

## 嵌入式 Linux 常见开源工程的移植

第一节： tslib1.0移植

第二节： QT4.7.0 在ARM平台移植

第三节： boa web服务器在PC上移植

第四节： boa web服务器在S3C2440开发板上移植

第五节： boa移植过程中出现的问题

第六节： sqlite-3.7.6.2在PC及S3C2440平台移植

第七节： ffmpeg视频编码库S3C2440移植

第八节： mplayer在PC、ARM上移植

第九节： mplayer字符界面操作方法

第十节： mplayer从模式说明

第十一节： mplayer从模式翻译文档

第十二节： 使用 busybox 制作 Linux 根文件系统

第一节： tslib1.0移植

移植 QT 需要 tslib 触摸屏校正程序的支持，下面我们进行 tslib1.0 版本的编译、移植讲解

下载地

址: <http://www.rayfile.com/zh-cn/files/a01f838a-clbb-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置脚本

```
#CC=arm-linux-gcc //选择交叉编译器
```

```
#!/configure--host=arm-linux
```

```
--prefix=$PWD/../../tslib1.0_target --enable-inputapi=no
```

```
//在当前路径的前一级目录建立目标文件夹
```

```
#exportPLUGIN_DIR=$PWD/plugins
```

### 3. 编译

在tslib-1.0\config.h中有一行定义“#define malloc rpl\_malloc”，直接注释

```
#make
```

```
#make install
```

则会在与当前目录平级的文件夹下建立tslib\_target文件夹，里面包含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贝到tslib目录下

```
#cd /usr/
```

```
#mkdir /usr/tslib
```

```
#cp tslib_target/* /usr/tslib -r
```

2) 修改/usr/tslib/etc/ts.conf将第一句话的屏蔽去掉

```
#module_raw input
```

改为

```
module_raw input (注意前面不要有空格)
```

3) 修改开发板 /etc目录下/profile 文件，添加以下启动脚本并运行

```
#vi /etc/profile
```

```
export T_ROOT=/usr/tslib #目录
```

```
export PATH=$PATH:$T_ROOT/bin #可执行程序目  
录加入环境变量
```

```
export LD_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 #指定触摸屏校准  
文件pintercal的存放位置
```

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

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

```
echo "ts init success!"
```

#### 4、使用与测试

##### 1) ts\_calibrate

执行后，屏幕上出现 5 个点，依次点击进行校准，校准完毕后，在/etc 下会产生 pointercal 文件，供其它程序使用，ts\_calibrate 是一个应用程序，在屏幕上画几个按钮，将用户点击后从 ts 驱动获得的数据和屏上的坐标位置通过一套算法来获得校准数据写到一个校准文件里

##### 2) ts\_print            将屏幕坐标打印出来

横坐标 纵坐标

```
1796351007.320859: 156 116 500 按下
```

```
1796351007.355145: 156 116 0 抬起
```

##### 3) ts\_print\_raw        将屏幕坐标的原始数据打印出来

```
1796350964.530867: 216 865 1 按下
```

```
1796350989.946831: 0 0 0 抬起
```

##### 4) ts\_test 检测一下校正结果，如果touch的时候，十字光标随着手指移动

#### 5、常见错误的解答

##### 1)

在make的过程提示以下错误解决：

```
libtool:link: only absolute run-paths are allowed
```

忘记指定环境变量：`export PLUGIN_DIR=$PWD/plugins`

##### 2)

ts\_config: Success

追查后应该是在打开ts\_config中出现了错误

tslib插件库目录 没有指定对

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

3)

ts\_open:Inappropriate ioctl 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
```

解决办法:

```
#echo $LD_LIBRARY_PATH //查看lib路径,
```

```
#export LD_LIBRARY_PATH=$T_ROOT/lib
```

5)

```
ts_open: No such file or directory
```

解决办法:

