Android 双应用进程工控方案（一）

------Android 开机启动 C/C++应用程序

英创公司

2017 年 11 月

Android 是移动设备的主流操作系统，近年来越来越多的工业领域的客户开始关注基于 Android 操作系统的设备在工控领域的应用。鉴于 Android 是基于 Linux 内核的事实，我们发展了一种以双应用进程为特色的 Android 工控应用方案，并在 ESM6802 工控主板上加以实现。具体说来，就是在 Linux 平台上运行一个直接操作硬件接口的控制通讯管理进程，为保证运行效率，该进程采用 C/C++语言编写（以下简称 C 进程或控制进程）；另一方面在 Android 平台采用标准 Java 语言编写一个人机界面进程（以下简称 Java 进程）。底层的控制进程并不依赖与上层的 Java 进程而独立运行，两个进程之间通过本地 IP 进行通讯，控制进程处于服务器侦听模式，Java 进程则为客户端模式。本方案的主要优点是客户可以直接继承已有的现成应用程序作为底层控制进程的基础，仅仅增加标准的 Socket 侦听功能，即可快速完成新的底层应用程序的设计。而界面的 Java 程序，由于不再涉及具体的工控硬件接口，属于单纯的 Android 程序，编程难度也大大降低。

我们将通过多篇技术报告来具体介绍双进程方案在 ESM6802 主板上实现的相关技术。本文是《Android 双应用进程工控方案》的第一篇，主要介绍在 Android 环境中，如何编译 C/C++应用程序，下载并配置为开机启动程序。

一、重新编译 C/C++应用程序

如图 1 所示，由于传统的 Linux 程序依赖的是 glibc 库，而 Android 程序需要的是谷歌公司在 AOSP（Android Open Source Project）中提供的 Bionic 库（比 glibc 小，提供了 Android 特定的函数）。所以，原来 Linux 上的 C/C++程序要运行在 Android 系统上，必须要在 Android 的编译环境中重新编译。英创推荐使用 Android 官方开发工具 Android Studio，下载 CMake 和 NDK 工具，进行 C/C++程序的重新编译。

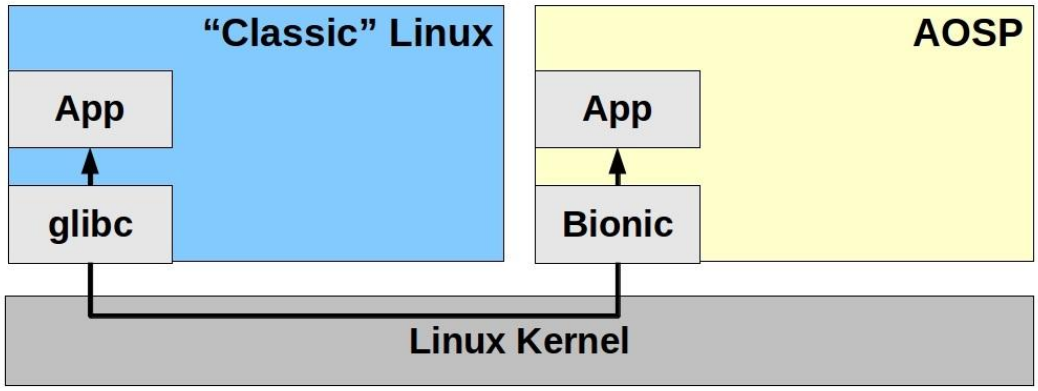

图 1 Android 和 Linux 依赖库区别

下面开始介绍使用 Android Studio 的 NDK 编译工具重新编译 C/C++程序的过程。

1.1 搭建 Android Studio NDK 编译环境

Android Studio 的安装具体过程请参考文档《Android Studio 应用开发简介》的第一章，在 SDK Tools 页面中一定要勾选 NDK 和 CMake。

