

Welink your smart



ME3630

# WelinkOpen™ SDK 安装开发向导

版本: V1.0

日期: 2018-02-09

LTE 模块



Website: [www.ztewelink.com](http://www.ztewelink.com)

E-mail: [ztewelink@zte.com.cn](mailto:ztewelink@zte.com.cn)

## 修订历史

---

版本	日期	描述
V1.0	2018-02-09	第一次发布

827455031@qq.com 123.156.199.12 2019/7/18 11:50:08

GOSUNCN Confidential

## 目录

修订历史.....	I
1. 概述 .....	3
2. WELINKOPEN™ SDK 安装 .....	4
3. WELINKOPEN™ SDK 开发目录.....	10
4. 客户程序编译 .....	12
5. 制作 UBI 文件系统.....	17
6. 烧录 UBI 文件 .....	22

## 1. 概述

本文档介绍使用 VMware 虚拟机开发可以运行在 ME3630 模块上的应用程序。

WelinkOpen™ SDK 提供交叉编译链、所需的库文件和头文件、API 和 API 调用示例程序，这些 API 可以实现客户所有的需求。所有的内容将以 VM 虚拟机配置文件的形式提供给客户，客户只需要安装 VMware，然后加载 WelinkOpen™ SDK 提供的 VMX 配置文件，加载成功后启动 ZTEWELINK-Ubuntu16.04 操作系统便可以开发。

WelinkOpen™ SDK 的开发环境位于 /opt/welinkopen-x86\_64 目录下，该目录下的 sysroots 目录包含了交叉编译链、库文件、头文件和 API 等。Example 目录包含了 API 的调用示例程序，客户可以参考或直接拷贝所需的代码。示例程序如下：

atcop_client:	用于发送 AT 命令
data_client:	用于数据拨号，可以使用的 ME3630 Linux 获取 IP 地址上网
dm_client:	用于设备管理
gps_client:	用于获取定位信息
nw_client:	用于获取模块的网络注册状态
sim_client:	用于获取 SIM 卡的相关信息
log_client:	用于将客户程序的 LOG 信息打印到 ME3630 模块的日志系统中，编译保存跟踪
voice_client:	用于电话语音操作
sms_client:	用于收发短信
uart_client:	用于 mcu 和 ME3630 Linux 之间的通信

## 2. WELINKOPEN™ SDK 安装

如果需要安装虚拟机，按照如下步骤依次执行。如果是真实的 ubuntu 操作系统，则直接跳到第 7 步。

① WelinkOpen™ SDK 项目将提供虚拟机的相关文件，要使用 WelinkOpen™ SDK 项目虚拟机，首先必须安装 VMware Workstation。



图 2-1 VMware Workstation 主界面

WelinkOpen™ SDK 项目虚拟机的配置文件包括 vmx 和 vmdk 文件。

名称	修改日期	类型	大小
ZTEWELINK-Ubuntu16.04.vmdk	2018/1/24 9:42	VMware 虚拟磁...	1 KB
ZTEWELINK-Ubuntu16.04.vmx	2018/1/25 8:47	VMware 虚拟机...	3 KB
ZTEWELINK-Ubuntu16.04-s001.vmdk	2018/1/25 8:47	VMware 虚拟磁...	2,015,552...
ZTEWELINK-Ubuntu16.04-s002.vmdk	2018/1/25 8:47	VMware 虚拟磁...	1,142,784...
ZTEWELINK-Ubuntu16.04-s003.vmdk	2018/1/25 8:47	VMware 虚拟磁...	681,856 KB
ZTEWELINK-Ubuntu16.04-s004.vmdk	2018/1/25 8:47	VMware 虚拟磁...	1,000,768...
ZTEWELINK-Ubuntu16.04-s005.vmdk	2018/1/25 8:47	VMware 虚拟磁...	382,208 KB
ZTEWELINK-Ubuntu16.04-s006.vmdk	2018/1/19 11:05	VMware 虚拟磁...	64 KB