```
#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
```

解决办法:

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

7)

Couldnt load module pthres

No raw modules loaded.

ts\_config: Success

解决办法:

#export TSLIB\_PLUGINDIR=\$T\_ROOT/lib/ts

8)

No raw modules loaded.

Ts\_config: No such file or directory

解决办法:

修改\$T\_ROOT/etc/ts.conf, 至少放开一个module\_raw, 并去掉前面空格

## 第二节: QT4.7.0 在 ARM 平台移植

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

下载地址: <http://qt.nokia.com/downloads-cn>

把下面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：带红色的字体部分要改成对应的目录 -I /-L 指定前面编译出的 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：编译之前使用 unset CC 取消CC变量的定义，否则编译会出错**

**step 2:**

拷贝/usr/local/Trolltech/QtEmbedded-4.7.0-arm/lib目录下的以下文件到根文件系统/usr/lib目录下：

**#cp -a /usr/local/Trolltech/QtEmbedded-4.7.0-arm/lib/\*so\* rootfs/usr/lib**

（也可以不用全部拷贝，可以根据需要进行拷贝，这里为了方便我们全拷贝过去）

**step 3:**

在根文件系统下建立目录

**#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/**

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

**step 5:修改环境变量及LCD驱动连接**

修改 /etc/profile 文件，增加以下声明：

```
export set QWS_MOUSE_PROTO="TSLIB:/dev/event0 Intellimouse:/dev/mouse0" #鼠标  
键盘均支持  
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  
置屏幕参数  
export set QT_QWS_FONTDIR=/usr/lib/fonts/ #字体库环境变  
量
```

#### step 6:

拷贝一个应用程序到根文件系统的/home目录下，

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

```
#. / appchooser -qws
```

程序在显示屏上显示窗口。

## 第三节：boa web 服务器在 PC 上移植

boa 是一个非常小巧的 Web 服务器，可执行代码只有约 60KB。它是一个单任务 Web 服务器，只能依次完成用户的请求，而不会 fork 出新的进程来处理并发连接请求。但 Boa 支持 CGI，能够为 CGI 程序 fork 出一个进程来执行。下面我们讲解以下 boa 如何在 PC 机（虚拟机 RedHatAS5 环境）完成移植修改工作。

boa 下载地

址：<http://www.rayfile.com/zh-cn/files/d7a7b368-c236-11e0-a55a-0015c55db73d/>

### 一、配置编译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生效目录设置

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

第 30 行 #define SERVER\_ROOT "/etc/boa", 默认为/etc/boa 文件夹

我们为了统一路径起见, 改为: "/home/boa", 接下来所有跟 boa 有关的文件我们都放在/home/boa 下

5. 修改src/boa.c

注释掉下面语句:

```
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
```

否则运行boa时会提示

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的可执行程序，./boa运行即可，ps能看到boa的进程即说明boa正常工作。

注：有些编译器会提示以下错误，按相应方法解决即可。

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

解决方法：

方法>. 修改compat.h中的

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

为：

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

## 二> 修改配置文件boa.conf

1. 建立/home/boa目录，并复制boa.conf到/home/boa目录下，并按照以下步骤进行修改。

```
#mkdir /home/boa
```

```
#cp boa.conf /home/boa
```

2. 访问端口号设置

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

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`

一般在同一个电脑上运行多个boa服务器时可采用此方法

我们这里保持默认即可

### 3. 修改访问权限:

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。

所以我们统一修改为:

ErrorLog /home/boa/error\_log

AccessLog /home/boa/access\_log

(注意:这里/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. 设置html文件目录:

默认为: DocumentRoot /var/www

我们统一修改为:

DocumentRoot /home/boa/www

### 6. 设置默认首页: DirectoryIndex index.html

这是访问网页时若不指定访问的网页名称, 服务器默认返回的网页, 我们不做修改

## 7. 设置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，使得网页具有交互能力，后面我们在做讲解.**

## 8.测试

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

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

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

证明整个boa搭建成功.

## 第四节： boa web 服务器在 S3C2440 开发板上移植

前面已经在 PC 机上完成了 boa 服务器的移植工作，下面介绍一下在 S3C2440 硬件平台上如何完成相应工作，两者大同小异，细节上稍有不同,不同之处我用紫色标注了出来

### 一、配置编译 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 生效目录设置

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

第 30 行 #define SERVER\_ROOT "/etc/boa", 默认为/etc/boa 文件夹

我们为了统一路径起见, 改为: **"/usr/boa"**,接下来所有跟 boa 有关的文件我们都放在 **/usr/boa** 下

#### 5.修改 src/boa.c

注释掉下面语句:

```
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
```

否则运行 **boa** 时会提示

**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 的可执行程序, ./boa 运行即可, ps 能看到 boa 的进程即说明 boa 正常工作.

注: 有些编译器会提示以下错误, 按相应方法解决即可.

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

解决方法:

方法>. 修改 compat.h 中的

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

为:

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

## 二> 修改配置文件 boa.conf

1. 建立/usr/boa 目录，并复制 boa.conf 到/usr/boa 目录下，并按照以下步骤进行修改。

```
#mkdir /usr/boa
```

```
#cp boa.conf/usr/boa
```

### 2. 访问端口号设置

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

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>

一般在同一个电脑上运行多个 boa 服务器时可采用此方法

我们这里保持默认即可

### 3. 修改访问权限：

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。

所以我们统一修改为：

```
ErrorLog /usr/boa/error_log
```

```
AccessLog /usr/boa/access_log
```

(注意：这里/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. 设置 html 文件目录:

默认为: `DocumentRoot /var/www`

我们统一修改为:

`DocumentRoot /usr/boa/www`

### 6. 设置默认首页: `DirectoryIndex index.html`

这是访问网页时若不指定访问的网页名称, 服务器默认返回的网页, 我们不做修改

### 7. 设置 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, 使得网页具有交互能力, 后面我们在做讲解.

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` 大约第 94 行的注释去掉

`ServerName www.your.org.here`



## 10.测试

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

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

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

证明整个 **boa** 搭建成功.

## 第五节：boa 移植过程中出现的问题

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

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

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

解决办法：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

解决办法：修改src/boa.c

注释掉下面这段程序:

```
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

解决办法: 修改src/boa.c

注释掉下面语句:

```
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

解决方法:

方法 1> 确定日志目录对与所有用户都具有可读/写的权限

方法 2> 修改src/log.c (建议采用方法 1)

注释掉

```
if(dup2(error_log, STDERR_FILENO) == -1) {  
  
DIE("unable to 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
```

## 第六节：sqlite-3.7.6.2 在 PC 及 S3C2440 平台移植

SQLite，是一款轻型的数据库，是遵守ACID的关联式数据库管理系统，它的设计目标是嵌入式的，而且目前已经在很多嵌入式产品中使用了它，它占用资源非常的低，在嵌入式设备中，可能只需要几百K的内存就够了。它能够支持 Windows/Linux/Unix等等主流的操作系统，同时能够跟很多程序语言相结合，比如 Tcl、C#、PHP、Java等，还有ODBC接口，同样比起Mysql、PostgreSQL这两款开源世界著名的数据库管理系统来讲，它的处理速度比他们都快。SQLite 第一个Alpha版本诞生于 2000 年 5 月。至今已经有 10 个年头，SQLite也迎来了一个版本 SQLite 3 已经发布

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

下载地址：

<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 行的 “ 3.7.6.2” 空格导致，去掉’ 3’ 之前的空格，编译通过。**

## 第七节：ffmpeg 视频编解码库 S3C2440 移植

FFmpeg 是一个开源免费跨平台的视频和音频流方案，属于自由软件，采用 LGPL 或 GPL 许可证（依据你选择的组件）。它提供了录制、转换以及流化音视频的完整解决方案。它包含了非常先进的音频/视频编解码库 libavcodec，为了保证高可移植性和编解码质量，libavcodec 里很多 codec 都是从头开发的。

官方网站：

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

下载地址：

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

## 一、配置编译

```
#!/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、使用：

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

注意： -i 后面指定摄像头的设备文件，摄像头必须遵循 v412 协议

## 第八节：mplayer 在 PC、ARM 上移植

MPlayer是一款开源[多媒体播放器](#)，以GNU通用公共许可证发布。此款软件可在各主流作业系统使用，例如Linux和其他类Unix系统、[微软](#)的视窗系统及苹果电脑的Mac OS X系统。MPlayer是建基于命令行界面，在各作业系统可选择安装不同的图形界面。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，而且你也能使用GGI和SDL和一些低级的硬件相关的驱动模式(比如Matrox, 3Dfx和Radeon,

