**暑期实习课题文档**

**目 录**

[1. 课题内容 2](#_Toc23070)

[2. 课题要求 2](#_Toc11954)

[3. 课题时间安排 2](#_Toc24345)

[4. 方案设计 2](#_Toc24685)

[4.1 总体方案设计 2](#_Toc3593)

[4.2 具体方案设计 4](#_Toc29921)

[4.2.1 阻塞队列 4](#_Toc9218)

[4.2.2 读取图片线程 6](#_Toc5011)

[4.2.3 图片处理线程 7](#_Toc5684)

[4.2.4 图片写出线程 8](#_Toc455)

[4.2.5 线程中的算法 9](#_Toc28250)

[5. 线程执行步骤 9](#_Toc15789)

[5.1 线程创建 10](#_Toc26559)

[5.2 线程启动 10](#_Toc23511)

[5.3 线程同步 10](#_Toc14681)

[5.4 线程退出 10](#_Toc354)

[6. 课题评价 11](#_Toc7625)

[7. 总结 11](#_Toc26754)

# 课题内容

实现一个预览图像实时处理模块，里面有若干算法，要求采用多线程流水线实现算法的流水线处理。

# 课题要求

1. 使用C++编写，代码符合代码规范；
2. 流水线采用多线程；
3. 从文件系统读取YUV数据，30ms一张，处理完后YUV数据按顺序写入文件系统；
4. 若干算法以在图像上插入水印标签代替，算法内运行/等待总时间大于20ms，小于30ms。

# 课题时间安排

|  |  |
| --- | --- |
| 7月12日 — 7月13日 | C++ 代码规范学习 |
| 7月13日 — 7月15日 | YUV图片知识学习 |
| 7月17日 — 7月21日 | 深入学习Linux C++ 多线程 |
| 7月22日 — 7月23日 | 程序总体结构设计 |
| 7月24日 — 7月25日 | 读写图片线程编写 |
| 7月26日 — 7月29日 | 图片处理线程基类设计 |
| 7月30日 — 8月15日 | 结合前面工作，完成初步程序编写 |
| 8月16日 — 8月20日 | 调试、优化规范代码、编写总结PPT |
| 8月21日 — 8月30日 | 答辩 |

表1 课题时间安排

# 方案设计

## 4.1 总体方案设计

根据课题要求，需要完成图片的读入，线程处理，图片写出到文件系统，整个结构呈流水线状。设计的组成部分有：图片读入线程、阻塞队列、图片处理线程和图片写出线程。它们的总体结构如图1.

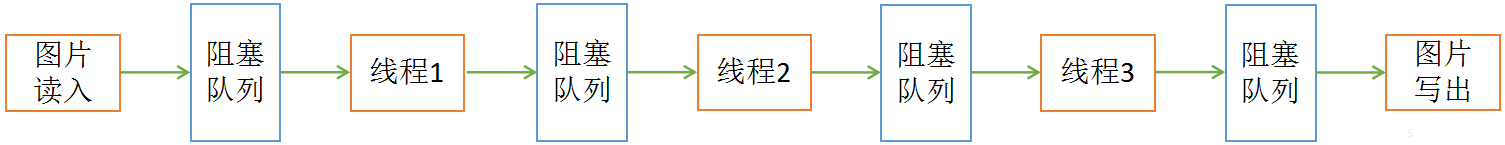


图1 课题结构设计

通过一个“图片读入线程”将图片从文件系统中读入，然后将其放入阻塞队列，“线程1”检测到阻塞队列中有元素将阻塞队列中的元素取出然后处理此元素(打水印)，然后将其放入下一个阻塞队列，“线程2”、“线程3”操作步骤同“线程1”，最后“图片写出线程”从临近的阻塞队列取元素并将其写成文件保存到文件系统。

