Gabby

android learner

7-framework--详解 8 Android平台开发-WIFI 驱动移植 -- 详细 9 Android WIFI 驱动移植

http://blog.csdn.net/wh_19910525/article/details/7397619

WifiService：

由SystemServer启动的时候生成的ConnecttivityService创建，负责 启动关闭wpa_supplicant, 启动和关闭WifiMonitor线程，把命令下发给wpa_supplicant以及 更新WIFI的状态

WifiMonitor负责从wpa_supplicant接收事件通知

8 Android平台开发-WIFI 驱动移植 -- 详细

http://blog.csdn.net/wh_19910525/article/details/7392199

一、WIFI的基本架构（代码路径）

1、WIFI Settings应用程序：
packages/apps/Settings/src/com/android/settings/wifi/
2、JAVA部分（framework）：
frameworks/base/services/java/com/android/server/
frameworks/base/wifi/java/android/net/wifi/
3、JNI部分：
frameworks/base/core/jni/android_net_wifi_Wifi.cpp
4、wpa_supplicant的适配器 部分
hardware/libhardware_legary/wifi/主要是wifi.c。
5、wifi用户空间的程序和库:
external/wpa_supplicant/ 和 external/wap_supplicant_6/ 我不清楚是哪一个目录生成库libwpaclient.so和 守护进程wpa_supplicant。
6、kernel 驱动

二、WIFI在Android中如何工作

Android使用一个修改版wpa_supplicant作为daemon来控制WIFI，代码位于external/wpa_supplicant。wpa_supplicant是通过socket与hardware/libhardware_legacy/wifi/wifi.c通信。UI(APP)通过android.net.wifi package（frameworks/base/wifi/java/android/net/wifi/）发送 命令 给wifi.c。相应的JNI实现位于frameworks/base/core/jni/android_net_wifi_Wifi.cpp。更高一级的网络管理位于frameworks/base/core/java/android/net。

三、配置Android支持WIFI

*在device/rootfs-project/BoardConfig.mk中添加：

LINUX_PRO (6)
OS_Install_Tips (12)
C++ (60)
HR (9)
health (1)
tv&&music&&fun (9)
compile (9)
并发处理相关 (8)
const&mutable (2)
callback_func (1)
ipc (1)
camera_framework (9)
sp&wp (3)
selinux (1)
initrc (3)
cmd (2)
virtual_func (2)
stl (1)
log (2)
mediaplayer (1)
parcel (1)
property (2)
debug (1)
busybox (1)
bionic (1)
zygote (2)
surface (2)
jni (1)
笔试题目 (3)
平台设备总线模型 (1)
iic (0)
字符驱动 (1)

文章存档

2016年04月 (32)
2016年03月 (6)
2016年02月 (13)
2015年12月 (43)
2015年11月 (9)

展开

阅读排行

请教：过量喝咖啡对心脏 (4385)
C++和JAVA的区别 -- 给初 (3934)
svn如何取消认证缓存设置 (3792)
android TP (2739)
Android LCD (2562)
高通平台android开发总结 (2280)
Android图形合成和显示系 (2258)
一个离职员工对中兴的回 (2237)
高通QPST Download使用 (2208)
android camera (2088)

推荐文章

* CSDN日报20170828——《4个方法快速打造你的阅读清单》
* Android检查更新下载安装
* 动手打造史上最简单的Recycleview 侧滑菜单
* TCP网络通讯如何解决分包粘包问题
* 程序员的八重境界

WPA_SUPPLICANT_VERSION := VER_0_8_X //wpa_supplicant的版本
WIFI_DRIVER := ar6003//驱动名字（自己定义的宏），主要在*hardware/平台名称/wlan/芯片名/Android.mk* 文件里使用
***BOARD_WPA_SUPPLICANT_PRIVATE_LIB* := lib_driver_cmd_wext**
BOARD_WPA_SUPPLICANT_DRIVER := WEXT //驱动类型，决定wap_supplicant的底层接口类型
这将使external/wpa_supplicant/Android.mk设置WPA_BUILD_SUPPLICANT为true，默认使用驱动driver_wext.c。
如果使用定制的wpa_supplicant驱动(例如 madwifi)，可以设置:
BOARD_WPA_SUPPLICANT_DRIVER := MADWIFI

------- 如果 wifi 具有 SoftAP(即虚拟无线AP) 的功能，需要以下两个宏 -------
WIFI_DRIVER_FW_STA_PATH := /system/wifi/fw.bin
WIFI_DRIVER_FW_AP_PATH :=/system/wifi/fw_ap.bin