1.2 在 Android Studio 中新建 C++项目

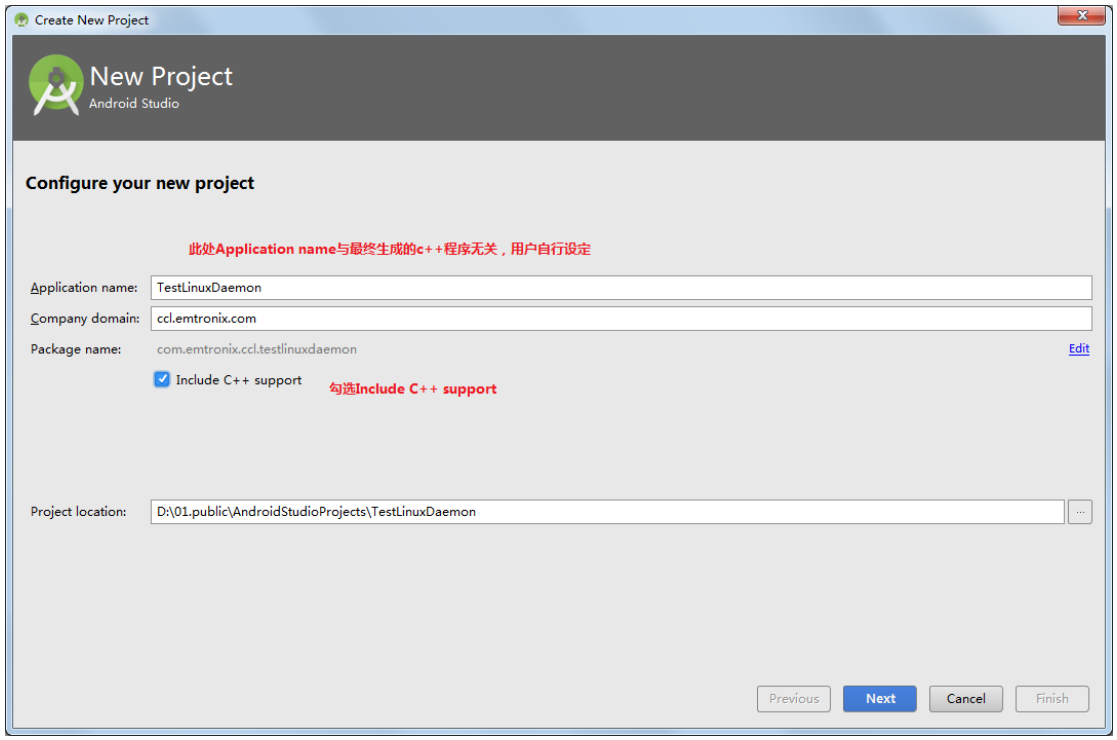

图 2 新建 C++项目

首先新建一个 Android Studio 项目，并勾选 Include C++ support 选项，此处的 Application name 是 Android app 的名字，与最终需要的 C++程序无关，用户可随意设定。然后一直点击下一步“Next”，直到图 3 页面，使用默认的工具链，点击 Finish。

