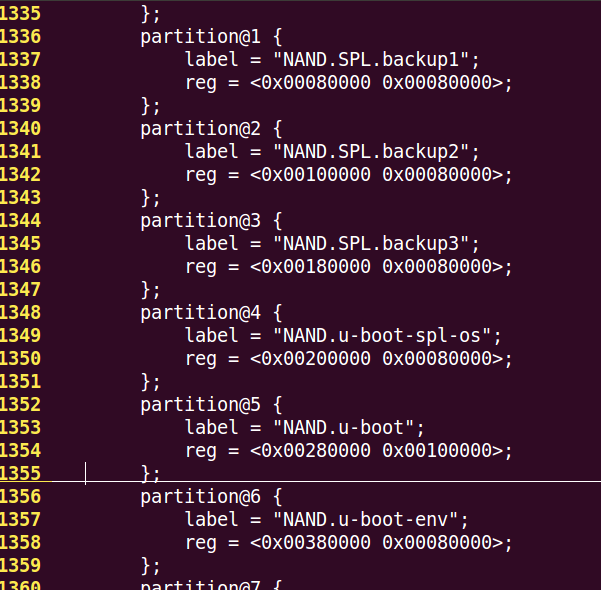
**1.修改MTD分区**

**修改linux-3.14/arch/arm/boot/dts/am437x-gp-evm.dts**

**修改前:（适用于512M nand flash）**



**修改后(适用于1G nand flash )**



**修改完成后 重新编译生成am437x-gp-evm.dtb**

**从新制作SD启动卡**

1. **修改u-boot 里的NAND 启动信息**

**修改u-boot目录下**include/configs/am43xx\_evm.h

#define CONFIG\_SYS\_NAND\_PAGE\_SIZE 2048

#define CONFIG\_SYS\_NAND\_OOBSIZE 128

改为

#define CONFIG\_SYS\_NAND\_PAGE\_SIZE 4096

#define CONFIG\_SYS\_NAND\_OOBSIZE 224

#define CONFIG\_SYS\_NAND\_ECCPOS { 2, 3, 4, 5, 6, 7, 8, 9, \

10, 11, 12, 13, 14, 15, 16, 17, 18, 19, \

20, 21, 22, 23, 24, 25, 26, 27, 28, 29, \

30, 31, 32, 33, 34, 35, 36, 37, 38, 39, \

40, 41, 42, 43, 44, 45, 46, 47, 48, 49, \

50, 51, 52, 53, 54, 55, 56, 57, 58, 59, \

60, 61, 62, 63, 64, 65, 66, 67, 68, 69, \

70, 71, 72, 73, 74, 75, 76, 77, 78, 79, \

80, 81, 82, 83, 84, 85, 86, 87, 88, 89, \

90, 91, 92, 93, 94, 95, 96, 97, 98, 99, \

100, 101, 102, 103, 104, 105, \

}

改为

#define CONFIG\_SYS\_NAND\_ECCPOS \ { 2, 3, 4, 5, 6, 7, 8, 9, \ 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, \ 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, \ 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, \ 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, \ 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, \ 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, \ 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, \ 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, \ 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, \ 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, \ 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, \ 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, \ 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, \ 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, \ 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, \ 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, \ 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, \ 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, \ 190, 191, 192, 193, 194, 195, 196, 197, 198, 199, \ 200, 201, 202, 203, 204, 205, 206, 207, 208, 209, }

"256k(NAND.SPL)," \

"256k(NAND.SPL.backup1)," \

"256k(NAND.SPL.backup2)," \

"256k(NAND.SPL.backup3)," \

"512k(NAND.u-boot-spl-os)," \

"1m(NAND.u-boot)," \

"256k(NAND.u-boot-env)," \

"256k(NAND.u-boot-env.backup1)," \

"7m(NAND.kernel)," \

"-(NAND.file-system)"

改为

"512k(NAND.SPL)," \

"512k(NAND.SPL.backup1)," \

“512k(NAND.SPL.backup2)," \

"512k(NAND.SPL.backup3)," \

"512k(NAND.u-boot-spl-os)," \

"1m(NAND.u-boot)," \

"512k(NAND.u-boot-env)," \

"512k(NAND.u-boot-env.backup1)," \

"7m(NAND.kernel)," \

"-(NAND.file-system)"

#define CONFIG\_SYS\_NAND\_U\_BOOT\_OFFS 0x00180000

改为

#define CONFIG\_SYS\_NAND\_U\_BOOT\_OFFS 0x00280000

#define CONFIG\_SYS\_NAND\_SPL\_KERNEL\_OFFS 0x00300000 /\* kernel offset \*/

改为

#define CONFIG\_SYS\_NAND\_SPL\_KERNEL\_OFFS 0x00480000 /\*kernel offset\*/

**四 .SSH无法登陆**

**解决方法：**

在开发板操作系统下：

打开文件系统的 /etc/passwd 文件

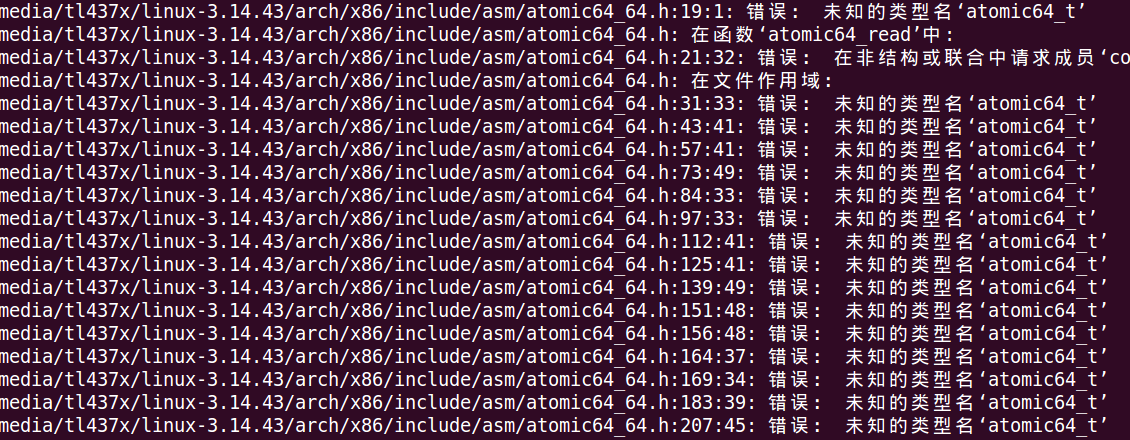
检查第一行是否是

root::0:0:root:/home/root:/bin/ash

如果是，将该行替换成如下内容

root::0:0:root:/home/root:/bin/sh

重新测试 ssh 连接

编译驱动模块时出现

问题是编译平台错误

解决方法

1. 修改内核顶层目录下的 Makefile  
$ vm Makefile  
修改：

**ARCH ?= $(SUBARCH)  
CROSS\_COMPILE ?= $(CONFIG\_CROSS\_COMPILE:”%”=%)**

为  
**ARCH ?= arm  
CROSS\_COMPILE ?=** arm-linux-gnueabihf-

1. 在make 后面添加arm-linux-gnueabihf-

$ make arm-linux-gnueabihf-

串口 5个全部打开 修改linux3.14.43/arch/arm/boot/dts/am437x-gp-evm.dts

修改1204行 把 #if 0 去掉