软件开发

调试总结文档

版本号: 0.1

发布日期: 2020-09-30

版本历史

版本号	日期	制/修订人	内容描述
0.1	2020.9.28	汤健雄	1. 建立文档

目　录

1 前言

1.1 文档简介

此文档为 P166 项目开发过程 Debug 调试文档。

1.2 目标读者

P166 项目开发人员和维护人员。

2 调试工具

2.1 GDB 工具

GDB 是一个由 GNU 开源组织发布的、UNIX/LINUX 操作系统下的、基于命令行的、功能强大的程序调试工具。对于一名 Linux 下工作的 c++ 程序员，gdb 是必不可少的工具。

2.1.1 启动 gdb

对 C/C++ 程序的调试，需要在编译前就加上-g 选项:

```
$g++ -g hello.cpp -o hello
```

调试可执行文件:

```
$gdb <program>
```

program 也就是你的执行文件，一般在当前目录下。
调试 core 文件 (core 是程序非法执行后 core dump 后产生的文件)。

```
$gdb <program> <core dump file>
$gdb program core.11127
```

调试服务程序。

```
$gdb <program> <PID>
$gdb hello 11127
```

如果你的程序是一个服务程序，那么你可以指定这个服务程序运行时的进程 ID。gdb 会自动 attach 上去，并调试他。program 应该在 PATH 环境变量中搜索得到。

2.2 ADB 工具

- adb 简介
  adb 全称为 Android Debug Bridge，是 Android SDK 里的一个工具，用于操作管理 Android 模拟器或真实的 Android 设备。其主要功能有：运行设备的 shell（命令行）；管理模拟器或设备的端口映射；在计算机和设备之间上传/下载文件。
- 运行 adb Windows PC 端的 adb 使用方法和 adb 应用程序，请自行从网络搜索。
- adb 常用命令
- pc 端查看当前连接的设备

```
adb devices
```

