# IOCP 실습

## MM4220 게임서버 프로그래밍

정내훈

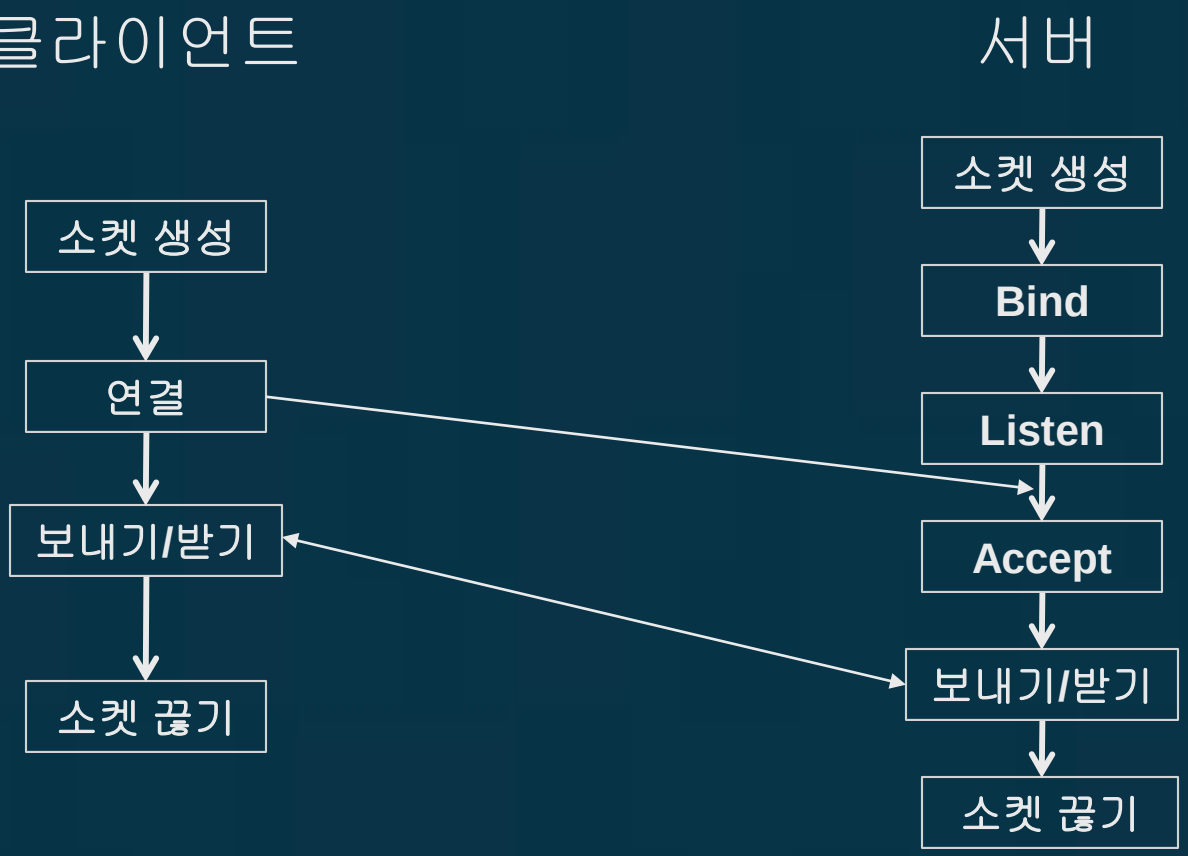
# 실습

- IOCP를 사용한 서버 제작
- 제출한 숙제 같이 해보기

# 실습

- 클라이언트 :
  - Overlapped I/O 클라이언트를 그대로 사용
- 서버 :
  - IOCP 사용
  - 다중 접속 처리

# 기본 프로그래밍


클라이언트
서버
소켓 생성
연결
보내기/받기
소켓 끊기
소켓 생성
Bind
Listen
Accept
보내기/받기
소켓 끊기

# IOCP

- IOCP 서버 작동 순서
  1. 초기화
     - IOCP 핸들 생성
  2. Worker thread 생성
  3. Accept Loop 실행

# IOCP

- Accept Loop
  - 새로 접속해 오는 클라이언트를 IOCP로 넘기는 역할
  - 무한 루프를 돌면서
    - Accept() 호출
    - 새 클라이언트가 접속했으면 **클라이언트 정보 구조체**를 만든다.
    - IOCP에 소켓을 등록한다.
      - Send/recv가 IOCP를 통해 수행됨
    - WSARecv()를 호출한다.
      - Overlapped I/O recv 상태를 항상 유지해야 한다.

