



**Android RIL代码分析**

版权声明

版权所有©深圳市广和通实业发展有限公司 2012。保留一切权利。

非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

注意

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

商标申明

 为深圳市广和通实业发展有限公司的注册商标，由所有人拥有。

版本记录

|  |  |  |  |
| --- | --- | --- | --- |
| 文档版本 | 更新日期 | 说明 | 编写人 |
| V1.0.0 |  | 初始版本 | 刘其峰 |

**目录**

[1 基础知识普及 5](#_Toc402166200)

[2 RIL整体架构 6](#_Toc402166201)

[3 hardware/ril/reference-ril目录主要文件说明 7](#_Toc402166202)

[4 RIL接口调用方式 8](#_Toc402166203)

[5 涉及到主要源代码文件 10](#_Toc402166204)

[5.1 reference-ril.c 文件涉及的重要内部函数及功能 10](#_Toc402166205)

[5.2 ril.cpp 文件主要提供ril.java调用接口，实现RIL的JNI调用方式、RIL Event管理机制 10](#_Toc402166206)

[5.3 rild.c 主要初始化工作都在main函数中进行。 11](#_Toc402166207)

[6 RIL如何处理AT命令 12](#_Toc402166208)

[7 如何编译在android源码上编译RIL代码 13](#_Toc402166209)

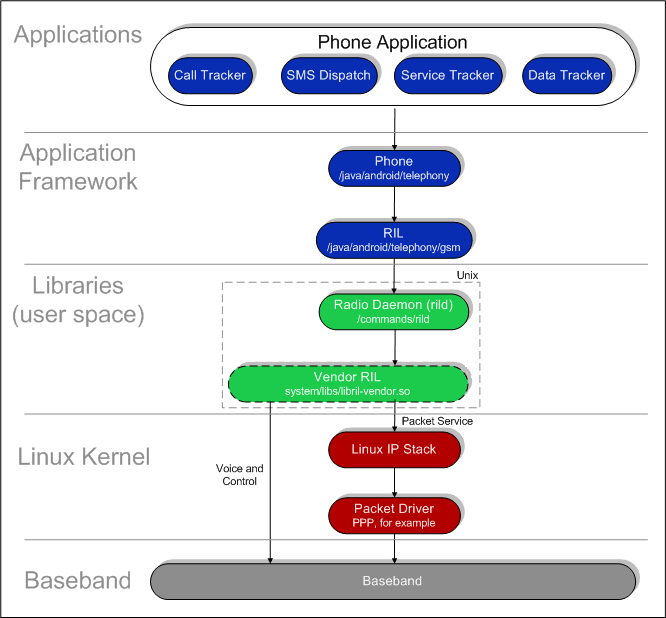
# 基础知识普及

RIL(Radio Interface Layer)，是Android电信业务系统和底层硬件之间的抽象层，Android提供该层方便用户移植。手机每次请求相应的电信服务，如电话、短信、网络连接等，都是通过AT指令与底层硬件进行交互的，拿电话来讲，用户请求拨打电话，需要向RIL发送相关的请求，接着RIL按照用户的请求，调用相应的AT指令发送接口给底层发送AT指令，完成用户拨打电话的流程。简单的说，对RIL的移植，主要是针对不同的设备针对AT指令做相应的修改。

RIL负责数据的可靠传输、AT命令的发送以及response的解析。一般的，应用处理器（AP）通过AT命令集与无线通讯模块（基带/BP）通信。通信的方式又分为主动请求的request(诸如拨号、发短信……)，以及Modem主动上报的例如信号强度、基站信息、来电、来短信等，称之为unsolicited response。

# RIL整体架构

下图为RIL在Android系统中所处位置：



根据上图，Vendor RIL是提供AT指令发送接口，RIL Daemon是加载RIL相关的库同时初始化RIL，注册RIL相关函数接口。同时RILD也是上层和底层RIL的调用中转站。RIL JNI Layer，提供上层Telephony APP Framework调用。我司的RIL一般集中在Vendor RIL部分，即对reference-ril.c的处理。