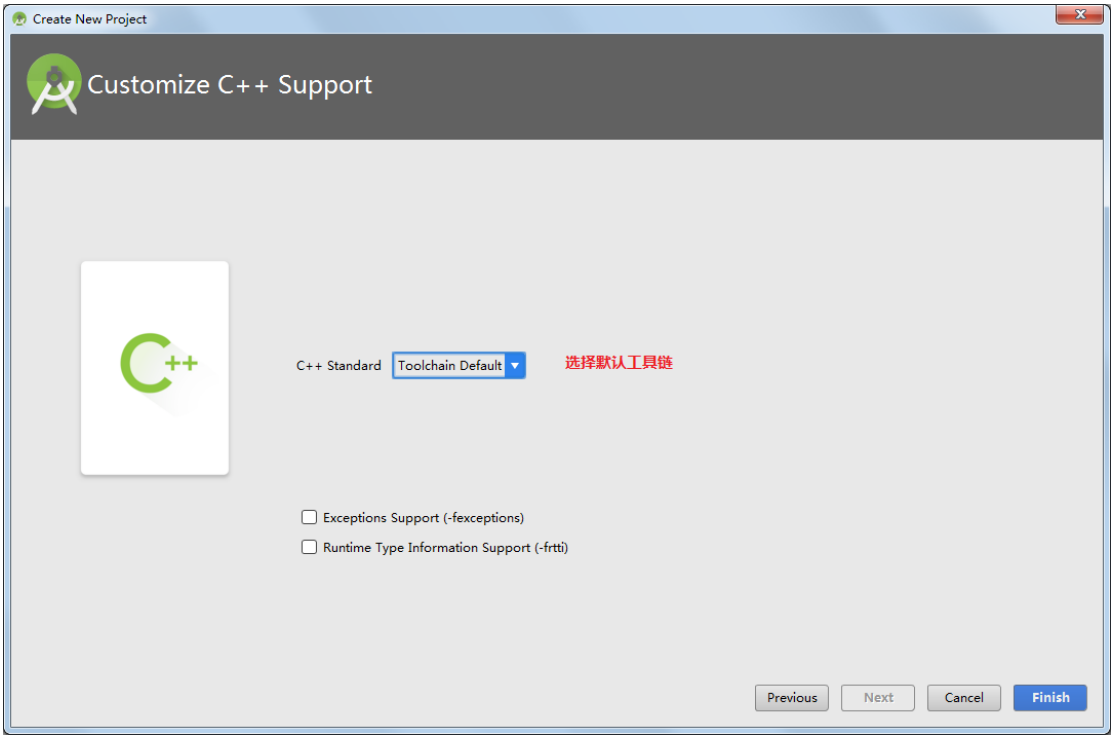

图 3 默认工具链

点击 finish 后会进入项目编辑页面，进入到图 4 所示的项目视图，可以看到所有的目录结构，**其中 app/src/main/cpp 目录、app/build.gradle 和 app/CMakeLists.txt 是用户需要编辑修改的**。然后，点击左上角 File >> Project Structure 进入图 5 的页面，检查 NDK 环境路径是否正确设置。

