HALLYM UNIVERSITY

CIC Convergence Information & Communications

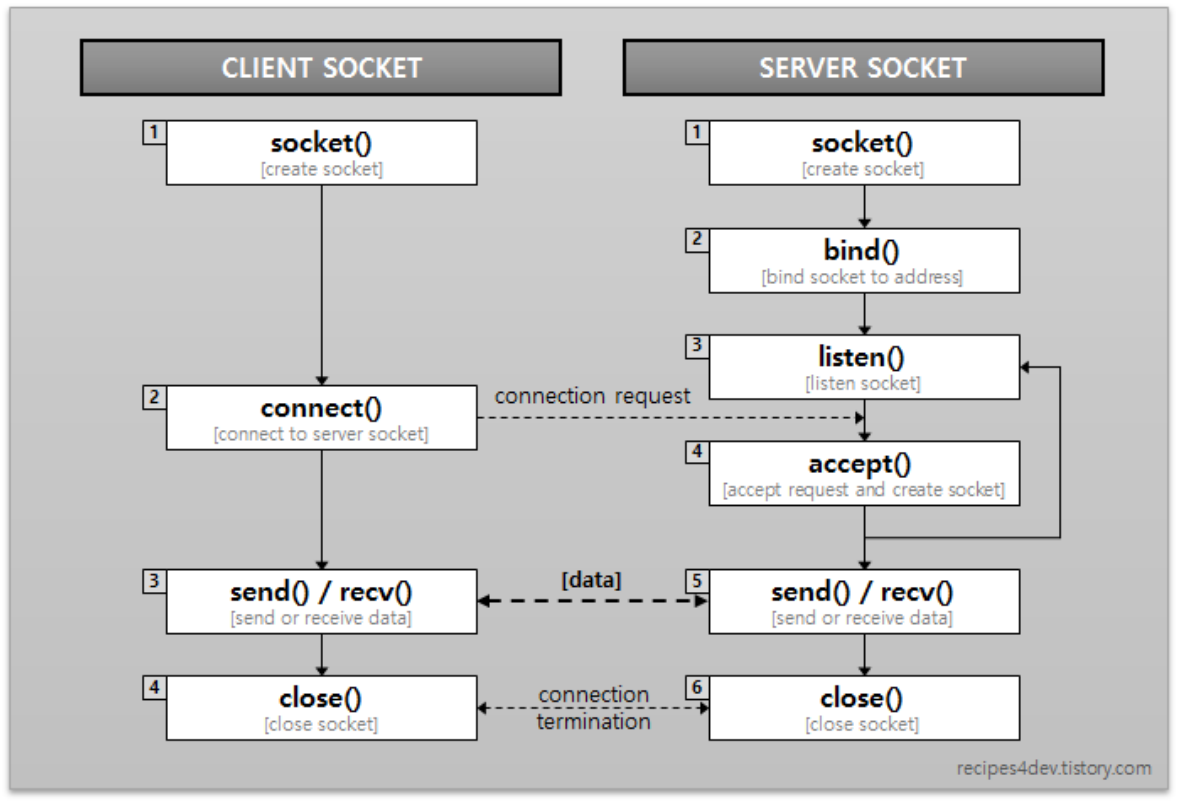
(Operating System) Practice -11-

Socket

Index

Socket

- **Socket**
  - 네트워크 혹은 인터넷 통신의 종착점 (end-point)를 의미함
  - 네트워크를 위해 사용되는 다양한 통신 프로토콜에 대한 이해 없이, end-point interface를 제공하여 연결 및 데이터 송신을 가능하게 함



Socket

- **Socket Functions**
  - socket → 소켓 생성

유형	내용
헤더	#include <sys/socket.h>
함수	int socket(int domain, int type, int protocol);
매개변수	domain: IP 프로토콜 버전 type: 연결 종류 (TCP or UDP) protocol: 사용하는 통신 프로토콜 (0 이면 자동선택)
반환 값	성공: socket 파일 식별자 실패: -1

  - bind → 생성한 소켓에 IP 주소 & 포트번호 부여

유형	내용
헤더	#include <sys/socket.h>
함수	Int bind(int socket_fd, SOCKADDR_ARG addr, socklen_t len);
매개변수	socket_fd: 소켓 파일 식별자 addr: 소켓의 IP 주소, 포트 등 소켓의 정보를 담은 구조체 len : addr의 크기
반환 값	성공: 0 실패: -1