# hardware/ril/reference-ril目录主要文件说明

1、文件reference-ril.c：

此文件核心是两个函数：onRequest和onUnsolicited

1) onRequest 函数：在这个函数里，对每一个RIL\_REQUEST\_XXX请求，都转化成相应的ATcommand，发送给modem，然后睡眠等待。当收到此AT command的最终响应后，线程被唤醒，将响应传给客户进程（RIL\_onRequestComplete -> sendResponse）。

2) onUnsolicited函数：这个函数处理modem从网络端收到的各种事件，如网络信号变化，拨入的电话，收到短信等。然后将时间传给客户进程（RIL\_onUnsolicitedResponse -> sendResponse）。

2、文件atchannel.c：

负责向modem读写数据。其中，写数据(主要是AT command)功能运行在主线程中，读数据功能运行在一个单独的读线程中。

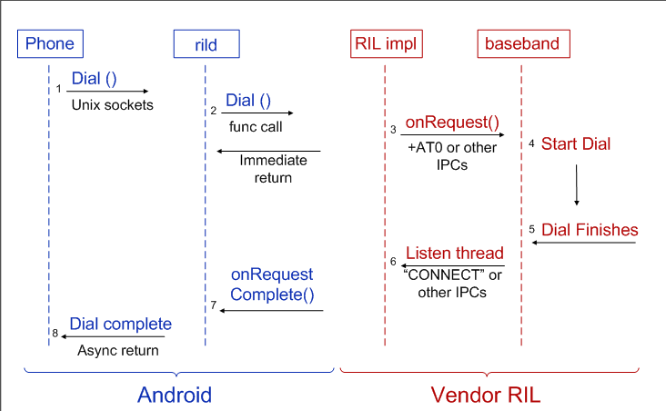
函数at\_send\_command\_full\_nolock：运行在主线程里面。将一个AT command命令写入modem后进入睡眠状态（使用 pthread\_cond\_wait或类似函数），直到modem读线程将其唤醒。唤醒后此函数获得了AT command的最终响应并返回。函数readerLoop运行在一个单独的读线程里面，负责从modem中读取数据。读到的数据可分为三种类型：网络端传入的事件；modem对当前AT command的部分响应；modem对当前AT command的全部响应。对第三种类型的数据（AT command的全部响应），读线程唤醒（pthread\_cond\_signal）睡眠状态的主线程。

3、文件at\_tok.c 提供AT响应的解析函数

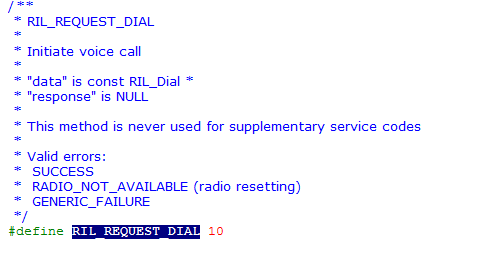
4、misc.c 字面意思杂项，里面就提供一个字符串匹配函数

# RIL接口调用方式

两种方式：主动请求(Solicited)和被动请求(Unsolicited)

1. 主动请求处理流程

主动请求一般由上层应用发起，到RIL层就是以一个Request的方式下来,再根据对应该的Request处理，查看ril.h可知道每个请求对应的response所需要参数及request所下的参数。以上图为例：上层Phone发起Dial(打电话)动作，经过RILD再到RIL的时候就是一个RIL\_REQUEST\_DIAL的请求，再调用requestDial函数下发ATD电话号码;拨打电话。查看ril.h中对RIL\_REQUEST\_DIAL的定义可知



Response为NULL，即不需要参数。Requset所带的data参数是一个结构体RIL\_Dial，在ril.h文件同样可以查询到其成员变量为

/\*\* Used by RIL\_REQUEST\_DIAL \*/

typedef struct {

char \* address; -🡪指的是电话号码

int clir;

/\* (same as 'n' paremeter in TS 27.007 7.7 "+CLIR"

\* clir == 0 on "use subscription default value"

\* clir == 1 on "CLIR invocation" (restrict CLI presentation)