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

下载地址:

<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里, 在任意目录下敲mplayer即可启动mplayer**

**使用./mplayer file.name 即可播放指定视频或音频**

### 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可执行程序

其中**make**会出错，需要更改mplayer\_src/libmpeg2/motion\_comp\_arm\_s.S中的代码，

```
#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：主要配置选项说明**

**a、--host-cc=gcc**

//是用来编译一些需要在host上执行的中间文件的，如codec-cfg，出现“codec-cfg无法运行”等错误均是因为少了这句话的原因

**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` //设定静态连接，省去考很多库的麻烦，建议添加
- f、`--prefix=/tmp/mplayer` //指定编译出的可执行文件放到什么地方，默认放到源文件所在文件夹下
- g、`--disable-mp3lib --enable-mad`  
//禁用mplayer自带的音频解码库，使用mad解码库，自带的解码库CPU占用率较高的问题，实际经验证，不加之选项，编译可以通过，但是没有什么声音输出
- h、`--language=zh_CN` //编译出的Mplayer版本 帮助、提示信息均为中文

## 第九节：mplayer 字符界面操作方法

### 一、启动播放时参数：

在终端下敲入：`./mplayer` 提示各种使用帮助信息

- `-vo <drv[:dev]>` 选择视频输出模式和设备(用'`-vo help`' 查看列表)
- `-ao <drv[:dev]>` 选择音频输出模式和设备(用'`-ao help`' 查看列表)
- `-ss <timepos>` 寻找指定的(多少秒或 hh:mm:ss)位置
- `-nosound` 不播放声音
- `-fs -vm -zoom` 全屏播放选项(fullscr, vidmode chg, softw.scale)
- `-x <x> -y <y>` 设置播放的分辨率(用于改变 vidmode 或软件缩放)
- `-sub <file>` 指定使用的字幕文件(参见-subfps, -subdelay)
- `-playlist <file>` 指定使用播放列表文件
- `-vid x -aid y` 选择用于播放的视频(x)和音频(y)流
- `-fps x -srate y` 改变视频(x fps)和音频(y Hz)率
- `-pp <quality>` 使用后期处理滤镜(详细内容参见 manpage/docs)
- `-framedrop` 使用 frame-dropping (用于慢机器)

### 二、播放时控制键

基本控制键：(完整的列表参见 manpage, 同时也要检查一下 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首/下一首  
p or SPACE       暂停播放 (按任意键继续)  
q or ESC          停止播放并推出  
+ or -            调整音频延迟 +/-0.1 秒  
o                  循环 OSD 模式: none/seekbar/seekbar+timer  
\* or /            增加或减少 pcm 音量

z or x            调整字幕延迟 +/-0.1 秒

r or t            上/下调整字幕位置, 参见-vf expand!

### 三、mplayer 的使用实例:

```
#mplayer filename (PC 版)
```

```
#!/mplayer -ac mad filename (ARM 版)
```

//即可播放音频, 经测试过可播放 .mp3 .wmv .mpeg .mp4 等格式的音视频

```
#!/mplayer -ss 10
```

//从第 10s 开始播放

```
#!/mplayer -ss 00:01:00
```

//从第 1 分钟开始播放

```
#!/mplayer -fs -zoom -x 160 -y 120
```

//以 160\*120 大小居中播放视频

`//-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` 设置缓存为 8M

```
#!/mplayer -loop 5 file
```

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

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

`//制作一个文件播放列表`

```
#mplayer -playlist tom.lst -shuffle
```

`//-playlist` 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
```

