# TODO

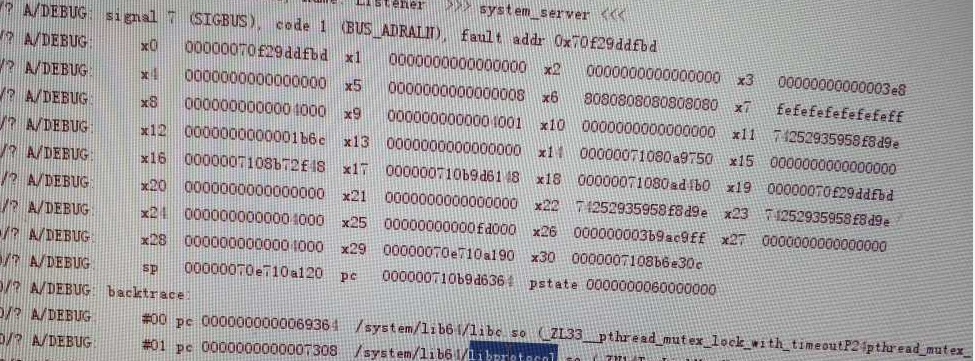
## 看书

C++ 规范： [Google C++ 代码规范](https://zh-google-styleguide.readthedocs.io/en/latest/google-cpp-styleguide/)

C++ 上手教程 ：[Effective C++中文版（第三版）.pdf](https://confluence.djicorp.com/download/attachments/109933687/Effective%20C%2B%2B%E4%B8%AD%E6%96%87%E7%89%88%EF%BC%88%E7%AC%AC%E4%B8%89%E7%89%88%EF%BC%89.pdf?version=1&modificationDate=1563337399000&api=v2)

signal 7 (SIGBUS), code 1, fault add android

<https://www.bbsmax.com/A/x9J2xALVd6/>



stCmd\* CmdQueue::waitCmd(uint16\_t seq, **int** timeoutInMs) {  
 **struct** timeval tTime;  
 **struct** timespec timer = {0, 0};  
 **long long** duration;  
 **int** ret ;  
  
 stCmd \*pCmd = NULL;  
  
 gettimeofday(&tTime,NULL);  
 duration = ((**long long**)tTime.tv\_sec)\*1000000+tTime.tv\_usec + timeoutInMs\*1000;  
 timer.tv\_sec=duration/1000000;  
 timer.tv\_nsec=(duration%1000000)\*1000;  
  
 pthread\_mutex\_lock(&cmdLock);  
 uint16\_t index = seq % MAX\_CMD\_COUNT;  
 pCmd = &mCmds[index];  
  
 ret = pthread\_cond\_timedwait(&pCmd->cond, &cmdLock, &timer);  
  
 pthread\_mutex\_unlock(&cmdLock);  
 pthread\_cond\_destroy(&pCmd->cond);  
 **if** (!ret)  
 **return** pCmd;  
 **return** NULL;  
}

<https://blog.csdn.net/fingding/article/details/36653139>

#pragma pack

<https://www.jianshu.com/p/90a6eef329ec>

#include **<pthread.h>在哪个库中？**

<https://blog.csdn.net/koozxcv/article/details/51728793>

android C++常用库

# 开始

当前C++使用版本：C++14

## 代码规范

Google C++ Code Style([**英文版**](https://google.github.io/styleguide/cppguide.html)，[**中文版**](https://zh-google-styleguide.readthedocs.io/en/latest/google-cpp-styleguide/)）

请以英文版为准，中文版很久没有更新了，两边内容有不一致的地方。

对 Google C++ Code Style 有各种好坏的讨论，觉得问题不大，重要的是一致性。所有代码/开发同事有一致的风格，对后续的维护有很好的帮助。

格式化工具：[**ClangFormat**](https://clang.llvm.org/docs/ClangFormat.html)

LLVM中的代码格式化工具，下列命令可以生成 Google 格式的 配置文件 .clang-format。把这个文件放到代码的根目录，大部分IDE（XCode、CLion、Visual Studio）都支持按照这个配置来格式化代码。（**TODO**后续更新具体的步骤，请先google，问题不大）。

clang-format -style=google -dump-config > .clang-format

1. 推荐一个 clang-format 与 git 整合的脚本，来源 llvm 项目：<https://github.com/llvm-mirror/clang/blob/master/tools/clang-format/git-clang-format>