/* 以下两项 可以在 hardware/libhardware_legary/wifi/wifi.c 里边直接定义 */
WIFI_DRIVER_MODULE_PATH := /system/wifi/ar6000.ko //wifi 驱动路径
WIFI_DRIVER_MODULE_NAME := ar6000//驱动名字
WIFI_DRIVER_MODULE_ARG:=DBG = 7 //该宏是用于insmod时传参数

四、使能wpa_supplicant调试信息

默认wpa_supplicant设置为MSG_INFO，为了输出更多信息，可修改：
1、在/external/wpa_supplicant/common.c中设置wpa_debug_level = MSG_DEBUG;
2、在common.c中把#define wpa_printf宏中的
if ((level) >= MSG_INFO)
改为
if ((level) >= MSG_DEBUG)

五、修改system/etc/wifi/wpa_supplicant.conf（在源码中是修改external/wpa_supplicant/wpa_supplicant.conf）

wpa_supplicant是通过 rootfs-src/external/wpa_supplicant/wpa_supplicant.conf中的ctrl_interface=来指定控制socket的，这个路径在wifi.c中用到。
一般的/external/wpa_supplicant/**wpa_supplicant.conf**配置为：
ctrl_interface=DIR=/data/system/wpa_supplicant GROUP=wifi
update_config=1
fast_reauth=1
eapol_version=1
有时，驱动需要增加：
ap_scan=1
如果遇到AP连接问题，需要修改ap_scan=0来让驱动连接，代替wpa_supplicant。

如果要连接到non-WPA or open wireless networks，要增加：
network={
key_mgmt=NONE
}

六、配置路径和权限

Google修改的wpa_supplicant要运行在wifi用户和组下的。代码可见/external/wpa_supplicant/os_unix.c中的os_program_init()函数。

关闭

* 四大线程池详解

如果配置不对，会出现下面错误：
E/WifiHW ( ): Unable to open connection to supplicant on "/data/system/wpa_supplicant/wlan0": No such file or directory will appear.

确认init.rc中有如下配置：

```
mkdir /system/etc/wifi 0771 wifi wifi
chmod 0771 /system/etc/wifi
chmod 0660 /system/etc/wifi/wpa_supplicant.conf
chown wifi wifi /system/etc/wifi/wpa_supplicant.conf #wifi的原始配置文件

# wpa_supplicant socket
mkdir /data/system/wpa_supplicant 0771 wifi wifi
chmod 0771 /data/system/wpa_supplicant #放置wifiinterface的地方

#wpa_supplicant control socket for android wifi.c
mkdir /data/misc/wifi 0770 wifi wifi
mkdir /data/misc/wifi/sockets 0770 wifi wifi  #与上层通过socket通信的路径
chmod 0770 /data/misc/wifi
chmod 0660 /data/misc/wifi/wpa_supplicant.conf #wifi的配置文件，将由wpa_supplicant根据实际配置写入该文件
setprop wifi.interfaceeth0  # intreface名称设置，这在framework/base/wifi/java/android/net/wifi /WifiStateTracker.java中会用到，以处理dhcp。ar6000.ko用eth0。
```

目录权限的处理是为了所有用户能对下一级进行搜索，/data/misc/wifi/sockets目录不仅为wifi拥有者服务，还因为通信的原因要和其他用户联系，要不然，将会出现Unable to open connection to supplicant on "/data/system/wpa_supplicant/ra0": Connection refused，或permission denied的错误。很多人干脆将上述所有的权限都设为0777，当然也行，但总觉得有些粗糙。

七、运行wpa_supplicant和dhcpcd

在init.rc中确保有如下语句：

```
service wpa_supplicant /system/bin/wpa_supplicant  -Dwext -iwlan0 -c /data/misc/wifi/wpa_supplicant.conf
    user root
    group system wifi
    socket wpa_wlan0 dgram 0666 wifi wifi   //这项是用于UDP连接的
    disabled
    oneshot

service dhcpcd /system/bin/logwrapper /system/bin/dhcpcd -d -B eth0
    group system dhcp
    disabled
    oneshot
```