图 2-2 vmx 和 vmdk 文件

② 打开已安装好的 VMware Workstation，点击 VMware Workstation 主界面的“打开虚拟机”选项。



图 2-3 打开虚拟机选项

③ 选择虚拟机文件目录下的 vmx 文件，点击打开按钮。

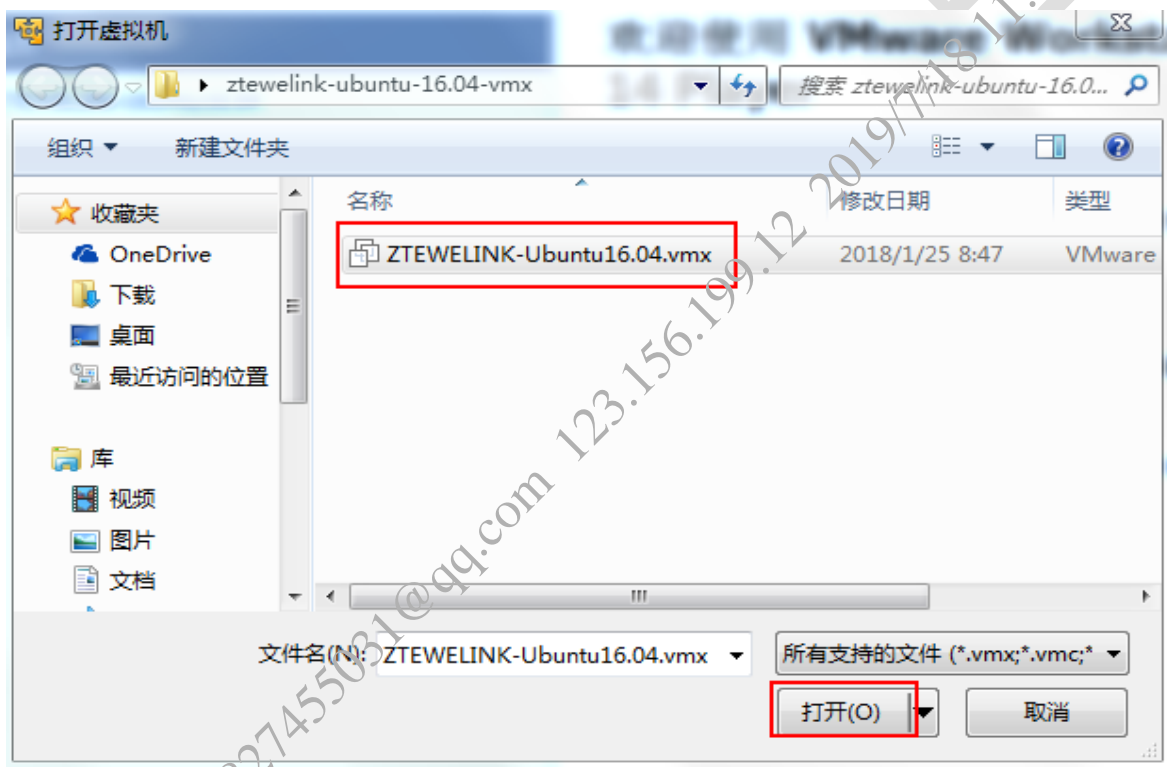


图 2-4 选择 vmx 文件

打开成功后，可看到如下界面。



图 2-5 ZTEWELINK 虚拟机

- ④ 选中已加载成功的虚拟机“ZTEWELINK-Ubuntu16.04”,右键选择“开机”。



图 2-6 打开 ZTEWELINK 虚拟机

- ⑤ 如果弹出如下界面，请选择“我已复制该虚拟机”。

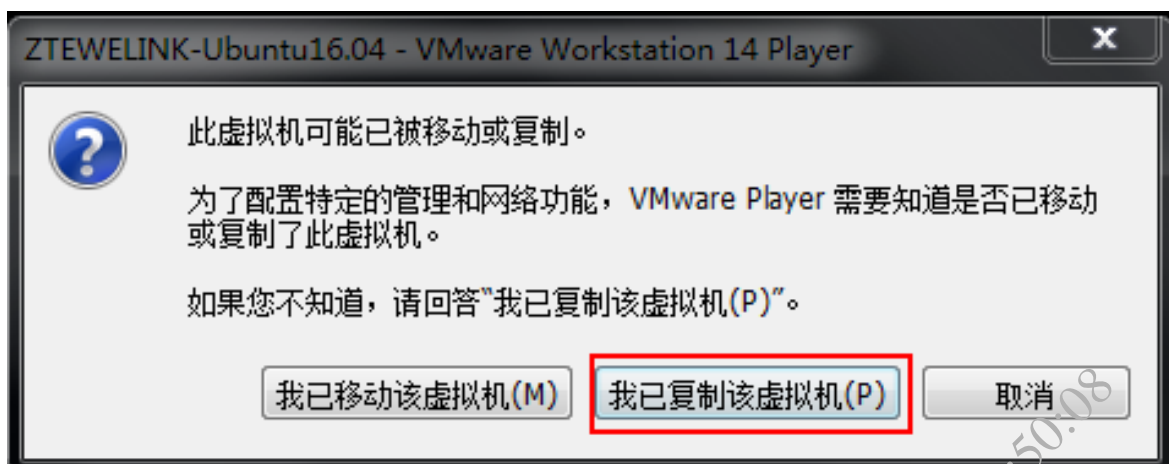


图 2-7 虚拟机开机弹窗

⑥ 登陆虚拟机。选择虚拟机“ztewelink”，输入密码，按下“Enter”键进入虚拟机。密码请联系张云涛，邮箱 zhangyuntao@gosuncn.com 获取。

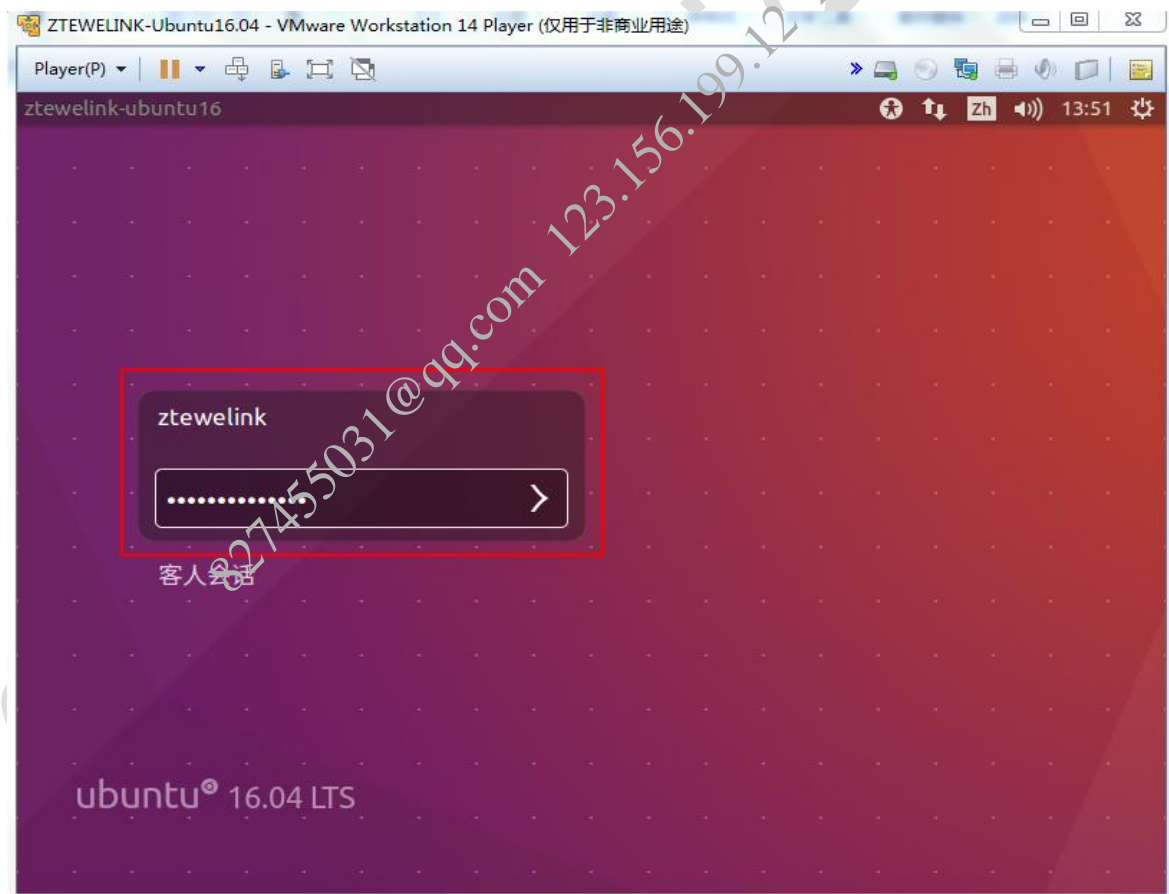


图 2-8 登陆界面

⑦ 查看 WelinkOpen™ SDK 安装包文件。



安装包“welinkopen-x86\_64-armv7a-vfp-neon-toolchain-0.1.sh”如下图所示：

```
ztewelink@ztewelink-ubuntu16: ~/ztewelink_sdk
ztewelink@ztewelink-ubuntu16:~/ztewelink_sdk$
ztewelink@ztewelink-ubuntu16:~/ztewelink_sdk$ pwd
/home/ztewelink/ztewelink_sdk
ztewelink@ztewelink-ubuntu16:~/ztewelink_sdk$ ls
welinkopen-x86_64-armv7a-vfp-neon-toolchain-0.1.sh
ztewelink@ztewelink-ubuntu16:~/ztewelink_sdk$
ztewelink@ztewelink-ubuntu16:~/ztewelink_sdk$
```

图 2-9 安装包文件

⑧ 在 ubuntu 系统安装 WelinkOpen™ SDK 包。直接运行 WelinkOpen™ SDK 安装包文件“welinkopen-x86\_64-armv7a-vfp-neon-toolchain-0.1.sh”。出现“Enter target directory for SDK (default: /opt/welinkopen-x86\_64):”，则表示需要客户输入安装目录，安装文件的默认目录是“/opt/welinkopen-x86\_64”，这里我们选择默认目录，直接按“Enter”按键确认。

```
ztewelink@ztewelink-ubuntu16: ~/ztewelink_sdk
ztewelink@ztewelink-ubuntu16:~/ztewelink_sdk$ cd ztewelink_sdk/
ztewelink@ztewelink-ubuntu16:~/ztewelink_sdk$ ls
welinkopen-x86_64-armv7a-vfp-neon-toolchain-0.1.sh
ztewelink@ztewelink-ubuntu16:~/ztewelink_sdk$ ./welinkopen-x86_64-armv7a-vfp-neon-toolchain-0.1.sh
Enter target directory for SDK (default: /opt/welinkopen-x86_64):
```

