LINUX KERNEL,ROOTFS裁剪与构建

V1.0



四川长虹网络科技有限责任公司

变更记录

|  |  |  |  |
| --- | --- | --- | --- |
| 版本号 | 修改点说明 | 记录人 | 变更日期 |
| V1.0 | 创建 | 张强 |  |
|  |  |  |  |
|  |  |  |  |

*修改点说明的内容有如下几种：创建、修改（+修改说明）、删除（+删除说明）*

**目录**

[LINUX KERNEL**裁剪原则** 4](#_Toc522717058)

[LINUX ROOTFS**基本构件** 5](#_Toc522717059)

[libc**的交叉编译** 5](#_Toc522717060)

[libc**编译结果** 8](#_Toc522717061)

[Busybox**配置说明** 11](#_Toc522717062)

[mdev**说明** 13](#_Toc522717063)

[mdev原理 13](#_Toc522717064)

[mdev使用 13](#_Toc522717065)

[系统设置 13](#_Toc522717066)

[配置文件/etc/mdev.conf 14](#_Toc522717067)

[LINUX ROOTFS**目录** 18](#_Toc522717068)

[1. 创建根文件系统的目录 18](#_Toc522717069)

[2. 创建设备文件 18](#_Toc522717070)

[3. 配置/etc 18](#_Toc522717071)

[4. libc so库文件 19](#_Toc522717072)

[5. Busybox bin**文件** 20](#_Toc522717073)

[LINUX ROOTFS**打包和使用** 21](#_Toc522717074)

[LINUX ROOTFS**裁剪原则** 22](#_Toc522717075)

# LINUX KERNEL**裁剪原则**

目前，通用KERNEL的大小压缩前5~6MB左右，压缩后2~3M左右。裁剪的一般原则如下:

1. Menuconfig中去掉不需要功能的编译配置。例如，项目不需要网络部分，可以去掉网络的相关配置。
2. 如果对启动速度不是太敏感，可以采用压缩kernel的方式，这样可以直接省掉2~3M的存储空间占用。这么处理会牺牲掉启动时间大概1~2s。
3. 目前，LINUX DVB项目高安CA对Linux config有严格的要求，其保证安全的主要原则如下:

1). 去掉所有的终端打印信息和调试信息及端口。去掉所有通过网络连接进系

统的配置及端口，比如telnet，ssh2等。

2). 去掉不必要的功能配置。

3). 采用沙箱进行进程隔离，增加权限管理，按功能划分进程模块等。

4). 如果有相应网络应用，需要添加防火墙和严格的端口管理，关闭不必要的

端口，网络及防火墙按照需求进行配置。

1. 实验证明，基础DVB功能在极限情况下，部分平台可以将kernel裁剪到2M以下，这点还是要根据kernel内包含驱动的情况而定。

# LINUX ROOTFS**基本构件**

对于LINUX嵌入式系统，一个重要的工作就是构建ROOTFS文件系统，目前，一般芯片厂商会提供基础的根文件系统版本，我们需要根据项目的实际需求对ROOTFS进行调整和修改，接下来是其中两个重要部分，libc和busybox。

## libc**的交叉编译**

libc是程序运行的基础库，一般芯片厂商提供的rootfs已经包含了相应libc的库，如果其提供的库是strip版本的,即去掉了库中的debug信息，那么我们在gdb调试问题的时候就会非常不方便，会出现corrupted stack问题。如果我们掌握交叉编译libc库的方法对于自己构建文件系统和调试问题会起到有益的作用。

下面介绍我们在MStar Linux平台上交叉编译Libc的方法,如下:

1. Frist,download sourcecode from GNU website.

X G The GNU C Library 
C G) vwm.gnu.org/software/libc/ 
X Index of /gnu/libc/ X 
The current stable version of g c is 2.27, released on February 1, 2018. 
The current development versio of glibc 2.28, releasing on or around August 1, 2018. 
Latest News 
01: glibc 2.27 released. 
2018-02- 
-02: glibc 2.26 released. 
2017-08 
glibc 2.25 released. 
2017-02-05 
Test results mailing list is active. 
2016-11-24: 
-05: glibc 2.24 released. 
2016-08 

