데이터 통신
Socket Programming 동영상학습

컴퓨터 전공
김다빈 2015004375

Socket Programming 동영상학습

1. 서론

소켓 프로그래밍이 무엇인지 고민하기 전에 소켓이란 무엇인가? 소켓은 두 프로그램이 네트워크를 통해 서로 통신을 수행할 수 있도록 양쪽에 생성되는 링크의 단자이다. 두 소켓이 연결되면 서로 다른 프로세스끼리 데이터를 전달할 수 있다는 이야기다. 그렇다면, 소켓 프로그래밍이란? “to build any networked application”. 즉, 서로 데이터를 주고받을 수 있는 어플리케이션을 만드는 것이다. 이제, ‘소켓’에 대해 대략적으로 알게 되었으니, ‘소켓 프로그래밍’ 에 대해 좀 더 알아보려 한다.

2. Socket Programming

1) 서버와 클라이언트

소켓은 위에서 언급했듯이, 서버 소켓과 클라이언트 소켓이 존재한다. 서버는 정보를 제공하는 모든 기기를 뜻하고, 클라이언트는 정보를 찾는 모든 기기를 뜻한다. 소켓 프로그래밍에서는 거의 모든 통신들이 클라이언트-서버 모델을 베이스로 두고있기 때문에, 서버와 클라이언트가 서로 어떻게 작용하는 지는 매우 중요한 부분이다. 간단하게 **서버와 클라이언트의** 소켓통신 구조의 흐름을 전화기를 예로 들어 설명하자면, 먼저 서버 소켓과 클라이언트 소켓은 서로 전화를 하고 싶은 상황이다. 그럼 서버 소켓은 전화기를 구입(소켓 생성)해서, 전화번호를 할당해 주고(주소 및 포트 할당), 케이블을 연결해서(연결요청 대기), 전화가 왔을때 전화를 받을 것이다(연결 수락). 또 전화를 걸고 싶은 클라이언트 소켓은 전화기를 구입하고(소켓생성), 서버에게 전화를 걸게 될 것이다(연결요청). 이렇게, 클라이언트와 서버는 서버가 먼저 클라이언트를 받을 준비를 마친후 클라이언트가 연결을 요청하기 전까지 대기하다가 요청이 들어오면 요청을 수행하게 된다.

2) Socket Coding

소켓 프로그래밍을 진행하기 위해선 몇가지 중요한 함수들이 있다. 그 함수들이 어떤 일을 하고 쓰이는지 설명해 보려한다.

- Socket()

소켓을 생성하는 함수. 즉, 호스트가 통신을 하기 위해 필요한 리소스를 할당하는 것을 의미한다. 함수 원형은

int Socket(int domain, int type, int protocol);

로, domain이 생성할 소켓이 통신을 하기 위해 사용할 프로토콜 체계를 설정해주고, type가 소켓이 데이터를 전송하는 데 있어서, 사용하게 되는 전송 타입을 설정해 주고, protocol을 통해 두 호스트간에 통신을 하는 데 있어서 특정 프로토콜을 지정해 주게 된다.

socket(PF_INET,SOCK_STREAM,0);

이런식으로 서버 또는 클라이언트 소켓이 생성되게 된다.

- Bind()

주소 정보 구조체의 변수를 설정해 주고 초기화에 이은 소켓에 주소정보를 할당해주는 함수이다. 즉, IP 및 포트를 할당해주는 함수이다. 함수 원형은

int Bind(int sockfd, struct sockaddr *my_addr, socklen_t addrlen);

로, sockfd는 주소를 할당하고자 하는 소켓의 파일 디스크립터, my_addr는 할당하고자 하는 주소 정보를 지니고 있는 sockaddr 구조체 변수의 포인터 인자 값을 나타내고, addrlen은 인자로 전달된 주소 정보 구조체의 길이를 알려준다.

```
struct sockaddr_in my;

my.sin_family = PF_INET;
my.sin_port = htons(80);
my.sin_addr.s_addr = INADDR_ANY;
bzero(&my, 8)

bind(sock, (struct sockaddr *)&my, sizeof(my));
```

이런식으로 bind함수는 사용이 되며, 각 주소정보를 할당해 주게 된다.