图 2-10 SDK 路径

⑨ 出现“You are about to install the SDK to "/opt/welinkopen-x86\_64". Proceed[Y/n]?”，在后面输入“y”，然后按“Enter”按键确认。

```
ztewelink@ztewelink-ubuntu16: ~/ztewelink_sdk
ztewelink@ztewelink-ubuntu16:~/ztewelink_sdk$ cd ztewelink_sdk/
ztewelink@ztewelink-ubuntu16:~/ztewelink_sdk$ ls
welinkopen-x86_64-armv7a-vfp-neon-toolchain-0.1.sh
ztewelink@ztewelink-ubuntu16:~/ztewelink_sdk$ ./welinkopen-x86_64-armv7a-vfp-neon-toolchain-0.1.sh
Enter target directory for SDK (default: /opt/welinkopen-x86_64):
You are about to install the SDK to "/opt/welinkopen-x86_64". Proceed[Y/n]y
```

图 2-11 安装 SDK

- ⑩ 如果出现输入管理员密码，请输入密码，然后按“Enter”键安装。

```
ztewelink@ztewelink-ubuntu16: ~/ztewelink_sdk
ztewelink@ztewelink-ubuntu16:~/ztewelink_sdk$ ls
welinkopen-x86_64-armv7a-vfp-neon-toolchain-0.1.sh
ztewelink@ztewelink-ubuntu16:~/ztewelink_sdk$
ztewelink@ztewelink-ubuntu16:~/ztewelink_sdk$ ./welinkopen-x86_64-armv7a-vfp-neo
n-toolchain-0.1.sh
Enter target directory for SDK (default: /opt/welinkopen-x86_64):
You are about to install the SDK to "/opt/welinkopen-x86_64". Proceed[Y/n]?y
[sudo] ztewelink 的密码: 
```

图 2-12 管理员密码

安装成功后，出现“SDK has been successfully set up and is ready to be used.”字符串。

```
ztewelink@ztewelink-ubuntu16: ~/ztewelink_sdk
ztewelink@ztewelink-ubuntu16:~/ztewelink_sdk x ztewelink@ztewelink-ubuntu16:~/atcop_client x
5155 秒之后
tar: ./sysroots/x86_64-oesdk-linux: 时间戳 2018-01-25 19:26:39 是未来的 16974.445967537
秒之后
tar: ./sysroots: 时间戳 2018-01-25 19:26:39 是未来的 16974.445960766 秒之后
tar: ./: 时间戳 2018-01-25 19:36:35 是未来的 17570.44595418 秒之后
done
Setting it up...done
SDK has been successfully set up and is ready to be used.
ztewelink@ztewelink-ubuntu16:~/ztewelink_sdk$
ztewelink@ztewelink-ubuntu16:~/ztewelink_sdk$
ztewelink@ztewelink-ubuntu16:~/ztewelink_sdk$
ztewelink@ztewelink-ubuntu16:~/ztewelink_sdk$
ztewelink@ztewelink-ubuntu16:~/ztewelink_sdk$
```

图 2-13 安装 SDK 成功

到此 WelinkOpen™ SDK 安装完成，可直接开发程序。

### 3. WELINKOPEN™ SDK 开发目录

① 打开一个终端，进入到“/opt”目录。opt 目录下的“welinkopen-x86\_64”目录，就是 Openlinux 项目的编译环境和代码。

```

ztewelink@ztewelink-ubuntu16: /opt
ztewelink@ztewelink-ubuntu16:~$
ztewelink@ztewelink-ubuntu16:~$ cd /opt/
ztewelink@ztewelink-ubuntu16:/opt$ ls
welinkopen-x86_64
ztewelink@ztewelink-ubuntu16:/opt$
ztewelink@ztewelink-ubuntu16:/opt$ pwd
/opt
ztewelink@ztewelink-ubuntu16:/opt$
ztewelink@ztewelink-ubuntu16:/opt$

```

图 3-1 代码目录

② 进入到“/opt/welinkopen-x86\_64”目录：

“sysroots”目录是 Openlinux 项目的交叉编译环境。

“oem\_start.sh”用于启动客户程序，必须存放在/oemapp/etc/目录下。

“ubi\_tools”是 oemapp 和 oemdata 分区的 ubi 文件制作工具。

“examples”目录是 atcop、data、gps、nw、dm、sim、sms、voice 客户端的示例程序。

“examples”里面的示例程序展示 API 的使用方法，客户可以参考或拷贝。

“examples”下面的文件目录和功能见下表：

Example 下目录名称	目录功能
atcop_client	用于发送 AT 命令
data_client	用于数据拨号，可以使的 ME3630 Linux 获取 IP 地址上网
dm_client	用于设备管理
gps_client	用于获取定位信息
nw_client	用于获取模块的网络注册状态
sim_client	用于获取 SIM 卡的相关信息
log_client	用于将客户程序的 LOG 信息打印到 ME3630 模块的日志系统中，编译保存跟踪
voice_client	用于电话语音操作
Sms_client	用于收发短信