其中的四个阻塞队列都是线程之间的临界资源，若要访问此资源必须先获得这个临界资源的锁之后，才能访问这个资源。每经过一个线程就给图片打一个水印。

课题实际类、对象关系如图2，4.2节将详细描述这些类、函数以及使用到的算法。

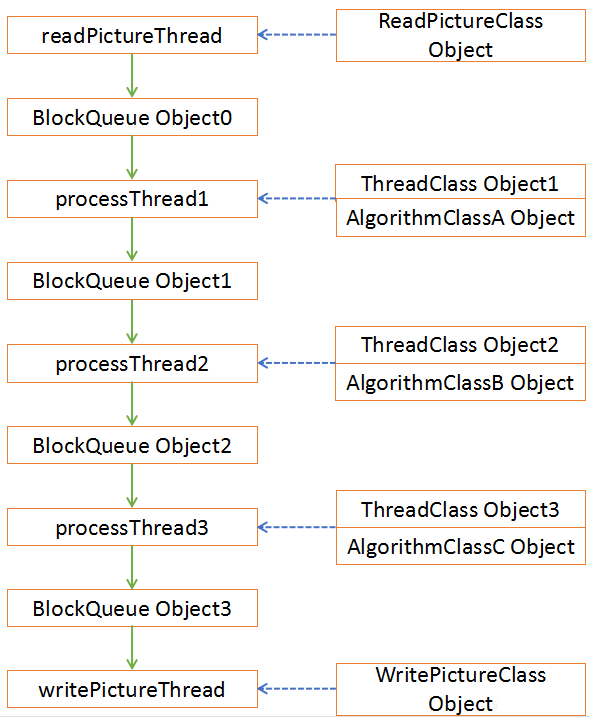


图2 线程关系图

## 4.2 具体方案设计

根据上述部分结构设计，下面对每个部分进行详细描述。

### 4.2.1 阻塞队列

阻塞队列主要用于线程和线程之间的通信。当队列为空时，从队列中获取元素的线程将会被阻塞；当队列是满时，往队列里添加元素的线程将会阻塞。本课题程序中主要通过 pthread 库中的线程锁、线程等待、线程唤醒函数来实现线程获得临界资源、线程阻塞以及唤醒。

1. pthread库简介
2. pthread\_t : 用于创建一个线程标识符。
3. pthread\_mutex\_t : 用于创建一个临界资源锁。
4. pthread\_cond\_t : 用于创建一个条件等待变量，用于线程阻塞。
5. pthread\_mutex\_init : 用于初始化临界资源锁。
6. pthread\_cond\_init : 用于初始化一个条件等待变量。
7. pthread\_create : 线程创建函数。
8. pthread\_join : 等待线程返回。
9. 阻塞队列类 BlockQueue
10. 构造函数 BlockQueue();

用于初始化阻塞队列的容量大小，初始化阻塞队列状态，初始状态为空(true)，满(false)，初始化此阻塞队列的锁、队列满阻塞条件变量和队列空阻塞条件变量。

1. 元素入队列函数 elemPush();

内部执行过程为：先尝试锁住此临界资源队列的锁，如果获取不到锁，那就阻塞，如果获取到锁，那就继续检查队列是否为满，如果为满则将队列满标志置为“true”,如果不为满，进入while循环，检查队列满标志，如果为 “true”，那么使用此函数的线程阻塞等待直到被唤醒，如果被阻塞唤醒后再次检查队列是否为满。经过上面的检查之后，就可以元素入队列，这里使用 C++ STL 中的双端队列中的 push\_back 函数入队列，然后将队列空标志置为(false)，并唤醒由于队列空造成阻塞的线程，最后释放锁。图3是元素入队列流程图。