根据所用的WIFI驱动名字，修改**wlan0**为自己驱动的名字。

八、编译WIFI驱动为module或kernel built in

1、编译为module

在**device/rootfs-project/BoardConfig.mk**中添加：

```
WIFI_DRIVER_MODULE_PATH := "/system/lib/modules/ar6000.ko"  //驱动的具体位置
WIFI_DRIVER_MODULE_ARG := ""  #for example nohwcrypt
WIFI_DRIVER_MODULE_NAME := "ar6000"  #for example wlan0
WIFI_FIRMWARE_LOADER := ""
```

关闭

2、编译为kernel built in

1）在hardware/libhardware_legacy/wifi/wifi.c要修改interface名字，

2）在init.rc中添加：

setprop wifi.interface "wlan0"

3）在hardware/libhardware_legacy/wifi/wifi.c中当insmod/rmmod时，

直接return 0。

九、WIFI需要的firmware

Android不使用标准的hotplug binary，WIFI需要的firmware要复制到/etc/firmware

或者复制到WIFI驱动指定的位置，然后WIFI驱动会自动加载。

十、修改**WIFI**驱动适合**Android**

Google修改的**wpa_supplicant**要求SIOCSIWPRIV**ioctl** 发送 命令 到驱动，及接收信息，例如signal strength, mac address of the AP, link speed等。所以要正确驱动，需要从 SIOCSIWPRIV ioctl返回RSSI (signal strength)和MACADDR信息。

如果没实现这个ioctl，会出现如下错误：

```
E/wpa_supplicant( ): wpa_driver_priv_driver_cmd failed
                     wpa_driver_priv_driver_cmd RSSI len = 4096
E/wpa_supplicant( ): wpa_driver_priv_driver_cmd failed
D/wpa_supplicant( ): wpa_driver_priv_driver_cmd LINKSPEED len = 4096
E/wpa_supplicant( ): wpa_driver_priv_driver_cmd failed
I/wpa_supplicant( ): CTRL-EVENT-DRIVER-STATE HANGED
```

十一、设置dhcpcd.conf

一般/system/etc/dhcpcd/dhcpcd.conf的配置为：

```
interface eth0
option subnet_mask, routers, domain_name_servers
```

9 Android WIFI 驱动移植

http://blog.csdn.net/wh_19910525/article/details/7395306

在 Android 2.2上移植了2个wifi模块，vt6656和rt2070，总结一下要点。

首先，将wifi linux驱动编译成模块，并将驱动（vntwusb.ko或rt3070sta.ko放到/system/lib/modules/中。

然后，做如下修改：

1。修改 init.rc：很多文章都有描述，但还是有些说明不清的地方，我先列出增加项，然后作些说明。

增加： mkdir /system/etc/wifi 0771 wifi wifi

```
chmod 0771 /system/etc/wifi
chmod 0660 /system/etc/wifi/wpa_supplicant.conf
chown wifi wifi /system/etc/wifi/wpa_supplicant.conf  # ...
# wpa_supplicant socket
mkdir /data/system/wpa_supplicant 0771 wifi wifi
chmod 0771 /data/system/wpa_supplicant  # 放置wifi interface的地方
mkdir /data/misc/wifi 0771 wifi wifi
chmod 0771 /data/misc/wifi
chmod 0660 /data/misc/wifi/wpa_supplicant.conf  # wifi的配置文件，将由wpa_supplicant根据实际配置写入该文件
mkdir /data/misc/wifi/sockets 0777 wifi wifi  # 与上层通过socket通信的路径
# Prepare for wifi
```

关闭

setprop wifi.interface ra0 # **intreface**名称设置，这在framework/base/wifi/java/android/net/wifi /WifiStateTracker.java中会用到，以处理dhcp。**rt2070**用**ra0**，而vt6656使用eth1。

这里0771对 目录权限的处理是为了所有用户能对下一级进行搜索，而红字特别提醒的权限配置，是因为/data/misc/wifi/sockets目录不仅为wifi拥有者服务，还因为通信的原因要和其他用户联系，要不然，将会出现Unable to open connection to supplicant on "/data/system/wpa_supplicant/ra0": Connection refused，或permission denied的错误。很多人干脆将上述所有的权限都设为0777，当然也行，但总觉得有些粗糙。

service的修改：

```
service wpa_supplicant /system/bin/logwrapper /system/bin/wpa_supplicant /
   -Dwext -ira0 -c/data/misc/wifi/wpa_supplicant.conf  #也可以用/system/etc/wifi
/wpa_supplicant.conf代替
   user root
   group system wifi inet
#    socket wpa_wlan0 dgram 660 wifi wifi    # 屏蔽该项是因为这项是用于UDP连接的
   disable
   oneshot

service dhcpcd /system/bin/logwrapper /system/bin/dhcpcd -d -B ra0
   group system dhcp wifi
   disabled
   oneshot
```

