class ServerSocket

{

protected:

static const int CLNT\_SIZE = 512;

public:

SOCKET servSocket;

SOCKADDR\_IN servAddr;

char \* port;

public:

virtual bool ConnectServer()=0;

};

TCP와 UPD 서버 소켓을 만들기 위한 최상위 클래스

추상 클래스입니다.

class TCPServerSocket : public ServerSocket

{

public:

TCPServerSocket(char \* p);

~TCPServerSocket();

virtual bool ConnectServer();

};

TCPServerSocket을 만듭니다.

초기화하며 socket()을 호출합니다.

ConnectServer()를 호출해 bind()와 listen()을 호출합니다.

class SocketEx

{

public:

SOCKET socket;

SOCKADDR\_IN sockAddr;

int szSockAddr;

public:

~SocketEx();

bool Accept(ServerSocket\* servSock);

int Receive(char \* message, int bufsize);

int Send(char \* message, int len);

};

Accept() 함수는 accept() 함수에 대한 래퍼 함수 입니다.

Receive() 함수는 recv() 함수에 대한 래퍼 함수 입니다.

Send() 함수는 send() 함수에 대한 래퍼 함수 입니다.

class TCPServerManager

{

public:

static const int CLNT\_SIZE = 512;

static const int BUF\_SIZE = 1024;

private:

TCPServerManager(const TCPServerManager& sv) {}

TCPServerManager& operator=(const TCPServerManager& sv) {}

public:

SocketEx\* clntSocks[CLNT\_SIZE];

HANDLE hMutex;

int clntCnt;

char message[BUF\_SIZE];

private:

WSADATA wsaData;

public:

TCPServerManager();

~TCPServerManager();

void StartThread(SocketEx \* sock, unsigned (\_stdcall\*threadMain)(void\*));

};

서버를 만들 때 부가적으로 필요한 상수와 WSADATA, 멀티스레딩을 위한 뮤텍스, 메시지 전송에 필요한 메시지 값 등을 핸들링합니다.

StartThread() 함수를 호출해 스레드를 생성합니다.

ThreadMain을 직접 디자인해 매개변수 값으로 넣어주십시오.

class Application

{

public:

static TCPServerManager \* manager;

};

TCPServerManager를 static 지역변수로 선언해 서버에서 만들어지는 유일한 manager를 등록합니다. 내부적으로 작동하기 때문에 응용프로그램을 만들 때에는 사용할 경우가 없습니다.