1. Unzip and enter the glibc-2.27(for example) folder.
2. configure

set MSTAR MIPS cross-compile toolchain path:

C:\Users\ZQiang\AppData\Local\Packages\Microsoft.Office.OneNote_8wekyb3d8bbwe\TempState\msohtmlclip\clip_image002.png

then, configure like below:configure --host=mips-linux "CC=mips-linux-gnu-gcc -EL -mno-compact-eh" "CXX=mips-linux-gnu-g++ -EL -mno-compact-eh" "LDFLAGS= -EL" --prefix=/home/amd64/opensource/glibc-2.20/installpath

But get below error:

-g++ -EL -mno-compact-eh 
" "LDFLRGS= 
configure: command not Found 
. 'configure 
ensource/g1ibc-2.2ß/insta11path 
yes 
EL" --preFix=/home/amd64/opensource/g1ibc-2.2ß/insta11path 
--host=mips-linux 
"CC=mips-1inux-gnu-gcc -EL 
checking 
checking 
checking 
checking 
checking 
checking 
checking 
checking 
checking 
checking 
checking 
checking 
build system type... 
x86 64-pc-linux-gnu 
mips-unknown-linux-gnu 
host system type.. 
For mips-linux-gcc... 
mips-linux-gnu-gcc -EL -mno- 
For suFFix OF object Files... o 
whether we are using the GNU C compiler 
• yes 
act-eh 
whether mips-linux-gnu-gcc -EL -mno-compact-eh accepts -g... 
For gcc... 
gcc 
For mips-linux-reade1F... 
For reade1F... 
reade1F 
whether we are using the GNU C++ compiler... 
yes 
whether mips-linux-gnu-g++ -EL -mno-compact-eh accepts -g 
whether mips-linux-gnu-g++ -EL -mno-compact-eh can link progra 
configure: error: you must configure in a separate build directory 
amd64@ubuntu :N/opensource/g1ibc-2.27$ 

so make "build" dir in glibc-2.27, and "installpath" dir for lib and bin files install path,which is set with"--prefix" in configure.

RECOMMEND: make a shell script with commands "export" and "configure" to folder "build".

build mstar.sh 
1 #!/bin/sh 
export PA TH 
*make distclean 
../configure —host—mips-linux "CC=mips4inux-gnu-gcc -EL -mno-compact-eh" "CXX=mips4inux-gnu-g++ -EL -mno-compact-eh" 
*make 
*make install 

but still failed,bcs my toolchain is too old,so we download glibc ver2.20:

•hecking 
:hecking 
:hecking 
:hecking 
:hecking 
:hecking 
:hecking 
:hecking 
:hecking 
:hecking 
:hecking 
:hecking 
:hecking 
:hecking 
:hecking 
:hecking 
:hecking 
:hecking 
For make... 
make 
version OF make... 
For gnumsgFmt... 
For gmsgFmt... n 
For msgFmt... 
msgFmt 
version OF msgFmt... 
For makeinFo... 
makeinFo 
version OF makeinFo... 
For sed... 
sed 
version OF sed... 
For gawk... 
version OF gawk... 
For bison... 
bison 
version OF bison... 
if mips-linux-gnu-gcc -EL 
For mips-linux-nm... n 
-mno-compact-eh is sufficient to build 
For nm... 
For python3... 
python3 
:onFigure: error: 
These critical programs are missing or too old: as GNU Id compiler 
Check the INSTALL File For required versions. 

