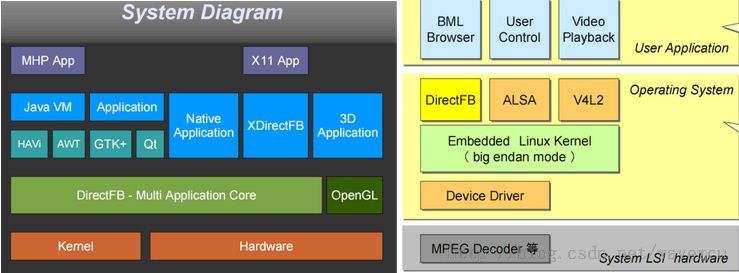
**一:DirectFB简介**

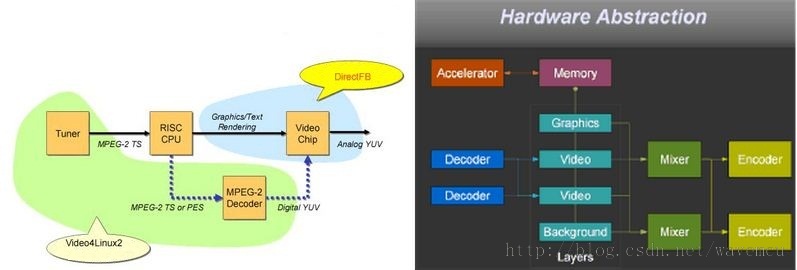
      DirectFB是一个轻量级的提供硬件图形加速，输入设备处理和抽象的图形库，它集成了支持半透明的视窗系统以及在LinuxFramebuffer驱动之上的多层显示。它是一个用软件封装当前硬件无法支持的图形算法来完成硬件加速的层。DirectFB是为嵌入式系统而设计。它是以最小的资源开销来实现最高的硬件加速性能。

      DirectFB是图形API存在于Frame Buffer层之上与高级图形管理层如GTK+等之下的。它可以以很小的系统资源占用来提供图形硬件加速功能，提供类如多路a通道渲染模型等高级图像操作。它不修改Linux内核，除了标准C库没有其他库的依赖。应用在了基于Linux系统的DTV显示系统的研发和其他有关Linux显示界面的项目上。支持市面上绝大多数显示卡，支持键盘、鼠标、遥控器、游戏手柄、触摸屏等输入设备。支持JPEG、PNG、GIF、mpeg1/2、AVI、MOV、Flash、Video4Linux、DirectFB bitmap font和TrueType等音视频文件和字体。

**二：DirectFB在系统的位置**  
      从GUI整个系统架构来讲，对于一般的MMP，DTV和STB来说，下面图中是比较经典的结构。 DirectFB都处于QT，GTK+这样的高级GUI框架的下面，Linux系统驱动的上面。如下图所示：

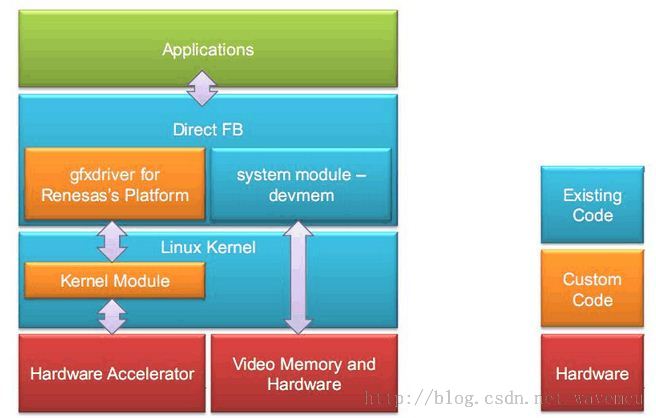


   从功能上面来讲，下面图中比较形象的说明了DirectFB的主要作用：图形，文字的处理（DirectFB把TS经过Decoder处理之后的码流，再加上图形和文字，一起传给视频的芯片），如下图所示：



**三：DirectFB GUI的架构**

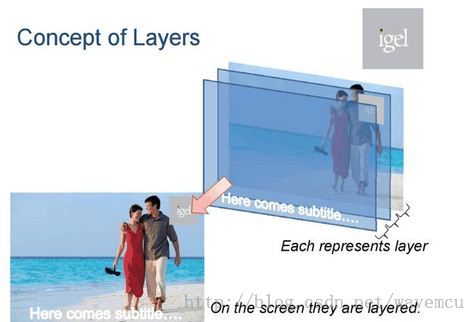
      对于上层的GUI高级框架来说，DirectFB是透明的，下图是一个更加细化的DirectFB的结构图：