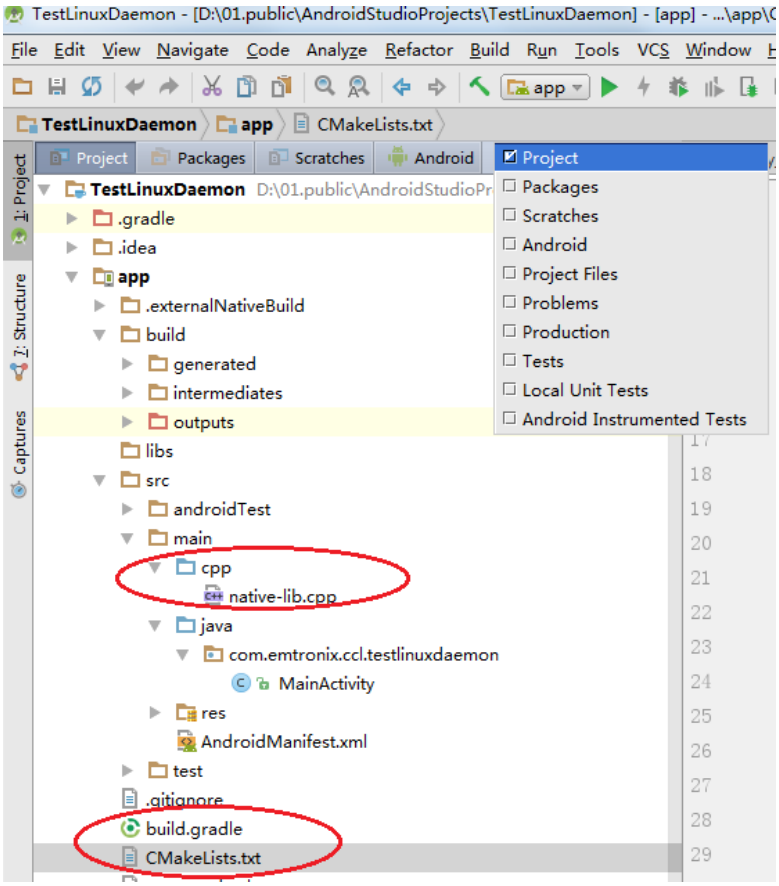

图 4 项目目录结构及要修改的文件

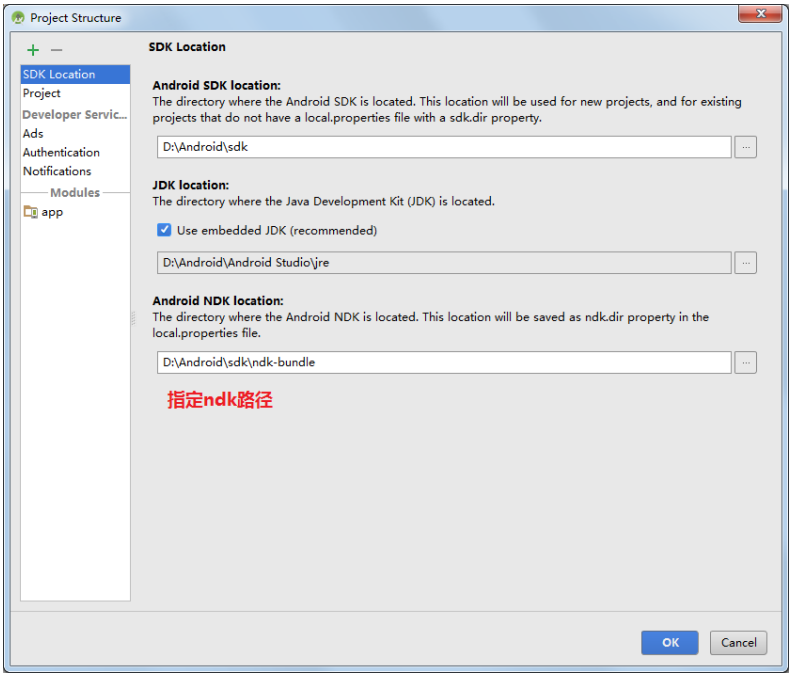

图 5 环境路径设置检查

1.3 复制 C/C++应用程序源码

将原 C/C++应用程序的所有源文件拷贝到 app/src/main/cpp 目录。

1.4 修改 CMakeLists.txt

新的 Android Studio 已经支持使用 cmake 编译 c++项目，这里提供对于简单项目使用的 CMakeLists.txt，对于更复杂的需求，用户可以参考 cmake 官网文档 https://cmake.org/cmake/help/v3.4/ 自行修改。

app/CMakeLists.txt：

```
cmake_minimum_required(VERSION 3.4.1)
# Android 5.0 以上需要在此处设置 PIE
set(CMAKE_C_FLAGS "${CMAKE_C_FLAGS} -fPIE")
set(CMAKE_EXE_LINKER_FLAGS "${CMAKE_EXE_LINKER_FLAGS} -fPIE -pie")
# 头文件目录
include_directories(
src/main/cpp
)
# 源文件目录
aux_source_directory(src/main/cpp DIR_SRCS)
# 添加要编译的可执行文件
add_executable(serialControlDaemon ${DIR_SRCS})
```

其中，add_executable 表明要生成的是可执行文件，名字为 SerialControlDaemon，源文件为 DIR_SRCS 变量代表的文件，而 aux_source_directory 将 src/main/cpp 目录下的所有文件赋给了变量 DIR_SRCS。

1.5 修改 build.gradle

app/build.gradle 文件主要是设置构建 app 的一些参数，这里主要往 android>>defaultConfig>>externalNativeBuild>>cmake 添加 targets 和 abiFilters 两个参数。其中，targets 表示生产目标文件的名字，与 CmakeLists.txt 中的相同；abiFilters 表示要生产哪种 cpu 架构下的目标文件，这里使用 armeabi-v7a。修改之后会在右上角提示需要同步项目，点击即可。