1. Repeat step 1~3 with ver2.20, below is the correct configure log.

checking For C cleanup handling... 
yes 
running configure Fragment For sysdeps/gnu 
running configure Fragment For sysdeps/mips 
checking For grep that handles long lines and -e... 
checking For egrep... 
(cached) 'bin/grep -E 
(cached) 'bin/grep 
checking whether the compiler is using the 2908 NaN encoding... 
checking For old glibc 2. g.x headers... n 
checking whether -EPIC is default... 
configure: creating ./conFig.status 
(cached) 
no 
config 
config 
config 
config 
• creating config-make 
-status. 
• creating Makefile 
-status. 
• creating conFig.h 
-status. 
• executing default commands 
-status. 

1. Pls note below options, “-EL”(mips little endian), “-mno-compact-eh”(no compact   
   exception handler),

CC=mips4inux-gnu-gcc -EL -mno-compact-eh" 

Otherwise, if no option "-mno-compact-eh",will encounter below error,



1. But still "compact exception handler" errors occor,while compile "sln" and "ldconfig" elf files. I just delete them from makefile.

elf/Makfile:

#others = sprof sln

others = sprof

install-bin   = sprof

#others-static = sln

#install-rootsbin = sln

others-static =

install-rootsbin =

#others-static += ldconfig

#others         += ldconfig

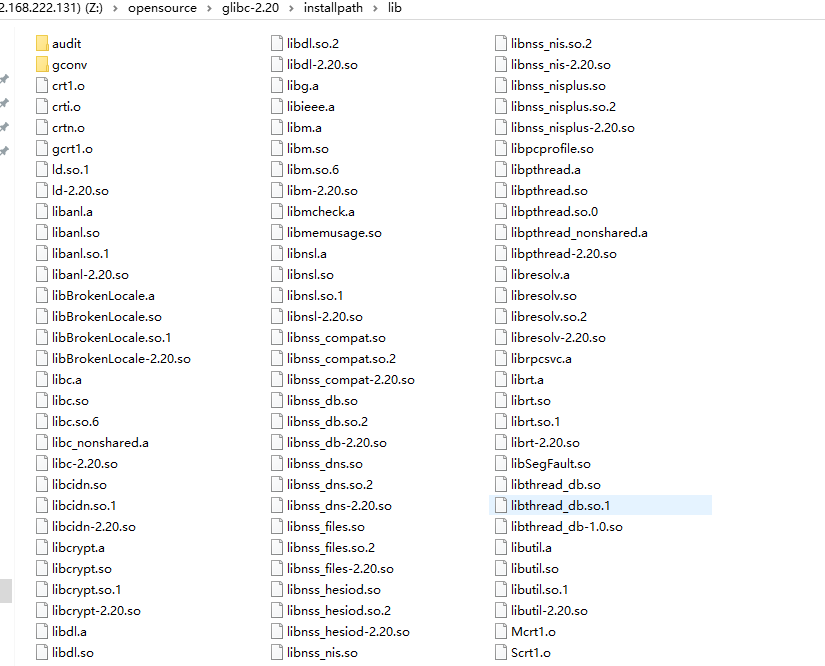
#install-rootsbin += ldconfig

NOTES: "compact exception handler" error issue, it is on mips platfrom, bcs of DWARF version are not compatible between gcc and glibc, I guess.

7. ok, then, "make;make install",and you can check glibc lib and bin files in folder "installpath".

## libc**编译结果**

libc编译文成后会生产所有的libc库，如下图所示:



可见生成动态库的同时也生成了静态库，不过，一般linux开发均把动态库放到rootfs的/lib目录，那么，我们就可以将自己编译libc库替换掉根文件系统/lib中的库。

以下事项请注意:

1. 不建议使用自己编译的libc库进行生产，建议最终软件还是使用源厂提供的libc库。
2. 源厂提供的libc库如果未经裁剪，可以通过交叉编译XXX-linux-gnu-strip进行裁剪。
3. 自己编译的libc库，可以加入-g 命令配合gdb问题调试使用(相关c库内容会显示源码)。也可以添加打印分析相关问题原因。
4. 如果需要自己构建rootfs，那么libc库生成后还需要进行详细的测试验证。