       对于底层的驱动来说，DirectFB里面负责和硬件打交道的显卡的驱动（gfxdriver）和显示系统（system），这里面gfxdriver是和各个硬件平台相关的，有可能需要自己写。 为了更易于理解DirectFB，需要先介绍几个DirectFB里面的概念：

* Layers：代表互相独立的图形缓存。大多数嵌入式设备都有多个layer。多个layer根据对应的alpha值混合 在一起，从而显示出来。
* Surface：代表一块预留的内存，来保存像素数据。DirectFB中的Drawing，Bilitting操作就是基于 surface的。Surface的内存根据设定，可以从系统中分配，也可以从显卡的内存中分配。
* Primary Surface：代表一个特殊layer的特殊surface。如果primary surface是单缓冲的，任何对于 primary surface的操作都会直接反应到显示屏上。
* Subsurface：是一个surfac的子集，但是自身并不占有任何内存。
* Window/Windowstack：控制一个layer中的surface该显示什么。Window属于某个背景可以设定的layer。 每个window有自己的surface，window用这个surface来组合图像。

       比如说下图中有三个Layer，最下面的Layer是一幅背景图，中间的Layer是一个透明的带有一个“igel”的Layer，最上面是一个透明的带有subtitle的Layer，最后我们在显示器中看到的就是各个Layer的混合。



        再比如说下面这幅图，Lower Layer是一个全屏模式的Layer，它的背景图片渲染在自己的SurFace上面。Video area是一个SurFace，用来显示video。Upper Layer通过window stack/window 来显示应用程序的内容，而且Upper Layer是一个透明的Layer，这样混合之后，可以透过上面的Layer看到下面的video。



**四：DirectFB的移植**

 如果要移植DirectFB需要用到以下的第三方库，如下图所示：



DirectFB需要用到第三方库简介：

* fontconfig-2.10.92：管理系统安装的字体，根据应用程序的字体配置，match（匹配）一种字体，填充match字体的各种属性。
* freetype-2.5.0：FreeType库是一个完全免费（开源）的、高质量的且可移植的字体引擎，它提供统一的接口来访问多种字体格式文件，包括TrueType, OpenType, Type1, CID, CFF, Windows FON/FNT, X11 PCF等。
* jpeg-8d：jpeg开源库
* libpng-1.5.8：PNG 库是用来创立和操作PNG 格式的图像文件，PNG 格式是设计来替代GIF，他对于更小范围的TIFF(标记图象文件格式)来说，有了非常多的进步和拓展并且减少了关于专利权的麻烦。
* libsigc++-2.2.8：libsigc++是实现类型安全回调的一个C++模板库。libsigc++提供了信号(signal)和槽(slot)的概念，实现了信号机制。slot对应一个回调函数，信号(signal)与槽(slot)相关联(attach)，当一个信号被发射(emit)时，相对应的槽被调用。
* libxml2-2.6.28：包含了对 XPath 表达式使用的支持来检索匹配一个指定准则的结点集。libxml2软件包提供允许用户操控XML文件的函数库,包含有读、修改和写XML和HTML文件支持。

4.1： jpeg-8d的编译配置

jpeg-8d的build.sh的脚本配置如下所示：

**[plain]** [view plain](http://blog.csdn.net/wavemcu/article/details/39251805) [copy](http://blog.csdn.net/wavemcu/article/details/39251805)

[print](http://blog.csdn.net/wavemcu/article/details/39251805)[?](http://blog.csdn.net/wavemcu/article/details/39251805)[在CODE上查看代码片](https://code.csdn.net/snippets/468240)

1. #!/bin/bash
3. if [ "$1" == "" ]; then
4. INSTALL\_PATH=`pwd`/\_install
5. else
6. if [ -d $1 ]; then
7. INSTALL\_PATH=$1
8. else
9. exit 1
10. fi
11. fi
13. mkdir -p $INSTALL\_PATH
15. #autoreconf -ivf
16. if [ -f config.log ]; then
17. make distclean
18. fi
20. ./configure --prefix=$INSTALL\_PATH \
21. --host=arm-none-linux-gnueabi \
22. --target=arm-none-linux-gnueabi \
23. --build=i486-linux-gnu \
24. --disable-option-checking \
25. --disable-silent-rules \
26. --disable-dependency-tracking \
27. --enable-maintainer-mode \
28. --enable-ld-version-script \
29. --enable-shared=yes \
30. --enable-static=no \
31. --enable-fast-install=yes \
32. --disable-libtool-lock \
33. --with-gnu-ld
35. if [ $? -eq 0 ]; then
36. make
37. if [ $? -eq 0 ]; then
38. make install
39. exit 0
40. fi
41. fi
42. exit 1

