# IOCP模型解读

## 概述:

IOCP全称I/O Completion Port，中文译为I/O完成端口。IOCP是一个异步I/O的API，它可以高效地将I/O事件通知给应用程序。一个套接字[socket]与一个完成端口关联了起来，然后就可继续进行正常的Winsock操作了。然而，当一个事件发生的时候，此完成端口就将被操作系统加入一个队列中。然后应用程序可以对核心层进行查询以得到此完成端口。

获取关键字：端口、API、异步、套接字、队列(不用管)

## 应用IOCP需要包含的头和库:

//IOCP需要包含的头文件

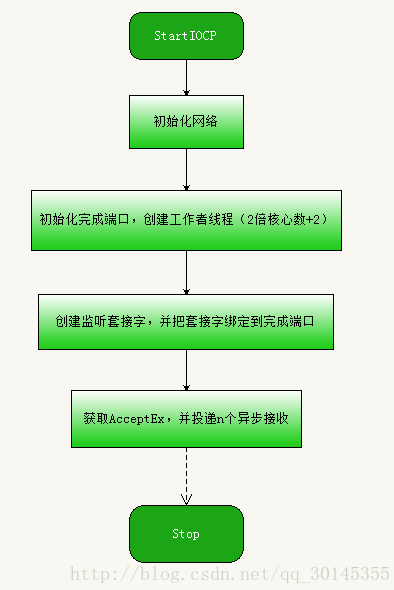
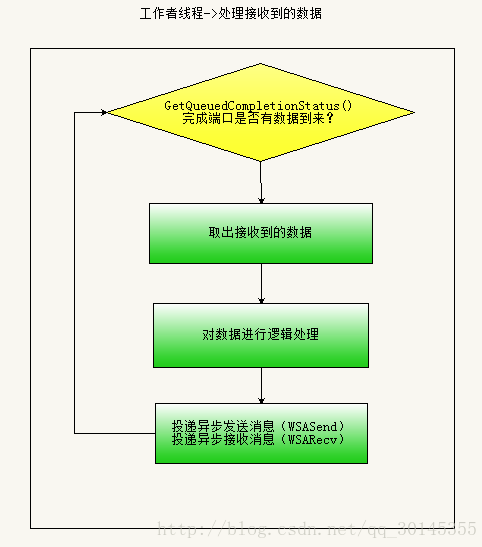
#include <WinSock2.h

#include <Windows.h>

#pragma comment(lib,"Ws2\_32.lib")//Socket编程需用的动态链接库

#pragma comment(lib,"Kernel32.lib")//IOCP需要用到的动态链接库

## IOCP的流程图：

## IOCP代码逻辑详解：

1、初始化网络：IOCP也是基于Winsock2套接字库的，所以在开始先必须初始化套接字库

//加载socket动态链接库

WORDwVersionRequested=MAKEWORD(2,2);//请求2.2版本的WinSock库

WSADATA wsaData;//接收WindowsSocket的结构信息

DWORD err = WSAStartup(wVersionRequested,&wsaData);

if(0!=err){//检查套接字库是否申请成功

cerr<<"RequestWindowsSocketLibraryError!\n";

system("pause");

return-1;

}

if(LOBYTE(wsaData.wVersion)!=2||HIBYTE(wsaData.wVersion)!=2){//检查是否申请了所需版本的套接字库

WSACleanup();

cerr<<"RequestWindowsSocketVersion2.2Error!\n";

system("pause");

return-1;

}

2、初始化完成端口(首次创建参数就是为空)

m\_hIOCompletionPort = CreateIoCompletionPort(INVALID\_HANDLE\_VALUE, *NULL*, 0, 0);

3、创建工作者线程

(*HANDLE*)*\_beginthreadex*(*NULL*, 0, ThreadPoolFunc, (void\*)this, 0, &nThreadID);

4、我们熟悉的bind、*listen*系列服务器端操作，做完这些事情后将套接字绑定到完成端口上

CreateIoCompletionPort((*HANDLE*)socket, m\_hIOCompletionPort, dwCompletionKey, 0);

5、获取AcceptEx、投递N个异步接收(N可以作为服务端的一个预设缓存区看)

注：AcceptEx要用获取的方式而不是用直接应用是因为为了不受限于平台

//需要用到的结构体

/\*\*

\*结构体名称：PER\_IO\_DATA

\*结构体功能：重叠I/O需要用到的结构体，临时记录IO数据

\*\*/

Const int DataBuffSize = 2\*1024;

typedefstruct

{

OVERLAPPED overlapped;

WSABUF databuff;

char buffer[DataBuffSize];

int BufferLen;

int operationType; //标记操作类型的可以用枚举

}PER\_IO\_OPERATEION\_DATA,\*LPPER\_IO\_OPERATION\_DATA,\*LPPER\_IO\_DATA,PER\_IO\_DATA;

AcceptEx +1，官网是用的 这里要注意光只连接不发送数据这里是不知道的

## 工作线程中做的事情

GetQueuedCompletionStatus(// 从完成端口中获取消息

pThis->m\_hIOCompletionPort,

&dwIoSize,

(PULONG\_PTR)&pIocpParam,

&pOverlapped,

INFINITE);

从参数中获取重叠结构数据：

pIoContext = *CONTAINING\_RECORD*(pOverlapped, PER\_IO\_CONTEXT, overlapped);

退出操作：

*PostQueuedCompletionStatus*(m\_hIOCompletionPort, 0, (*DWORD*)EXIT\_CODE, *NULL*);

注：有几个线路程就要进行几次调用，一次只会触发一次

## 其它：

* 1. **Socket的通信缓冲区设置成多大合适**
  2. **关于完成端口通知的次序问题【后入先出】**
  3. **数据销毁的时间节点**
  4. **应用链接池**
  5. **防止内存泄漏**
  6. **同步的问题须慎重**