嵌入式 Linux 常见开源工程的移植

一 ：tslib1.0移植

移植 QT tslib 触摸屏 正程序的支持，下 我们 tslib1.0 版本的编译、移植 解

下　地
址：http://www.rayfile.com/zh-cn/files/a01f838a-c1bb-11e0-aaa6-0015c55db73d/

#tar　zxvf　tslib-1.0.tar.bz2

1、生成.configure文件

./autogen.sh

2、指定交叉编译器及配　脚本

#CC=arm-linux-gcc　　//　择交叉编译器

#./configure--host=arm-linux
--prefix=$PWD/../tslib1.0_target　--enable-inputapi=no

//在当前路径的前一　目录建立目　文件夹

#exportPLUGIN_DIR=$PWD/plugins

3.编译

在tslib-1.0\config.h中有一　定义“#define malloc rpl_malloc”，直接注释

#make

#make install

则会在与当前目录平　的文件夹下建立tslib_target文件夹，　　包含bin/ etc/ include/ lib/ 等几个文件夹.

3、移植到开发板上

1）在开发板的usr/目录下建立文件夹tslib，将上述四个文件夹拷　到tslib目录下

#cd /usr/

#mkdir /usr/tslib

#cp tslib_target/* /usr/tslib -r

2）修改/usr/tslib/etc/ts.conf将 一句 的屏蔽去掉

#module_raw input

改为

module_raw input （注意前 不 有空格）

3）修改开发板 /etc目录下/profile 文件，添加以下启动脚本并运

#vi /etc/profile

export T_ROOT=/usr/tslib #目录

exportPATH=$PATH:$T_ROOT/bin #可执 程序目录加入环境变

exportLD_LIBRARY_PATH=/lib:/lib:/usr/lib:$T_ROOT/lib #库文件目录

export TSLIB_CONSOLEDEVICE=none

export TSLIB_FBDEVICE=/dev/fb0 #framebuffer设备文件

export TSLIB_TSDEVICE=/dev/event0 #触摸屏设备文件

export TSLIB_CALIBFILE=/etc/pointercal #指定触摸屏 准文件pintercal的存放位

export TSLIB_CONFFILE=$T_ROOT/etc/ts.conf #tslib模块配文件

export TSLIB_PLUGINDIR=$T_ROOT/lib/ts/ #tslib插件库目录

echo "ts init success!"

4、使　与测试

1）ts_calibrate

执　后，屏幕上出　5 个点，依次点击　　准，　准完毕后，在/etc 下会产生 pointercal 文件，供其它程序使　，ts_calibrate 是一个应　程序，在屏幕上画几个按钮，将　户点击后从 ts 驱动　得的数据和屏上的坐　位　通过一套　　来　得　准数据写到一个　准文件

2）ts_print　　将屏幕坐　打印出来

坐　　坐

1796351007.320859:　156　116　500　按下

1796351007.355145:　156　116　0　抬

3）ts_print_raw　将屏幕坐　的原始数据打印出来

1796350964.530867:　216　865　1　按下

1796350989.946831:　0　0　0　抬

4）ts_test　测一下　正　果，如果touch的时候，十字光　随着手指移动

5、常见错误的解

1）

在make的过程提　以下错误解决：

libtool:link: only absolute run-paths areallowed

忘　指定环境变　：exportPLUGIN_DIR=$PWD/plugins

2）

ts_config: Success

追查后应该是在打开ts_config中出 了错误

tslib插件库目录 有指定对

exportTSLIB_PLUGINDIR=$T_ROOT/share/ts/plugins/ #tslib插件库目录

3)

ts_open:Inappropriate ioctol for device

自己板子的触摸屏驱动不支持ioctl操作

.configure 时加上 --enable-inputapi=no 参数

4)

./ts_calibrate: error while loading shared libraries: libts-0.0.so.0: cannotopen shared object file: Error 40

./ts_calibrate: error while loading shared libraries: libts-0.0.so.0: cannotope

n shared object file: No such file or directory

解决办 ：

#echo $LD_LIBRARY_PATH //查看lib路径，

#export LD_LIBRARY_PATH=$T_ROOT/lib

5)

ts_open: No such file or directory

解决办 ：

#export TSLIB_TSDEVICE=/dev/input/event0 //触摸屏设备路径

6)

Couldnt open tslib config file: No such file or directory ts_config: Illegalseek

解决办 ：

#export TSLIB_CONFFILE=$T_ROOT/etc/ts.conf

7）

Couldnt load module pthres

No raw modules loaded.

ts_config: Success

解决办 ：

#export TSLIB_PLUGINDIR=$T_ROOT/lib/ts

8）

No raw modules loaded.

Ts_config: No such file or directory

解决办 ：

修改$T_ROOT/etc/ts.conf, 至少放开一个module_raw,并去掉前 空格

二 ：QT4.7.0 在 ARM 平台移植

在前 tslib1.0 移植完成的基础上我们可以开始 QT 的移植，以下 测试在 4.6.2、4.6.3、4.7.0、4.7.2 平台上均可顺利通过

下 地址：http://qt.nokia.com/downloads-cn

把下 文档中相应的数字改为对应的版本即可

step 1:

echo yes |../qt-everywhere-opensource-src-4.7.0/configure -opensource -release -fast -embedded arm -xplatform qws/linux-arm-g++ -depths 8,16,32 -no-stl -no-qt3support -no-nis -no-cups -no-iconv -no-qdbus -make libs -nomake docs -qt-freetype -qt-sql-sqlite -qt-kbd-tty -qt-libtiff -qt-libjpeg -qt-gif -qt-libpng -continue -silent -no-mouse-linuxtp -qt-mouse-tslib-I/home/dengwei/QT_test/tslib_src/tslib-1.0/tslib1.0_target/include -L/home/dengwei/QT_test/tslib_src/tslib-1.0/tslib1.0_target/lib

注意：带　色的字体部分　改成对应的目录 -I /-L 指定前　编译出的 tslib 的路径

#make
#make install
/*会安装库、字体等文件到/usr/local/Trolltech/QtEmbedded-4.7.2-arm目录。*/
注意：编译之前使　unset CC 取　CC变　的定义，否则编译会出错

step 2:
拷　/usr/local/Trolltech/QtEmbedded-4.7.0-arm/lib目录下的以下文件到根文件系　/usr/lib目录下：
#cp -a /usr/local/Trolltech/QtEmbedded-4.7.0-arm/lib/*so* rootfs/usr/lib
（也可以不　全部拷　，可以根据　　拷　，这　为了方便我们全拷　过去）

step 3:

在根文件系　下建立目录

#mkdir /usr/lib/fonts

将QtEmbedded-4.7.0-arm/lib/fonts目录下的字体unifont_160_50.qpf拷到开发板对应的目录
#cp /usr/local/Trolltech/QtEmbedded-4.7.0-arm/lib/fonts/unifont_160_50.qpf /usr/local

step 4:

使QT支持jpg格式的图片

cp /usr/local/Trolltech/Qt-4.7.0/plugins/imageformats/* /usr/lib/

同时在应　程序main程序中加入　app.addLibraryPath("/usr/lib/");

step 5:修改环境变　及LCD驱动　接

修改 /etc/profile 文件，增加以下声明：

export set QWS_MOUSE_PROTO="TSLIB:/dev/event0 Intellimouse:/dev/mouse0" #鼠键 均支持
export QWS_SIZE=320x240 #根据屏幕分辨率改成对应的值
export QWS_DISPLAY="LinuxFb:/dev/fb0:mmWidth35:mmHeight45:0" #设屏幕参数
export set QT_QWS_FONTDIR=$/usr/lib/fonts/ #字体库环境变

step 6:

拷 一个应 程序到根文件系 的/home目录下，

```
#cp QtEmbedded-4.7.0-arm/examples/animation/appchooser

#./ appchooser -qws
```

程序在显 屏上显 窗口。

三 ：boa web 务器在 PC 上移植

boa 是一个非常小巧的 Web 务器，可执 代 只有 60KB。它是一个单任务 Web 务器，只 依次完成 户的 求， 不会 fork 出新的 程来处理并发接 求。但 Boa 支持 CGI， 够为 CGI 程序 fork 出一个 程来执 。下 我们 解以下 boa 如何在 PC 机（虚拟机 RedHatAS5 环境）完成移植修改工作。

boa下 地址:http://www.rayfile.com/zh-cn/files/d7a7b368-c236-11e0-a55a-0015c55db73d/

一、配 编译boa

1. #tar zxvf boa-0.94.13.tar.gz

2. #mv boa-0.94.13 boa_pc

3. #cd boa_pc/src

4.boa.conf生效目录设

boa启动过程中　　读取一个配　文件:boa.conf,它的路径由以下文件决定

30　#define SERVER_ROOT "/etc/boa"，默认为/etc/boa 文件夹

我们为了　一路径　见，改为："/home/boa",接下来所有　boa 有关的文件我们都放在/home/boa 下

5.修改src/boa.c

注释掉下　语句:

```
if (setuid(0) !=-1)
{
   DIE(" icky Linux kernel bug!");
}
```

即修改为:

```
#if 0
 if(setuid(0) != -1)
{
    DIE(" icky Linux kernel bug!");
 }
#endif
```

否则运 **boa时会提**

boa.c:226 - icky Linux kernel bug!: No suchfile or directory错误

6. #./configure

7. make

在当前目录下生成一个boa的可执 程序，./boa运 即可，ps 看到boa的程即说明boa正常工作.

注：有些编译器会提 以下错误，按相应方 解决即可.

util.c: 100: 1: pasting “t” and “->” does not give a valid preprocessing token make: [util.o] Error1

解决方 :

方 >. 修改compat.h中的

#define TIMEZONE_OFFSET(foo) foo##->tm_gmtoff

为:

#define TIMEZONE_OFFSET(foo)foo->tm_gmtoff

二> 修改配 文件boa.conf

1.建立/home/boa目录，并复制boa.conf到/home/boa目录下，并按照以下修改。

#mkdir /home/boa

#cp boa.conf /home/boa

2.访问端口号设

大概 25 左右：Port 80，可以设定我们访问网　时的端口号默认为 80—访问时无　指定

假如改变了此端口号为 8080，必须以下列格式访问：http://192.168.1.3:8080

一　在同一个电　上运　多个boa　务器时可采　此方

我们这　保持默认即可

3. 修改访问权　：

修改User nobody 为 user 0

修改Group nogroup 为 group 0

4. 设定日志目录：boa日志有两部分，Errorlog　和 AccessLog

默认为/var/log/boa/error_log和/var/log/boa/access_log两个文件。

所以我们　一修改为：

ErrorLog /home/boa/error_log

AccessLog /home/boa/access_log

(注意:这　/home/boa目录必须为可写，否则会出项错误:log.c:73 - Unable to dup2 the error log: Bad file r.)

5. 设　html文件目录:

默认为：DocumentRoot /var/www

我们　一修改为：

DocumentRoot /home/boa/www

6. 设　默认　　: DirectoryIndex index.html

这是访问网　时若不指定访问的网　名称，　务器默认返回的网　，我们不做修改

7. 设 cgi脚本目录： 将

ScriptAlias /cgi-bin/ /usr/lib/cgi-bin

修改为

ScriptAlias /cgi-bin/ /home/boa/cgi-bin

注：**boa提供cgi编程接口，使得网 具有交互 力，后 我们在做 解.**

8.测试

在/home/boa/www 中随意放一个测试网 （必须包含 index.html 文件）

我们这 随便建一个文件取名为 index.html， 写一个：hello boa...

在IE中敲入：http://192.168.1.101/ 即可看到刚才的helloo boa

明整个boa搭建成功.

四 ：boa web 务器在 S3C2440 开发板上移植

前 已 在 PC 机上完成了 boa 务器的移植工作，下 介 一下在 S3C2440 硬件平台上如何完成相应工作，两者大同小异， 上稍有不同,不同之处我色 注了出来

一、配 编译 boa

1. #tar zxvf boa-0.94.13.tar.gz

2. #mv boa-0.94.13 boa_arm

3. #cd boa_arm/src

4.boa.conf 生效目录设

boa 启动过程中　　读取一个配　文件:boa.conf,它的路径由以下文件决定

30　　#define SERVER_ROOT "/etc/boa"，默认为/etc/boa 文件夹

我们为了　一路径　见，改为："/usr/boa",接下来所有　boa 有关的文件我们都放在 /usr/boa 下

5.修改 src/boa.c

注释掉下　语句:

```
if (setuid(0) !=-1)
{
    DIE("icky Linux kernel bug!");
}
```

即修改为:

```
#if 0
 if(setuid(0) != -1)
{
    DIE("icky Linux kernel bug!");
}
#endif
```

否则运　boa 时会提

boa.c:226 - icky Linux kernel bug!: No suchfile or directory 错误

6. #./configure

7.修改交叉编译器

修改 Makefile 文件:

将:

CC = gcc

CPP = gcc -E

该为:

CC = arm-linux-gcc

CPP = arm-linux-gcc -E

8. make

在当前目录下生成一个 boa 的可执 程序，./boa 运 即可，ps 看到 boa 的 程即说明 boa 正常工作.

