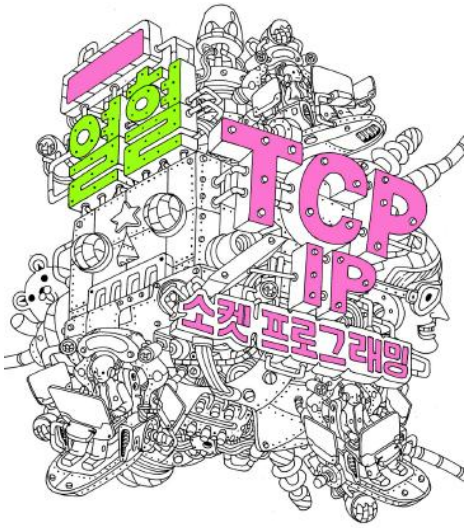
윤성우의 열혈 TCP/IP 소켓 프로그래밍

윤성우 저 열혈강의 TCP/IP 소켓 프로그래밍 개정판

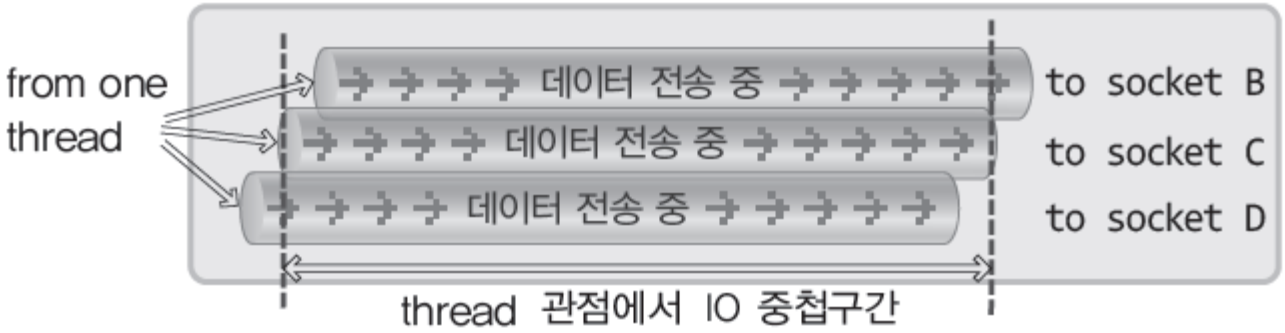
Chapter 22. Overlapped IO 모델



Chapter 22-1. Overlapped IO 모델의 이해

윤성우 저 열혈강의 TCP/IP 소켓 프로그래밍 개정판

IO(입출력)의 중첩이란?



IO의 중첩이라는 것은 쓰레드의 관점에서 동시에 둘 이상의 데이터 전송을 중첩시키는 것을 뜻한다. 그리고 데이터의 전송을 중첩시키기 위해서는 데이터 입출력 함수가 넌-블로킹 모드로 동작해야 한다. 그래서 연 이은 데이터의 입력 및 출력이 가능해진다.

위 그림에는 데이터 전송의 대상이 소켓 B, C, D로 구분이 되지만, IO의 중첩은 하나의 소켓을 대상으로 진행이 될수도 있다.

비동기 IO와 윈도우의 Overlapped IO의 비교

비동기 IO

- IO의 비동기란 넌 블로킹 모드의 IO를 뜻한다.
- IO가 비동기 방식으로 동작해야 IO를 중첩시킬 수 있다.
- 윈도우의 Overlapped IO는 IO를 중첩시키는 입출력 모델이다.

Overlapped IO

- Overlapped IO가 아니더라도 IO를 중첩시킬 수 있다.
- Overlapped IO의 포커스는 IO가 아닌, 입출력의 완료 확인방법에 있다.

Overlapped IO의 소켓 생성

```
#include <winsock2.h>

SOCKET WSASocket(
  int af, int type, int protocol, LPWSAPROTOCOL_INFO lpProtocolInfo, GROUP g, DWORD dwFlags);
```

➜ 성공 시 소켓의 핸들, 실패 시 INVALID_SOCKET 반환

- af — 프로토콜 체계 정보 전달.
- type — 소켓의 데이터 전송방식에 대한 정보 전달.
- protocol — 두 소켓 사이에 사용되는 프로토콜 정보 전달.
- lpProtocolInfo — 생성되는 소켓의 특성 정보를 담고 있는 WSAPROTOCOL_INFO 구조체 변수의 주소 값 전달, 필요 없는 경우 NULL 전달.
- g — 함수의 확장을 위해서 예약되어 있는 매개변수, 따라서 0 전달.
- dwFlags — 소켓의 속성정보 전달.