# IOCP

- 클라이언트 정보 구조체
  - IOCP에서 오고 가는 것은 completion_key와 overlapped I/O pointer 뿐
  - GetQueuedCompletionStatus를 받았을때 어느 클라이언트인지 가장 쉽게 아는 법?
    - Completion_key를 구조체 배열의 index로 한다.
  - Overlapped I/O pointer를 확장한다 던데?
    - 하나의 클라이언트당 recv용으로 overlapped I/O 구조체가 1개 필요.

# IOCP

- Overlapped I/O pointer를 확장
  - Overlapped I/O 구조체 자체는 쓸만한 정보가 없다.
  - 따라서 정보들을 더 추가할 필요가 있다.
    - 뒤에 추가하면 IOCP는 있는지 없는지도 모르고 에러도 나지 않는다. (pointer만 왔다 갔다 하므로)
  - 꼭 필요한 정보
    - 지금 이 I/O가 send인지 recv인지????
  - 추가로 필요한 정보
    - 실제 데이터가 저장되는 버퍼 (모아서 관리하면 편리, new/delete를 따로 하지 않아도 됨)

# IOCP

- WorkerThread
  - 무한루프
    - GetQueuedCompletionStatus를 부른다.
    - 에러처리/접속종료처리를 한다.
    - Send/Recv처리를 한다.
      - **확장** Overlapped I/O 구조체를 유용하게 사용한다.
      - Recv
        - 패킷이 다 왔나 검사 후 다 왔으면 패킷 처리
        - 여러 개의 패킷이 한번에 왔을 때 처리
        - 계속 Recv호출
      - Send
        - 구조체 및 버퍼 Free

# IOCP 서버 구현

- 먼저 할 일
  - 다중 접속 관리
    - 클라이언트 접속 시 마다 ID 부여
    - ID는 소켓 값

# IOCP 서버 구현

- Recv의 구현

Start :
받은 데이터를 출력
클라이언트에 그대로 다시 전송
Recv를 호출

# IOCP 서버 구현

- overapped 구조체의 확장

```
struct OVER_EX {
        WSAOVERLAPPED over;
        WSABUF  wsabuf[1];
        char    net_buf[MAX_BUFFER];
        bool    is_recv;
};
```

# IOCP 서버 구현

- Worker Thread의 구현

```
void do_worker()
{
   while (true) {
      DWORD num_byte;
      ULONG key;
      PULONG p_key = &key;
      WSAOVERLAPPED *p_over;

      GetQueuedCompletionStatus(g_iocp, &num_byte, p_key, &p_over, INFINITE);

      OVER_EX *over_ex = reinterpret_cast<OVER_EX *> (p_over);

      if (true == over_ex->is_recv) {
            /* RECV 완료 처리 */
      }
      else {
            /* SEND 완료 처리 */
      }
   }
}
```

# IOCP 서버 구현

- Recv 완료의 구현

```
SOCKET client_s = static_cast<SOCKET>(key);
if (num_byte == 0) {
        closesocket(client_s);
        clients.erase(client_s);
        continue;
}  // 클라이언트가 closesocket을 했을 경우
over_ex->net_buf[num_byte] = 0;
cout << "From client[" << client_s << "] : ";
cout << over_ex->net_buf << " (" << num_byte << ") bytes)\n";

OVER_EX *send_over = new OVER_EX;
memset(send_over, 0x00, sizeof(OVER_EX));
send_over->is_recv = false;
memcpy(send_over->net_buf, over_ex->net_buf, num_byte);
send_over->wsabuf[0].buf = send_over->net_buf;
send_over->wsabuf[0].len = num_byte;
WSASend(client_s, send_over->wsabuf, 1, 0, 0, &send_over->over, 0);
DWORD flags = 0;
memset(&over_ex->over, 0x00, sizeof(WSAOVERLAPPED));
WSARecv(client_s, over_ex->wsabuf, 1, 0, &flags, &over_ex->over, 0);
```