编译配置成动态库，如上面的build.sh脚本所示，具体的配置可以通过configure --help命令来查看！

4.2：libpng-1.5.8的编译和配置

libpng-1.5.8的build.sh的脚本配置如下所示：

**[plain]** [view plain](http://blog.csdn.net/wavemcu/article/details/39251805) [copy](http://blog.csdn.net/wavemcu/article/details/39251805)

[print](http://blog.csdn.net/wavemcu/article/details/39251805)[?](http://blog.csdn.net/wavemcu/article/details/39251805)[在CODE上查看代码片](https://code.csdn.net/snippets/468240)

1. #!/bin/bash
3. if [ "$1" == "" ]; then
4. INSTALL\_PATH=`pwd`/\_install
5. else
6. if [ -d $1 ]; then
7. INSTALL\_PATH=$1
8. else
9. exit 1
10. fi
11. fi
13. mkdir -p $INSTALL\_PATH
15. #autoreconf -ivf
16. if [ -f config.log ]; then
17. make distclean
18. fi
20. ./configure --prefix=$INSTALL\_PATH \
21. --host=arm-none-linux-gnueabi \
22. --target=arm-none-linux-gnueabi \
23. --build=i486-linux-gnu \
24. --disable-option-checking \
25. --disable-dependency-tracking \
26. --enable-maintainer-mode \
27. --disable-arm-neon \
28. --enable-shared=yes \
29. --enable-static=no \
30. --enable-fast-install=yes \
31. --disable-libtool-lock \
32. --with-gnu-ld
34. if [ $? -eq 0 ]; then
35. make
36. if [ $? -eq 0 ]; then
37. make install
38. exit 0
39. fi
40. fi
41. exit 1

编译配置成动态库，如上面的build.sh脚本所示，具体的配置可以通过configure --help命令来查看！

4.3：libxml2-2.6.28的编译和配置

libxml2-2.6.28的build.sh的脚本配置如下所示：

**[plain]** [view plain](http://blog.csdn.net/wavemcu/article/details/39251805) [copy](http://blog.csdn.net/wavemcu/article/details/39251805)

[print](http://blog.csdn.net/wavemcu/article/details/39251805)[?](http://blog.csdn.net/wavemcu/article/details/39251805)[在CODE上查看代码片](https://code.csdn.net/snippets/468240)

1. #!/bin/bash
3. if [ "$1" == "" ]; then
4. INSTALL\_PATH=`pwd`/\_install
5. else
6. if [ -d $1 ]; then
7. INSTALL\_PATH=$1
8. else
9. exit 1
10. fi
11. fi
13. mkdir -p $INSTALL\_PATH
15. #autoreconf -ivf
16. if [ -f config.log ]; then
17. make distclean
18. fi
20. ./configure --prefix=$INSTALL\_PATH \
21. --host=arm-none-linux-gnueabi \
22. --build=i486-linux-gnu \
23. --disable-option-checking \
24. --disable-dependency-tracking \
25. --enable-maintainer-mode \
26. --disable-silent-rules \
27. --enable-shared=yes \
28. --enable-static=no \
29. --enable-fast-install=yes \
30. --disable-libtool-lock \
31. --disable-documentation \
32. --enable-warnings=min \
33. --with-sax1=yes \
34. --with-catalog=yes \
35. --with-gnu-ld
37. if [ $? -eq 0 ]; then
38. make
39. if [ $? -eq 0 ]; then
40. make install
41. exit 0
42. fi
43. fi
44. exit 1

编译配置成动态库，如上面的build.sh脚本所示，具体的配置可以通过configure --help命令来查看！

4.4：freetype-2.5.0的编译和配置

freetype-2.5.0的build.sh的脚本配置如下所示：

**[plain]** [view plain](http://blog.csdn.net/wavemcu/article/details/39251805) [copy](http://blog.csdn.net/wavemcu/article/details/39251805)

[print](http://blog.csdn.net/wavemcu/article/details/39251805)[?](http://blog.csdn.net/wavemcu/article/details/39251805)[在CODE上查看代码片](https://code.csdn.net/snippets/468240)

1. #!/bin/bash
3. if [ "$1" == "" ]; then
4. INSTALL\_PATH=`pwd`/\_install
5. else
6. if [ -d $1 ]; then
7. INSTALL\_PATH=$1
8. else
9. exit 1
10. fi
11. fi
13. mkdir -p $INSTALL\_PATH
15. #autoreconf -ivf
16. if [ -f config.log ]; then
17. make distclean
18. fi
20. ./configure --prefix=$INSTALL\_PATH \
21. --host=arm-none-linux-gnueabi \
22. --build=i486-linux-gnu \
23. --disable-option-checking \
24. --disable-dependency-tracking \
25. --enable-biarch-config \
26. --disable-mmap \
27. --enable-shared=yes \
28. --enable-static=no \
29. --enable-fast-install=yes \
30. --disable-libtool-lock \
31. --without-bzip2 \
32. --without-zlib \
33. --without-png \
34. --with-gnu-ld
36. if [ $? -eq 0 ]; then
37. make
38. if [ $? -eq 0 ]; then
39. make install
40. exit 0
41. fi
42. fi
43. exit 1