Overlapped IO가 가능한 소켓의 생성

```
WSASocket(PF_INET, SOCK_STREAM, 0, NULL, 0, WSA_FLAG_OVERLAPPED);
```

WSASocket 함수의 마지막 전달인자로 WSA_FLAG_OVERLAPPED가 전달되어야 한다.

Overlapped IO를 진행하는 WSASend 함수

```
#include <winsock2.h>

int WSASend(
     SOCKET s, LPWSABUF lpBuffers, DWORD dwBufferCount,
     LPDWORD lpNumberOfBytesSent, DWORD dwFlags, LPWSAOVERLAPPED lpOverlapped,
     LPWSAOVERLAPPED_COMPLETION_ROUTINE lpCompletionRoutine);
```

➜ 성공 시 0, 실패 시 SOCKET_ERROR 반환

s	소켓의 핸들 전달, Overlapped IO 속성이 부여된 소켓의 핸들 전달 시 Overlapped IO 모델로 출력 진행.
lpBuffers	전송할 데이터 정보를 지니는 WSABUF 구조체 변수들로 이뤄진 배열의 주소 값 전달.
dwBufferCount	두 번째 인자로 전달된 배열의 길이정보 전달.
lpNumberOfBytesSent	전송된 바이트 수가 저장될 변수의 주소 값 전달(이는 잠시 후 별도로 설명).
dwFlags	함수의 데이터 전송특성을 변경하는 경우에 사용, 예로 MSG_OOB를 전달하면 OOB 모드 데이터 전송.
lpOverlapped	WSAOVERLAPPED 구조체 변수의 주소 값 전달, Event 오브젝트를 사용해서 데이터 전송의 완료를 확인하는 경우에 사용되는 매개변수
lpCompletionRoutine	Completion Routine이라는 함수의 주소 값 전달, 이를 통해서도 데이터 전송의 완료를 확인할 수 있다.

가장 관심을 둬야 할 두 개의 매개변수

Overlapped Send의 진행과정

```
typedef struct __WSABUF
{
    u_long len;    // 전송할 데이터의 크기
    char FAR* buf; // 버퍼의 주소 값
} WSABUF, *LPWSABUF;
```

```
typedef struct _WSAOVERLAPPED
{
    DWORD Internal;
    DWORD InternalHigh;
    DWORD Offset;
    DWORD OffsetHigh;
    WSAEVENT hEvent;
} WSAOVERLAPPED, *LPWSAOVERLAPPED;
```

운영체제 내부적으로 사용되는 멤버

```
WSAEVENT event;
WSAOVERLAPPED overlapped;
WSABUF dataBuf;
char buf[BUF_SIZE]={"전송할 데이터"};
int recvBytes=0;
. . . . .
event=WSACreateEvent();
memset(&overlapped, 0, sizeof(overlapped)); // 모든 비트 0으로 초기화!
overlapped.hEvent=event;
dataBuf.len=sizeof(buf);
dataBuf.buf=buf;
WSASend(hSocket, &dataBuf, 1, &recvBytes, 0, &overlapped, NULL);
. . . . .
```

Overlapped IO의 진행과정!

완료의 확인방법은 잠시 후 소개

함수의 호출과 동시에 데이터의 전송이 완료되지 못하면 의미를 갖지 않는다.

데이터 송수신 결과의 확인방법

```
#include <winsock2.h>

BOOL WSAGetOverlappedResult(
   SOCKET s, LPWSAOVERLAPPED lpOverlapped, LPDWORD lpcbTransfer, BOOL fWait, LPDWORD lpdwFlags);
```

➜ 성공 시 TRUE, 실패 시 FALSE 반환

- **s** — Overlapped IO가 진행된 소켓의 핸들.
- **lpOverlapped** — Overlapped IO 진행 시 전달한 WSAOVERLAPPED 구조체 변수의 주소 값 전달.
- **lpcbTransfer** — 실제 송수신된 바이트 크기를 저장할 변수의 주소 값 전달.
- **fWait** — 여전히 IO가 진행중인 상황의 경우, TRUE 전달 시 IO가 완료될 때까지 대기를 하게 되고, FALSE 전달 시 FALSE를 반환하면서 함수를 빠져 나온다.
- **lpdwFlags** — WSARecv함수가 호출된 경우, 부수적인 정보(수신된 메시지가 OOB 메시지인지와 같은)를 얻기 위해 사용된다. 불필요하면 NULL을 전달한다.

데이터의 전송이 계속 진행되는 상황에서는 WSASend 함수가 SOCKET_ERROR를 반환하고, WSAGetLastError 함수호출을 통해서 확인 가능한 오류코드로는 WSA_IO_PENDING이 등록된다. 그리고 이 경우에는 위의 함수호출을 통해서 실제 전송된 데이터의 크기를 확인해야 한다.