配置的修改请参考 [Clang-Format Style Options](https://clang.llvm.org/docs/ClangFormatStyleOptions.html)（有中文的翻译，google）。

注意下 clang-format 的版本，不同的软件版本和配置文件之间可能有兼容性问题。后续考虑整个部门维护一个统一的配置。

格式化检查工具：[**cpplint**](https://github.com/google/styleguide/tree/gh-pages/cpplint)

Python 脚本，可以对代码格式的问题进行检查。（**TODO** 具体的集成、使用说明）。

[**C++ Core Guidelines**](https://github.com/isocpp/CppCoreGuidelines)

C++ 大佬维护的 guideline，关注 C++ 11/14/17。很好的学习资料。

## 图书

选择图书的标准，一定要支持C++11标准之后，尽量不要阅读C++98标准的书，两者差别比较大。大部分书（除了测试驱动开发哪本）部门图书角都有，或者我有（可以找我）。

|  |  |  |
| --- | --- | --- |
| 入门/复习 | [C++ 程序设计语言（第 1 - 3 部分）（原书第 4 版）](https://book.douban.com/subject/26857943/) | 阅读 “第一部分 引言”，第一章到第五章，大概100多页，覆盖了主要的概念。看完了就可以开始干活了。和 [C++语言导学](https://book.douban.com/subject/26857943/)内容一致。 |
| 参考书/垫显示器专用 | [C++ 程序设计语言（第 1 - 3 部分）（原书第 4 版）](https://book.douban.com/subject/26857943/)  [C++ 程序设计语言（第 4 部分：标准库）（原书第 4 版）](https://book.douban.com/subject/26900000/)  [C++标准库（第2版）](https://book.douban.com/subject/26419721/) | 参考书，通读比较花时间。可以遇到问题/感兴趣的时候，阅读相应的章节。 |
| 深入必读 | [Effective Modern C++ 简体中文版](https://book.douban.com/subject/30178902/) | C++ 11 之后，有些概念还是略复杂，有些很好/很方便的特性依然有坑。详细/深入的解释新特性，值得反复阅读/理解。  [Effective\_Modern\_C++.pdf](https://confluence.djicorp.com/download/attachments/111233588/Effective_Modern_C%2B%2B.pdf?version=1&modificationDate=1563531712000&api=v2) |
| 专题：并发编程 | [C++并发编程实战](https://book.douban.com/subject/26386925/) | **书很好，但是翻译烂到爆。**  网上[这个人](https://chenxiaowei.gitbooks.io/cpp_concurrency_in_action/content/)的翻译还不错。  英文第二版：[C++ Concurrency in Action, 2nd Edition.pdf](https://confluence.djicorp.com/download/attachments/111233588/C%2B%2B%20Concurrency%20in%20Action%2C%202nd%20Edition.pdf?version=1&modificationDate=1563510506000&api=v2) |
| 专题：C++11 | [深入理解C++11](https://book.douban.com/subject/24738301/) | 关注C++11中新增内容。如果只想看看C++11新增了哪些内容，可以阅读这本书。 |
| 专题：API 设计 | [C++ API设计](https://book.douban.com/subject/24869855/) | API 设计的原则、常用方法、注释、文档、测试等等。 |
| 专题：C++ 模板 | [C++ Templates 第2版 英文版](https://book.douban.com/subject/30226708/) | 如果上面的内容都不能满足你的话，这本书一定可以。 |
| 专题：测试驱动开发 | [C++程序设计实践与技巧 测试驱动开发](https://book.douban.com/subject/26927749/) | 使用 Google Test 讲解C++ TDD。 |
| CMake | [CMake Cookbook.pdf](https://confluence.djicorp.com/download/attachments/111233588/CMake%20Cookbook.pdf?version=1&modificationDate=1563523971000&api=v2)  [Craig Scott - Professional CMake A Practical Guide (2018).pdf](https://confluence.djicorp.com/download/attachments/111233588/Craig%20Scott%20-%20Professional%20CMake%20A%20Practical%20Guide%20%282018%29.pdf?version=1&modificationDate=1563524000000&api=v2) | 两本在讲 Modern CMake 的书。还可以参考下面的“Effective CMake”的演讲。 |

## 在线资源

### 在线文档

C++ 标准文档: [Working Draft, Standard for Programming Language C++](http://eel.is/c++draft/)

嗯，标准文档。估计大家也不想读，请看下面：

[C++ reference](https://en.cppreference.com/w/)