表 1 目录功能

```

ztewelink@ztewelink-ubuntu16: /opt/welinkopen-x86_64
ztewelink@ztewelink-ubuntu16: /opt/weli... x  ztewelink@ztewelink-ubuntu16: ~/oemap... x
ztewelink@ztewelink-ubuntu16:/opt/welinkopen-x86_64$
ztewelink@ztewelink-ubuntu16:/opt/welinkopen-x86_64$ pwd
/opt/welinkopen-x86_64
ztewelink@ztewelink-ubuntu16:/opt/welinkopen-x86_64$ ls
environment-setup-armv7a-vfp-neon-oe-linux-gnueabi
examples
oem_start.sh
site-config-armv7a-vfp-neon-oe-linux-gnueabi
sysroots
lib_tools
version-armv7a-vfp-neon-oe-linux-gnueabi
ztewelink@ztewelink-ubuntu16:/opt/welinkopen-x86_64$
ztewelink@ztewelink-ubuntu16:/opt/welinkopen-x86_64$

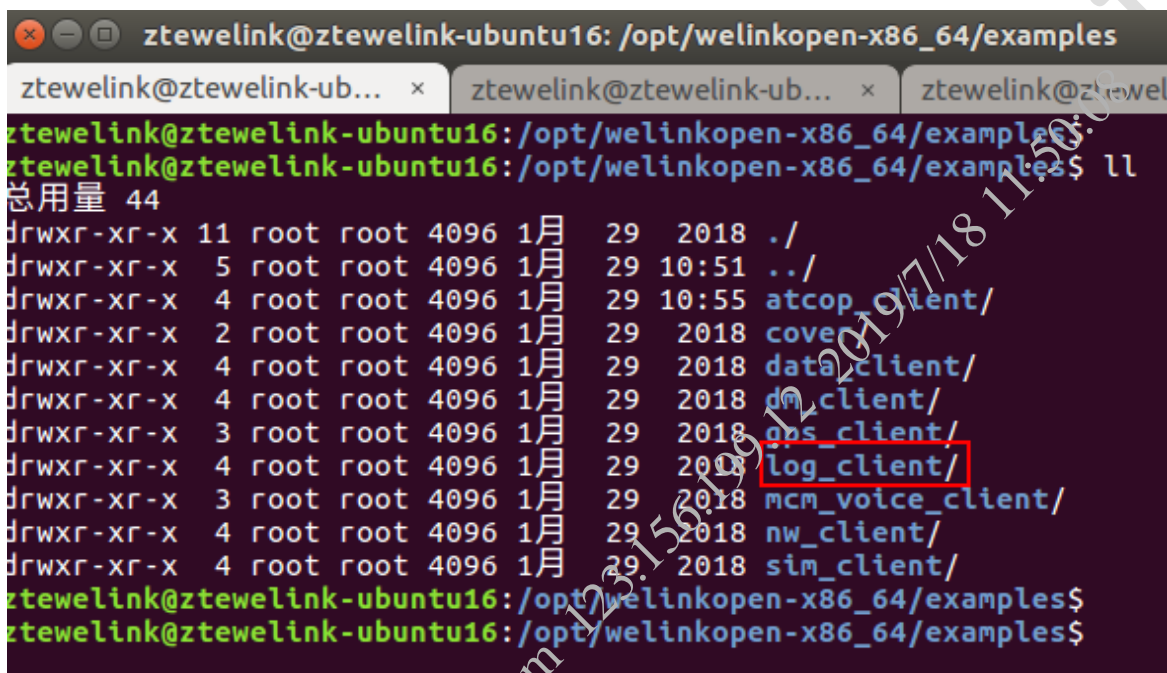
```

图 3-2 编译环境

## 4. 客户程序编译

下面以 log\_test 可执行程序编译为例。

①将“/opt/welinkopen-x86\_64/examples”下的“log\_client”目录，拷贝一份到“/home/zetwelink”目录下面。拷贝完成后，将 log\_client 目录改为 log\_test。详细见图 4-1 到图 4-3。

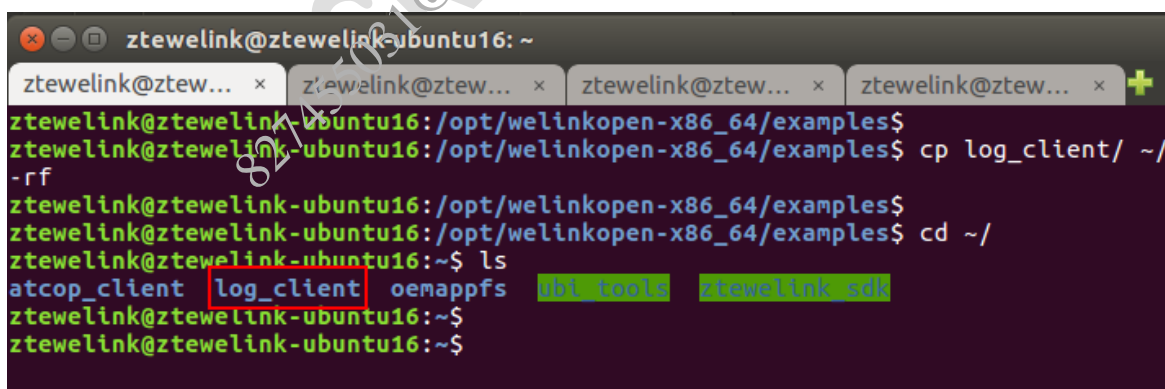


```

ztewelink@ztewelink-ubuntu16: /opt/welinkopen-x86_64/examples
ztewelink@ztewelink-ubuntu16: /opt/welinkopen-x86_64/examples$ ll
总用量 44
drwxr-xr-x 11 root root 4096 1月 29 2018 ./
drwxr-xr-x  5 root root 4096 1月 29 10:51 ../
drwxr-xr-x  4 root root 4096 1月 29 10:55 atcop_client/
drwxr-xr-x  2 root root 4096 1月 29 2018 cover
drwxr-xr-x  4 root root 4096 1月 29 2018 data_client/
drwxr-xr-x  4 root root 4096 1月 29 2018 dm_client/
drwxr-xr-x  3 root root 4096 1月 29 2018 gps_client/
drwxr-xr-x  4 root root 4096 1月 29 2018 log_client/
drwxr-xr-x  3 root root 4096 1月 29 2018 mcm_voice_client/
drwxr-xr-x  4 root root 4096 1月 29 2018 nw_client/
drwxr-xr-x  4 root root 4096 1月 29 2018 sim_client/
ztewelink@ztewelink-ubuntu16: /opt/welinkopen-x86_64/examples$
ztewelink@ztewelink-ubuntu16: /opt/welinkopen-x86_64/examples$

```

图 4-1 log\_client 源码



```

ztewelink@ztewelink-ubuntu16: ~
ztewelink@ztewelink-ubuntu16: /opt/welinkopen-x86_64/examples$ cp log_client/ ~/
-rf
ztewelink@ztewelink-ubuntu16: /opt/welinkopen-x86_64/examples$ cd ~/
ztewelink@ztewelink-ubuntu16: ~$ ls
atcop_client log_client oemappfs ubi tools zetwelink sdk
ztewelink@ztewelink-ubuntu16: ~$
ztewelink@ztewelink-ubuntu16: ~$

```

图 4-2 拷贝 log\_client 目录



```
ztewelink@ztewelink-ubuntu16: ~
ztewelink@ztew... x ztewelink@ztew... x ztewelink@ztew... x ztewelin
ztewelink@ztewelink-ubuntu16:~$ ls
atcop_client log_client oemappfs ubi_tools ztewelink_sdk
ztewelink@ztewelink-ubuntu16:~$ mv log_client/ log_test
ztewelink@ztewelink-ubuntu16:~$ ls
atcop_client log_test oemappfs ubi_tools ztewelink_sdk
ztewelink@ztewelink-ubuntu16:~$
```

图 4-3 log\_test 目录

② log\_test 目录内容如下：

“mcm\_log\_client.c”是 log\_test 客户端的源代码。

“build”是编译之后，生成 log\_test 二进制文件的目录。

“Makefile”是 log\_test 编译规则。

```
ztewelink@ztewelink-ubuntu16:~$
ztewelink@ztewelink-ubuntu16:~$ cd log_test/
ztewelink@ztewelink-ubuntu16:~/log_test$
ztewelink@ztewelink-ubuntu16:~/log_test$ ls
build Makefile mcm_log_client.c
ztewelink@ztewelink-ubuntu16:~/log_test$
ztewelink@ztewelink-ubuntu16:~/log_test$
ztewelink@ztewelink-ubuntu16:~/log_test$
ztewelink@ztewelink-ubuntu16:~/log_test$
```

图 4-4 log\_test 目录

③进入“/home/zetwelink/log\_test”目录，修改 Makefile 中 SDKTARGETSYSROOT、SDKBUILDROOT 所指的路径和 C\_SOURCES、TARGET 表示的源文件和目标文件。

