一、 mjpeg-streamer移植

MJPG-streamer是一个轻量级的视频服务器软件，一个可以从单一输入组件获取图像并传输到多个输出组件的命令行应用程序，可应用在基于IP协议的网络中，从网络摄像头中获取并传输JPEG格式的图像到浏览器。该工具源码简洁，注释清晰，使用Linux　C语言进行开发，可移植到不同的计算机平台。

1.准备阶段

主机环境	Ubuntu 14.04
目标机	fs4412
目标机系统	linux 3.14
编译工具链	arm-none-linux-gnueabi-gcc
摄像头	USB摄像头

MJPG-stream的移植需要jpeg库，所以必须先移植jpeg库,源码下载：
Jpeg源码包：http://www.ijg.org/files/jpegsrc.v8b.tar.gz
MJPG-stream源码包： http://sourceforge.net/projects/mjpg-streamer/

2. Jpeg移植

<1>解压

tar -xvf jpegsrc.v8b.tar.gz

<2>配置编译

第一步:配置

./configure --host=arm-none-linux-gnueabi --prefix=/home/linux/workdir/fs4412-bak/camera/jpeg-install

这里的--host=交叉开发工具链前缀　--prefix=编译完之后，安装的路径

第二步:编译

make

第三步:安装

make　install

安装成功之后，在--prefix 参数指定的路径下，可以看到如下内容:

```
linux@ubuntu:~/workdir/fs4412-bak/camera/jpeg-install$ ls
bin  include  lib  share
```

3.mjpg-stramer移植

<1>解压

tar -xvf mjpg-streamer-r63.tar.gz

<2>修改mjpg-streamer-r63源码顶层目录下的Makefile

```
 8 ###############################################################
 9
10 CC = arm-none-linux-gnueabi-gcc
11
12 CFLAGS += -O2 -DLINUX -D_GNU_SOURCE -Wall
13 #CFLAGS += -O2 -DDEBUG -DLINUX -D_GNU_SOURCE -Wall
14 LFLAGS += -lpthread -ldl
15
16 APP_BINARY=mjpg_streamer
17 OBJECTS=mjpg_streamer.o utils.o
18
19 all: application plugins
20
21 clean:
22     make -C plugins/input_uvc $@
23     make -C plugins/input_testpicture $@
24     make -C plugins/output_file $@
25     make -C plugins/output_http $@
26     make -C plugins/output_autofocus $@
27     make -C plugins/input_gspcav1 $@
28     rm -f *.a *.o $(APP_BINARY) core *~ *.so *.lo
29
30 plugins: input_uvc.so output_http.so
31
```

指定开发平台相关的gcc编译器

我们这里只需要一个input_uvc和 output_http插件，其他的都去掉

<2>修改插件下的Makefile

我们这里只使用了input_uvc 和 output_http两个插件，所以只需要修改这两个插件的Makefile。

plugins/input_uvc/Makefile修改如下:

```
CC = arm-none-linux-gnueabi-gcc

OTHER_HEADERS = ../../mjpg_streamer.h ../../utils.h ../output.h ../input.h

CFLAGS += -I /home/linux/workdir/fs4412-bak/camera/jpeg-install/include
CFLAGS += -O2 -DLINUX -D_GNU_SOURCE -Wall -shared -fPIC
#CFLAGS += -DDEBUG
LFLAGS += -lpthread -ldl
```

开发平台编译器gcc指定

```
all: input_uvc.so

clean:
    rm -f *.a *.o core *~ *.so *.lo

input_uvc.so: $(OTHER_HEADERS) input_uvc.c v4l2uvc.lo jpeg_utils.lo dynctrl.lo
    $(CC) $(CFLAGS) -L /home/linux/workdir/fs4412-bak/camera/jpeg-install/lib -ljpeg -o $@ input_uv
```

指定jpeg库的路径

plugins/output_http/Makefile修改如下:

```
CC = arm-none-linux-gnueabi-gcc

OTHER_HEADERS = ../../mjpg_streamer.h ../../utils.h ../output.h ../input.h

CFLAGS += -O2 -DLINUX -D_GNU_SOURCE -Wall -shared -fPIC
#CFLAGS += -DDEBUG
LFLAGS += -lpthread -ldl

all: output_http.so

clean:
    rm -f *.a *.o core *~ *.so *.lo

output_http.so: $(OTHER_HEADERS) output_http.c httpd.lo
    $(CC) $(CFLAGS) -o $@ output_http.c httpd.lo
```

<3>编译

make的时候，会提示**fatal error: linux/videodev.h: No such file or directory**这样的错误，找到对应的文件，这个文件中包含的**#include <linux/videodev.h>**修改**成#include <linux/videodev2.h>**就可以了。