**busybox的交叉编译**

1). busybox是Linux上的一个应用程序(application)，即只有一个ELF文件头。

2). 它整合了许多Linux上常用的工具和命令（utilities)， 如rm, ls, gzip, tftp等。对于这些工具和命令，busybox中的实现可能不是最全的，但却是最常用的，因此它的特点就是短小精悍，特别适合对尺寸很敏感的嵌入式系统。

3). busybox的官方网站是[http://www.busybox.net/](http://www.busybox.net/" \t "_blank)，在这里你可以找到与busybox相关的所有资料。

其编译说明如下:

1. Frist,download sourcecode from GNU website.

Busygox 
. FAQ 
Command Help 
Get BusyBox 
Download Source 
• 22May 2018 
usyBox 1.28.4 (stable) 
BusyBox 1.28_4_ (gitv patches, how to add a Qatch) 
Bug fix release. 1 _28_4 has fixes for nsenter option pars 

1. Set toolchain path like Glibc,but there is no "configure" tool in busybox.

3. "make menuconfig" ,set compile options below, roduce additional busybox binary linked against libbusybox 
(mips-linux-gnu-) ross compiler prefix 
( opt/mstar/mips-2ß14. g5/bin/ 'mips-linux-gnu/libc) to sysroot 
(-EL) dditional CFLRGS 
(-EL) dditional LDFLRGS 
dditional LDLIBS 
o 

If you don’t know how to set "path to sysroot", you can using this command "mips-linux-gnu-gcc-print-sysroot",pls note you should set you own gcc command,which I used is "mips-linux-gnu-gcc".

If the sysroot is not set, something unpredictable could happen,like below,

file "limits.h" open error in version 1.22.1 , modify method ->

command "mips-linux-gnu-gcc -print-sysroot" to get sysroot，and "make menuconfig".

CONFIG\_CROSS\_COMPILER\_PREFIX="mips-linux-gnu-"

CONFIG\_SYSROOT="/opt/mstar/mips-2014.05/bin/../mips-linux-gnu/libc"

CONFIG\_EXTRA\_CFLAGS="-EL"

CONFIG\_EXTRA\_LDFLAGS="-EL"

In addtion，"CONFIG\_EXTRA\_LDFLAGS = "-EL"” in .config file seems no effect，so modifiy Makefile below

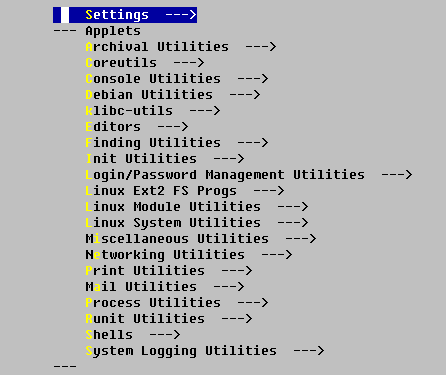
"LD = $(CC) -nostdlib -EL".

Busybox Settings ---> build Options ---> 选中Build BusyBox as a static binary(no shared libs)

4. ok, then, "make;make install",and you can check busy lib and bin files in folder "installpath".

## Busybox**配置说明**

Busybox需要仔细配置config以适合单板rootfs文件系统的构建,busybox可以像Uboot和Kernel一样通过make menuconfig 来配置编译，下面是该配置的一些基本配置方法。