注：有些编译器会提 以下错误，按相应方 解决即可.

util.c: 100: 1: pasting “t” and “->” does not give a valid preprocessing token make: [util.o] Error1

解决方 :

方 >. 修改 compat.h 中的

#define TIMEZONE_OFFSET(foo) foo##->tm_gmtoff

为:

#define TIMEZONE_OFFSET(foo)foo->tm_gmtoff

二> 修改配　文件 boa.conf

1.建立/usr/boa 目录，并复制 boa.conf 到/usr/boa 目录下，并按照以下修改。

#mkdir /usr/boa

#cp boa.conf/usr/boa

2.访问端口号设

大概 25 左右：Port 80，可以设定我们访问网　时的端口号默认为 80—访问时无　指定

假如改变了此端口号为 8080，必须以下列格式访问: http://192.168.1.3:8080

一　在同一个电　上运　多个 boa　务器时可采　此方

我们这　保持默认即可

3. 修改访问权　：

修改 User nobody 为 user 0

修改 Group nogroup 为 group 0

4. 设定日志目录：boa 日志有两部分，Errorlog　和 AccessLog

默认为/var/log/boa/error_log 和/var/log/boa/access_log 两个文件。

所以我们　一修改为：

ErrorLog /usr/boa/error_log

AccessLog /usr/boa/access_log

(注意:这　/home/boa 目录必须为可写，否则会出项错误:log.c:73 - Unable to dup2 the error log: Bad file r.)

5. 设 html 文件目录:

默认为：DocumentRoot /var/www

我们 一修改为：

DocumentRoot /usr/boa/www

6. 设 默认 : DirectoryIndex index.html

这是访问网 时若不指定访问的网 名称， 务器默认返回的网 ，我们不做修改

7. 设 cgi 脚本目录: 将

ScriptAlias /cgi-bin/ /usr/lib/cgi-bin

修改为

ScriptAlias /cgi-bin/ /usr/boa/cgi-bin

注：**boa 提供 cgi 编程接口，使得网 具有交互 力，后 我们在做解.**

8. mime.types 存放目录, 可在 boa.conf 文件中修改。

如: **MimeTypes /usr/boa/mime.types**

或者将 boa.conf 文件中

MimeTypes /etc/mime.types

修改为:

MimeTypes /dev/null

9.把 boa.conf 大 94 的注释去掉

ServerName www.your.org.here

10.测试

在/usr/boa/www 中随意放一个测试网　（必须包含 index.html 文件）

我们这　随便建一个文件取名为 index.html，　　写一个：hello boa...

在 IE 中敲入：**http://192.168.1.101/** 即可看到刚才的 **helloo boa**

明整个 **boa** 搭建成功.

五　：boa 移植过程中出　的问题

1>. 错误 1: gethostbyname:: No such file or directory

解决办　：修改boa.conf　去掉 ServerName www.your.org.here 前的注释　号(#)

2>. 错误 1: util.c:100:1: error: pasting "t" and "->"does not give a valid preprocessing token make: *** [util.o]

解决办　：修改 src/compat.h

找到

#defineTIMEZONE_OFFSET(foo) foo##->tm_gmtoff

修改成

#defineTIMEZONE_OFFSET(foo) (foo)->tm_gmtoff

3>. 错误 2: boa.c:211 - getpwuid: No such file or directory

解决办　：修改src/boa.c

注释掉下　这段程序:

```
if (passwdbuf == NULL) {
DIE(" getpwuid");
}
if(initgroups(passwdbuf->pw_name, passwdbuf->pw_gid) == -1) {
DIE(" initgroups");
}
```

即修改为:

```
#if 0
if (passwdbuf == NULL) {
DIE(" getpwuid");
}
if(initgroups(passwdbuf->pw_name, passwdbuf->pw_gid) == -1) {
DIE(" initgroups");
}
#endif
```

4>. 错误 3: boa.c:228 - icky Linux kernel bug!: No such file or directory

解决办　: 修改src/boa.c

注释掉下　语句:

```
if(setuid(0) != -1) {
DIE(” icky Linux kernel bug!” );
}
```

即修改为:

```
#if 0
 if(setuid(0) != -1) {
 DIE(” icky Linux kernel bug!” );
 }
#endif
```

5>. 错误 4: log.c:73 unable to dup2 the error log:bad file descriptor

解决方 :

方 1> 确定日志目录对与所有 户都具有可读/写的权

方 2> 修改src/log.c (建议采 方 1)

注释掉

```
if(dup2(error_log, STDERR_FILENO) == -1) {
DIE("unableto dup2 the error log");
}
```

即修改为:

```
#if 0
```

```
    if (dup2(error_log, STDERR_FILENO) == -1) {
    DIE("unable to dup2 the error log");
     }
   #endif
```

六 ：sqlite-3.7.6.2 在 PC 及 S3C2440 平台移植

SQLite，是一款轻型的数据库，是 守ACID的关联式数据库管理系 ，它的设 目 是嵌入式的， 且目前已 在很多嵌入式产品中使 了它，它占 资源非常的低，在嵌入式设备中，可 只 几百K的内存就够了。它 够支持 Windows/Linux/Unix等等主流的操作系 ，同时 够 很多程序语 相 合，比如 Tcl、C#、PHP、Java等， 有ODBC接口，同 比 Mysql、PostgreSQL这两款开源世 名的数据库管理系 来 ，它的处理 度比他们都快。SQLite 一个Alpha版本 生于 2000 年 5 月. 至今已 有 10 个年头，SQLite也迎来了一个版本 SQLite 3 已 发布