## 第十节: mplayer 从模式说明

**slave**模式协议

### 一、简介:

默认**mplayer**是从键盘上获得控制信息

**mplayer**另外提供了一种更为灵活的控制方式, 用来进行播放控制——**slave**模式

在**slave**模式下, **MPlayer**为后台运行其他程序, 不再截获键盘事件,

**MPlayer**会从标准输入读一个换行符 (`\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

### 方法一：从控制台输入控制命令(测试使用)

运行**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

### 常用到的 Mplayer指令：

**loadfile string** //参数**string** 为 歌曲名字。

**volume 100 1**//设置音量 中间的为音量的大小。

**mute1/0**//静音开关

**pause**//暂停/取消暂停

**get\_time\_length**//返回值是播放文件的长度，以秒为单位。

**seek value** //向前查找到文件的位置播放 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置用秒表示，采用浮点数

**volume <value> [abs]** //增大/减小音量，或将其设置为<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'标题'的元数据

**get\_meta\_year**//打印出当前文件的'年份'的元数据

### 方法二：从有名管道(fifo)输入控制命令（应用编程中使用）

```
#mkfifo </tmp/fifofile>
```

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

//用户可以通过往管道里写入slave命令来实现对应的功能

例：主进程创建一个无名管道和一个有名管道

### 1:开一个子进程

在子进程中：

启动Mplayer，参数规定通过命名管道进行通信；

把子进程的标准输出重定向无名管道的写端；

Mplayer从命名管道读到主进程发送的命令；

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

### 2: 在父进程中：

启动两个线程

第一个线程，不断使用fgets从键盘获取一个字符串命令，并写入命名管道中

第二个线程，循环检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](#)

```
1. #include <stdio.h>
2. #include <stdlib.h>
3. #include <unistd.h>
4. #include <fcntl.h>
5. #include <sys/stat.h>
6. #include <sys/types.h>
7. #include <string.h>
8.
9. /*****全局变量定义区*****/
10. int fd_fifo; //创建有名管道,用于向mplayer发送命令
11. int fd_pipe[2]; //创建无名管道,用于从mplayer读取命令
12.
13. void *get_pthread(void *arg)
14. {
15.     char buf[100];
16.     while(1)
17.     {
18.         printf("please input you cmd:");
```

```
19.         fflush(stdout);
20.         fgets(buf, sizeof(buf), stdin);           //从标准输入获取数据
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");                       //如果命名管道存在，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)                                //子进程播放mplayer
63.     {
64.         close(fd_pipe[0]);
65.         dup2(fd_pipe[1],1);                    //将子进程的标准输出重定向到管
道的写端
66.         fd_fifo=open("/tmp/my_fifo",O_RDWR);
67.         execlp("mplayer","mplayer","-slave","-quiet","-input","file=/t
mp/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_thread,NULL);        //从键盘获
取控制信息
78.         pthread_create(&tid2,NULL,print_thread,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. }
```

## 第十一节：mplayer 从模式翻译文档

**mplayer** 源码中有一个关于 **mplay slave** 模式的说明文档：  
**/DOCS/tech/slave.txt**，下面是中文翻译

slave 模式协议

-----

在关于 **slave** 模式，**MPlayer** 为后台运行其他程序，不再截获键盘事件，**MPlayer** 会从标准输入读一个换行符（**\n**）分隔开的命令。

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

您也可以使用一个 **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行选项，但并非一定要在相同的名称。详细说明中可以在手册中找到