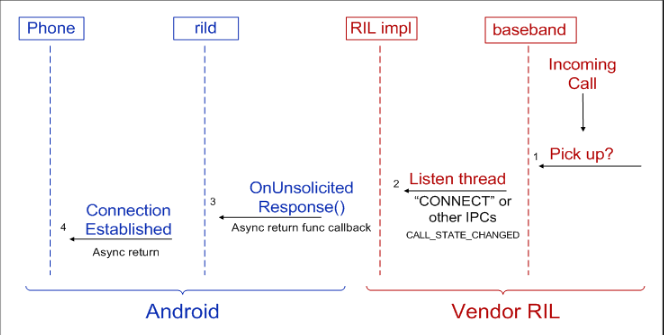
\* clir == 2 on "CLIR suppression" (allow CLI presentation)

\*/

RIL\_UUS\_Info \* uusInfo; /\* NULL or Pointer to User-User Signaling Information \*/

} RIL\_Dial;

1. 被动请求处理流程

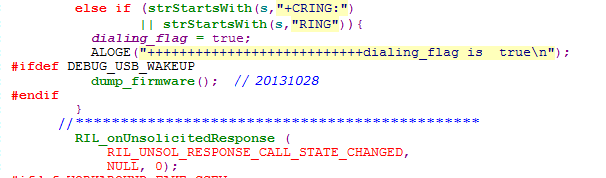


被动请求由模块的主动上报发起，在RIL里面处理函数为

static void onUnsolicited (const char \*s, const char \*sms\_pdu)//主动上报响应

{//onUnsolicited只解析出头部(一般是+XXXX的形式)，然后按类型决定下一步操作

以来电为例，来电时模块会上动上报RING的动作。



上图为对应函数处理，当RING上报时RIL需要给上层反馈CALL状态发生变化。上层根据这个变化将显示来电界面。对应的RIL\_UNSOL\_RESPONSE\_CALL\_STATE\_CHANGED参数可以在ril.h中查询到相关的解释。

# 涉及到主要源代码文件

## reference-ril.c 文件涉及的重要内部函数及功能

requestRadioPower 开启通讯模块

requestQueryNetworkSelectionMode 查询网络模式

requestGetCurrentCalls 获取当前通话

requestDial 请求呼叫

requestWriteSmsToSim 写入短信至SIM卡

requestHangup 挂断电话

requestSignalStrength 获取信号强度

requestRegistrationState 获取卡注册状态

requestOperator 获取运营商名称

requestSendSMS 请求发送短信

requestSetupDataCall 拨号上网

requestSMSAcknowledge 短信回报

requestEnterSimPin 请求输入SIM PIN码

requestSendUSSD 请求USSD业务

## ril.cpp 文件主要提供ril.java调用接口，实现RIL的JNI调用方式、RIL Event管理机制

实现了供上层调用的接口

typedef struct {

int requestNumber;

void (\*dispatchFunction) (Parcel &p, struct RequestInfo \*pRI);

int(\*responseFunction) (Parcel &p, void \*response, size\_t responselen);

} CommandInfo;

具体执行的操作可以看头文件ril\_commands.h

举一个例子： 拨打电话，对应的requestNumber为RIL\_REQUEST\_DIAL，对应的dispatchFunction为dispatchDial，对应的responseFunction为responseFunction。

RIL Event管理

与ril\_event.cpp的Event接口配合实现RIL Event管理

RIL\_startEventLoop

rilEventAddWakeup

eventLoop

ril\_event\_init

ril\_event\_set

ril\_event\_add ril\_event\_del

## rild.c 主要初始化工作都在main函数中进行。

# RIL如何处理AT命令

以查询当前设置的PDP上下文参数AT+CGDCONT? 命令为例，看RIL如何处理此AT。

先看CGDCONT?的格式：

|  |  |  |  |
| --- | --- | --- | --- |
| Read | AT+CGDCONT? | +CGDCONT: <cid>, <PDP\_type>, <APN>,<PDP\_addr>, <d\_comp>, <h\_comp>[,<pd1>[,…[,pdN]]]  OK | 读取当前设置的PDP上下文参数。  如果没有任何设置，将只返回OK |

示例：

AT+CGDCONT?