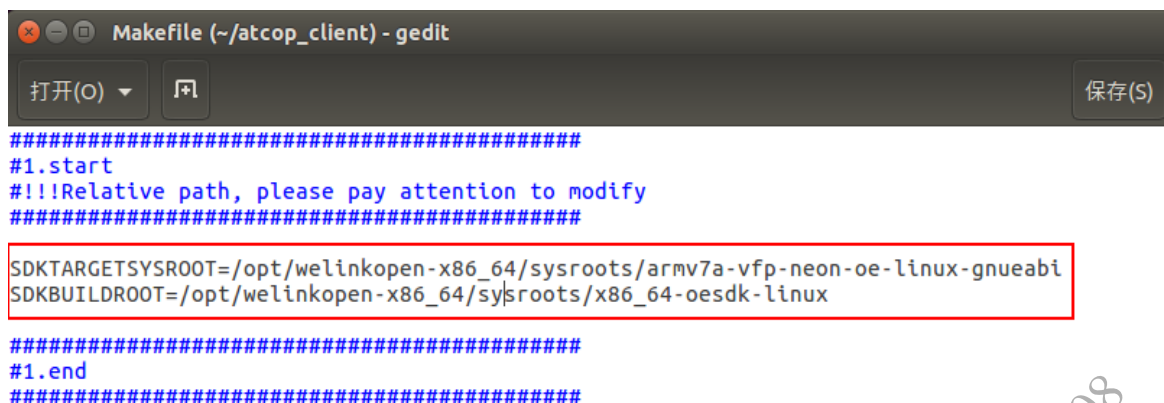
SDKTARGETSYSROOT 必须指向“armv7a-vfp-neon-oe-linux-gnueabi”文件目录。

SDKBUILDROOT 必须指向“x86\_64-oesdk-linux”文件目录。

i. 修改 Makefile 中 SDKTARGETSYSROOT、SDKBUILDROOT 所指的路径。

SDKTARGETSYSROOT=/opt/welinkopen-x86\_64/sysroots/armv7a-vfp-neon-oe-linux-gnueabi;

SDKBUILDROOT=/opt/welinkopen-x86\_64/sysroots/x86\_64-oesdk-linux;



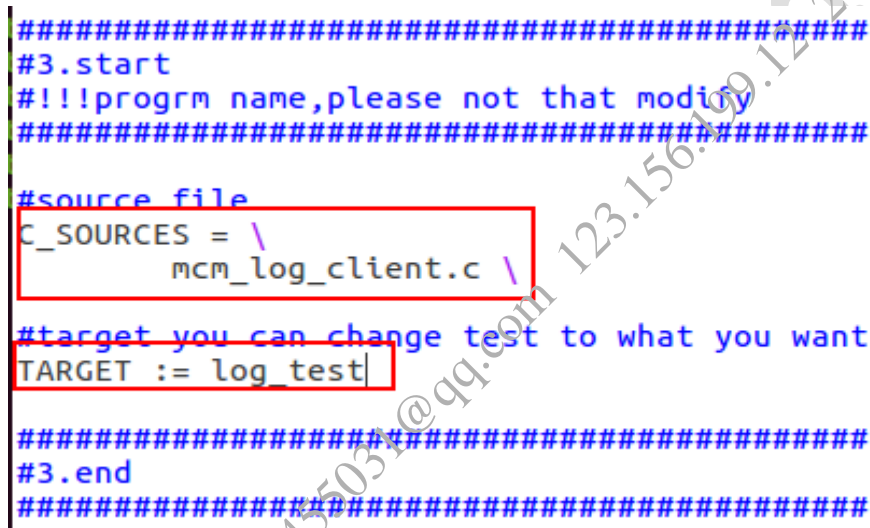
```
#####
#1.start
#!!!Relative path, please pay attention to modify
#####
SDKTARGETSYSROOT=/opt/welinkopen-x86_64/sysroots/armv7a-vfp-neon-oe-linux-gnueabi
SDKBUILDR00T=/opt/welinkopen-x86_64/sysroots/x86_64-oesdk-linux
#####
#1.end
#####
```

图 4-5 两个路径

- ii. 修改 Makefile 中 C\_SOURCES 指向的源文件，TARGET 指向的目标文件名。

C\_SOURCES = mcm\_log\_client.c

TARGET := log\_test



```
#####
#3.start
#!!!progrm name,please not that modify
#####
#source file
C_SOURCES = \
    mcm_log_client.c \
#target you can change test to what you want
TARGET := log_test
#####
#3.end
#####
```

图 4-6 源文件和目标文件

- iii. 修改 mcm\_log\_client.c 的相关代码，当 log\_test 在开机运行之后，每隔 3 秒时间，在 logcat 中打印一句“Hello Word!”。

```
int main
(
    int    argc,
    char **argv
)
{
    char *str_tmp = "Hello Word!";

    while (1)
    {
        LOGI("%s", str_tmp);

        sleep(3);
    }

    return MCM_LOG_SUCCESS;
}
```

图 4-7 mcm\_log\_client.c 文件内容

- ④ 在终端输入“make clean”,清除编译文件。用“ls”查看编译目录“build”是否被删除。

```
ztewelink@ztewelink-ubuntu16:~/log_test$
ztewelink@ztewelink-ubuntu16:~/log_test$ make clean
rm -fR .dep build
ztewelink@ztewelink-ubuntu16:~/log_test$ ls
Makefile  mcm_log_client.c
ztewelink@ztewelink-ubuntu16:~/log_test$
ztewelink@ztewelink-ubuntu16:~/log_test$
ztewelink@ztewelink-ubuntu16:~/log_test$
```

图 4-8 清除界面

- ⑤ 在终端输入“make”,编译 log\_test 可执行程序。

```
ztewelink@ztewelink-ubuntu16:~/log_test$
ztewelink@ztewelink-ubuntu16:~/log_test$ make
mkdir -p build
CC build/mcm_log_client.o
arm-oe-linux-gnueabi-gcc -march=armv7-a -mfloat-abi=soft -mfpu=vfp
t/welinkopen-x86_64/sysroots/armv7a-vfp-neon-oe-linux-gnueabi -c
open-x86_64/sysroots/armv7a-vfp-neon-oe-linux-gnueabi/usr/include
d=c99 -MD -MP -MF .dep/mcm_log_client.o.d -Wa,-a,-ad,-alms=build/m
lst mcm_log_client.c -o build/mcm_log_client.o
LN build/log_test
arm-oe-linux-gnueabi-gcc -march=armv7-a -mfloat-abi=soft -mfpu=vfp
```

图 4-9 编译界面

- ⑥ 如果“make”执行成功,那么编译目录“build”下会编译出二进制文件“log\_test”。如下图。



```
ztewelink@ztewelink-ubuntu16:~/log_test$  
ztewelink@ztewelink-ubuntu16:~/log_test$ cd build/  
ztewelink@ztewelink-ubuntu16:~/log_test/build$  
ztewelink@ztewelink-ubuntu16:~/log_test/build$ ls  
log_test mcm_log_client.lst mcm_log_client.o  
ztewelink@ztewelink-ubuntu16:~/log_test/build$  
ztewelink@ztewelink-ubuntu16:~/log_test/build$  
ztewelink@ztewelink-ubuntu16:~/log_test/build$
```