本文 讨论sqlite在PC机(RedHatAS5 环境)及S3C2440 平台的移植工作

下 地址：
http://www.rayfile.com/zh-cn/files/5c8d420a-c25c-11e0-b32c-0015c55db73d/

1.准备文件夹

```
    #mkdir /home/user_name/sqlite_pc
```

#mkdir /home/user_name/sqlite_arm

2.//编译PC版的sqlite

#tar zxvf sqlite-amalgamation-3.7.6.2.tar.gz

#cd sqlite3.7.6.2

#./configure －prefix=/home/user_name/sqlite/sqlite_pc

#make

#make install

3.//编译ARM版的sqlite

 # tar zxvf sqlite-amalgamation-3.7.6.2.tar.gz

 # cd sqlite-3.7.6.2

 # ./configure - prefix=/home/user_name/sqlite/sqlite_arm --host=arm-linux

#make

#make install

注意：

make时会出错：

“arm-none-linux-gnueabi-gcc: 3.7.6.2": No such file or directory”

Makefile 127 的“ 3.7.6.2”空格导致，去掉’3’之前的空格，编译通过。

七 ：ffmpeg 频编 库 S3C2440 移植

FFmpeg 是一个开源免 平台的 频和音频流方案，属于自由 件，采 LGPL 或 GPL 可 （依据你 择的 件）。它提供了录制、转换以及流化音频的完整解决方案。它包含了非常先 的音频/ 频编解 库 libavcodec，为了保 高可移植性和编解 质 ，libavcodec 很多 codec 都是从头开发的。

官方网站：

http://www.ffmpeg.org/index.html

下 地址：

http://www.rayfile.com/zh-cn/files/0a236159-c26d-11e0-84b7-0015c55db73d/

一、配　编译

```
#./configure --cpu=armv4t --cross-prefix=arm-linux- --cc=arm-linux-gcc
--host-cc=gcc --enable-cross-compile --enable-static
--disable-optimizations --disable-mmx --disable-iwmmxt
--disable-altivec --disable-mmx2 --disable-armv5te --disable-armv6
--disable-armv6t2  --arch=arm --target-path=output
--target-os=linux  --prefix=./_install --enable-ffplay
#make
#make install
```

二、使　：

使　：./ffmpeg　-f video4linux2 -s 320*240 -r 10 -i
/dev/video1　test.asf

注意：-i 后　指定摄像头的设备文件，摄像头必须　循 v4l2 协议

八　：mplayer 在 PC、ARM 上移植

MPlayer是一款开源多媒体播放器，以GNU通　公共　可　发布。此款　件可在各主流作业系　使　，例如Linux和其他类Unix系　、微　的　窗系　及　果电的Mac OS X系　。MPlayer是建基于命令　　，在各作业系　可　择安装不同的图形　　。mplayer的另一个大的特色是广泛的输出设备支持。它可以在X11，Xv，DGA，　OpenGL，SVGAlib，fbdev，AAlib，DirectFB下工作，　且你也　使　GGI和SDL和一些低　的硬件相关的驱动模式(比如Matrox，3Dfx和Radeon，

Mach64，Permedia3)。MPlayer 支持通过硬件MPEG解 卡显 ，诸如DVB 和DXR3与Hollywood+。

下 地址：

http://www.mplayerhq.hu/design7/dload.html

或：

一、编译PC版的mplayer

#tar jxvf MPlayer-1.0rc4.tar.bz2

#mv MPlayer-1.0rc4 MPlayer_pc

#cd MPlayer-1.0rc4

#./configure

#make

#make install

Mplayer会被默认安装到/usr/local/bin ，在任意目录下敲mplayer即可启动mplayer

使 ./mplayer file.name 即可播放指定 频或音频

二、编译ARM版的mplayer

#tar zxvf MPlayer-1.0rc4.tar.gz

#mv MPlayer_arm

#cd MPlayer_arm

#./configure --host-cc=gcc --cc=arm-linux-gcc
--target=armv4l--enable-static --disable-win32dll --disable-dvdread
--disable-dvdread-internal --disable-dvdnav --disable-libdvdcss-internal
--enable-fbdev --disable-mencoder --disable-live --disable-mp3lib --enable-mad
--enable-libavcodec_a --language=zh_CN

#make //会在当前文件下生成mplayer可执 程序

其中make会出错， 更改mplayer_src/libmpeg2/motion_comp_arm_s.S中的代
，

#vim mplayer_src/libmpeg2/motion_comp_arm_s.S

在最开始的地方添加：

#ifndef HAVE_PLD

.macro pld reg

.endm

#endif

保存之后，再次make，就OK了

备注：主 配 项说明

a、--host-cc=gcc

//是 来编译一些 在host上执 的中间文件的，如codec-cfg，出 “codec-cfg
无 运 ”等错误均是因为少了这句 的原因

b、--cc=arm-linux-gcc //指定交叉编译器

c、--target=arm-armv4l //指定编译平台

e、--enable-static　　//设定　态　接，省去考很多库的麻　，建议添加

f、--prefix=/tmp/mplayer　　//指定编译出的可执　文件放到什么地方，默认放到源文件所在文件夹下

g、 --disable-mp3lib --enable-mad

　//禁　mplayer自带的音频解　库，使　mad解　库，自带的解　库CPU占　率较高的问题，实际　验　，不加此　项，编译可以通过，但是　有什么声音输出

h、--language=zh_CN　　//编译出的Mplayer版本 帮助、提　信息均为中文

九　：mplayer 字　　操作方

一、启动播放时参数：

在终端下敲入：./mplayer　提　各种使　帮助信息

```
-vo <drv[:dev]>   择　频输出模式和设备(　'-vo help'查看列表)
-ao <drv[:dev]>   择音频输出模式和设备(　'-ao help'查看列表)
-ss <timepos>   寻找指定的(多少秒或 hh:mm:ss)位
-nosound        不播放声音
-fs -vm -zoom   全屏播放　项(fullscr,vidmode chg,softw.scale)
-x <x> -y <y>   设　播放的分辨率(　于改变 vidmode 或　件缩放)
-sub <file>     指定使　的字幕文件(参见-subfps, -subdelay)
-playlist <file> 指定使　播放列表文件
-vid x -aid y     择　于播放的　频(x)和音频(y)流
-fps x -srate y 改变　频(x fps)和音频(y Hz)率
-pp <quality>   使　后期处理滤镜(详　内容参见 manpage/docs)
-framedrop      使　 frame-dropping (　于慢机器)
```

二、播放时控制键

基本控制键：（完整的列表参见 manpage，同时也　　查一下 input.conf）

<- or ->	向后/向前搜索 10 秒
up or down	向后/向前搜索 1 分钟
pgup or pgdown	向后/向前搜索 10 分钟
< or >	跳到播放列表中的前一　/下一
p or SPACE	暂停播放(按任意键继续)
q or ESC	停止播放并推出
+ or -	调整音频延　+/-0.1 秒
o	循环 OSD 模式：　none/seekbar/seekbar+timer
* or /	增加或减少 pcm 音
z or x	调整字幕延　+/-0.1 秒
r or t	上/下调整字幕位　，参见-vf expand!