  
1.Busybox Settings --->  //BusyBox的通用配置，一般采用默认值即可。     
2.---Applets    
3.Archival Utilities --->  //压缩、解压缩相关工具。     
4.Coreutils --->           //最基本的命令，如cat、cp、ls等。     
5.Console Utilities --->       //控制台相关命令。     
6.Debian Utilities --->        //Debian操作系统相关命令。     
7.Editors --->         //编辑工具，如vi、awk、sed等。     
8.Finding Utilities --->       //查找工具，如find、grep、xargs。     
9.Init Utilities --->      //BusyBox init相关命令。     
10.Login/Password Management Utilities---> //登陆用户账号/密码等方面的命令   
11.Linux Ext2 FS Progs ---> //ext2文件系统的一些工具。     
12.Linux Module Utilities --->  //加载/卸载模块等相关的命令。     
13.Linux System Utilities --->  //一些系统命令。     
14.Miscellaneous Utilities ---> //一些不好分类的命令，如crond、crontab。     
15.Networking Utilities --->    //网络相关的命令和工具。     
16.Print Utilities --->     //print spool服务及相关工具。     
17.Mail Utilities --->      //mail相关命令。     
18.Process Utilities --->       //进程相关命令，如ps、kill等。     
19.Runit Utilities --->     //runit程序。     
20.Shells --->              //shell程序。     
21.System Logging Utilities --->    //系统日志相关工具，如syslogd、klogd

## mdev**说明**

mdev是busybox自带的一个简化版的udev，适合嵌入式应用场合。其具有使用简单的特点。它的作用就是在**系统启动**和**热插拔**或**动态加载驱动程序**时，自动产生驱动程序所需要的节点文件。在以busybox为基础构建嵌入式linux根文件系统时，使用它时最优的选择。

### mdev原理

1. mdev不加参数时就是利用hotplug机制来决定创建什么样的设备文件。
2. mdev -s程序通过扫描/sys/class和/sys/block中所有的类设备目录，利用这些信息在/dev下创建设备节点。

### mdev使用

mdev在使用过程中分为两部分，由于系统启动时mdev需要根据/sys目录下的信息来生成设备节点，因此在使用前需要对系统进行一些配置。mdev同过hotplug机制来创建设备节点是根据/etc/mdev.conf文件来生成设备节点的。

### 系统设置

在系统执行mdev -s前需要执行如下操作：

1. 配置内核

make menuconfig

General setup ---->

Configure standard kernel features (for small systems) ---->

[\*] load all symbols for debugging/ksymoops

[\*] Include all symbols in kallsyms

[\*] Support for hot-pluggable devices

[\*] Enable support for printk

1. 配置busybox

make menuconfig

Linux System Utilities ---->

[\*] mdev

[\*] Support /etc/mdev.conf

[\*] Support subdirs/symlinks

[\*] Support regular expressions substitutions when renaming device

[\*] Support command execution at device addition/removal

[\*] Support loading of firmwares

1. 修改系统启动时脚本

vi /etc/init.d/rcS

mount -t tmpfs tmpfs /dev

mkdir /dev/pts

mount -t devpts devpts /dev/pts

mount -t proc proc /proc

mount -t sysfs sysfs /sys

echo /sbin/mdev>/proc/sys/kernel/hotplug//启动热插拔事件；

mdev –s

首先挂载/dev、/dev/pts、/proc和/sys文件系统，mdev需要用到这些文件系统。然后告诉系统当有设备热插拔时，使用mdev来处理。最后执行mdev -s来扫描系统中的设备和驱动等。

### 配置文件/etc/mdev.conf

系统中的hotplug是通过mdev.conf文件来生成设备节点的，该配置文件格式如下：

1. 基本格式

<device regex> <uid>:<gid> <octal permissions>

<device regex> :设备名称，支持正则表达式如hd[a-z][0-9]\*等

<uid>:<gid> :用户ID和组ID

<octal permissions> :八进制表示的设备属性

1. 执行脚本格式

<device regex> <uid>:<gid> <octal permissions> [=|>path] [@|$|\*]