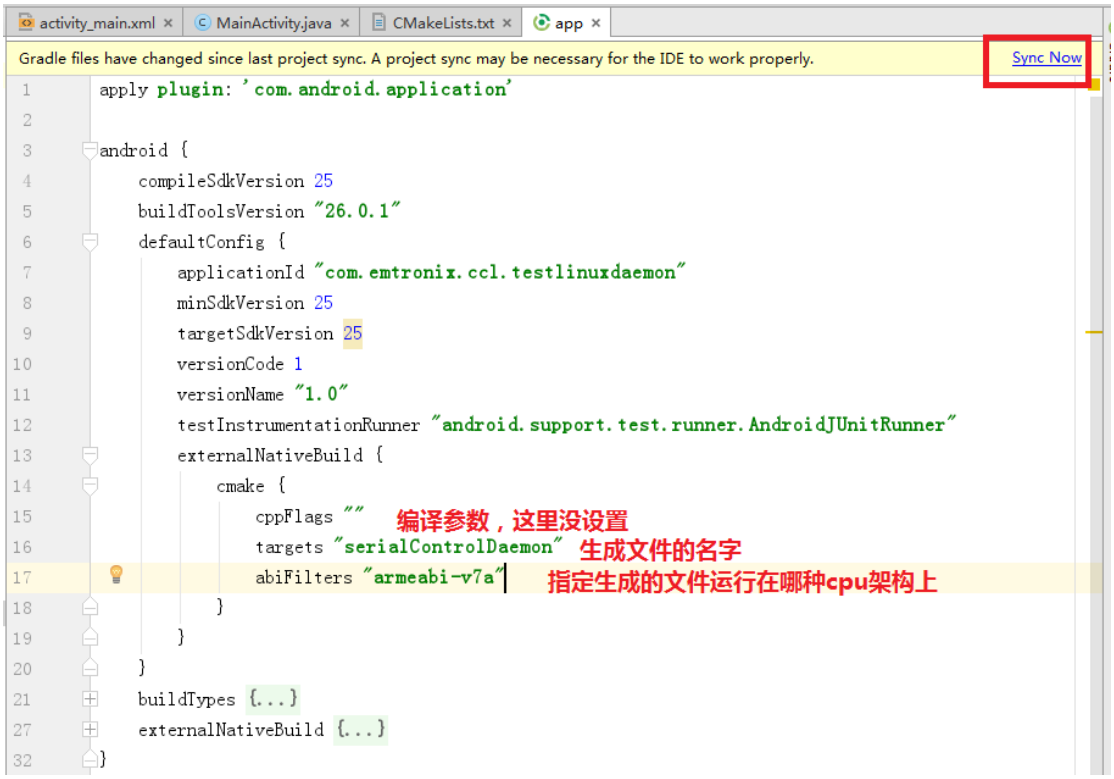

图 6 修改 build.gradle

1.6 编译 cpp 项目

在 Android Studio 中直接使用 Build>>Make Project 即可编译整个项目，包括 cpp 和 java。生成的目标文件在目录 app/.externalNativeBuild/cmake/debug/armeabi-v7a 目录下，名字为 serialControlDaemon。

1.7 下载目标文件到 Android

Android Studio 集成了 Android 开发的所有工具，在 Android Studio 中使用 adb push 命令可以将编译得到的目标文件下载到 Android 目标板上。首先，要使用 usb otg 的调试线连接 PC 和目标板；然后点击左下角的 Terminal 窗口会弹出所在项目的命令行窗口；输入命令：

adb push app\.externalNativeBuild\cmake\debug\armeabi-v7a\serialControlDaemo /data/local

这样，serialControlDaemon 便下载到了目标板的/data/local 目录下。这时，使用 adb shell 登陆到 Android 目标板的命令行，修改目标文件的运行权限并运行，整个过程如图 7 所示。程序正常运行起来后，表明整个编译过程没有问题，用户可以在命令行中按 Ctrl+c 停止运行应用程序，并输入 exit 命令退出 adb shell 登陆，然后进行下一步的开机自启动配置。

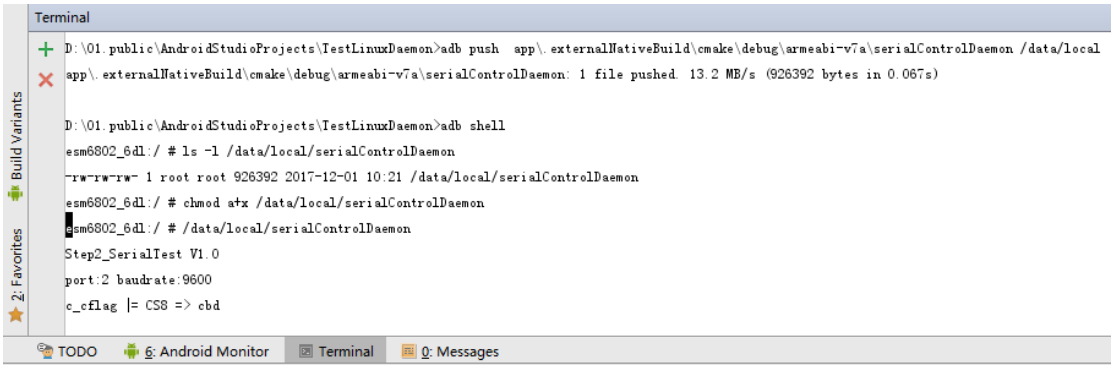

图 7 下载目标文件到 Android

二、开机自启动配置

ESM6802 上电后通过 uboot 引导进入 linux 内核，内核完成一系列系统配置后会启动第一个用户进程：init 进程。Android 相关的启动过程也是从 init 开始的。在 init 进程中会挂载 Android 的文件系统，运行 init.rc 脚本。init 进程启动过后，会 fork 出子进程去开启 init.rc 文件中配置的 service。