三、mplayer 的使　实例：

#mplayer filename(PC 版)

#./mplayer -ac mad filename（ARM 版）

//即可播放音频，　测试过可播放 .mp3　.wmv .mgp .mp4 等格式的音　频

#./mplayer -ss 10

//从　10s 开始播放

#./mplayer　-ss　00:01:00

//从　1 分钟开始播放

#./mplayer -fs -zoom -x 160 -y　120

//以 160*120 大小居中播放　频

//-fs 配合-zoom 居中播放， -zoom -x 宽度 -y 高度 指定播放窗口的宽度与高度

#mplayer filename -caceh 8192

//-caceh 8192 设 缓存为 8M

#./mplayer -loop 5 file

//-loop 5 让这个文件循环播放 5 遍，如果为 0 就表 不停的播放。

#ls /video/tom/*.avi > tom.lst

//制作一个文件播放列表

#mplayer -playerlist tom.lst -shuffle

//-playerlist 指定播放列表、-shuffle 指定随机播放

#./mplayer -ac mad -vop rotate=1 matrix.mpg 旋转 90 度

十 ：mplayer 从模式说明

slave模式协议

一、 介：

默认**mplayer**是从键 上 得控制信息

mplayer另外提供了一种更为灵 的控制方式， 来 播放控制——**slave**模式

在**slave**模式下，**MPlayer**为后台运 其他程序，不再截 键 事件，

MPlayer会从 准输入读一个换 （\n）分隔开的命令。

二、操作：

#mplayer -input cmdlist

//会打印出一份当前mplayer所支持的所有slave模式的命令

方　一：从控制台输入控制命令(测试使　)

　运　mplayer -slave -quiet <movie>，并在控制台窗口输入slave命令。

//-slave 启动从模式

//-quiet 不输出冗余的信息

常　到的 Mplayer指令：

loadfile　string　　//参数string 为 歌　名字。

volume 100 1//设　音　　中间的为音　的大小。

mute1/0//　音开关

pause//暂停/取　暂停

get_time_length//返回值是播放文件的　度，以秒为单位。

seek value　//向前查找到文件的位　播放 参数value为秒数。

get_percent_pos//返回文件的百分比（0--100）

get_time_pos//打印出在文件的当前位　　秒表　，采　浮点数

volume <value> [abs] //增大/减小音　，或将其设　为<value>，如果[abs]不为零

get_file_name//打印出当前文件名

get_meta_album//打印出当前文件的'专辑'的元数据

get_meta_artist//打印出当前文件的'艺术家'的元数据

get_meta_comment//打印出当前文件的'评论'的元数据

get_meta_genre//打印出当前文件的'流派'的元数据

get_meta_title//打印出当前文件的'　题'的元数据

get_meta_year//打印出当前文件的'年份'的元数据

方　二：从有名管道(fifo)输入控制命令（应　编程中使　）

#mkfifo </tmp/fifofile>

#mplayer -slave -input file=</tmp/fifofile> <movie>

// 户可以通过往管道 写入slave命令来实 对应的功

例：主 程创建一个无名管道和一个有名管道

1:开一个子 程

在子 程中：

启动Mplayer，参数 定通过命名管道 通信；

把子 程的 准输出重定向无名管道的写端；

Mplayer从命名管道读到主 程发送的命令;

Mplayer发出的内容发送到无名管道中，父 程通过读管道就可以读到Mplayer发出的信息。

2：在父 程中：

启动两个线程

一个线程，不断使 fgets从键 取一个字 串命令，并写入命名管道中

二个线程，循环 测无名管道是否有信息可读，有信息将其打印输出在屏幕上

[cpp] view plaincopy

```cpp
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <string.h>

/**********************全局变 定义区*****************/
int fd_fifo;                    //创建有名管道， 于向mplayer发送命令
int fd_pipe[2];                 //创建无名管道， 于从mplayer读取命令

void *get_pthread(void *arg)
{
    char buf[100];
    while(1)
    {
        printf("please input you cmd:");
```

```
19.         fflush(stdout);
20.         fgets(buf,sizeof(buf),stdin);        //从  准输入  取数据
21.         buf[strlen(buf)]='\0';
22.         printf("*%s*\n",buf);
23.         if(write(fd_fifo,buf,strlen(buf))!=strlen(buf))
24.             perror("write");                    //将命令写入命名管道
25.     }
26. }
27.
28. void *print_pthread(void *arg)
29. {
30.     char buf[100];
31.     close(fd_pipe[1]);
32.     int size=0;
33.     while(1)
34.     {
35.         size=read(fd_pipe[0],buf,sizeof(buf));  //从无名管道的写端读取信息
    打印在屏幕上
36.         buf[size]='\0';
37.         printf("th msg read form pipe is %s\n",buf);
38.     }
39. }
40.
41. int main(int argc, char *argv[])
42. {
43.     int fd;
44.     char buf[100];
45.     pid_t pid;
46.
47.     unlink("/tmp/my_fifo");                 //如果明明管道存在，则先删除
48.     mkfifo("/tmp/my_fifo",O_CREAT|0666);
49.     perror("mkfifo");
50.
51.     if (pipe(fd_pipe)<0 )                    //创建无名管道
52.     {
53.         perror("pipe error\n");
54.         exit(-1);
55.     }
56.
57.     pid=fork();
58.     if(pid<0)
59.     {
60.         perror("fork");
61.     }
```

```
62.     if(pid==0)                                  //子  程播放mplayer
63.     {
64.         close(fd_pipe[0]);
65.         dup2(fd_pipe[1],1);                     //将子  程的  准输出重定向到管道的写端
66.         fd_fifo=open("/tmp/my_fifo",O_RDWR);
67.         execlp("mplayer","mplayer","-slave","-quiet","-input","file=/tmp/my_fifo","juhuatai.mpg",NULL);
68.     }
69.     else
70.     {
71.         pthread_t tid1;
72.         pthread_t tid2;
73.         fd_fifo=open("/tmp/my_fifo",O_RDWR);
74.         if(fd<0)
75.             perror("open");
76.
77.         pthread_create(&tid1,NULL,get_pthread,NULL);          //从键取控制信息
78.         pthread_create(&tid2,NULL,print_pthread,NULL);        //打印从无名管道收到的信息
79.         pthread_join(tid1,NULL);
80.         pthread_join(tid2,NULL);
81.     }
82.     return 0;
83. }
```