所有命令都可以以前缀“**pausing**”，“**pausing\_keep**”，或“**pausing\_toggle**”为前缀。

“**pausing**”告诉 MPlayer 暂停尽快正在处理的命令。

“**pausing\_keep**”告诉 MPlayer 暂定保持只要它已在暂停模式。

“**pausing\_toggle**”告诉 MPlayer 暂定保持只要它尚未暂停模式。

请注意，“尽可能”可以在命令完全执行之前。

各种提示和技巧（帮助扩展！）：

- 尝试使用例如

```
pausing_keep_force pt_step 1
```

```
get_property pause
```

切换到下一个文件，它避免在转换到新的音频文件之前旧文件播放一小段时间

可用的命令（'**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选择下一个/前一个。

#### **audio\_delay <value> [abs]**

设置/调整音频延迟。

如果[abs]不提供或为零，调整延迟<value>秒。

如果[abs]不为零，将延迟到<value>秒。

#### **[brightness|contrast|gamma|hue|saturation] <value> [abs]**

设置/调整视频参数。

如果[abs]不提供或为零，修改参数为<value>。

如果[abs]不为零，参数设置为<value>。

<value>的范围是[-100, 100]。

#### **change\_rectangle <val1> <val2>**

更改矩形滤波器矩形的坐标。

**<val1>**

必须是下列之一：

**0 = 宽度**

**1 = 高度**

**2 = x 坐标**

**3 = y 坐标**

**<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**：

相对矩形左上角的整数值。正值移动矩形向右/向下和负值移动矩形向左/向上。

#### **dvb\_set\_channel <channel\_number> <card\_number>**

设置的 DVB 通道。

#### **dvdnav <button\_name>**

给定 dvdnav 按钮。

**up**

**down**

**left**

**right**

**menu**

**select**

**prev**

**mouse**

#### **edl\_mark**

将当前位置写入 EDL 文件。

### **frame\_drop [value]**

切换/设置帧的模式。

### **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码器的名称。

### **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'标题'的元数据。

### **get\_meta\_track**

打印出当前文件的'音轨的数量'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置为整数百分比[0-100)。

### **get\_property <property>**

打印出的属性的当前值。

**get\_sub\_visibility**

打印出字幕能见度（1 ==开启，0 ==关闭）。

**get\_time\_length**

打印出当前文件的长度用秒表示。

**get\_time\_pos**

打印出在文件的当前位置用秒表示，采用浮点数。

**get\_vo\_fullscreen**

全屏状态打印出来（1 == 全屏，0 ==窗口）。

**get\_video\_bitrate**

打印出当前文件的视频比特率。

**get\_video\_codec**

打印出当前视频文件的编解码器的名称。

**get\_video\_resolution**

打印出当前文件的视频分辨率。

**screenshot <value>**

截屏。要求屏幕过滤器加载。

0 以一个单独的截图。

1 启动/停止服用，每帧画面。

**gui\_[about|loadfile|loadsubtitle|play|playlist|preferences|skinbrowser|stop]**

图形用户界面行动

**key\_down\_event <value>**

注入<value>到 MPlayer 的关键代码的事件。

**loadfile <file|url> <append>**

加载给定的文件/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>**

加载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/设置怎样的电影应该是循环多次。 -1 代表不循环，永远的 0。

### 菜单命令>

执行上显示 OSD 菜单命令。

**up** 移动光标向上。

**down** 移动光标向下。

**ok** 接受的选择。

**cancel** 取消选择。

**hide** 隐藏的 OSD 菜单。

### set\_menu <menu\_name>

显示菜单命名<menu\_name>。

### mute [value]

切换声音输出静音或将其设置为[value] (value>=0)

(1 ==开启, 0 ==关闭)。

### osd [level]

切换 OSD 模式或将其设置为[level]在[level]>= 0。

### osd\_show\_property\_text <string> [duration] [level]

显示一项关于 OSD 扩展属性的字符串, 看到-playing-msg 用于描述可用的扩展。如果[duration]>=0,显示为[duration]ms。 [level]设置所需的最低水平 OSD 该消息可见 (默认是: 0 -始终显示)。

### 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>的范围, 1.0 是最高的。

负值降低 pan-and-scan 范围。

如果<abs> != 0, 那么 pan-and-scan 范围被解释为绝对的范围。

### pause

暂停/取消暂停播放。

### frame\_step

播放一帧, 然后暂停。

### pt\_step <value> [force]

转到下一个/上的播放树项。标志的<value>讲述

该方向。如果没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。可选的整数[value]的值作为返回代码为 mplayer 的进程（默认值：0）。

### **radio\_set\_channel <channel>**

切换到<channel>。在'channel'的广播参数需要设置。

### **radio\_set\_freq <frequency in MHz>**

设置广播频率调谐器。

### **radio\_step\_channel <-1|1>**

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

### **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>%在电影里。
- 2 是寻求一个绝对位置的<value>秒。

### **seek\_chapter <value> [type]**

定位一章的开始。

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

### **switch\_angle <value>**

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

### **set\_mouse\_pos 的<X> <y>**

告诉 MPlayer 的窗口中鼠标坐标。  
此命令不移动鼠标！

### **set\_property <property> <value>**

设置属性。

### **speed\_incr <value>**

增加<value>当前回放速度。

### **speed\_mult <value>**

目前速度乘以<value>。

**speed\_set <value>**

设定速度为<value>。

**step\_property <property> [value] [direction]**

通过 **value** 来改变属性，或者，如果没给定或为 **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/设置对齐字幕。