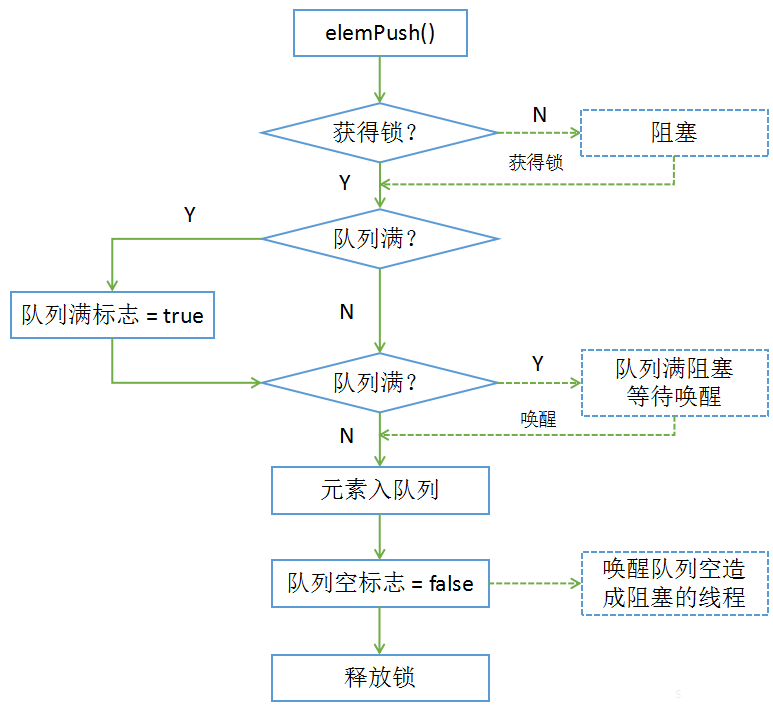


图3 元素入队列流程图

1. 元素出队列函数 elemPop();

内部执行过程为：先尝试锁住此临界资源队列的锁，如果获取不到锁，那就阻塞，如果获取到锁，那就继续检查队列是否为空，如果为满则将队列空标志置为“true”,如果不为空，进入while循环，检查队列空标志，如果为 “true”，那么使用此函数的线程阻塞等待直到被唤醒，如果被阻塞唤醒后再次检查队列是否为空。经过上面的检查之后，就可以元素出队列，这里使用 C++ STL 中的双端队列中的 pop\_front 函数出队列，然后将队列满标志置为(false)，并唤醒由于队列满造成阻塞的线程，最后释放锁。

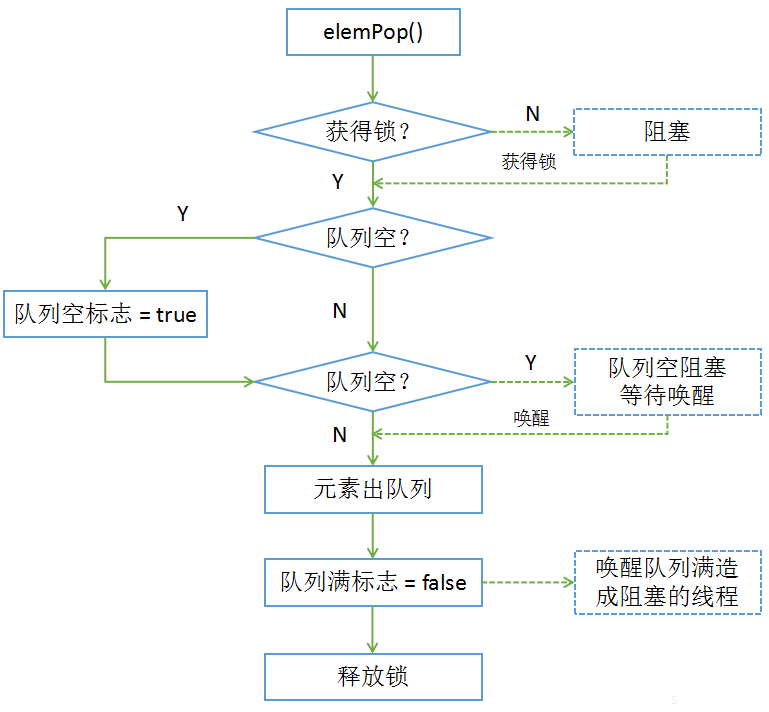


图4 元素出队列流程图

### 4.2.2 读取图片线程

读取图片线程分为两部分。第一部分是读取图片类，第二部分是线程函数。

1. 图片读取类 ReadPictureClass
2. 构造函数 ReadPictureClass();

