IT CookBook, 쉽게 배우는 데이터 통신과 컴퓨터 네트워크(개정) 12장 연습문제 해답

1. ① AF_UNIX, ② AF_INET
2. ① sockaddr_in, ② sin_family, ③ sin_addr, ④ sin_port
3. ① (struct sockaddr *)
4. ① SOCK_STREAM, ② SOCK_DGRAM
5. ① 포트, ② 포트 번호
6. ① TCP, ② UDP
7. ① listen(), ② accept(), ③ connect()
8. ① send(), ② recv(), ③ sendto(), ④ recvfrom()
9. ① INADDR_ANY, ② IP 주소
10. ① Well-known 포트 (혹은 Well-known Address), ② IP 주소, ③ 포트 번호 (혹은 소켓 주소)

11. ①, ②, ④, ⑤

(설명③) 인터넷 주소 체계를 지원하는 sockaddr_in 구조체는 32 비트의 IP 주소와 16 비트의 포트 번호를 저장할 수 있다.

12. ①, ②, ③, ④, ⑤

13. ①, ②, ④

(설명③) accept() 함수는 보통 서버 프로그램에서 실행되며, 클라이언트의 연결 요구가 들어올 때까지 대기한다. accept() 함수에서 대기 중이던 서버는 클라이언트의 연결 요구가 들어오면 둘 사이에 연결이 설정된다.

(설명⑤) send()와 recv()는 데이터를 보내고 받는 기능을 수행하는데, 주로 연결형 서비스에서 사용된다.

14. ①, ⑤

(설명①) bind() 함수는 소켓에 주소를 부여하는 기능을 수행한다. 예를 들어서

AF_INET 도메인에서는 호스트의 IP 주소와 포트 번호 조합의 주소를 소켓에 부여한다.

(설명⑤) 일반인들의 인터넷 주소 표기는 211.223.201.30 등과 같은 형식을 취한다. 그러나 컴퓨터 내부에서는 32 비트 크기의 2진수를 사용하므로 변환 과정이 필요하다. inet_addr() 함수는 이러한 목적으로 사용한다.

15. ①, ④, ⑤

(설명②) 서버 프로세스는 교신점에 해당하는 소켓에 (IP 주소, 포트 번호) 조합을 부여하는데, 이 조합 값은 인터넷에서 유일하다.

(설명③) 일반적으로 클라이언트와 서버의 동작 과정에서 서버가 먼저 대기 상태로 들어가고, 클라이언트의 연결 요청이 있으면 연결이 설정된다.

16. ②

(설명②) 서버는 생성된 소켓에 주소를 부여하는 절차를 수행하지만, 클라이언트는 이 절차를 생략할 수 있다.

17.

소켓 주소 체계는 사용하는 프로토콜의 종류에 따라 달라진다. 따라서 운영체제에서 제공하는 통신 프로토콜의 수가 증가하면 이에 비례하여 주소 체계의 표현 방식도 증가한다. 소켓 주소의 사용은 동일한 소켓 시스템 콜을 통해 이루어지는데 의미는 같으나 형식이 다른 여러 구조체를 하나의 함수에서 수용하는 것은 문법적으로 불가능하다. 따라서 여러 소켓 구조체를 통합해 일반 구조체 하나로 정의할 필요가 있다.

18.

• socket() : 매개변수로 지정한 유형에 따라 소켓을 생성한다. 시스템 콜이 성공적으로 실행되어 소켓이 만들어지면 해당 소켓의 디스크립터를 반환하고, 이 값을 파일 I/O의 디스크립터와 똑같은 방식으로 사용한다. / socket(int domain, int type, int protocol)

• bind() : socket() 함수로 생성한 소켓에 bind() 함수가 주소를 부여한다. /bind(int s, const struct sockaddr *name, socklen_t *namelen)

• listen() : 첫 번째 매개변수로 표시한 소켓을 활성화하는 역할을 하며, 보통 서버 프로그램에서 실행된다. 시스템에서 연결을 거부하지 않고 대기할 수 있는 연결 설정 요구의 최대 수를 지정하는 데 사용한다. / listen(int s, int backlog)

• accept() : 보통 서버 프로그램에서 실행되며, 첫 번째 매개변수로 지정한 소켓으로 클라이언트의 연결 요구가 들어올 때까지 대기한다. / accept(int s, struct sockaddr *addr, socklen_t *addrlen)