标准文档的通俗化描述，很好的在线文档。描述的内容可能等同于标准文档。还可以在线跑一些测试代码。

[C++ Patterns](https://cpppatterns.com/)

有一些 C++ 惯用法的实现，可以通读下，考察下 C++ 水平。

### 视频&会议

[CppCon](https://www.youtube.com/channel/UCMlGfpWw-RUdWX_JbLCukXg)

可以按照 popular 排序，或者关注早几年的内容（13、14、15），介绍的内容会是现在的编译器支持的。近两年的内容，都比较新，实验性质、或者编译器还没有广泛的支持。

一些好的演讲（**TODO**持续更新）：

[CppCon 2014: Herb Sutter "Back to the Basics! Essentials of Modern C++ Style"](https://www.youtube.com/watch?v=xnqTKD8uD64&t=4648s)

cmake 在新的版本中也逐渐有一些 modern design，逐渐走向面向对象的设计，很多东西都是一个对象（比如库、可执行文件），然后设置对应的属性，也甚至有了public、private的概念（当然，和编程语言的有所不同）：

[C++Now 2017: Daniel Pfeifer “Effective CMake"](https://www.youtube.com/watch?v=bsXLMQ6WgIk&t=4265s)

### Asesome C++

GitHub 上找到三个链接，都还不错。涉及到各种库、文档、书、在线资源、研究、工具等等。

<https://github.com/fffaraz/awesome-cpp>

<https://github.com/rigtorp/awesome-modern-cpp>

<https://github.com/fenbf/AwesomePerfCpp>

## 工具

### 动态扫描工具

Google Sanitizer <https://github.com/google/sanitizers>

支持运行时内存越界、内存泄露、多线程问题等的检查。

### 单元测试

### 静态扫描工具

### CPPCheck

参见：[使用cppcheck查找代码缺陷](https://confluence.djicorp.com/pages/viewpage.action?pageId=73949176)

# 常量方法

system/connectivity/wificond/ipc\_constants.h

**namespace** android {  
**namespace** wificond {  
**namespace** ipc\_constants {  
  
**extern const char** kServiceName[];  
  
} *// namespace ipc\_constants*} *// namespace wificond*} *// namespace android*

system/connectivity/wificond/ipc\_constants.cpp

**namespace** android {  
**namespace** wificond {  
**namespace** ipc\_constants {  
  
**const char** kServiceName[] = **"wificond"**;  
  
} *// namespace ipc\_constants*} *// namespace wificond*} *// namespace android*

## 使用

**using** android::wificond::ipc\_constants::kServiceName;

**void** RegisterServiceOrCrash(**const** android::sp<android::IBinder>& service) {  
 android::sp<android::IServiceManager> sm = android::defaultServiceManager();  
 CHECK\_EQ(sm != NULL, **true**) << **"Could not obtain IServiceManager"**;  
  
 CHECK\_EQ(sm->addService(android::String16(kServiceName), service),  
 android::NO\_ERROR);  
}

# 多线程变编程

pthread\_cond\_t

pthread\_mutex\_t： 互斥锁，多线程中对共享变量的包保护

pthread\_cond\_t： 线程间同步，一般和pthread\_mutex\_t一起使用，以防止出现逻辑错误，即如果单独使用条件变量，某些情况下（条件变量前后出现对共享变量的读写）会出现问题

pthread\_mutexattr\_init(&tAttr);  
pthread\_mutexattr\_settype(&tAttr,PTHREAD\_MUTEX\_RECURSIVE);  
pthread\_mutex\_init(&tLock, &tAttr);  
pthread\_mutexattr\_destroy(&tAttr);

*PTHREAD\_MUTEX\_RECURSIVE*

如果一个线程对这种类型的互斥锁重复上锁，不会引起死锁，一个线程对这类互斥锁的多次重复上锁必须由这个线程来重复相同数量的解锁，这样才能解开这个互斥锁，别的线程才能得到这个互斥锁。如果试图解锁一个由别的线程锁定的互斥锁将会返回一个错误代码。如果一个线程试图解锁已经被解锁的互斥锁也将会返回一个错误代码。这种类型的互斥锁只能是进程私有的（作用域属性为*PTHREAD\_PROCESS\_PRIVATE*）。

# Wunused-parameter

/pipe/mainpipe.c:6:10: error: unused parameter 'argc' [-Werror,-Wunused-parameter]

LOCAL\_CFLAGS += -Wno-unused-parameter