- PC 端进入设备 shell 命令行模式

```
adb shell
```

- 将电脑上的文件上传至设备

```
adb push <local path> <remote path>
```

- 下载设备里的文件到电脑

```
adb pull <remote path> <local path>
```

3 各模块 netlink 初始化

当 netlink 初始化失败时，会有失败错误打印，根据打印可定位原因。主要分为两类：netlink 套接字创建失败、地址绑定失败。

debug 流程如下图：

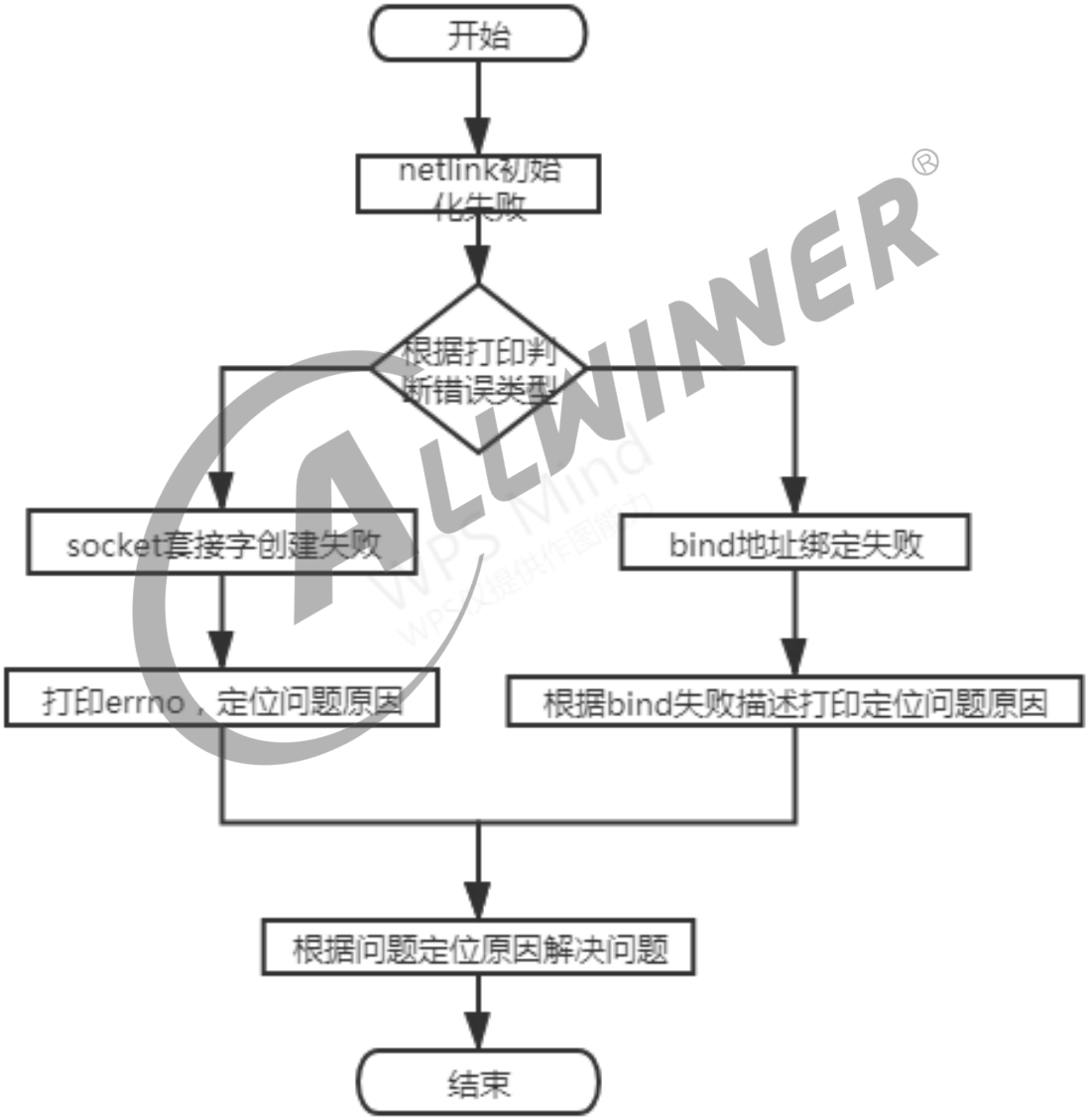

图 3-1: netlink 初始化 debug 流程

- 套接字创建失败

```
create socket error!
[errno]93
```

当套接字创建失败时，会返回相应的错误码，根据错误码可定位创建套接字失败的原因。如上所示，可以通过 errno 值 93 知道，此错误码的原因为 Protocol not supported，根据原因即可定位问题。* bind 地址绑定失败

```
sh-4.4# ./client_b -f
bind: Address already in use
```