图 4-10 编译生成的文件

827455031@qq.com 123.156.199.12 2019/7/18 11:50:08

GOSUNCN Confidential

## 5. 制作 UBI 文件系统

① ME3630 将提供两个分区给客户：“oemapp”和“oemdata”。其中，“oemapp”是只读分区。

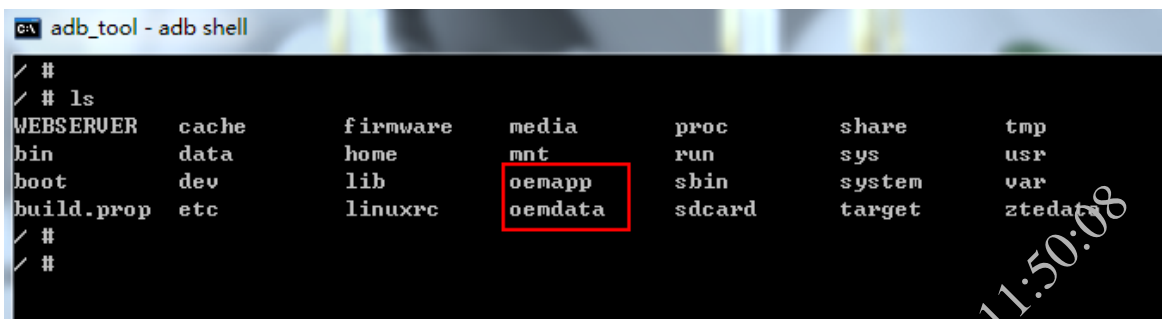


图 5-1 两个分区文件

i. “oemapp”分区文件系统，目前默认有三个文件目录“bin、lib、etc”。客户可以重新分配“oemapp”分区的文件系统，但 oemapp 下的“etc”目录必须存在。

“bin” 存放可执行文件。

“etc” 存放配置文件。此目录必须存在，目录里面的“oem\_start.sh”用于启动客户程序，模块开机时会调用该脚本。

“lib” 存放库文件。



图 5-2 oemapp 分区文件系统

ii. “oemdata”分区主要用于存储客户的过程文件。

② 在“/home/zteWelink”下创建 oemapp 分区对应的源目录 oemappfs 和 oemappfs 目录下的 3 个子目录“etc、lib、bin”。

```
ztewelink@ztewelink-ubuntu16:~$  
ztewelink@ztewelink-ubuntu16:~$ mkdir oemappfs  
ztewelink@ztewelink-ubuntu16:~$  
ztewelink@ztewelink-ubuntu16:~$ cd oemappfs/  
ztewelink@ztewelink-ubuntu16:~/oemappfs$  
ztewelink@ztewelink-ubuntu16:~/oemappfs$ mkdir ./{bin,etc,lib}  
ztewelink@ztewelink-ubuntu16:~/oemappfs$ ls  
bin  etc  lib  
ztewelink@ztewelink-ubuntu16:~/oemappfs$
```

图 5-3 oemappfs 目录

- ③ 将编译好的 log\_test 程序，拷贝到“/home/ztewelink/oemappfs/bin”目录下。

```
ztewelink@ztewelink-ubuntu16:~/oemappfs/bin$  
ztewelink@ztewelink-ubuntu16:~/oemappfs/bin$ pwd  
/home/ztewelink/oemappfs/bin  
ztewelink@ztewelink-ubuntu16:~/oemappfs/bin$  
ztewelink@ztewelink-ubuntu16:~/oemappfs/bin$ cp /home/ztewelink/log_test/build/l  
og_test ./  
ztewelink@ztewelink-ubuntu16:~/oemappfs/bin$  
ztewelink@ztewelink-ubuntu16:~/oemappfs/bin$ ls  
log_test  
ztewelink@ztewelink-ubuntu16:~/oemappfs/bin$  
ztewelink@ztewelink-ubuntu16:~/oemappfs/bin$
```

图 5-4 oemappfs 的 bin 目录

- ④ 将客户程序的启动脚本 oem\_start.sh 拷贝到“/home/ztewelink/oemappfs/etc”目录下。oem\_start.sh 的内容默认为空，需要客户手动添加。

```
ztewelink@ztewelink-ubuntu16:~/oemappfs/etc$  
ztewelink@ztewelink-ubuntu16:~/oemappfs/etc$ pwd  
/home/ztewelink/oemappfs/etc  
ztewelink@ztewelink-ubuntu16:~/oemappfs/etc$  
ztewelink@ztewelink-ubuntu16:~/oemappfs/etc$ cp /opt/welinkopen-x86_64/oem_start  
.sh ./  
ztewelink@ztewelink-ubuntu16:~/oemappfs/etc$ ls  
oem_start.sh  
ztewelink@ztewelink-ubuntu16:~/oemappfs/etc$  
ztewelink@ztewelink-ubuntu16:~/oemappfs/etc$
```

图 5-5 oemappfs 的 etc 目录

- ⑤ 将“log\_test”可执行程序，添加在开机启动脚本列表中。修改启动脚本“/home/ztewelink/oemappfs/etc/oem\_start.sh”。