Socket

- **Socket Functions**
  - listen → 소켓 접속 요청을 기다림

유형	내용
헤더	#include <sys/socket.h>
함수	int listen(int socket_fd, int n);
매개변수	socket_fd: 소켓 파일 식별자 n: n개의 접속 요청이 queue에 담겨서 순서대로 처리됨, 그 이상의 요청은 거절됨
반환 값	성공: 0 실패: -1

  - accept → 소켓 접속 요청을 허가함

유형	내용
헤더	#include <sys/socket.h>
함수	Int accept(int socket_fd, SOCKADDR_ARG &addr, socklen_t len);
매개변수	socket_fd: 소켓 파일 식별자 addr: 접속한 Client의 IP 주소, 포트 등 소켓의 정보를 담을 구조체 len : addr의 크기
반환 값	성공: 0 실패: -1

Socket

- **Socket Functions**
  - connect → 소켓을 통해 서버에 접속함

유형	내용
헤더	#include <sys/socket.h>
함수	int connect(int socket_fd, , SOCKADDR_ARG addr, socklen_t len);
매개변수	socket_fd: 소켓 파일 식별자 addr: 접속할 서버의 IP 주소, 포트 등 소켓의 정보를 담을 구조체 len : addr의 크기
반환 값	성공: 0 실패: -1

  - send → 소켓을 통해 데이터를 전송함

유형	내용
헤더	#include <sys/socket.h>
함수	Int send(int socket_fd, void buf, size_t size, int flags);
매개변수	socket_fd: 소켓 파일 식별자 buf: 보내고자하는 데이터 size: buf의 크기 flags: 전송 옵션 (0 → 일반적인 전송)
반환 값	성공: 보낸 데이터의 크기 실패: -1

Socket

- **Socket Functions**
  - recv → 소켓을 통해 서버에 접속함

유형	내용
헤더	#include <sys/socket.h>
함수	Int recv(int socket_fd, void &buf, size_t size, int flags);
매개변수	socket_fd: 소켓 파일 식별자 buf: 수신할 데이터를 담을 변수 size: buf의 크기 flags: 전송 옵션 (0 → 일반적인 전송)
반환 값	성공: 수신한 데이터의 크기 실패: -1

  - close → FD를 종료함

유형	내용
헤더	#include <unistd.h>
함수	Int close(int socket_fd);
매개변수	socket_fd: 파일 식별자
반환 값	성공: 0 실패: -1

Socket Practice

- **Socket Practice (Server)**

```c
#include <unistd.h>
#include <stdio.h>
#include <sys/socket.h>
#include <stdlib.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <string.h>

void child_proc(int sock);
struct sending_packet receive_sock(int sock);
void send_sock(int sock, struct sending_packet pck);

 struct sending_packet{
    char sender[1024];
    char receiver[1024];
    char msg[1024];
};

void main(){
    int s_sock_fd, new_sock_fd;
    struct sockaddr_in s_address, c_address;
    int addrlen = sizeof(s_address);
    int check;

    // 서버 소켓 선언
    s_sock_fd = socket(AF_INET, SOCK_STREAM, 0);// AF_INET: IPv4 인터넷 프로토콜, SOCK_STREAM: TCP/IP
    if (s_sock_fd == -1){
        perror("socket failed: ");
        exit(1);
    }

    // 서버 소켓 셋팅
    memset(&s_address, '0', addrlen); // s_address의 메모리 공간을 0으로 셋팅 / memset(메모리 공간 시작점, 채우고자 하는 값, 메모리 길이)
    s_address.sin_family = AF_INET; // IPv4 인터넷 프로토콜
    s_address.sin_addr.s_addr = inet_addr("127.0.0.1"); // 내부 IP 주소
    s_address.sin_port = htons(8080); // 포트번호 지정
    check = bind(s_sock_fd, (struct sockaddr *)&s_address, sizeof(s_address));
    if(check == -1){
        perror("bind error: ");
        exit(1);
    }
```