编译配置成动态库，如上面的build.sh脚本所示，具体的配置可以通过configure --help命令来查看！

4.5：fontconfig-2.10.92的编译和配置

fontconfig-2.10.92的build.sh的脚本配置如下所示：

**[plain]** [view plain](http://blog.csdn.net/wavemcu/article/details/39251805) [copy](http://blog.csdn.net/wavemcu/article/details/39251805)

[print](http://blog.csdn.net/wavemcu/article/details/39251805)[?](http://blog.csdn.net/wavemcu/article/details/39251805)[在CODE上查看代码片](https://code.csdn.net/snippets/468240)

1. #!/bin/bash
3. if [ "$1" == "" ]; then
4. INSTALL\_PATH=`pwd`/\_install
5. else
6. if [ -d $1 ]; then
7. INSTALL\_PATH=$1
8. else
9. exit 1
10. fi
11. fi
13. mkdir -p $INSTALL\_PATH
15. export LDFLAGS="-L$INSTALL\_PATH/lib"
16. export CFLAGS="-I$INSTALL\_PATH/include"
17. export PKG\_CONFIG\_PATH="$INSTALL\_PATH/lib/pkgconfig"
18. export LIBS="-ldl -lpthread -lpng"
20. if [ -f config.log ]; then
21. make distclean
22. fi
24. ./configure --prefix=$INSTALL\_PATH \
25. --host=arm-none-linux-gnueabi \
26. --build=i486-linux-gnu \
27. --disable-option-checking \
28. --disable-dependency-tracking \
29. --disable-silent-rules \
30. --disable-largefile \
31. --enable-shared=yes \
32. --enable-static=no \
33. --enable-fast-install=yes \
34. --disable-libtool-lock \
35. --enable-libxml2 \
36. --disable-iconv \
37. --with-gnu-ld \
38. FREETYPE\_CFLAGS="-I$INSTALL\_PATH/include/freetype2" \
39. FREETYPE\_LIBS="-L$INSTALL\_PATH/lib -lfreetype" \
40. LIBXML2\_CFLAGS="-I$INSTALL\_PATH/include/libxml2" \
41. LIBXML2\_LIBS="-L$INSTALL\_PATH/lib -lxml2"
43. if [ $? -eq 0 ]; then
44. make
45. if [ $? -eq 0 ]; then
46. make install
47. exit 0
48. fi
49. fi
50. exit 1

编译配置成动态库，如上面的build.sh脚本所示，具体的配置可以通过configure --help命令来查看！

4.6：libsigc++-2.2.8的编译和配置

libsigc++-2.2.8的build.sh的脚本配置如下所示：

**[plain]** [view plain](http://blog.csdn.net/wavemcu/article/details/39251805) [copy](http://blog.csdn.net/wavemcu/article/details/39251805)

[print](http://blog.csdn.net/wavemcu/article/details/39251805)[?](http://blog.csdn.net/wavemcu/article/details/39251805)[在CODE上查看代码片](https://code.csdn.net/snippets/468240)

1. #!/bin/bash
3. if [ "$1" == "" ]; then
4. INSTALL\_PATH=`pwd`/\_install
5. else
6. if [ -d $1 ]; then
7. INSTALL\_PATH=$1
8. else
9. exit 1
10. fi
11. fi
13. mkdir -p $INSTALL\_PATH
15. #autoreconf -ivf
16. if [ -f config.log ]; then
17. make distclean
18. fi
20. ./configure --prefix=$INSTALL\_PATH \
21. --host=arm-none-linux-gnueabi \
22. --build=i486-linux-gnu \
23. --disable-option-checking \
24. --disable-dependency-tracking \
25. --enable-maintainer-mode \
26. --enable-shared=yes \
27. --enable-static=no \
28. --enable-fast-install=yes \
29. --disable-libtool-lock \
30. --enable-ipv6=no \
31. --with-gnu-ld
33. if [ $? -eq 0 ]; then
34. make
35. if [ $? -eq 0 ]; then
36. make install
37. exit 0
38. fi
39. fi
40. exit 1

编译配置成动态库，如上面的build.sh脚本所示，具体的配置可以通过configure --help命令来查看！

**五：移植总结**

     DirectFB的移植留到下一篇来详细介绍，这一篇只详细的介绍DirectFB所需要的第三方库的编译和配置。