- 0 顶部对齐
- 1 居中对齐
- 2 底部对齐

**sub\_delay <value> [abs]**

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

**sub\_load <subtitle\_file>**

从<subtitle\_file>加载字幕。

**sub\_log**

当前日志上显示的字幕或连同文件名和时间信息的 `~/mplayer/subtitle_log`。

**sub\_pos <value> [abs]**

调整/设置字幕的位置。

**sub\_remove [value]**

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

**sub\_select [value]**

显示字幕的索引[**value**]。关闭字幕显示，如果关闭[**value**]的值为-1 或比更高可用的字幕指数更大。

可用的字幕周期，如果[**value**]省略或低于-1。支持字幕来源是 **-sub** 选项在命令行，**VOBsubs**，**DVD** 字幕和 **Ogg** 和 **Matroska** 文本流。

这主要是循环所有字幕命令，如果要设置一个特定的字幕，使用 **sub\_file**，**sub\_vob**，或 **sub\_demux**。

### **sub\_source [source]**

显示第一个字幕，从[source]。这里[source]是一个整数：

**SUB\_SOURCE\_SUBS (0)** 用于文件字幕

**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，将关闭字幕显示。如果[source]低于-1，将循环每个之间的现有资源第一个字幕。

### **sub\_file [value]**

显示字幕 **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，将关闭字幕显示。如果[value]小于-1，将循环的所有文件 **subs**。

### **sub\_vob [value]**

显示字幕 **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，将关闭字幕显示。如果[value]小于-1，将循环的所有 **vobsubs**。

### **sub\_demux [value]**

显示字幕 **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，将关闭字幕显示。

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>或将其设置为<value>时，[abs]不为零。

### **vobsub\_lang**

这是与 **sub\_select** 为了向后兼容。

### **sub\_step<value>**

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

### **sub\_visibility [value]**

切换/设置字幕。

### **forced\_subs\_only [value]**

强制切换/设置字幕。

### **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

歌曲，如果[value]省略或负数。

**switch\_angle [value]** (DVD 光盘只)

切换到 DVD 的角度通过 ID[value]。循环  
如果可用角度，如果[value]省略或负数。

**switch\_ratio [value]**

在运行时改变长宽比。 [value]是表示新的长宽比  
作为浮动 16 / 9 (例如 1.77778)。  
这可能与某些视频过滤器的问题。

**switch\_title [value]** (DVDNAV only)

切换到 DVD 标题通过 ID[value]。循环可用标题，如果[value]的值省略或负数。

**switch\_vsync [value]**

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

**teletext\_add\_digit <value>**

进入/离开字幕的页面号编辑模式，并追加提供的以前输入的数字。

0 .. 9 - 附加 appropriate 数字。(启用编辑模式，如果从一般要求模式，并  
切换到正常模式时。)

- - 删除最后的页码数字。(退格仿真，只能在页码编辑模式。)