2。修改system/etc/wifi/wpa_supplicant.conf（在源码中是修改external/wpa_supplicant /wpa_supplicant.conf)

将ctrl_interface=wlan0改成ctrl_interface=DIR=/data/system/wpa_supplicant GROUP=wifi # 这个路径在wifi.c中用到。

3。修改system/etc/dhcpcd/dhcpcd.conf

将其中的interface名称改成ra0

4。修改芯片厂商的配置 BoardConfig.mk。

例如，Freeescale 是在device/fsl/imx51_bbg/BoardConfig.mk，加入：

```
HAVE_CUSTOM_WIFI_DRIVER_2 := true
BOARD_WPA_SUPPLICANT_DRIVER := WEXT
```

5。修改hardware/libhardware_legacy/wifi/wifi.c

已经定义的wifi驱动的路径我感到还是屏蔽为好，直接在wifi.c中修改不是更直观些。

```
ifndef WIFI_DRIVER_MODULE_PATH
#define WIFI_DRIVER_MODULE_PATH         "/system/lib/modules/rt3070sta.ko"
#endif
#ifndef WIFI_DRIVER_MODULE_NAME
#define WIFI_DRIVER_MODULE_NAME         "rt3070sta"
#endif
```

在 文件中还可以看到其他一些信息，如IFACE_DIR[] = "/data/system/wpa_supplicant"， 就是interface的安放的路径。MODULE_FILE[] = "/proc/modules"，这是insmod安放module的路径。在调试时可以查询是否已经安装了模块，接口是否启动。

6。源码修改

修改 external/wpa_supplicant/wpa_ctrl.c: 找到chmod那行，将 chmod(ctrl->local.sun_path, S_IRUSR|S_IWUSR|S_IRGRP|S_IWGRP);改成chmod(ctrl->local.sun_path, S_IRUSR|S_IWUSR|S_IRGRP|S_IWGRP|S_IROTH|S_IWOTH); #增加权限这是为了防止出现ioctl 写message的错误。

修改 external/wpa_supplicant/driver_wext.c：这 是为了避免wpa_supplicant与 下层驱动 通讯时出现ioctl[SIOCSIWPRIV]错误，因为现在大部分wifi模块对 SIOCSIWPRIV命令不处理，而这个命令要用于侦测wifi强度RSSI的，比较简单的方法是在wifi驱动中增加个空函数。例如，对于 rt2070，有一个sta_ioctl.c，找到SIOCSIWPRIV，将其对应个空函数

```
static int handler_SIOCSIWPRIV(struct net_device *dev, struct iw_request_info *info,
                               union iwreq_data *wrqu, char *extra)
{
```

关闭

```
        return 0;
}
```

不过，这样做就有些信息，如**RSSI**，**MAC**地址等就没法在上层显示了。比较好的方法如下：

在struct wpa_driver_wext_data 中增加

```
    u8 ssid[32];
    unsigned int ssid_len;
```

2个变量。将原来的wpa_driver_priv_driver_cmd函数改成如下函数（我从Porting wifi driver to Android抄来稍作修改）：

```
static int wpa_driver_priv_driver_cmd(void *priv, char *cmd, char *buf, size_t buf_len)
{

    struct wpa_driver_wext_data *drv = priv;
    int ret = -1;

    wpa_printf(MSG_DEBUG, "AWEXT: %s %s", __func__, cmd);

    if (os_strcasecmp(cmd, "start") == 0) {
        wpa_printf(MSG_DEBUG,"Start command");
        return (ret);
    }

    if (os_strcasecmp(cmd, "stop") == 0) {
        wpa_printf(MSG_DEBUG,"Stop command");
    }
    else if (os_strcasecmp(cmd, "macaddr") == 0) {
        struct ifreq ifr;
        os_memset(&ifr, 0, sizeof(ifr));
        os_strncpy(ifr.ifr_name, drv->ifname, IFNAMSIZ);

        if (ioctl(drv->ioctl_sock, SIOCGIFHWADDR, &ifr) < 0) {
            perror("ioctl[SIOCGIFHWADDR]");
            ret = -1;
        } else {
            u8 *macaddr = (u8 *) ifr.ifr_hwaddr.sa_data;
            ret = snprintf(buf, buf_len, "Macaddr = " MACSTR "/n",
                        MAC2STR(macaddr));
        }
    }
    else if (os_strcasecmp(cmd, "scan-passive") == 0) {
        wpa_printf(MSG_DEBUG,"Scan Passive command");
    }
    else if (os_strcasecmp(cmd, "scan-active") == 0) {
        wpa_printf(MSG_DEBUG,"Scan Active command");
    }
    else if (os_strcasecmp(cmd, "linkspeed") == 0) {
        struct iwreq wrq;
        unsigned int linkspeed;
        os_strncpy(wrq.ifr_name, drv->ifname, IFNAMSIZ);
        wpa_printf(MSG_DEBUG,"Link Speed command");
        if (ioctl(drv->ioctl_sock, SIOCGIWRATE, &wrq) < 0) {
            perror("ioctl[SIOCGIWRATE]");
            ret = -1;
        } else {
            linkspeed = wrq.u.bitrate.value / 1000000;
            ret = snprintf(buf, buf_len, "LinkSpeed %d/n", linkspeed);
```