+CGDCONT: 1,"IP","CMWAP","0.0.0.0",0,0

+CGDCONT: 2,"IP","CMNET","0.0.0.0",0,0

OK

RIL里面处理函数为requestRegistrationState，先使用

at\_send\_command\_multiline (cmd, prefix, &p\_response);下发AT+CGDCONT？命令，

其中cmd 为CGDCONT? , prefix 为 +CGDCONT, p\_response即为下发此AT后，用来存储AT的回复的结构体。之后需要处理p\_response->p\_intermediates->line 变量，line即为 AT的响应+CGREG:XXXXXX;

at\_tok\_start(&line); --- 指针跳转到 : 位置

at\_tok\_nextint(&line, &cid); -- 指针跳转到 下一个整型位置，并将此整型变量存储至pdp\_type变量中

at\_tok\_nextstr(&line, &pdp\_type); -- 指针跳转至下一个字符串位置，并将字符串存储至pdp\_type变量中

at\_tok\_nextstr(&line, &apn); -- 指针跳转至下一个字符串位置，并将字符串存储至apn变量中

at\_tok\_nextstr(&line, &pdp\_address);指针跳转至下一个字符串位置，并将字符串存储至pdp\_address变量

至此AT处理完毕。那如何将这些处理结果反馈给上层呢。RIL通过下述函数将处理结果反馈给上层。

RIL\_onRequestComplete(\*t, RIL\_E\_SUCCESS, responses, n \* sizeof(RIL\_Data\_Call\_Response\_v6));

其中第二个参数responses为AT处理结果。Responses为 RIL\_Data\_Call\_Response\_v6结构体里面成员变量查看ril.h文件可知即为AT处理相应的结果，如cid,pdp\_type,apn,address等。