用于初始化此类对应的线程的编号。例如读取图片线程为第一个线程，则将其编号置为“0”。

1. 图片读取函数 readPicture();

内部执行过程为：线程函数调用此函数时会传入两个参数，将要读取图片的序号以及临时存放图片内容的变量。此函数获取到这两个参数函数后，从文件系统中找到对应的图片，将其读入到存图片的变量中。至此完成一张图片的读取。

1. 图片读取线程函数 readPictureThead();

内部执行过程为：线程函数在线程被创建的时候就开始执行，创建此线程时会将传入的线程参数强制转换为指向ReadPictureClass的指针，这样便可以访问其中的所有成员。然后进入 for 循环，调用 readPicture() 函数读取图片，调用elemPush() 函数将读取的图片内存存入队列，在此过程中线程有可能因为阻塞队列为满而被阻塞。然后进行下一次循环，线程循环的结束条件是“读取完所有图片”。

### 4.2.3 图片处理线程

图片处理线程一共有三个，它们之间共性又有一定的区别。共性在于它们的处理过程类似，都是从上一个阻塞队列取元素，然后将元素内容进行处理，处理完成之后放入下一个阻塞队列。区别在于它们进行图像处理的算法是不同的。下面将详述这种同时具有共性又具有区别的类，以便于流水线线程可增添性。此过程也分为两部分：图片处理类，图片处理线程函数。

1. 图片处理类 ThreadClass
2. 构造函数 ReadPictureClass();

用于初始化此类对应的线程的编号。

1. 图片处理算法选择函数 processPicture();

根据不同线程的编号选择不同的算法。算法类在下文中会详述。

1. 图片处理线程函数 processThread();

内部执行过程为：线程函数将创建函数传入的参数强制转化为 ThreadClass的类对象指针，这样片可以方位其中所有成员。进入 for 函数，调用elemPop()，取出上一个阻塞队列中的元素，将元素中的内容通过 processPicture() 函数选择不同的算法来处理，处理完成之后调用 elemPush() 放入下一个阻塞队列中。在取元素和放元素的过程中此线程均可能被阻塞。进入下一次循环，线程结束条件为处理完所有图片。

### 4.2.4 图片写出线程

图片写出线程分为两部分。第一部分是写图片类，第二部分是线程函数。

1. 图片写出类 WritePictureClass
2. 构造函数 WritePictureClass();

用于初始化此类对应的线程的编号。

1. 图片写出函数 writePicture();

内部执行过程为：线程函数调用此函数时会传入两个参数，将要写图片的序号以及临时存放图片内容的变量。此函数获取到这两个参数函数后，将对应的图片用相应的格式以及序号存入文件系统

1. 图片写出线程函数 writePictureThead();

内部执行过程为：创建此线程时会将传入的线程参数强制转换为指向WritePictureClass的类对象指针然后进入 for 循环，调用elemPop() 函数取队列，在此过程中线程有可能因为阻塞队列为空而被阻塞，调用writePicture() 函数将图片写入到文件系统相应位置。然后进行下一次循环，线程循环的结束条件是“写完所有图片”。

### 4.2.5 线程中的算法

根据本课题要求要在三个线程中使用水印图片的方式来代替算法，但是也要注意流水线的模块化和可拓展性，以及方便后续人员在增加算法的时候，只关注算法的实现而不用很关注流水线是如何实现的。于是在设计中，将三个线程的算法分别封装到了三个独立的文件中，方便后续人员增加或修改，增加其独立性的同时又保持了原有流水线的结构。这些算法分别存放在AlgorithmClassA、AlgorithmClassB和AlgorithmClassC 中。下面详述给图片打水印的算法。

waterMark() 函数。函数接受两个参数，存放图片的内容的变量，打水印图片的位置。通过水印图片的位置(序号)在文件系统中找到用于水印的图片，然后使用内存拷贝方法 memcopy() 将水印图片替换原始图片的部分内容，即可完成水印添加。图3是效果图。右图是经过每个线程处理之后的图片，左图是原图。