**teletext\_go\_link <1-6>**

按照目前的字幕的页面给出链接。

**tv\_start\_scan**

电视频道开始自动扫描。

**tv\_step\_channel <channel>**

选择下一个/上一个电视频道。

**tv\_step\_norm**

更改电视制式。

**tv\_step\_chanlist**

改变频道列表。

**tv\_set\_channel <channel>**

设置当前的电视频道。

**tv\_last\_channel**



设置当前电视频道到最后一个。

**tv\_set\_freq <frequency in MHz>**

设置电视调谐器的频率。

**tv\_step\_freq <frequency offset in MHz>**

设置电视调谐器的频率相对于当前值。

**tv\_set\_norm <norm>**

电视调谐器设置规范（包括 PAL，SECAM，NTSC 制式，...）。

**tv\_set\_brightness <-100 - 100> [abs]**

设置电视调谐器的亮度或调整，如[abs]设置为 0。

**tv\_set\_contrast <-100 -100> [abs]**

设置电视调谐器的对比或调整，如[abs]设置为 0。

**tv\_set\_hue <-100 - 100> [abs]**

设置电视调谐器色调或调整，如[abs]设置为 0。

**tv\_set\_saturation <-100 - 100> [abs]**

设置电视调谐器饱和或调整，如[abs]设置为 0。

**use\_master**

主之间切换和 PCM 音量控制。

**vo\_border [value]**

切换/设置边界显示。

**vo\_fullscreen [value]**

切换/设置全屏模式。

**vo\_ontop [value]**

切换/设置保持在最上层。

**vo\_rootwin [value]**

切换/设置在根窗口播放。

**volume <value> [abs]**

增大/减小音量，或将其设置为<value>，如果[abs]不为零。

下面的命令，实际上只可用于 OSD 菜单控制台模式：

### help

帮助文本显示，目前还是空的。

### exit

从 OSD 菜单退出控制台。不像'quit'，不退出 MPlayer 的。

### hide

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

### run <value>

运行<value>的 shell 命令。在 OSD 菜单控制台模式标准输出和标准输入是通过视频输出。

## 第十二节：使用 busybox 制作 Linux 根文件系统

构建最简单的嵌入式 Linux 根文件系统，下面我们一步步教大家做起。。

硬件环境：

S3C2440

软件环境：

Busybox-1.16.1 <http://www.rayfile.com/zh-cn/files/2e477e14-clbb-11e0-b287-0015c55db73d/>

cross-4.3.2 <http://www.rayfile.com/zh-cn/files/99e727f3-cla5-11e0-822a-0015c55db73d/>

### STEP 1: 构建目录结构

创建根文件系统目录，主要包括以下目录

```
/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: 使用 busybox 构建/bin /sbin linuxrc

进入 busybox-1.16.1 目录，执行

```
#make defconfig
```

```
#make menuconfig
```

```
Busybox Setting ----->
```

```
Build Options ----->
```

```
//1 选择将busybox进行静态编译
```

```
[*]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. 选择上 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命令解释器支持显示当前路径及主机信
```

息

保存退出

```
#make
```

```
#make install
```

在 busybox 目录下会看见 \_install 目录，里面有/bin /sbin linuxrc 三个文件  
将这三个目录或文件拷到第一步所建的 rootfs 文件夹下。

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

切记一定要带上-a的参数，因为bin目录里大部分都是链接，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 构建etc目录:

#### 1)进入根文件系统rootfs的etc目录，执行如下操作:

拷贝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，把第二项改为::respawn:-/bin/login

删除第三、第四行代码

#### 2)拷贝虚拟机上的/etc/passwd, /etc/group, /etc/shadow到rootfs/etc下

```
# cp /etc/passwd rootfs/etc
# cp /etc/group rootfs/etc
# cp /etc/shadow rootfs/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，而且最后改成/bin/ash。

修改group为root:x:0:root

修改shadow为

```
root:$1$x9yv1W1B$abJ2v9j010c9xW/y0QwPs.:14034:0:99999:7:::
```

登陆开发板时需输入用户名密码，同虚拟机相同

### 3) 修改profile

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

/bin/hostname sunplusedu

USER="`id -un`"

LOGNAME=$USER

HOSTNAME=' /bin/hostname'

PS1=' [\u@\h \W]# '                          //显示主机名、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服务需要

/bin/mkdir -p /dev/shm                          //telnet服务需要

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

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

```
/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 构建lib目录:

```
1)#cd
```

```
/usr/local/arm/4.3.2/arm-none-linux-gnueabi/libc/armv4t/lib
```

将以下动态库拷贝到rootfs/lib下

```
#cp *so* rootfs/lib -a
```

```
2)#cd
```

```
/usr/local/arm/4.3.2/arm-none-linux-gnueabi/libc/armv4t/usr/lib
```

将以下动态库拷贝到rootfs/lib下

```
#cp ./libstdc++.so.* rootfs/lib -a
```

以上资料来自凌阳教育嵌入式培训网，更多嵌入式linux学习资料免费申请：

<http://emb.sunplusedu.com/download/>