获

十一 ：mplayer 从模式翻译文档

mplayer 源 中有一个关于 mplay slave 模式的说明文档：/DOCS/tech/slave.txt，下 是中文翻译

slave 模式协议

在关于 slave 模式，MPlayer 为后台运 其他程序，不再截 键 事件，MPlayer 会从 准输入读一个换 （\n）分隔开的命令。

动手尝试 slave 模式，运　　mplayer -slave -quiet <movie>，并在控制台窗口输入 slave 命令。

您也可以使　一个 fifo 文件（命名管道）：

```
mkfifo </tmp/fifofile>
mplayer -slave -input file=</tmp/fifofile> <movie>
```

大多数 slave 模式命令相当于命令　　项，但并非一定　在相同的名称。详　说明中可以在手册中找到

所有命令都可以以前　“pausing”，“pausing_keep”，或“pausing_toggle”为前　。

“pausing”告　MPlayer 辆停尽快正在处理的命令。

“pausing_keep”告　MPlayer 辆定保持只　它已在辆停模式。

“pausing_toggle”告　MPlayer 辆定保持只　它尚未辆停模式。

注意，“尽可　“可以在命令完全执　之前。

各种提　和技巧（帮助扩展！）：

- 尝试使　例如

pausing_keep_force pt_step 1

get_property pause

切换到下一个文件，它　免在转换到新的音频文件之前旧文件播放一小段时间

可　的命令（'mplayer -input cmdlist 的'会打印出一份清单）：

alt_src_step <value> (ASX playlist only)

当有一个以上的源可以有　择下一个/前一个。

audio_delay <value> [abs]
设　/调整音频延　。
如果[abs]不提供或为零，调整　延<value>秒。
如果[abs]不为零，将延　到<value>秒。

[brightness|contrast|gamma|hue|saturation] <value> [abs]
设　/调整　频参数。
如果[abs]不提供或为零，修改参数为<value>。
如果[abs]不为零，参数设　为<value>。
<value>的　围是[-100，100]。

change_rectangle <val1> <val2>
更改　形滤波器　形的坐　。
<val1>
必须是下列之一：
0 =宽度
1 =高度
2 = x 坐
3 = y 坐
<val2>
如果<val1>为 0 或 1：
整数加/减去宽/高。
正值宽度/高度和负值减去它。
如果<val1>是 2 或 3：
相对　形左上角的整数值。正值移动　形向右/向下和负值移动形向左/向上。

dvb_set_channel <channel_number> <card_number>
设　的 DVB 通道。

dvdnav <button_name>
给定 dvdnav 按钮。
up
down
left
right
menu
select
prev
mouse

edl_mark

将当前位　写入 EDL 文件。

frame_drop [value]
切换/设　帧的模式。

get_audio_bitrate
打印出当前文件音频比特率。

get_audio_codec
打印出的音频当前文件的编解　器的名称。

get_audio_samples
打印出的音频和当前文件的声道数。

get_file_name
打印出当前文件名。

get_meta_album
打印出当前文件的'专辑'的元数据。

get_meta_artist
打印出当前文件的'艺术家'的元数据。

get_meta_comment
打印出当前文件的'评论'的元数据。

get_meta_genre
打印出当前文件的'流派'的元数据。

get_meta_title
打印出当前文件的'　题'的元数据。

get_meta_track
打印出当前文件的'音轨的数　'的元数据。

get_meta_year
打印出当前文件的'年份'的元数据。

get_percent_pos
打印出文件中的当前位　为整数百分比[0-100）。

get_property <property>
打印出的属性的当前值。

get_sub_visibility
打印出字幕 见度（1 ==开启，0 ==关闭）。

get_time_length
打印出当前文件的 度 秒表 。

get_time_pos
打印出在文件的当前位 秒表 ，采 浮点数。

get_vo_fullscreen
全屏状态打印出来（1 == 全屏，0 ==窗口）。

get_video_bitrate
打印出当前文件的 频比特率。

get_video_codec
打印出当前 频文件的编解 器的名称。

get_video_resolution
打印出当前文件的 频分辨率。

screenshot <value>
截屏。 求屏幕过滤器加 。
0 以一个单 的截图。
1 启动/停止 ，每帧画 。

gui_[about|loadfile|loadsubtitle|play|playlist|preferences|skinbrowser|stop]
图形 户 动

key_down_event <value>
注入<value>到 MPlayer 的关键代 的事件。

loadfile <file|url> <append>
加 给定的文件/网址，停止当前文件的播放/网址。
如果是<append>非零继续播放和文件/网址
追加到当前播放列表代替。

loadlist <file> <append>
加 给定的播放列表文件，停止当前文件的播放。
如果<append>是非零和继续播放文件,文件追加到当前播放列表。

loop <value> [abs]
调整/设 怎 的电影应该是循环多次。 -1 代表不循环，永远的 0。

单命令>
执 上显 OSD 单命令。
up 移动光 向上。
down 移动光 向下。
ok 接受的 择。
cancel 取 择。
hide 隐藏的 OSD 单。

set_menu <menu_name>
显 单命名<menu_name>。

mute [value]
切换声音输出 音或将其设 为[value]（value>=0）
（1 ==开启，0 ==关闭）。

osd [level]
切换 OSD 模式或将其设 为[level]在[level]>= 0。

osd_show_property_text <string> [duration] [level]
显 一项关于 OSD 扩展属性的字 串，看到-playing-msg 于描述可 的扩展。如果[duration]>=0,显 为[duration]ms。 [level]设 所 的最低水平 OSD 该 息可见（默认是：0 -始终显 ）。

osd_show_text <string> [duration] [level]
查看 OSD 的<string>。

panscan <-1.0 - 1.0> | <0.0 - 1.0> <abs>
增加或减少 pan-and-scan 的<value>的 围，1.0 是最高的。
负值降低 pan-and-scan 围。
如果<abs>！= 0， 么 pan-and-scan 围被解释为 对的 围。

pause
㆓停/取 ㆓停播放。

frame_step
播放一帧，然后㆓停。

pt_step <value> [force]
转到下一个/上的播放树项。 志的<value> 述
该方向。如果 有项目可在给定的方向不会做任何事，除非[force]不为零。

pt_up_step <value>[部队]
类似 pt_step，但跳转到下一个/父列表中的前一个项目。

有助于摆脱在播放树 内部循环。

quit [value]