[=|>path]:这个选项可以更改设备节点的命名和路径，如：

<1> =/driver: 可以将设备节点移动到driver目录下

<2> =newname: 可以讲设备节点改为newname命名

<3> >/driver/newname: 可以在/driver目录下创建一个设备节点的链接，并命名为newname

[@|$|\*]:这个选项当设备匹配成功时，执行指令，这个指令可以是自己编写的脚本。前面的符号含义如下：

<1>@:在设备节点创建完执行

<2>$:在设备节点删除前执行

<3>\*:在设备节点创建完和删除前执行

此外在mdev成功匹配设备后会设置两个系统变量$MDEV和$ACTION。其中$MDEV用来存放匹配到的设备名，$ACTION用来存放设备插拔状态其值为add和remove。这两个变量可以在脚本中使用。

脚本实例

mdev.conf

# system all-writable devices

full 0:0 0666

null 0:0 0666

ptmx 0:0 0666

random 0:0 0666

tty 0:0 0666

zero 0:0 0666

# console devices

tty[0-9]\* 0:5 0660

vc/[0-9]\* 0:5 0660

# serial port devices

s3c2410\_serial0 0:5 0666 =ttySAC0

s3c2410\_serial1 0:5 0666 =ttySAC1

s3c2410\_serial2 0:5 0666 =ttySAC2

s3c2410\_serial3 0:5 0666 =ttySAC3

# loop devices

loop[0-9]\* 0:0 0660 =loop/

# i2c devices

i2c-0 0:0 0666 =i2c/0

i2c-1 0:0 0666 =i2c/1

# frame buffer devices

fb[0-9] 0:0 0666

# input devices

mice 0:0 0660 =input/

mouse.\* 0:0 0660 =input/

event.\* 0:0 0660 =input/

ts.\* 0:0 0660 =input/

# rtc devices

rtc0 0:0 0644 >rtc

rtc[1-9] 0:0 0644

# misc devices

mmcblk0p1 0:0 0600 =sdcard \*/bin/hotplug.sh

mmcblk0 0:0 0600 =mmcblk0 \*/bin/hotplug.sh

sda1 0:0 0600 =udisk \* /bin/hotplug.sh

/bin/hotplug.sh

#!/bin/sh

case $MDEV in

sda1)

DEVNAME=udisk

MOUNTPOINT=/udisk

;;

mmcblk0p1)

DEVNAME=sdcard

MOUNTPOINT=/sdcard

;;

mmcblk0)

DEVNAME=mmcblk0

MOUNTPOINT=/sdcard

;;

\*)

exit 0

;;

esac

case $ACTION in

remove)

/bin/umount $MOUNTPOINT || true

rmdir $MOUNTPOINT >/dev/null 2>&1 || true

;;

\*)

/bin/mkdir $MOUNTPOINT > /dev/null 2>&1 || true

/bin/mount -o sync -o noatime -o nodiratime -t vfat /dev/$DEVNAME $MOUNTPOINT > /dev/null 2>&1 || true

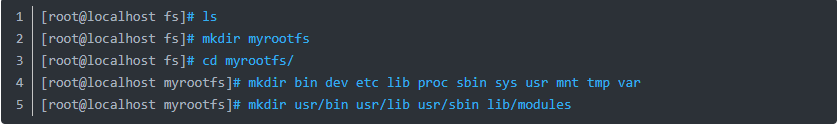
;;