关闭

```
    }
}
else if (os_strncasecmp(cmd, "scan-channels", 13) == 0) {
}
else if ((os_strcasecmp(cmd, "rssi") == 0) || (os_strcasecmp(cmd, "rssi-approx") == 0)) {
    struct iwreq wrq;
    struct iw_statistics stats;
    signed int rssi;
    wpa_printf(MSG_DEBUG, ">>>. DRIVER AWEXT RSSI ");
    wrq.u.data.pointer = (caddr_t) &stats;
    wrq.u.data.length = sizeof(stats);
    wrq.u.data.flags = 1; /* Clear updated flag */
    strncpy(wrq.ifr_name, drv->ifname, IFNAMSIZ);

    if (ioctl(drv->ioctl_sock, SIOCGIWSTATS, &wrq) < 0) {
        perror("ioctl[SIOCGIWSTATS]");
        ret = -1;
    } else {
        if (stats.qual.updated & IW_QUAL_DBM) {
            /* Values in dBm, stored in u8 with range 63 : -192 */
            rssi = ( stats.qual.level > 63 ) ?
                stats.qual.level - 0x100 :
                stats.qual.level;
        } else {
            rssi = stats.qual.level;
        }

        if (drv->ssid_len != 0 && drv->ssid_len < buf_len) {
            os_memcpy((void *) buf, (void *) (drv->ssid),
                    drv->ssid_len );
            ret = drv->ssid_len;
            ret += snprintf(&buf[ret], buf_len-ret,
                    " rssi %d/n", rssi);
            if (ret < (int)buf_len) {
                return( ret );
            }
            ret = -1;
        }
    }
}
else if (os_strncasecmp(cmd, "powermode", 9) == 0) {
}
else if (os_strncasecmp(cmd, "getpower", 8) == 0) {
}
else if (os_strncasecmp(cmd, "get-rts-threshold", 17) == 0) {
    struct iwreq wrq;
    unsigned int rtsThreshold;

    strncpy(wrq.ifr_name, drv->ifname, IFNAMSIZ);

    if (ioctl(drv->ioctl_sock, SIOCGIWRTS, &wrq) < 0) {
        perror("ioctl[SIOCGIWRTS]");
        ret = -1;
    } else {
        rtsThreshold = wrq.u.rts.value;
```