图 5-6 oemappfs 的 etc 目录

⑥“oemapp”和“oemdata”ubi 文件系统的制作方法。

i. ubi 文件系统制作工具在“/opt/welinkopen-x86\_64/ubi\_tools/”目录下。

“make\_ubi.sh ” 用于生成“oemapp”和“oemdata”分区的 ubi 文件。

“oemapp\_output” 存储“oemapp”分区 ubi 文件。

“oemdata\_output” 存储“oemdata”分区 ubi 文件。

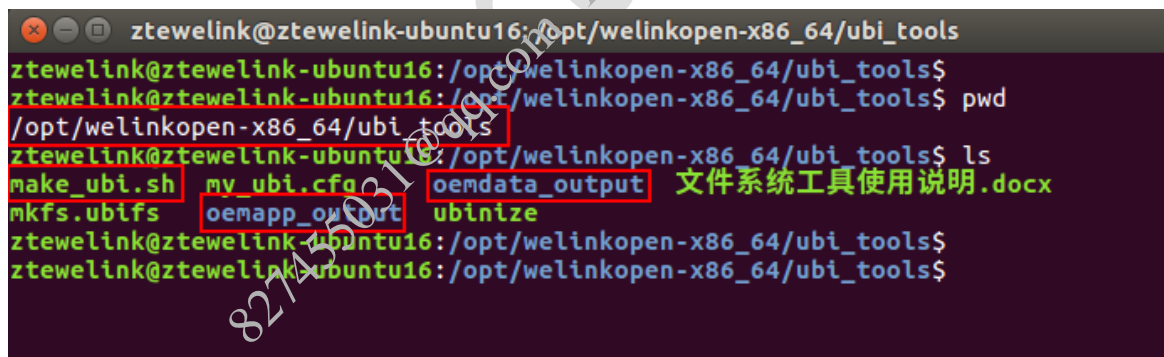


图 5-7 制作 ubi 文件目录

ii.“make\_ubi.sh”脚本用于自动生成“oemapp”和“oemdata”分区 ubi 文件。

“OEMAPP\_INPUT\_FILE” 指定生成 oemapp 分区文件系统的源目录，默认为空。

“OEMLDATA\_INPUT\_FILE” 指定生成 oemdata 分区文件系统的源目录，默认为空。

当“OEMAPP\_INPUT\_FILE”或“OEMLDATA\_INPUT\_FILE”变量的值为空时，不生成对应的 ubi 文件。

```
make_ubi.sh (/opt/welinkopen-x86_64/ubi_tools) - gedit
打开(O)  [icon]
#!/bin/bash
CUR_PATH=`pwd`

#### oemdata ####
OEMDATA_INPUT_FILE=
OEMDATA_OUTPUT_FILE_PATH="${CUR_PATH}/oemdata_output"
OEMDATA_OUTPUT_FILE_UBIFS="${OEMDATA_OUTPUT_FILE_PATH}/oem-datafs.ubifs"
OEMDATA_OUTPUT_FILE_FINAL_UBI="${OEMDATA_OUTPUT_FILE_PATH}/oem-datafs.ubi"
OEMDATA_UBINIZE_CFG="${OEMDATA_OUTPUT_FILE_PATH}/ubinize_oem-data.cfg"

MK_OEMDATA_UBIFS_ARGS="-m 2048 -e 126976 -c 280 -F"
MK_OEMDATA_UBINIZE_ARGS="-m 2048 -p 128KiB -s 2048 -c 280"
OEMDATA_VOLUME_SIZE="27MiB"
#--- oemdata ---

#### oemapp ####
OEMAPP_INPUT_FILE=
OEMAPP_OUTPUT_FILE_PATH="${CUR_PATH}/oemapp_output"
OEMAPP_OUTPUT_FILE_UBIFS="${OEMAPP_OUTPUT_FILE_PATH}/oem-appfs.ubifs"
OEMAPP_OUTPUT_FILE_FINAL_UBI="${OEMAPP_OUTPUT_FILE_PATH}/oem-appfs.ubi"
OEMAPP_UBINIZE_CFG="${OEMAPP_OUTPUT_FILE_PATH}/ubinize_oem-app.cfg"

MK_OEMAPP_UBIFS_ARGS="-m 2048 -e 126976 -c 680 -F"
MK_OEMAPP_UBINIZE_ARGS="-m 2048 -p 128KiB -s 2048 -c 680"
OEMAPP_VOLUME_SIZE="77MiB"
#--- oemapp ---
```

图 5-8 制作 ubi 的源文件路径

iii. 在“make\_ubi.sh”里面，添加“oemapp”和“oemdata”分区的源目录。

如果只添加“oemapp”分区的源目录，则只生成“oemapp”分区的 ubi 文件。

如果只添加“oemdata”分区的源目录，则只生成“oemdata”分区的 ubi 文件。

这里我们只添加“oemapp”分区的源目录

```
#### oemapp ####
OEMAPP_INPUT_FILE="/home/zte/welink/oemappfs"
OEMAPP_OUTPUT_FILE_PATH="${CUR_PATH}/oemapp_output"
OEMAPP_OUTPUT_FILE_UBIFS="${OEMAPP_OUTPUT_FILE_PATH}/oem-appfs.ubifs"
OEMAPP_OUTPUT_FILE_FINAL_UBI="${OEMAPP_OUTPUT_FILE_PATH}/oem-appfs.ubi"
OEMAPP_UBINIZE_CFG="${OEMAPP_OUTPUT_FILE_PATH}/ubinize_oem-app.cfg"

MK_OEMAPP_UBIFS_ARGS="-m 2048 -e 126976 -c 680 -F"
MK_OEMAPP_UBINIZE_ARGS="-m 2048 -p 128KiB -s 2048 -c 680"
OEMAPP_VOLUME_SIZE="77MiB"
#--- oemapp ---
```

图 5-9 oemapp ubi 依赖的源文件路径

iv. 执行“make\_ubi.sh”脚本，生成“oemapp”分区的 ubi 文件“oem-appfs.ubi”，可以进入“oemapp\_output”目录查看。

```

ztewelink@ztewelink-ubuntu16: /opt/welinkopen-x86_64/ubi_tools/oemapp_output
ztewelink@ztewelink-ubuntu16: /opt/weli... x ztewelink@ztewelink-ubuntu16: ~ x +
ztewelink@ztewelink-ubuntu16:/opt/welinkopen-x86_64/ubi_tools$ ls
make_ubi.sh my_ubi.cfg oemdata_output 文件系统工具使用说明.docx
mkfs.ubifs oemapp_output ubinize
ztewelink@ztewelink-ubuntu16:/opt/welinkopen-x86_64/ubi_tools$ cd oemapp_output/
ztewelink@ztewelink-ubuntu16:/opt/welinkopen-x86_64/ubi_tools/oemapp_output$ ls
ztewelink@ztewelink-ubuntu16:/opt/welinkopen-x86_64/ubi_tools/oemapp_output$ cd
../
ztewelink@ztewelink-ubuntu16:/opt/welinkopen-x86_64/ubi_tools$ ./make_ubi.sh
ztewelink@ztewelink-ubuntu16:/opt/welinkopen-x86_64/ubi_tools$
ztewelink@ztewelink-ubuntu16:/opt/welinkopen-x86_64/ubi_tools$ cd oemapp_output/
ztewelink@ztewelink-ubuntu16:/opt/welinkopen-x86_64/ubi_tools/oemapp_output$ ls
oem-appfs.ubi oem-appfs.ubifs ubinize_oem-app.cfg
ztewelink@ztewelink-ubuntu16:/opt/welinkopen-x86_64/ubi_tools/oemapp_output$
ztewelink@ztewelink-ubuntu16:/opt/welinkopen-x86_64/ubi_tools/oemapp_output$
ztewelink@ztewelink-ubuntu16:/opt/welinkopen-x86_64/ubi_tools/oemapp_output$
ztewelink@ztewelink-ubuntu16:/opt/welinkopen-x86_64/ubi_tools/oemapp_output$

```

图 5-10 生成 oemapp ubi 文件

## 6. 烧录 UBI 文件

- ①. 将“oemapp”分区的“oemapps-appfs.ubi”文件取出，放到 windows 电脑的某个位置。这里选择放在“D:\tmp\_file”下面。