WSAGetLastError 함수는 잠시 후 설명

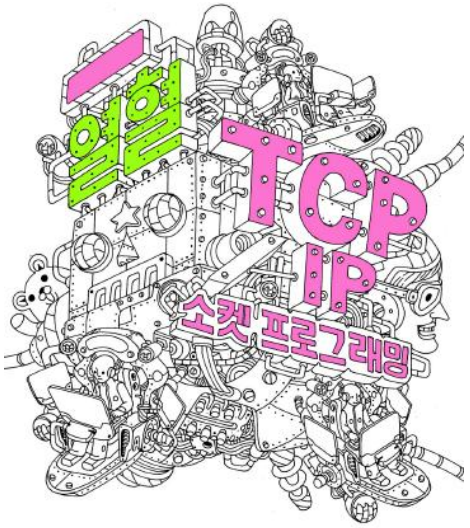
Overlapped IO를 진행하는 WSARecv 함수

```
#include <winsock2.h>

int WSARecv(
    SOCKET s, LPWSABUF lpBuffers, DWORD dwBufferCount,
    LPDWORD lpNumberOfBytesRecvd, LPDWORD lpFlags, LPWSAOVERLAPPED lpOverlapped,
    LPWSAOVERLAPPED_COMPLETION_ROUTINE lpCompletionRoutine
);
```

➡ 성공 시 0, 실패 시 SOCKET_ERROR 반환

s	Overlapped IO 속성이 부여된 소켓의 핸들 전달.
lpBuffers	수신된 데이터 정보가 저장될 버퍼의 정보를 지니는 WSABUF 구조체 배열의 주소 값 전달.
dwBufferCount	두 번째 인자로 전달된 배열의 길이정보 전달.
lpNumberOfBytesRecvd	수신된 데이터의 크기정보가 저장될 변수의 주소 값 전달.
lpFlags	전송특성과 관련된 정보를 지정하거나 수신하는 경우에 사용된다.
lpOverlapped	WSAOVERLAPPED 구조체 변수의 주소 값 전달.
lpCompletionRoutine	Completion Routine이라는 함수의 주소 값 전달.

WSASend 함수와 호출의 방식이 매우 유사하다.



Chapter 22-2. Overlapped IO에서의 입출력 완료의 확인

윤성우 저 열혈강의 TCP/IP 소켓 프로그래밍 개정판

Event 오브젝트 기반의 Send

Event 오브젝트 기반 IO 확인의 중요한 두 가지

- IO가 완료되면 WSAOVERLAPPED 구조체 변수가 참조하는 Event 오브젝트 signaled 상태가 됨.
- IO의 완료 및 결과를 확인하려면 WSAGetOverlappedResult 함수를 호출한다.

```
evObj=WSACreateEvent();
memset(&overlapped, 0, sizeof(overlapped));
overlapped.hEvent=evObj;
dataBuf.len=strlen(msg)+1;
dataBuf.buf=msg;

if(WSASend(hSocket, &dataBuf, 1, &sendBytes, 0, &overlapped, NULL)
    ==SOCKET_ERROR)
{
    if(WSAGetLastError()==WSA_IO_PENDING)
    {
        puts("Background data send");
        WSAWaitForMultipleEvents(1, &evObj, TRUE, WSA_INFINITE, FALSE);
        WSAGetOverlappedResult(hSocket, &overlapped, &sendBytes, FALSE, NULL);
    }
    else
    {
        ErrorHandling("WSASend() error");
    }
}
```

Signaled 상태를 확인한 다음에 실제 전송된 바이트 크기를 확인하는 코드

WSAGetLastError

```
#include <winsock2.h>

int WSAGetLastError(void);
```

➔ 오류상황에 대한 상태 값(오류의 원인을 알리는 값) 반환

오류 발생시 해당 오류 정보를 위의 함수호출을 통해서 확인할 수 있다.

위의 함수는 오류의 발생으로 인해서 등록된 오류의 상태 값을 반환한다.

Event 오브젝트 기반의 Recv

```
evObj=WSACreateEvent();
memset(&overlapped, 0, sizeof(overlapped));
overlapped.hEvent=evObj;
dataBuf.len=BUF_SIZE;
dataBuf.buf=buf;

if(WSARecv(hRecvSock, &dataBuf, 1, &recvBytes, &flags, &overlapped, NULL)
    ==SOCKET_ERROR)
{
    if(WSAGetLastError()==WSA_IO_PENDING)
    {
        puts("Background data receive");
        WSAWaitForMultipleEvents(1, &evObj, TRUE, WSA_INFINITE, FALSE);
        WSAGetOverlappedResult(hRecvSock, &overlapped, &recvBytes, FALSE, NULL);
    }
    else
    {
        ErrorHandling("WSARecv() error");
    }
}
```

이렇듯 WSAGetOverlappedResult 함수는 전송된 바이트 수의 확인뿐만 아니라,
수신된 바이트 수의 확인에도 사용된다.