编译成功之后，效果如下:

```
linux@ubuntu:~/workdir/fs4412-bak/camera/mjpg-streamer-r63$ ls
CHANGELOG     Makefile         mjpg_streamer.h  plugins     utils.c  www
input_uvc.so  mjpg_streamer    mjpg_streamer.o  README      utils.h
LICENSE       mjpg_streamer.c  output_http.so   start.sh    utils.o
```

start.sh 是运行应用程序的脚步文件

www 目录存放的是一个网页，这个网页可以通过网络的方式在浏览器上显示

input_uvc.so 插件库，完成的是从摄像头中采集数据

ouput_http.so插件库，完成的是将摄像头采集的数据通过http协议传输到客户端

mjpg_stremaer 就是我们需要运行的应用程序，它在运行的时候，会加载input_uvc.so 和 ouput_http.so

好了,到此为止，我们已经完成了mjpeg-stramer源码的编译，接下来我们把它拷贝到开发板上的文件系统中，最终在开发板上运行我们的应用程序。笔者这里开发板启动的时候，是通过NFS的方式挂载文件系统的。我们在开发板的文件系统中，新建一个camera目录，拷贝以下文件到camera目录下:

```
linux@ubuntu:~/workdir/fs4412/fs/rootfs/camera$ ls
input_uvc.so  libjpeg.so.8  mjpg_streamer  output_http.so  start.sh  www
```

在fs4412平台，摄像头采集出来的数据是YUYV格式的原始数据，是无法直接显示的，所以需要将YUYV格式转换成jpeg格式。

这里我们只需要修改start.sh，告诉应用程序需要将YUYV格式数据转换成jpeg格式就可以了。修改如下:

```
export LD_LIBRARY_PATH="$(pwd)"
./mjpg_streamer -i "input_uvc.so -y" -o "output_http.so -w ./www"
```

二、USB 摄像头驱动添加

USB驱动移植读者自己完成，FS4412平台可以参照实验手册来完成USB驱动移植。linux 3.14内核USB摄像头驱动添加如下:

```
Device Drivers  --->
    <*> Multimedia support  --->
        [*]  Media USB Adapters  --->
            <*>  USB Video Class (UVC)
            [*]    UVC input events device support
```

配置完成之后，重新编译Linux内核。

二、Qt应用程序显示Mjpeg-stramer应用程序通过http协议传输的数据

mjpeg-stramer应用程序采集完摄像头数据之后，是通过http协议的multipart/mixed报文进行数据传输的。关于multipart/mixed,读者自己去了解。这里给出Qt应用程序解析multipart/mixed报文的参考代码:

```
void Widget::on_pushButton_clicked()
{
    this->statu = START;
    this->currentLen = 0;
    QString httpUrl = "http://192.168.1.5:8080/?action=stream";
    networkReply = netwokManger->get(QNetworkRequest(QUrl(httpUrl)));
    connect(networkReply,SIGNAL(readyRead()),this,SLOT(readHttpCameraData()));
    connect(networkReply,SIGNAL(finished()),this,SLOT(finishedHttp()));
}
```

```
void Widget::readHttpCameraData()
{
     qint32 len;
     QByteArray contentTypeLine;
     QByteArray contentLenghtLine;
     QByteArray blankSpaceLine;
     QByteArray endBoundaryLine;

    switch(statu){
      case START:
           boundary = networkReply->readLine();
           statu = NEXT1;
      break;

      case NEXT1:
           contentTypeLine   = networkReply->readLine();
           contentLenghtLine = networkReply->readLine();
           blankSpaceLine    = networkReply->readLine();
           sscanf(contentLenghtLine.data(),"Content-Length: %d\r\n",&totalLen);
           buf.resize(totalLen);
           statu = NEXT2;
```

```
     case NEXT2:
        len = networkReply->read(buf.data()+ currentLen,totalLen - currentLen);
        currentLen = currentLen + len;
        if(currentLen == totalLen){
            image.loadFromData(buf);
            statu = END;
            update();
            break;
        }
        break;

    case END:
        blankSpaceLine = networkReply->readLine();
        endBoundaryLine = networkReply->readLine();
        statu = NEXT1;
        currentLen = 0;
        break;
    }
}
```

```
void Widget::paintEvent(QPaintEvent *event)
{
    QPainter painter(this);
    painter.setRenderHint(QPainter::SmoothPixmapTransform,true);
    painter.drawPixmap(this->rect().adjusted(50,50,-100,-100),QPixmap::fromImage(image));
}
```