# IOCP 서버 구현

- Send완료의 구현

```
SOCKET client_s = static_cast<SOCKET>(key);

if (num_byte == 0) {
        closesocket(client_s);
        clients.erase(client_s);
        delete p_over;
        continue;
}  // 클라이언트가 closesocket을 했을 경우

OVER_EX *over_ex = reinterpret_cast<OVER_EX *> (p_over);
cout << "TRACE - Send message : [" << client_s << "] "
     << over_ex->net_buf
     << " (" << num_byte << " bytes)\n";

delete p_over;
```

# 에러 처리

- 네트워크 관련 에러 검출

```
void error_display(char *msg, int err_no )
{
	WCHAR *lpMsgBuf;
	FormatMessage(
		FORMAT_MESSAGE_ALLOCATE_BUFFER |
		FORMAT_MESSAGE_FROM_SYSTEM,
		NULL, err_no,
		MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT),
		(LPTSTR)&lpMsgBuf, 0, NULL );
	std::cout << msg;
	std::wcout << L"에러 " << lpMsgBuf << std::endl;
	while(true);
	LocalFree(lpMsgBuf);
}
```

# 3번째 시간

- 네트워크 관련 에러 검출

```
int ret = WSASend(g_clients[cl].client_socket,
        &over->wsabuf, 1, NULL, 0,
        &over->over, NULL);
if (0 != ret) {
   int err_no = WSAGetLastError();
   if (WSA_IO_PENDING != err_no)
      error_display("Error in SendPacket:", err_no);
}
```

# 3번째 시간(2019-화목)

- 한글이 나오지 않는데???
  - 다음 추가

```
std::wcout.imbue(std::locale("korean");
```

  - VisualStudio -> 솔루션탐색기 -> 프로젝트 -> 오른클릭 -> 설정 -> 구성속성 -> 일반 -> 프로젝트 기본값 -> 문자 집합 -> 유니코드 문자 집합 사용

# 3번째 시간(2019-화목)

- WSARecv 에러 10055

| | |
|---|---|
| **WSAENOBUFS** 10055 | No buffer space available. An operation on a socket could not be performed because the system lacked sufficient buffer space or because a queue was full. |

  - 하나의 소켓에 WSARecv를 여러 번 호출했을 때 생기는 error
  - 원인 : send_packet을 할 때 is_recv = false를 안함.
  - 문제점 : Windows에서 표시하는 에러메세지의 엉뚱함.

# 3번째 시간(2019-수목)

- 서버의 오류
  - Clients[id].over_ex.dataBuffer.buf가 엉뚱한 값을 가짐
  - 원인
    - clients[new_id] = SOCKETINFO{clientSocket};
    - COPY constructor이므로 내부 값들의 주소가 변경됨
  - 해결책
    - Emplace로 생성 : 될수도 있고 안될 수도 있다.
    - Over_ex를 new로 생성 : 코드가 복잡
    - Clients를 map이 아닌 배열로 관리
      - Null 생성자를 추가 해야 한다.

# 3번째 시간

- 문제해결
  - Accept할 때 INVALID_HANDLE이 나옴
    - 주소 구조체의 크기를 제대로 넣지 않음
  - 체스말이 이동하지 않음
    - GQCS에서 패킷을 받지 못함
    - **WSARecv시 초기화 한 flag을 넣어줘야 함**
    - WSASend시 WSABUF.len을 제대로 세트해야 한다.
    - 접속종료 처리시 continue;
    - Accept에서 WSARecv시 BUFCOUNT
    - MAX, MIN 오류
    - WSA_IO_PENDING 처리하지 않음
    - WSABUF의 크기는 1 이다.

# OLD_ERROR

- listen socket도 OVERLAPPED 소켓으로.
- Bind()대신 ::bind()

# 3번째 시간

- Error의 원인
  - WSASend/WSARecv
    - 버퍼 카운트는 1로 한다.!!!