- Listen()

클라이언트로 부터의 연결 요청을 처리할 수 있는 상태를 뜻하는 함수이다. 즉, 연결요청대기 상태 라고 볼 수있다. 함수 원형은

int Listen(int sock, int backlong);

로, sockfd는 클라이언트로 부터 연결 요청을 받아들이기 위한 소켓 파일 디스크립터 이고, backlog는 연결요청 대기 큐의 크기를 설정하기 위한 파라미터이다.

Listen(sock,10);

이런식으로 사용되게 되며, 이것은 최대 10개의 연결이 대기상태에 있을 수 있음을 의미한다.

- Accept()

클라이언트로 부터의 연결 요청을 수락하는 함수로, 이 함수가 성공하면 데이터를 주고 받을 수 있는 상태가 되게 된다. 함수 원형은

int Accept(int socket, (struct sockaddr *)&client, socklen_t *client_len);

로, socket는 서버 소켓의 파일 디스크립터, client는 연결 요청을 수락 할 클라이언트의 주소 정보를 저장할 변수의 포인터, client_len은 client가 가리키는 구조체의 크기를 저장하고 있는 변수의 포인터 이다.

```
struct sockaddr_in client;
int len = sizeof(client);
Accept(sock, (struct sockaddr *)&client, &len);
```

위의 예와 같이 사용되며, 위의 함수가 성공시 파일 디스크립터가 새로 생성되게 된다.

- Connect()

클라이언트 소켓을 생성하고, 서버로 연결을 요청하는 함수이다. 함수의 원형은

int Connect(int sock, (struct sockaddr *)&server_addr, socklen_t len);

로, sock는 클라이언트 소켓의 파일 디스크립터, server_addr는 연결 요청을 보낼 서버의 주소 정보를 지닌 구조체 변수의 포인터를 가르키고, len은 server_addr 포인터가 가리키는 주소 정보 구조체 변수의 크기를 의미한다.

Connect(sock, (struct sockaddr *)server_addr,len);

위와 같이 사용되며, 연결 요청이 서버에 의해 수락되거나 오류가 발생해서 연결 요청이 중단되면 함수가 리턴을 한다. 연결 요청이 바로 이루어지지 않은 상태(큐에서 대기)에는 리턴되지 않고 블로킹 상태에 존재한다.

- 그 외 함수

그 외에도 **Send(), Resv(), Read(), Write(), Close()** 등의 함수가 존재하는 데, Close이외의 함수는 데이터를 보내고 받아오기 위해 쓰이는 함수이다. 데이터를 보내고 받아올때는 바이트값으로 주고 받게 되며, 밑에와 같은 형식으로 연결할 소켓을 정해주고 4개의 함수 모두 사용되게 된다.

```
char send_buffer[1024];
char recv_buffer[1024];
int sent_bytes;
int recvd_bytes;

sent_bytes = send(sock, send_buffer, 1024, 0);
recvd_bytes = recv(sock, recv_buffer, 1024, 0);
```

또, Close 함수 같은 경우는 서버와 클라이언트 간의 통신을 종료하고 소켓을 닫는 함수로, 효과적으로 소켓을 닫을 수 있다. Close안에는 닫아야 할 소켓이 들어가게 된다.

3. 결론

소켓은 위와 같은 함수를 이용하여 구현되게 되고, 클라이언트-서버 소켓 통신을 통해 다양한 통신을 구현할 수 있다. 물론 이것이 소켓 프로그래밍의 전부는 아니지만, 소켓 프로그래밍의 튜토리얼, 가장 기본적인 내용으로 프로그래밍을 하고 싶다면 반드시 알아야 하는 부분에 속해있다고 볼 수 있다. 클라이언트-서버 소켓의 구조 흐름을 잘 이해하고, 더 나아간 소켓 프로그래밍을 할 수 있길 바란다.

4. 느낀점

소켓에 대해서 처음 접한 것은 시스템 프로그래밍이었습니다. 그때 처음 멋모르고 봤던 소켓 프로그래밍은 사실 잘 이해되지 않았으나, 지금 두번째 학습을 통하여 조금 더 제대로 이해하게 된 것 같습니다. 소켓 프로그래밍에 관련된 기본적인 지식이지만, 어떻게 서로 주고받고 상호간 어떻게 작용하는지 좀 더 자세히 알게될 수 있는 시간이었던 것 같습니다. 시스템 프로그래밍 시간에는 프로그래밍, 리눅스에 더 집중했다면 이번에는 소켓의 '통신'에 조금 더 집중해 보고 싶습니다.