关闭

```
            wpa_printf(MSG_DEBUG,"Get RTS Threshold command = %d",
                rtsThreshold);
            ret = snprintf(buf, buf_len, "rts-threshold = %u/n",
                rtsThreshold);
            if (ret < (int)buf_len) {
                return( ret );
            }
        }
    }
    else if (os_strncasecmp(cmd, "set-rts-threshold", 17) == 0) {
        struct iwreq wrq;
        unsigned int rtsThreshold;
        char *cp = cmd + 17;
        char *endp;

        strncpy(wrq.ifr_name, drv->ifname, IFNAMSIZ);

        if (*cp != '/0') {
            rtsThreshold = (unsigned int)strtol(cp, &endp, 0);
            if (endp != cp) {
                wrq.u.rts.value = rtsThreshold;
                wrq.u.rts.fixed = 1;
                wrq.u.rts.disabled = 0;

                if (ioctl(drv->ioctl_sock, SIOCSIWRTS, &wrq) < 0) {
                    perror("ioctl[SIOCGIWRTS]");
                    ret = -1;
                } else {
                    rtsThreshold = wrq.u.rts.value;
                    wpa_printf(MSG_DEBUG,"Set RTS Threshold command = %d", rtsThreshold);
                    ret = 0;
                }
            }
        }
    }
    else if (os_strcasecmp(cmd, "btcoexscan-start") == 0) {
    }
    else if (os_strcasecmp(cmd, "btcoexscan-stop") == 0) {
    }
    else if (os_strcasecmp(cmd, "rxfilter-start") == 0) {
        wpa_printf(MSG_DEBUG,"Rx Data Filter Start command");
    }
    else if (os_strcasecmp(cmd, "rxfilter-stop") == 0) {
        wpa_printf(MSG_DEBUG,"Rx Data Filter Stop command");
    }
    else if (os_strcasecmp(cmd, "rxfilter-statistics") == 0) {
    }
    else if (os_strncasecmp(cmd, "rxfilter-add", 12) == 0 ) {
    }
    else if (os_strncasecmp(cmd, "rxfilter-remove",15) == 0) {
    }
    else if (os_strcasecmp(cmd, "snr") == 0) {
        struct iwreq wrq;
        struct iw_statistics stats;
        int snr, rssi, noise;
```

关闭

```
        wrq.u.data.pointer = (caddr_t) &stats;
        wrq.u.data.length = sizeof(stats);
        wrq.u.data.flags = 1; /* Clear updated flag */
        strncpy(wrq.ifr_name, drv->ifname, IFNAMSIZ);

        if (ioctl(drv->ioctl_sock, SIOCGIWSTATS, &wrq) < 0) {
            perror("ioctl[SIOCGIWSTATS]");
            ret = -1;
        } else {
            if (stats.qual.updated & IW_QUAL_DBM) {
                /* Values in dBm, stored in u8 with range 63 : -192 */
                rssi = ( stats.qual.level > 63 ) ?
                    stats.qual.level - 0x100 :
                    stats.qual.level;
                noise = ( stats.qual.noise > 63 ) ?
                    stats.qual.noise - 0x100 :
                    stats.qual.noise;
            } else {
                rssi = stats.qual.level;
                noise = stats.qual.noise;
            }

            snr = rssi - noise;

            ret = snprintf(buf, buf_len, "snr = %u/n", (unsigned int)snr);
            if (ret < (int)buf_len) {
                return( ret );
            }
        }
    }
    else if (os_strncasecmp(cmd, "btcoexmode", 10) == 0) {
    }
    else if( os_strcasecmp(cmd, "btcoexstat") == 0 ) {
    }
    else {
        wpa_printf(MSG_DEBUG,"Unsupported command");
    }
    return (ret);
}
```

经过这些修改后，wifi应该可以上去了。

问题留存：

1。这次修改的是5.x版本的wpa_supplicant，android已经出了6.x版本，我在那里没有发现driver_wext.c！，那我上面的修改如何实现？看来还要研究。

2。修改好的driver_wext.c可以显示MAC地址，在上层的设置界面也能显示信号强度，但在主页面上却没有显示强度，估计还要在java程序中找找原因。

3。在设置界面，如果将wifi关闭后再打开，wifi连不上，需要手工在调试终端打ifconfig ra0 up才能触发连接上。估计是java程序在关闭wifi时没有将wifi设置进行初始化操作，要去看看设置的数据库修改情况。

关闭

顶 踩
0 0

上一篇 4wpa_supplicant适配层 -- 详解 5wpa_supplicant程序 --详解 6wpa_supplicant无线网络配置

下一篇 10Android Wifi 移植配置 11wifi 驱动 进阶

相关文章推荐

- Android平台开发-WIFI 驱动移植 -- 详细
- 7-framework--详解 8 Android平台开发-WI
- 轻松拿下Linux进程、线程和调度
- 一招学会Android自定义控件
- Android底层开发技术实战详解 内核、移植和驱动
- Android驱动开发与移植实战详解
- 30天掌握机器学习升级版
- 从零练就iOS高手
- 8 Android平台开发-WIFI 驱动移植 -- 详细 http://bl...
- Android驱动开发与移植实战详解_13142700.pdf
- Python网络爬虫快速入门实战
- iOS常用三方库、插件、知名技术博客、常
- Android平台移植与底层开发
- Android驱动开发与移植实战详解.pdf
- 最适合自学的C++基础知识
- Android平台开发-WIFI 驱动移植 -- 详细-不错

查看评论

暂无评论

该文章已被禁止评论！

* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

关闭