退出 MPlayer。可　的整数[value]的值作为返回代
为 mplayer 的　程（默认值：0）。

radio_set_channel <channel>

切换到<channel>。在‘channel’的广播参数　　设　。

radio_set_freq <frequency in MHz>

设　广播频率调谐器。

radio_step_channel <-1|1>

向前（1）或向后（-1 频道列表）。只有当'channel'的广播参数设　。

radio_step_freq <value>

调整频率的<value>（正数 - 向上，负数 - 向下）。

seek <value> [type]

定位电影的某些地方。

0 是一个相对定位+/- <value>（默认值）。
1 是定位<value>％在电影　。
2 是寻求一个　对位　的<value>秒。

seek_chapter <value> [type]

定位一　的开始。

0 是一个相对寻求+/- <value>　　（默认）。
1 定位到<value>　。

switch_angle <value>

转换 ID 为角度[value]。通过循环如果　角度[value]省略或负数。

set_mouse_pos 的<X> <y>

告　MPlayer 的窗口中鼠　坐　。
此命令不移动鼠　！

set_property <property> <value>

设　属性。

speed_incr <value>

增加<value>当前回放　度。

speed_mult <value>

目前　度乘以<value>。

speed_set <value>
设定 度为**<value>**。

step_property <property> [value] [direction]
通过 **value** 来改变属性，或者，如果 给定或为 **0** 则增加默认值。如果小于零，方向是相反的方向。

stop
停止播放。

sub_alignment [value]
切换/设 对齐字幕。
0 顶部对齐
1 居中对齐
2 底部对齐

sub_delay <value> [abs]
调整了字幕延 **+/- <value>**秒或将其设 **<value>**
秒时**[abs]**不为零。

sub_load <subtitle_file>
从**<subtitle_file>**加 字幕。

sub_log
当前日志上显 的字幕或 同文件名和时间信息的一 **/.mplayer/subtitle_log**。

sub_pos <value> [abs]
调整/设 字幕的位 。

sub_remove [value]
如果**[value]**参数是当前和非负，并取 了字幕文件的**[value]**索引。如果参数省略或负，除去
所有的字幕文件。

sub_select [value]
显 字幕的索引**[value]**。关闭字幕显 ，如果关闭**[value]**的值为**-1** 或比更高可 的字幕指数更大。
可 的字幕周期，如果**[value]**省略或低于**-1**。支持字幕来源是 **-sub** 项在命令 ，**VOBsubs**，**DVD** 字幕和 **Ogg** 和 **Matroska** 文本流。
这主 是循环所有字幕命令，如果 设 一个特定的字幕，使 **sub_file**，**sub_vob**，或 **sub_demux**。

sub_source [source]
显　一个字幕，从[source]。这　[source]是一个整数：
SUB_SOURCE_SUBS（0）　于文件字幕
SUB_SOURCE_VOBSUB（1）　VOBSub 文件
SUB_SOURCE_DEMUX（2）在媒体文件或 DVD 嵌入字幕。
如果[source]为-1，将关闭字幕显　。如果[source]低于-1，将循环每个之间的　有资源　一个字幕。

sub_file [value]
显　字幕 specifid 由[value]的文件 subs。在[value]的值
通过相应的 ID_FILE_SUB_ID'-identify'报告的值。
如果[value]的值-1，将关闭字幕显　。如果[value]小于-1，
将循环的所有文件 subs。

sub_vob [value]
显　字幕 specifid 由[value]的 vobsubs。在[value]的值
通过相应的 ID_VOBSUB_ID'-identify'报告的值。
如果[value]的值-1，将关闭字幕显　。如果[value]小于-1，
将循环的所有 vobsubs。

sub_demux [value]
显　字幕 specifid 由[value]从 DVD 字幕或嵌入在媒体文件。在[value]的值对应 ID_SUBTITLE_ID 值'-identify'，。如果[value]的值-1，将关闭字幕显　。
如果[value]小于-1，将循环所有的 DVD 字幕或嵌入字幕。

sub_scale <value> [abs]
调整字幕大小+/- <value>或将其设　为<value>时，[abs]
不为零。

vobsub_lang
这是与 sub_select 为了向后兼容。

sub_step<value>
在字幕列表前　<value>　，如果<value>
是为负，倒退<value>　。

sub_visibility [value]
切换/设　字幕。

forced_subs_only [value]
强制切换/设　字幕。

switch_audio [value]（目前的 MPEG*，AVI，的 Matroska 和 libav 库处理流）
切换到音频文件通过 ID[value]。循环

歌 ，如果[value]省略或负数。

switch_angle [value]（DVD 光 只）
切换到 DVD 的角度通过 ID[value]。循环
如果可 角度，如果[value]省略或负数。

switch_ratio [value]
在运 时改变 宽比。 [value]是表 新的 宽比
作为浮动 16 / 9（例如 1.77778）。
这可 与某些 频过滤器的问题。

switch_title [value]（DVDNAV only）
切换到 DVD 题通过 ID[value]。循环可 题，如果[value]的值省略或负数。

switch_vsync [value]
切换场同 （1 ==开启，0 ==关闭）。如果[value]的值 有提供，刷新同状态反转。

teletext_add_digit <value>
入/离开字幕的 号编辑模式，并追加提供的以前输入的数字。
0 .. 9 - 附加 apropriate 数字。 （启 编辑模式，如果从一 求模式，并切换到正常模式时。）
- - 删除最后的 数字。 （退格仿真，只 在 编辑模式。）

teletext_go_link <1-6>
按照目前的字幕的 给出链接。

tv_start_scan
电 频道开始自动扫描。

tv_step_channel <channel>
择下一个/上一个电 频道。

tv_step_norm
更改电 制式。

tv_step_chanlist
改变频道列表。

tv_set_channel <channel>
设 当前的电 频道。

tv_last_channel

设　当前电　频道到最后一个。

tv_set_freq <frequency in MHz>
设　电　调谐器的频率。

tv_step_freq <frequency offset in MHz>
设　电　调谐器的频率相对于当前值。

tv_set_norm <norm>
电　调谐器设　　（包括 PAL，SECAM，NTSC 制式，...).