esac

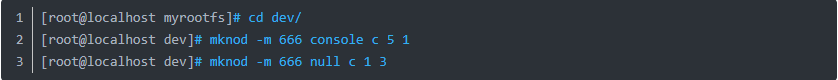
exit 0

# LINUX ROOTFS**目录**

## **创建根文件系统的目录**



## **创建设备文件**



**/dev/console /dev/null,这两个设备是通用的终端设备和null设备。**

## 配置/etc

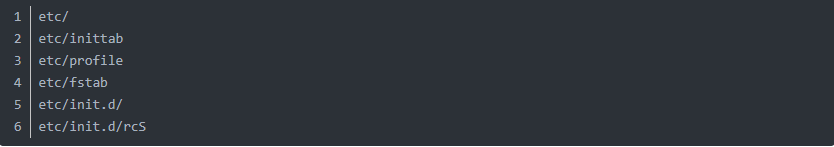
3.1 获取配置文件

     把busybox源码目录下的etc的内容拷贝到这里的/home/fs/myrootfs/etc下

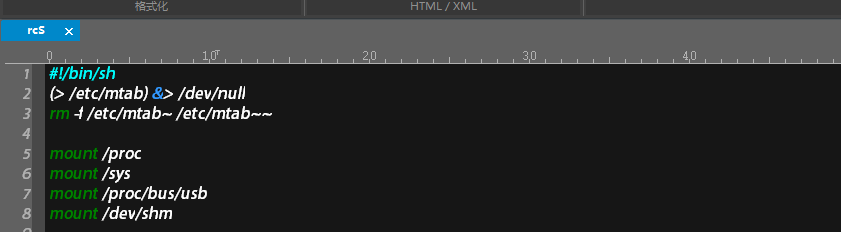


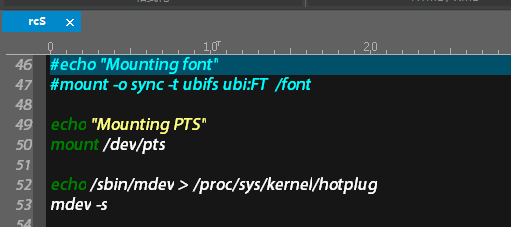
  3.2 修改配置文件

    我们可以根据我们需要，对下面列出的配置文件，做相应的修改，可参考网上资料。

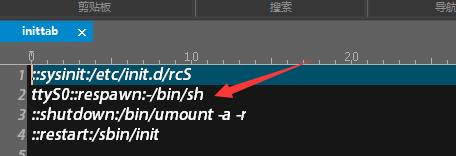


/etc/init.d/rcS (一般rcS中添加需要mount的分区,insmod相关的内核ko驱动)

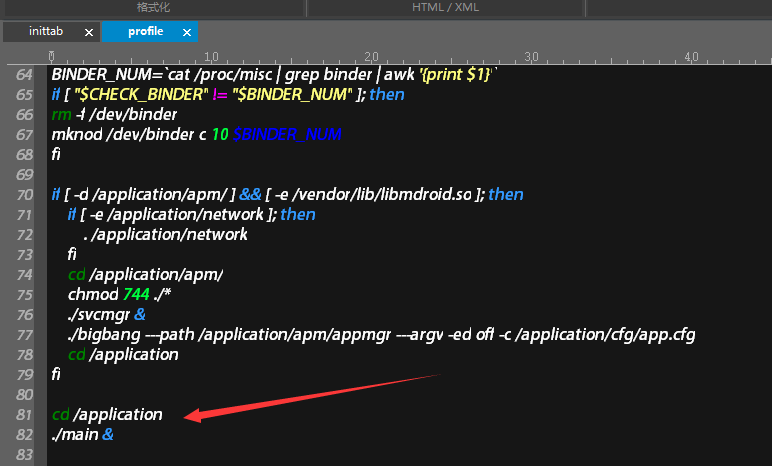




/etc/inittab(注意:respawn不管何时终止都重新启动进程，askfirst则需要在终端弹出提示，要确认后才会继续执行)