所以上层对于此请求需要什么结果都可以查看ril.h, ril.h详细列出所有请求的处理说明。

# 如何编译在android源码上编译RIL代码

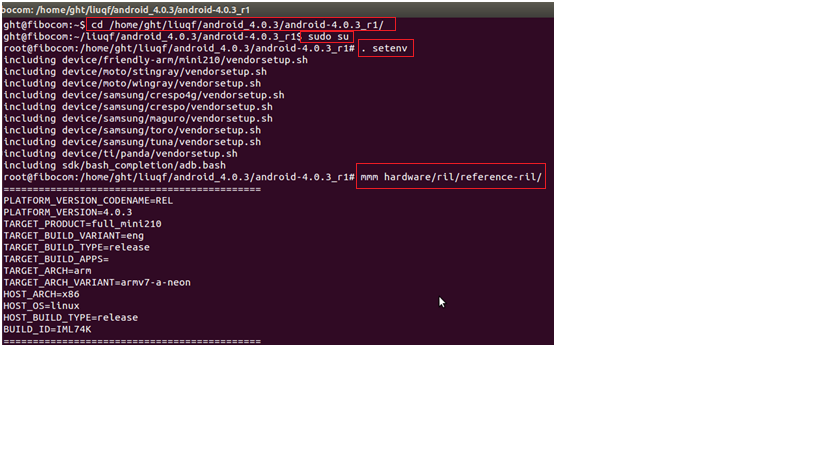
我司RIL分别支持android4.0 4.2 4.4等系统,源码路径分别为

Android4.0：/home/ght/liuqf/android\_4.0.3/android-4.0.3\_r1/

Android4.2: /home/ght/liuqf/intel\_sourcecode/ecs\_byt\_anzhen3/

Android4.4: /home/ght/liuqf/android\_4.4/android4.4/

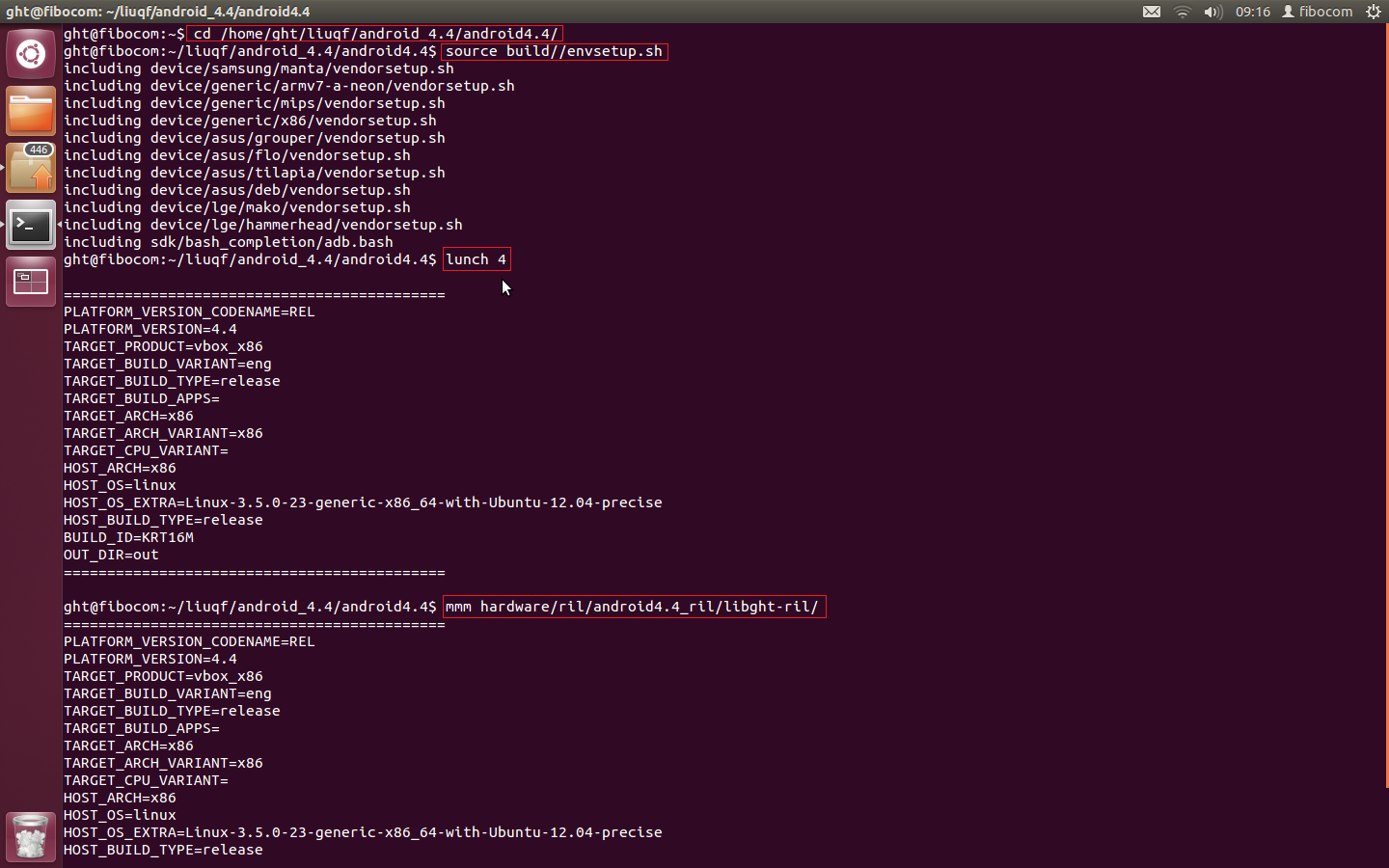
RIL代码编译：

Android4.0:

执行 gedit hardware/ril/reference-ril/reference-ril.c 可编辑源码文件，用ctrl+s保存修改

Android4.2编译：

执行 gedit hardware/ril/android4.2\_bak/android4.2\_ril/reference-ril/reference-ril.c 可编辑源码文件

Android4.4编译：

执行 gedit hardware/ril/android4.4\_ril/libght-ril/libght-ril.c 可编辑源码文件,ctrl+s保存