当地址绑定失败时，会返回相应的错误码，根据错误码可定位地址绑定失败的原因。如上所示，根据错误打印可知当前地址已经与一个 socket 描述符绑定，不可重复绑定。

4 消息通信 debug

消息通信出问题时，首先需要根据问题场景定位问题所在环节。一般可分为：公共模块问题和客户端与内核模块通信问题。一般 debug 流程如下图。

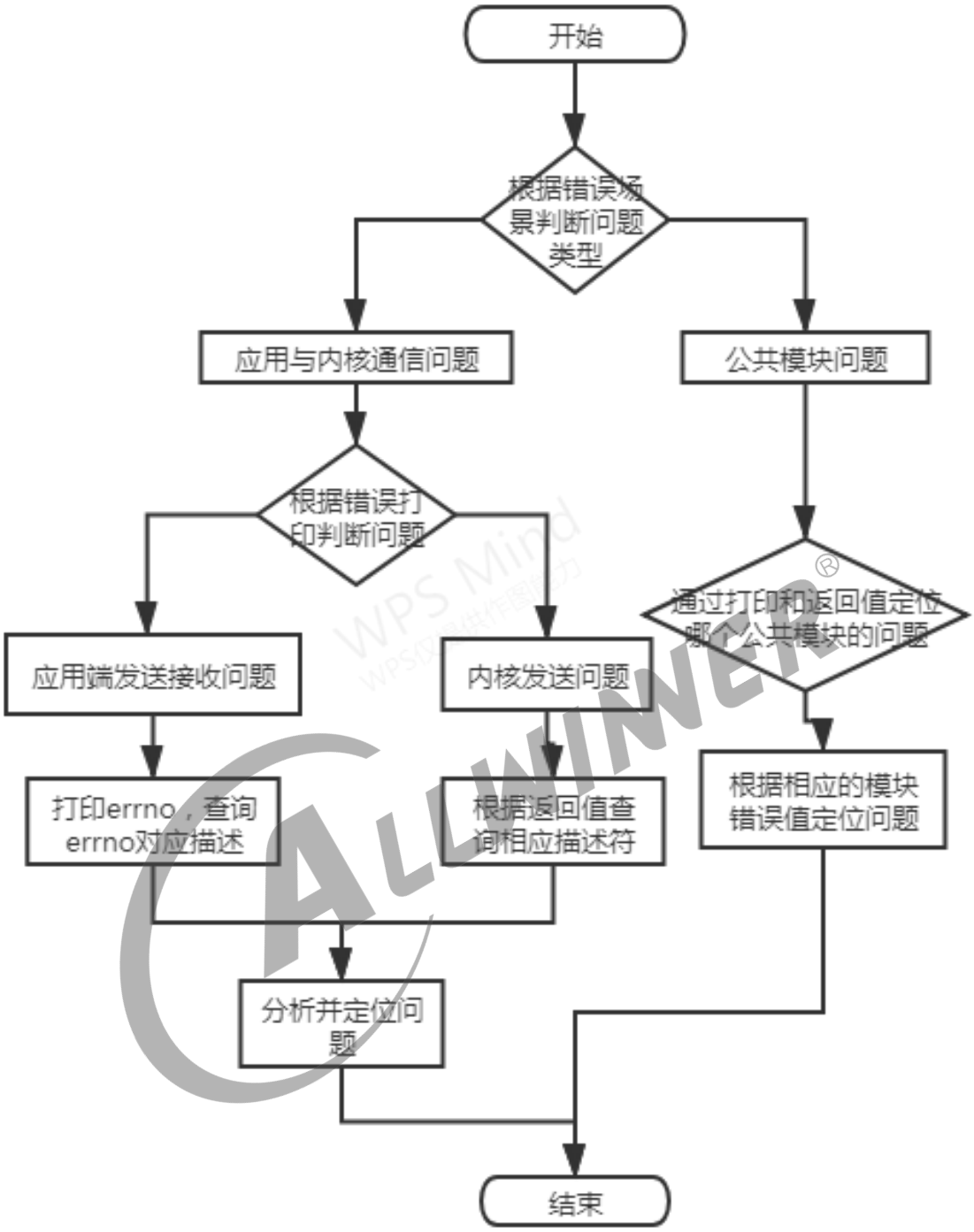

图 4-1: 消息通信 debug 流程

- 公共模块问题公共模块的问题可通过将每一次调用公共模块时的结果和返回值打印出来判断调用情况定位问题。
- 编码和解码功能
  编解码失败时，可以根据编解码函数的返回负值定位失败原因。

- 数据封装与解封
  编解码失败时，可以根据编解码函数的返回负值定位失败原因。

- hash 计算
  hash 计算错误时，可根据 hash 计算函数的返回负值定位失败原因。
- 应用层与内核模块通信问题应用层与内核模块的通信问题主要包括 3 种情况：应用层发送失败、应用层接收失败、内核模块发送失败。根据出错时的打印可判断是内核发送问题还是应用层发送和接收的问题。
- 应用层发送失败应用层发送失败之后，可打印 errno，可根据错误码描述快速定位问题。
- 应用层接收失败应用层接收失败之后，可打印 errno，可根据错误码描述快速定位问题。如下所示，当接收消息失败时，打印对应的 errno，查询得知对应的 errno 描述为 Bad file descriptor，查询资料可定位问题原因 socet 描述符是无效的，根据问题定位可解决问题。

```
[recvmsg error!]
[errno]9
```

- 内核发送失败内核的 netlink 发送函数会直接返回错误码的值，可根据错误码负值，查询源码中对应的原因定位问题。常见的发送失败错误码和原因如下：
  errno -11 内核发送 socket 队列已满，前面的消息应用层还未来得及接收，出现-11 的错误码就是内核发送的太快，应用端接收的太慢，想办法加快接收速度或者减慢发送速度即可解决。
  errno -111 #define ECONNREFUSED 111 /* Connection refused */
  errno -512 ERESTARTSYS 一般是内核发送函数为阻塞式的，当队列满了之后，接收端接收停止，内核模块会一直阻塞在发送消息的命令处，直至应用端被结束，此时就会出现-111 和-512 的错误。解决办法可以给发送消息进行超时处理。

著作权声明

商标声明

免责声明