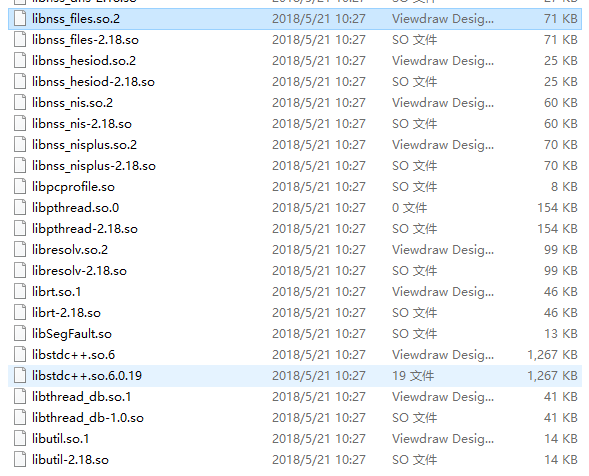


/etc/profile启动脚本，自动执行app一般加到该脚本下。



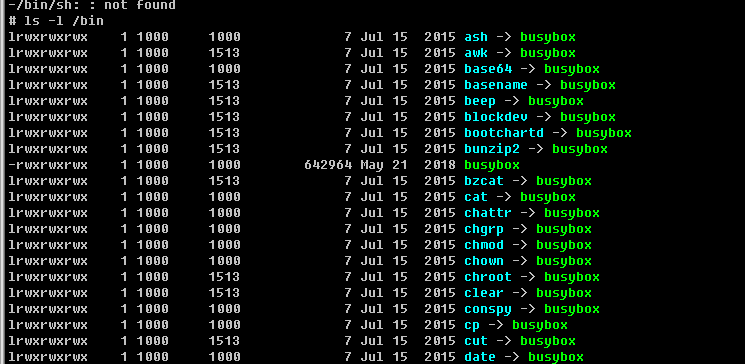
## libc so库文件

交叉编译的libc动态库，拷贝到rootfs的/lib目录。



## Busybox bin**文件**

编译的busybox bin放入rootfs /bin,其他命令在/bin /sbin目录中建立软链接指向busybox bin。



# LINUX ROOTFS**打包和使用**

1. **Initramfs**
   1. **安装initramfs根文件系统**

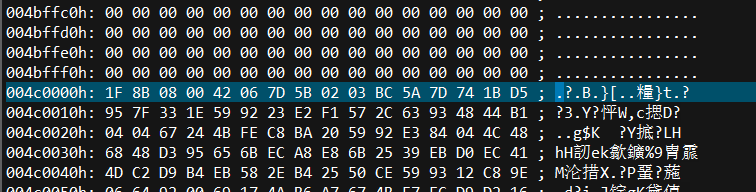
要使用initramfs做根文件系统，在内核中做如下配置，并重新编译内核镜像 即可。  
General setup --->  
[\*] Initial RAM filesystem and RAM disk (initramfs/initrd) support

  
(/rootfs\_dir) Initramfs source file(s) //输入根文件系统的所在目录，如果目录为空则需要另行打包文件系统image。不为空要保证文件系统正确，生成的image是一个包含kernel和rootfs的image。

* 1. **将根文件系统制作成ramdisk**

# cd ROOTFS  
# sudo find . | sudo cpio -o -H newc | gzip -9 > rootfs.cpio.gz

生成的rootfs cpio镜像还需要和kenrel image文件合并到一起。例如： STB MSTAR平台会将kernel image填充到0x4CFFFF,然后cpio镜像的位置从0x4C0000起，如下图:



Boot启动的时候会分别加载kernel镜像和cpio rootfs镜像到相应的内存位置使用。

1. UBIFS

Kernel配置支持UBIFS，同样需要打包成ubifs和ubi image。

# ./mkfs.ubifs -r ROOTFS -o ROOTFS.ubifs -m 2048 -e 126976 -c 123 -v

注意: -m -c需要和Nand Flash block大小和rootfs实际大小确认。该例中使用128MB nand，126976 = 128K – 2K； 123=0xf00000/126976。

1. 其他rootfs打包

squashfs,yaffs2等其他文件系统可以参考互联网上资料，这里不再赘述。

# LINUX ROOTFS**裁剪原则**

* 1. strip所使用的库和可执行文件(busybox),注意strip过后的库文件用于gdb调试会导致corrupt stack提示。
  2. 去掉不必要的功能模块和第三方库文件，字库文件等等。文件系统中只保留必要的文件。
  3. Busybox去掉不需要的功能，尽量减小busybox的大小。