为了满足用户运行不同名字的应用程序，英创在 init.rc 中配置了一个 usersh 服务。usersh 服务开机自动运行，具体过程用户不用关心。 要想开机自启动 C/C++程序，用户只需要做两件事：

- 编辑 userinfo.txt 文件
- 复制 userinfo.txt 以及 C/C++程序的目标文件到指定目录/sdcard/Download

2.1 编辑 userinfo.txt

Android 启动后，usersh 服务会自动检测/sdcard/Download/userinfo.txt 文件。如果 userinfo.txt 文件存在，usersh 会去解析并启动 userinfo.txt 文件中指定的应用；如果

userinfo.txt 不存在，则结束 usersh 服务。userinfo.txt 起到一个配置文件的作用，其格式如下：

Name=serialControlDaemon
Param=2

其中，Name 指定程序名字，Param 指定要带的参数，没有可以不写。用户可以直接在 Android Studio 中创建并编辑 userinfo.txt 文件。

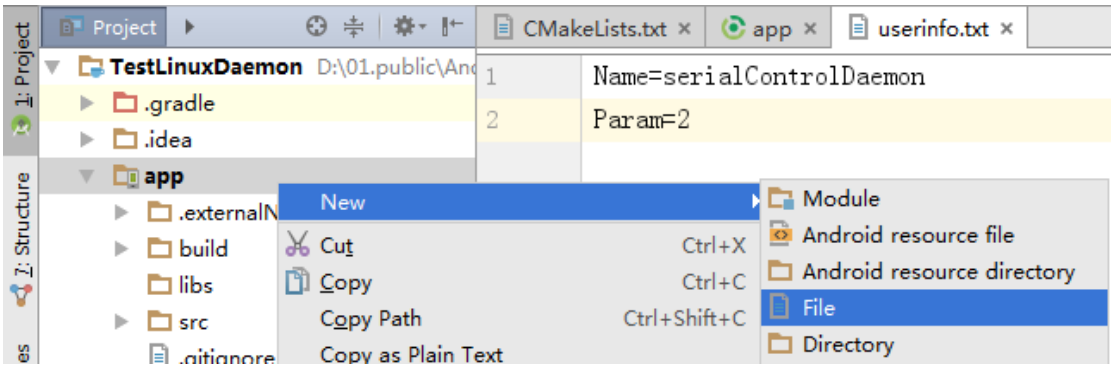

图 8 Android Studio 中新建 userinfo.txt

2.2 复制 userinfo.txt 以及 C/C++程序到指定目录/sdcard/Download

Android 系统中，不是每个目录都有读写以及可执行的权限，这里我们选择/sdcard/Download 作为存储 userinfo.txt 和 C/C++程序的指定目录。复制 userinfo.txt 以及 C/C++程序到指定目录有两种方法：通过 usb_otg 接口使用 Android Studio 的 adb push 命令下载到 ESM6802，或者通过 U 盘从 PC 端拷贝到 ESM6802。用户按其中一种方法下载文件到指定目录后，重启 ESM6802 即可以开机启动 userinfo.txt 中指定的 C/C++程序。

1、Android Studio 命令行下载 userinfo.txt 和 C/C++程序到 ESM6802

使用 Android Studio 命令行下载文件到 ESM6802，首先需要使用调试线连接 PC 和目标板的 usb_otg 接口。然后，在 Android Studio 的 Terminal 窗口输入：

```
adb push app\.externalNativeBuild\cmake\debug\armeabi-v7a\serialControlDaemo /sdcard/Download
adb push app\userinfo.txt /sdcard/Download
```

重启设备即可实现开机自启动 serialControlDaemon。

2、U 盘拷贝 userinfo.txt 和 C/C++程序到 ESM6802

使用 U 盘拷贝 userinfo.txt 和 C/C++程序到 ESM6802，只需要将 userinfo.txt 和目标文件（serialControlDaemon）拷贝到 U 盘，插到 ESM6802 的 USB 接口上，打开 Android 的文件管理应用 ES File Explorer，将 userinfo.txt 和 serialControlDaemon 拷贝到 /sdcard/Download 目录，重新启动即可。

2.3 查看程序是否开机运行

通过以上设置之后，Android 开机 boot_completed=1 之后会启动应用程序 serialControlDaemon，用户可以通过命令 adb shell 登陆 consolo 控制台，输入命令 getprop | grep init.svc | grep usersh 来查看 usersh 服务的运行状态；当然 usersh 实际运行的应用程序 serialControlDaemon 的进程状态可以通过 ps | grep serialControlDaemon 查看。