# 3번째 시간

- Error의 원인
  - WSASend/WSARecv
    - 버퍼 카운트는 1로 한다.!!!

# 멀티 쓰레드

- IOCP에서 멀티쓰레드 지원
  - 여러 쓰레드에서 동시에 GQCS를 호출할 수 있음.
- 하지만 Single Thread에서도 동작 가능
  - Accept도 IOCP를 통해서 받아야지만 다중 접속이 가능

# Single Thread IOCP 서버

```
…
#include <WS2tcpip.h>
#include <MSWSock.h>
#pragma comment(lib, "Ws2_32.lib")
#pragma comment(lib, "Mswsock.lib")
…
enum OPERATION { OP_RECV, OP_SEND, OP_ACCEPT};
struct OVER_EX {
                        WSAOVERLAPPED over;
                        WSABUF                  wsabuf[1];
                        char                    net_buf[MAX_BUFFER];
                        OPERATION  op;
};
…
int main()
{
…
                        g_iocp = CreateIoCompletionPort(INVALID_HANDLE_VALUE, NULL, NULL, 0);
                        CreateIoCompletionPort(reinterpret_cast<HANDLE>(listenSocket), g_iocp, listenSocket, 0);

                        SOCKET clientSocket1 = WSASocket(AF_INET, SOCK_STREAM, 0, NULL, 0, WSA_FLAG_OVERLAPPED);
                        OVER_EX *accept_over1 = new OVER_EX;
                        memset(accept_over1, 0, sizeof(OVER_EX));
                        accept_over1->op = OP_ACCEPT;

                        AcceptEx(listenSocket, clientSocket1, accept_over1->net_buf, NULL, sizeof(SOCKADDR_IN) + 16,
                                        sizeof(SOCKADDR_IN) + 16, NULL, &accept_over1->over);
                        CreateIoCompletionPort(reinterpret_cast<HANDLE>(clientSocket1), g_iocp, clientSocket1, 0);
…

                        while (true) {
                                        switch (over_ex->op) {
                                        case OP_ACCEPT: {
                                                        SOCKET clientSocket = clientSocket1;

                                                        clients[clientSocket] = SOCKETINFO{};
                                                        memset(&clients[clientSocket], 0, sizeof(struct SOCKETINFO));
                                                        clients[clientSocket].socket = clientSocket;
                                                        clients[clientSocket].recv_over.wsabuf[0].len = MAX_BUFFER;
                                                        clients[clientSocket].recv_over.wsabuf[0].buf = clients[clientSocket].recv_over.net_buf;
                                                        clients[clientSocket].recv_over.op = OP_RECV;
                                                        DWORD flags = 0;
                                                        CreateIoCompletionPort(reinterpret_cast<HANDLE>(clientSocket), g_iocp, clientSocket, 0);
                                                        WSARecv(clientSocket, clients[clientSocket].recv_over.wsabuf, 1, NULL,
                                                                        &flags, &(clients[clientSocket].recv_over.over), NULL);

                                                        clientSocket1 = WSASocket(AF_INET, SOCK_STREAM, 0, NULL, 0, WSA_FLAG_OVERLAPPED);
                                                        memset(&accept_over1->over, 0, sizeof(WSAOVERLAPPED));
                                                        AcceptEx(listenSocket, clientSocket1, accept_over1->net_buf, NULL, sizeof(SOCKADDR_IN) + 16,
                                                                        sizeof(SOCKADDR_IN) + 16, NULL, &accept_over1->over);
                                                        CreateIoCompletionPort(reinterpret_cast<HANDLE>(clientSocket1), g_iocp, clientSocket1, 0);
                                        }
                                                                                                                                                        break;

                                        case OP_RECV: {
                                        }
                                                                                                                                break;

                                        case OP_SEND: {
                                        }
                                                                                                                                break;

                                        }
                        }
                        closesocket(listenSocket);
                        WSACleanup();
}
```

# 정리

- IOCP MMOG 서버의 뼈대 완성
- 완벽하지 않음
  - 멀티 쓰레드 버그
  - 최적화 필요
  - id 재사용 금지
  - 등등등