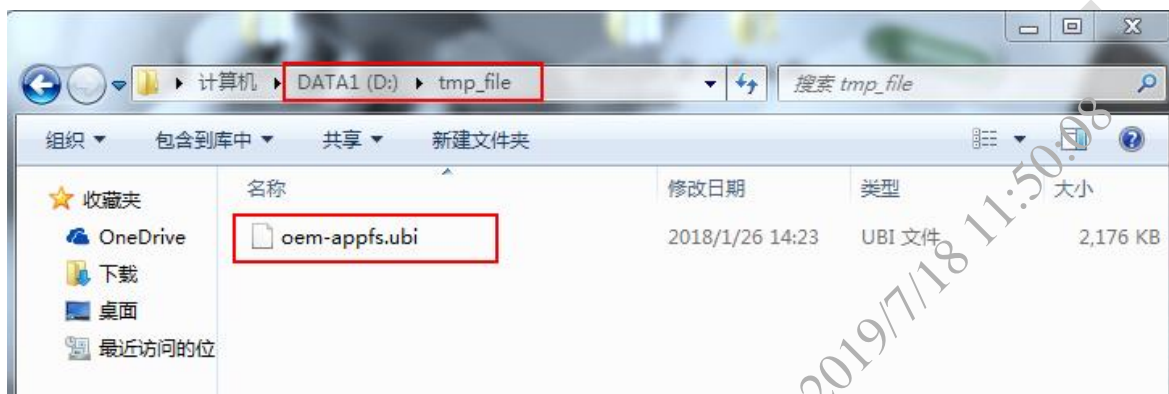


图 6-1 ubi 文件位置

- ②. 使用“adb reboot bootloader”，将 ME3630 切换到 fastboot 模式。

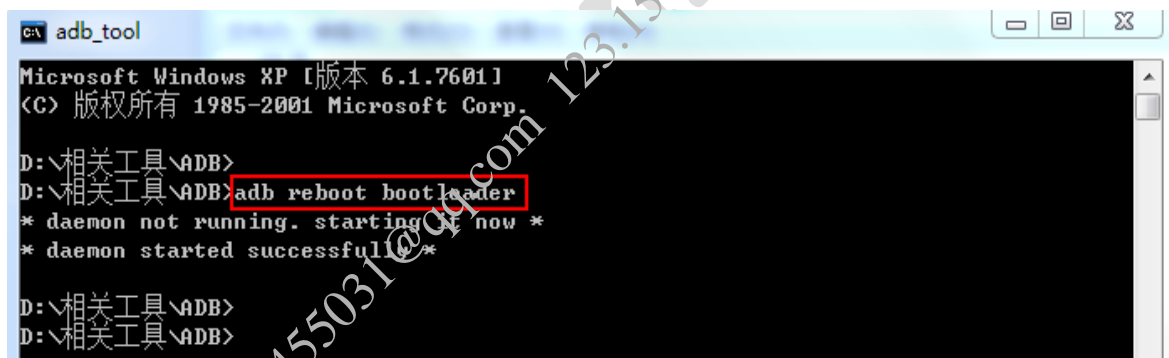


图 6-2 切换 fastboot 模式

- ③. 如果切换 fastboot 模式成功，会出现 ADB 口。





图 6-3 ADB 口

- ④. 使用“fastboot flash oemapp D:\tmp\_file\oem-appfs.ubi”命令，烧录 oemapp 分区的“oem-appfs.ubi”文件。

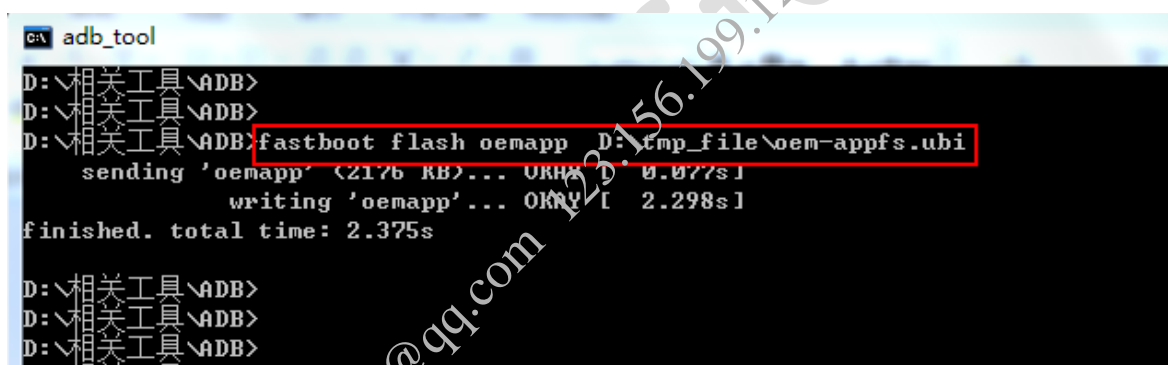


图 6-4 烧录 UBI 文件

- ⑤. 使用“fastboot reboot”重启设备。

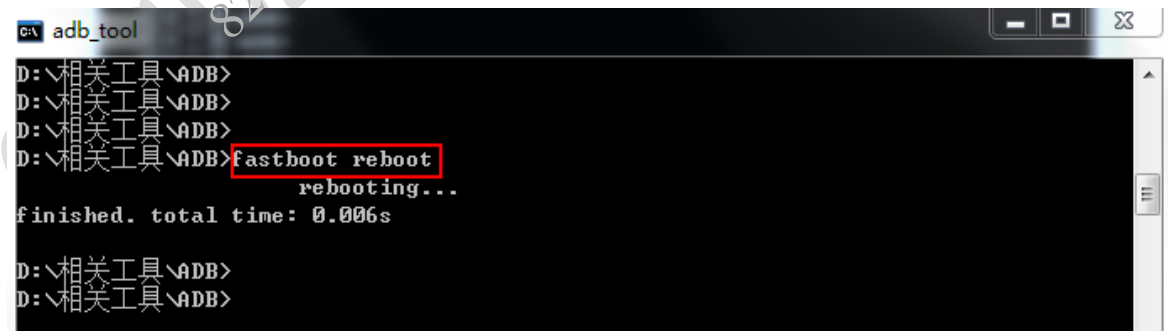


图 6-5 重启设备

- ⑥. 使用“adb shell”进入设备，然后到“/oemapp/bin”目录查看 log\_test 的位置是否正确。



```
C:\> adb_tool - adb shell

/ #
/ #
/ # cd /oemapp/bin/
/oemapp/bin #
/oemapp/bin # ls
log_test
/oemapp/bin #
/oemapp/bin # cd /oemapp/etc/
/oemapp/etc #
/oemapp/etc # ls
oem_start.sh
/oemapp/etc #
/oemapp/etc #
/oemapp/etc #
```

图 6-6 烧录的 log\_test 客户端

- ⑦. 使用“ps”命令，查看开机之后，log\_test 程序是否已经自启动。

```
C:\> adb_tool - adb shell

/ #
/ # ps | grep log
518 root      0:00 /sbin/syslogd -n -C64
521 root      0:00 /sbin/klogd -n
839 root      0:00 /oemapp/bin/log_test
1217 root     0:00 grep log
/ #
/ #
/ #
```

图 6-7 log\_test 程序

- ⑧. 使用“logcat”命令查看 log\_test 程序 log 输出,每隔 3 秒 log\_test 程序打印一句“Hello Word! ”。见下图。

```
C:\> adb_tool - adb shell

/ #
/ #
/ # logcat | grep Hello
I/ZTE_LOG < 839>: Hello Word!
I/ZTE_LOG < 839>: Hello Word!
I/ZTE_LOG < 839>: Hello Word!
I/ZTE_LOG < 839>: Hello Word!
I/ZTE_LOG < 839>: Hello Word!
I/ZTE_LOG < 839>: Hello Word!
```

图 6-8 log\_test 程序

以上通过图文结合的方式，详细说明了 WelinkOpen™ SDK 安装包从安装到开发，再到编译，最后将程序烧录到板子的四个过程。旨在指导客户更好更快地开发项目。