Completion Routine 사용하기

Completion Routine의 이해와 등록

- IO가 완료되었을때 호출되는 함수를 가리켜 Completion Routine이라 한다.
- IO가 완료되면, 미리 등록된 CompletionRoutine이 운영체제에 의해서 자동으로 호출된다.
- CompletionRoutine이 호출되기 위해서는 해당 쓰레드가 alertable wait 상태에 놓여야 한다.
- Alertable wait은 운영체제가 전달하는 메시지의 수신이 가능한 상태, 다시 말해서 특별한 일을 진행하지 않는 상태를 뜻한다.

Alertable wait 상태로의 진입에 사용되는 함수들

- WaitForSingleObjectEx
- WaitForMultipleObjectsEx
- WSAWaitForMultipleEvents
- SleepEx

Completion Routine 기반의 IO 완료확인은 함수의 등록과 등록된 함수의 호출을 통해서 이뤄진다. 단, 등록된 함수가 호출될 수 있도록 쓰레드는 Alertable wait 상태가 되어야 한다.

Completion Routine의 예

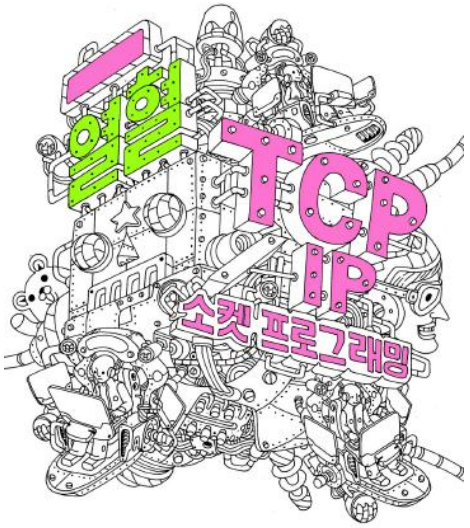
```
hRecvSock=accept(hLisnSock, (SOCKADDR*)&recvAdr,&recvAdrSz);
if(hRecvSock==INVALID_SOCKET)
    ErrorHandling("accept() error");

memset(&overlapped, 0, sizeof(overlapped));
dataBuf.len=BUF_SIZE;
dataBuf.buf=buf;
evObj=WSACreateEvent();    // Dummy event object

if(WSARecv(hRecvSock, &dataBuf, 1, &recvBytes, &flags, &overlapped, CompRoutine)
    ==SOCKET_ERROR)
{
    if(WSAGetLastError()==WSA_IO_PENDING)
        puts("Background data receive");
}

idx=WSAWaitForMultipleEvents(1, &evObj, FALSE, WSA_INFINITE, TRUE);
if(idx==WAIT_IO_COMPLETION)
    puts("Overlapped I/O Completed");
else    // If error occurred!
    ErrorHandling("WSARecv() error");
```

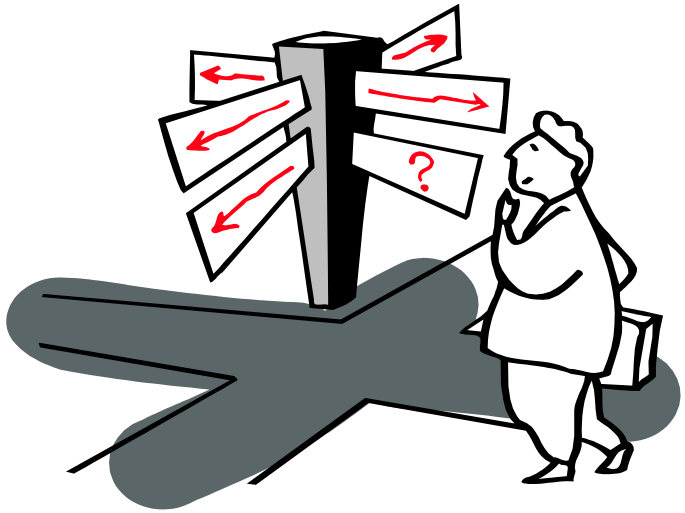
Completion Routine의 등록

Overlapped IO를 위한 인자의 전달

Alertable wait 상태로의 진입!

```
void CALLBACK CompRoutine(
    DWORD dwError, DWORD szRecvBytes, LPWSAOVERLAPPED lpOverlapped, DWORD flags)
{
    if(dwError!=0)
    {
        ErrorHandling("CompRoutine error");
    }
    else
    {
        recvBytes=szRecvBytes;
        printf("Received message: %s \n", buf);
    }
}
```





Chapter 22가 끝났습니다. 질문 있으신지요?