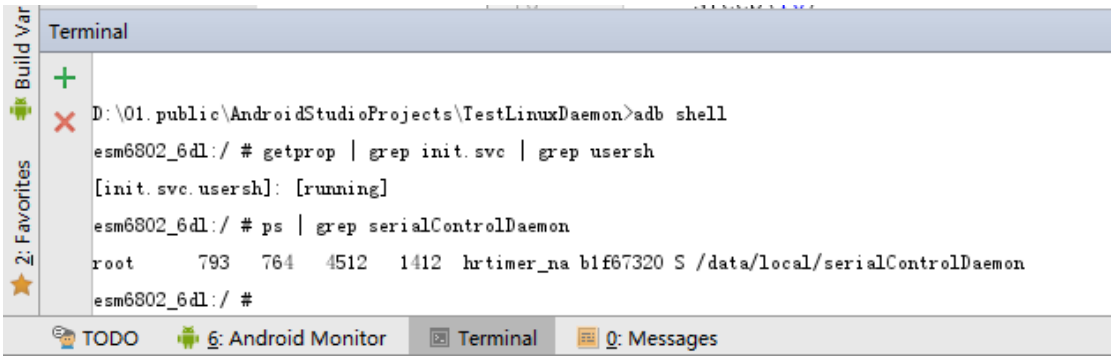

图 9 检测 usersh 服务运行状态

三、Q&A

Q1：查看 C/C++程序输出

在 Android 控制台上看不到开机启动的 C/C++程序输出信息，开发中如何在 Android 上调试 C/C++程序？

A1：使用 kill 命令终止掉已经启动的 C/C++程序；然后，在 Android 命令行中执行命令：user.sh，即可手动启动 C/C++应用程序，并且 C/C++应用程序的输出信息将打印到 Android 控制台。

Q2：关于 userinfo.txt 和 C/C++程序指定目录的说明

A2：userinfo.txt 和 C/C++程序指定目录要具有读写可执行权限，在 2.2 节中，adb push 命令将 C/C++应用程序（serialControlDaemon）下载到了/sdcard/Download 目录，其实下载到/data/local 也是可以的，而 U 盘却只能拷贝到/sdcard/Download/。这是因为 usersh 服务会比较/sdcard/Download/serialControlDaemon 是否比/data/local/serialControlDaemon 更新，如果是，则先用新文件覆盖旧文件，再运行/data/local/serialControlDaemon。因此，使用 adb push 命令的指定目录用/sdcard/Download/或者/data/local 都是可以的；而使用 U 盘，则受限于 ES File Manager 应用不能访问/data/local 目录，只能拷贝到/sdcard/Download。

Q3：关于 Android Studio 的 Terminal 窗口

A3：Android Studio 的 Terminal 窗口在进入的时候，工作在 PC 的文件系统上，操作的文件都是 PC 上的；当使用 adb shell 登陆 Android 目标板之后，工作在 Android 目标板的文件系统上，操作的文件、执行的命令都是 Android 目标板上的；在使用 adb shell 登陆之后，可以使用 exit 命令退出登陆状态，返回到 PC 端的工作目录。

Q4：adb 连接不上设备

使用 adb devices 查看一下是否有已连接的设备；检查 usb_otg 和 PC 端的物理连接；重新插拔一下调试线或者重启系统。

如果 ethernet 正常工作，可以使用 ethernet 代替 usb_otg，在 Terminal 中输入一下命令：

```
$ adb usb
restarting in USB mode
$ adb devices
List of devices attached
???????????       device
$ adb tcpip 5555
restarting in TCP mode port: 5555
$ adb connect YOUR_IP_ADDRESS
connected to YOUR_IP_ADDRESS:5555
$ adb devices
List of devices attached
???????????       device
```

YOUR_IP_ADDRESS:5555 device

退出：

adb disconnect YOUR_IP_ADDRESS

Q5：常用命令

查看所有 service 运行状态：getprop | grep init.svc

adb 相关：

adb devices　查看 usb_otg 已连接的设备

adb push localfile remotepath 将 PC 端的 localfile 下载到 Android 端的 remotepath 目录下。

adb pull remotefile 复制 Android 端的 remotefile 文件到 PC 端的当前目录