tv_set_brightness <-100 - 100> [abs]
设　电　调谐器的亮度或调整，如[abs]设　为 0。

tv_set_contrast <-100 -100> [abs]
设　电　调谐器的对比或调整，如[abs]设　为 0。

tv_set_hue <-100 - 100> [abs]
设　电　调谐器色调或调整，如[abs]设　为 0。

tv_set_saturation <-100 - 100> [abs]
设　电　调谐器　和或调整，如[abs]设　为 0。

use_master
主之间切换和 PCM 音　控制。

vo_border [value]
切换/设　　显　。

vo_fullscreen [value]
切换/设　全屏模式。

vo_ontop [value]
切换/设　保持在最上层。

vo_rootwin [value]
切换/设　在根窗口播放。

volume <value> [abs]
增大/减小音　，或将其设　为<value>，如果[abs]不为零。

下　的命令，实际上只可　于 OSD　单控制台模式：

help
帮助文本显 ，目前 是空的。

exit
从 OSD 单退出控制台。不像'quit'，不退出 MPlayer 的。

hide
隐藏了 OSD 单控制台。点击 单命令 unhides 它。其他按键绑定的 为一切如常。

run <value>
运 <value>的 shell 命令。在 OSD 单控制台模式 准输出和 准输入是通过 频输出。

十二 ：使 busybox 制作 Linux 根文件系

建最 单的嵌入式 Linux 根文件系 ，下 我们一 教大家做 。。

硬件环境：

S3C2440

件环境：

Busybox-1.16.1 http://www.rayfile.com/zh-cn/files/2e477e14-c1bb-11e0-b287-0015c55db73d/
cross-4.3.2 http://www.rayfile.com/zh-cn/files/99e727f3-c1a5-11e0-822a-0015c55db73d/

STEP 1： 建目录

创建根文件系 目录，主 包括以下目录
/dev /etc /lib /usr /var /proc /tmp /home /root /mnt
/bin /sbin /sys

```
#mkdir     /home/rootfs
#cd        /home/rootfs

#mkdir  dev  etc  lib  usr  var  proc  tmp  home  root  mnt   sys
```

STEP 2: 使 busybox 建/bin /sbin linuxrc

入 busybox-1.16.1 目录，执

```
#make defconfig

#make menuconfig

Busybox Setting ----->
    Build Options ----->
        //1  择将busybox      态编译
        [*]Build BusyBox as a static binary (no shared libs)
        //2.指定交叉编译器为
        (/usr/local/arm/4.3.2/bin/arm-linux-)Cross Compiler prefix

Installation Options ----->

        //3.  择上 Don’t use /usr
Busybox Library Tuning--->
    [*]Username completion
    [*]Fancy shell prompts
    [*]Query  cursor  position  from  terminal
        //4.编译出的busybox的shell命令解释器支持显  当前路径及主机信息
```

保存退出

```
#make
#make install
```

在 busybox 目录下会看见 _install 目录， 有/bin /sbin linuxrc 三个文件将这三个目录或文件拷到 一 所建的 rootfs 文件夹下。

#cp bin/ sbin/ linuxrc /home/rootfs -ra

切 一定 带上-a的参数，因为bin目录 大部分都是链接，如果不带-a的参数，拷过去之后会做相应的复制，不再是链接的形式

STEP 3 建etc目录：

1) 入根文件系 rootfs的etc目录，执 如下操作：

拷 Busybox-1.16.1/examples/bootfloopy/etc/* 到当前目录下

#cp -r busybox-1.16.1/examples/bootfloopy/etc/* rootfs/etc

修改 inittab，把 二项改为::respawn:-/bin/login

删除 三、 四 代

2)拷 虚拟机上的/etc/passwd, /etc/group, /etc/shadow到rootfs/etc下

cp /etc/passwd rootfs/etc
cp /etc/group rootfs/etc
cp /etc/shadow roofs/etc

对以下三个文件修改，只保存与root相关的项，根据具体情况内容会有所不同。

修改passwd为root:x:0:0:root:/root:/bin/sh，即只保存与root相关项， 且最后改成/bin/ash。

修改group为root:x:0:root

修改shadow为
root:1x9yv1WlB$abJ2v9jOlOc9xW/yOQwPs.:14034:0:99999:7:::

开发板时 输入 户名密 ，同虚拟机相同

3)修改profile

```
PATH=/bin:/sbin:/usr/bin:/usr/sbin          //可执  程序 环境变
export LD_LIBRARY_PATH=/lib:/usr/lib        //动态链接库 环境变

/bin/hostname sunplusedu

USER="`id -un`"

LOGNAME=$USER

HOSTNAME='/bin/hostname'

PS1='[\u@\h \W]# '                          //显  主机名、当前路径等
```
信息:

4)修改 etc/init.d/rc.S文件

```
/bin/mount -n -t ramfs ramfs /var

/bin/mount -n -t ramfs ramfs /tmp

/bin/mount -n -t sysfs none /sys

/bin/mount -n -t ramfs none /dev

/bin/mkdir /var/tmp

/bin/mkdir /var/modules

/bin/mkdir /var/run

/bin/mkdir /var/log

/bin/mkdir -p /dev/pts                      //telnet  务

/bin/mkdir -p /dev/shm                      //telnet  务

echo /sbin/mdev > /proc/sys/kernel/hotplug//USB自动挂
```

/sbin/mdev -s　　//启动mdev在/dev下自动创建设备文件　点

/bin/mount -a

5)修改etc/fstab文件，增加以下文件

```
none    /dev/pts     devpts    mode=0622       0 0
tmpfs   /dev/shm     tmpfs     defaults        0 0
```

STEP 4　建lib目录：

1)#cd /usr/local/arm/4.3.2/arm-none-linux-gnueabi/libc/armv4t/lib

将以下动态库拷　到rootfs/lib下

#cp　*so*　roofs/lib　-a

2)#cd /usr/local/arm/4.3.2/arm-none-linux-gnueabi/libc/armv4t/usr/lib

将以下动态库拷　到rootfs/lib下

#cp ./libstdc++.so.*　rootfs/lib　-a