图3 水印图片和原始图片对比

# 线程执行步骤

此课题中主要使用到pthread 即 POSIX threads, POSIX表示可移植操作系统接口(Portable Operating System Interface of UNIX), 所以pthread就是在这个标准下实现的线程。程序中主要用到其中的phread\_mutex\_t、pthread\_cond\_t、pthread\_cond\_signal、pthread\_mutex\_lock、pthread\_mutex\_unlock、pthread\_cond\_wait、pthread\_create和pthead\_join函数。以图片处理线程的执行过程为例进行介绍。

## 5.1 线程创建

创建一个图片处理线程，使用如下语句：

pthread\_create(&(m\_ProcessA->m\_Thread), NULL, processThread , (void \*) m\_ProcessA);

其中，第1个参数是指类ThreadClass 的对象指针 m\_ProcessA所指向的线程标识符，第2个参数是设置线程属性为默认，第3个参数为线程函数，第4个参数为线程函数参数。

## 5.2 线程启动

当processThread线程函数接收到pthread\_create传来的线程函数参数( (void \*)m\_ProcessA )后，线程就创建好并已经运行。

## 5.3 线程同步

当processThread线程想要访问阻塞队列的时候，例如执行取出队列元素(elemPop)，先要锁住此临界资源的锁，当有别的线程在使用此队列时当前线程因无法获得锁而被阻塞，当别的线程使用完此队列后，释放此队列临界资源锁，当前线程才可以获得锁来访问此队列。当获得锁之后执行取队列操作，检查队列元素是否空，如果为空，触发pthread\_cond\_wait函数使线程又进入阻塞状态,直到有pthread\_cond\_signal信号产生后，此线程才会重新进入运行状态。当processThread 线程执行入队列元素(elemPush)是同取队列类似的过程。总之，任何时刻都只能有一个线程获得访问临界资源的权限，否则就是线程不同步。

## 5.4 线程退出

pthread 库支持多种线程退出方式，例如pthread\_detach和pthread\_join。本程序使用pthread\_join方式，即：当函数调用线程函数后，会一直等待线程结束之后才会继续执行。本程序所有线程结束的方式是“处理完所有图片数据”。

# 课题评价

1. 课题完成度、代码规范自我评价

使用C++、按照《Android系统编码规范C、C++》规范编写代码；在程序中实现了多线程流水线；程序按照30fps读入图片；完成图片水印；算法内运行等待的时间为31ms左右，为保障算法运行时间大于20ms，在算法中延时了20ms,加上算法其他部分执行时间所以执行时间为31ms。基本完成所有课题要求。

1. 程序易修改性、可拓展性自我评价

本程序具有易修改性。本程序的算法部分，是独立在不同文件中的，如果有其他人员对本程序算法进行修改的话，不用太关心流水线的实现过程，容易修改。

本程序具有可扩展性。如果需要增加一个算法到流水线中，只需增加一个图片线程、增加一个阻塞队列并将此算法文件包含进ThreadClass.hpp后做一些小修改即可。

1. 程序优化及展望

本程序的图片处理算法都是使用的水印图片，可以为程序增加或者修改别的算法，例如滤镜等效果，使其更接近真实应用场景。此外应该优化架构设计，使其更合理，更易修改、易移植和易扩展。

# 总结

通过本次课题，学习到的不仅仅是在学校里没有接触过的知识，更多的是面对一个项目时应该如何去思考，如何去设计架构，如何去学习自己没有学习过的知识，这些在项目真正开始前的准备工作才是最重要的，没有这些准备作为导向，在项目实施过程成非常容易造成失败。此外也认识到自己知识的不足，需要再深入学习编程、操作系统、驱动方面的知识。