Socket Practice

- **Socket Practice (Server)**

```
43      // 아래부터 소켓 통신 시작
44      while(1){
45          // 클라이언트의 접속을 기다림 (한 클라이언트가 서버 소켓을 사용하고 있으면, 다른 클라이언트는 대기열에서 기다리게 됨)
46          check = listen(s_sock_fd, 16);
47          if(check==-1){
48              perror("listen failed: ");
49              exit(1);
50          }
51
52          // 클라이언트의 접속을 허가함 -> 접속에 성공한 클라이언트와의 통신을 위해 새로운 소켓을 생성함
53          new_sock_fd = accept(s_sock_fd, (struct sockaddr *) &c_address, (socklen_t*)&addrlen);
54          if(new_sock_fd<0){
55              perror("accept failed: ");
56              exit(1);
57          }
58
59          //접속에 성공한 클라이언트와의 통신은 자식 프로세스를 통해서 수행함
60          if(fork()>0){ // 자식 프로세스
61              child_proc(new_sock_fd);
62          }
63          else{ // 부모 프로세스
64              // 부모 프로세스는 새로운 소켓을 유지할 필요 없음.
65              close(new_sock_fd);
66          }
67      }
68  }
```

Socket Practice

- **Socket Practice (Server)**

```
70  void child_proc(int sock){
71      struct sending_packet pck;
72      int flag = 0;
73
74      while(1){
75          pck = receive_sock(sock); // 소켓을 통해 데이터 수신
76          printf("%s: %s\n", pck.sender, pck.msg); // 수신받은 데이터 출력
77          if (strcmp(pck.msg,"quit")==0){
78              flag = -1;
79          }
80
81          // packet 정보 수정
82          sprintf(pck.msg, "Message received!");
83          sprintf(pck.sender, "Server");
84          sprintf(pck.receiver, "Client");
85
86          send_sock(sock, pck); // 데이터 송신
87          if (flag == -1){
88              break;
89          }
90      }
91      shutdown(sock, SHUT_WR); // 소켓 Close
92      exit(0);
93  }
94
95  struct sending_packet receive_sock(int sock){
96      struct sending_packet pck;
97      recv(sock, (struct sending_packet*)&pck, sizeof(pck),0);
98      return pck;
99  }
100
101 void send_sock(int sock, struct sending_packet pck){
102     send(sock, (struct sending_packet*)&pck, sizeof(pck), 0);
103 }
```

Socket Practice

- **Socket Practice (Client)**

```c
#include <stdio.h>
#include <sys/socket.h>
#include <stdlib.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <string.h>

 struct sending_packet{
    char sender[1024];
    char receiver[1024];
    char msg[1024];
};

void main(){
    struct sockaddr_in s_addr;
    int sock_fd;
    char buffer[1024] = {0};
    struct sending_packet pck;
    int check;

    // 소켓 선언
    sock_fd = socket(AF_INET, SOCK_STREAM, 0);
    if (sock_fd <= 0){
        perror("socket failed: ");
        exit(1);
    }

    // 소켓 셋팅
    memset(&s_addr, '0', sizeof(s_addr));
    s_addr.sin_family = AF_INET;
    s_addr.sin_port = htons(8080);
    check = inet_pton(AF_INET, "127.0.0.1", &s_addr.sin_addr);
    if (check<=0){
        perror("inet_pton failed: ");
        exit(1);
    }
    check = connect(sock_fd, (struct sockaddr *) &s_addr, sizeof(s_addr));
    if(check < 0){
        perror("connect failed: ");
        exit(1);
    }
```

Socket Practice

- **Socket Practice (Client)**

```
42      int flag =0;
43      while(1){
44          // 명령어 창으로 부터 문자열 입력받기
45          scanf("%s", buffer);
46          sprintf(pck.msg, "%s", buffer);
47          sprintf(pck.sender, "Client");
48          sprintf(pck.receiver, "Server");
49
50          if (strcmp(pck.msg,"quit")==0){ // 프로세스 종료 조건
51              flag = -1;
52          }
53
54          // 소켓을 통해 데이터 송신
55          send(sock_fd, (struct sending_packet*)&pck, sizeof(pck), 0);
56          // 소켓을 통해 데이터 수신
57          recv(sock_fd, (struct sending_packet*)&pck, sizeof(pck), 0);
58          printf("%s: %s\n", pck.sender, pck.msg);
59
60          if (flag == -1){
61              break;
62          }
63      }
64      shutdown(sock_fd, SHUT_WR); // 송신부 소켓  Close
65  }
```