• connect() : 클라이언트 프로그램에서 사용하며, 두 번째 매개변수가 가리키는 서버와 연결 설정을 시도한다. 해당 주소의 서버가 존재하지 않으면 오류 처리되고, 서버가 연결 대기 중이면 연결이 설정된다. / connect(int s, const struct sockaddr *name, socklen_t namelen)

• send() : 연결형 서비스를 제공하는 환경에서 데이터를 전송하는 역할을 수행한다. 즉, 첫 번째 매개변수로 표시한 소켓을 통해 상대방 프로세스에게 두 번째 매개변수에 보관된 데이터를 전송한다. / send(int s, const void *msg, size_t len, int flags)

• recv() : 연결형 서비스에서 데이터를 수신하는 역할을 한다. 첫 번째 매개변수로 표시되는 소켓을 통해 데이터를 수신하고, 수신한 데이터를 두 번째 매개변수에 보관한다. / recv(int s, void *buf, size_t len, int flags)

19.

• TCP 용

```
sd = socket(AF_UNIX, SOCK_STREAM, 0); /* 유닉스 도메인 연결형 서비스 */
sd = socket(AF_INET, SOCK_STREAM, 0); /* 인터넷 도메인 연결형 서비스 */
```

• UDP 용

```
sd = socket(AF_UNIX, SOCK_DGRAM, 0); /* 유닉스 도메인 비연결형 서비스 */
sd = socket(AF_INET, SOCK_DGRAM, 0); /* 인터넷 도메인 비연결형 서비스 */
```

20.

```
int sd;
struct sockaddr_in addr;

sd = socket(AF_INET, SOCK_STREAM, 0);
if(sd == -1) {
        perror("socket");
        exit(1);
}
```

```
addr.sin_family = AF_INET;
addr.sin_addr.s_addr = htonl(inet_addr("211.223.201.30"))
addr.sin_port = htons(5000);

if(bind(sd, (struct sockaddr *)&addr, sizeof(addr)) == -1) {
        perror("bind");
        exit(1);
}
```

21.

개발된 프로그램이 여러 컴퓨터에서 실행되는 경우, 각 실행 파일에 개별 호스트의 IP 주소를 지정하여 컴파일하기는 현실적으로 불가능하다. 따라서 이와 같은 상황을 고려해 INADDR_ANY라는 호스트 주소 표기 방법을 제공한다. 이는 프로그램이 실행되는 로컬 호스트 자체를 의미하기 때문에 프로그램이 실행되는 호스트의 IP 주소가 자동으로 대체된다. 결과적으로 동일한 프로그램을 여러 호스트에서 실행시키는 경우에도 호스트마다 주소 변경과 재컴파일 작업을 하지 않아도 된다.

22.

컴퓨터마다 정수형 데이터를 저장하는 기억 장소 공간의 바이트 순서가 다를 수 있다. 이 차이를 극복하기 위해 네트워크 바이트 순서라는 전송 문법 데이터 유형이 필요하다. 데이터를 전송하기 전에 개별 컴퓨터의 바이트 순서를 네트워크 바이트 순서로 변환해야 하는데, htonl()과 htons() 함수가 이를 담당한다. 데이터를 수신할 때는 반대로 네트워크 바이트 순서를 개별 컴퓨터의 바이트 순서로 변환해야 하며, ntohl()과 ntohs()함수가 담당한다.

23.

accept( ) 함수

```
int sd, new;
struct sockaddr_in addr;
struct sockaddr_in client
int client_len;
sd = socket(AF_INET, SOCK_STREAM, 0);

if(sd == -1) {
        perror("socket");
        exit(1);
```

```
}

addr.sin_family = AF_INET;
addr.sin_addr.s_addr = htonl(INADDR_ANY);
addr.sin_port = htons(5010);

if(bind(sd, (struct sockaddr *)&addr, sizeof(addr)) == -1) {
        perror("bind");
        exit(1);
}

if(listen(sd, 5) == -1) {
        perror("listen");
        exit(1);
}

while((new = accept(sd, (struct sockaddr *)&client, &client_len)) != -1) {
        if(fork() == 0) {
                /* 자식 프로세스 */
                close(sd);
                work(new); /* new를 이용해 클라이언트와 통신 */
                close(new);
                exit(0);
        }

        /* 부모 프로세스 */
        close(new);
}
```

connect( ) 함수

```
int sd;
struct sockaddr_in addr;

sd = socket(AF_INET, SOCK_STREAM, 0);
if(sd == -1) {
        perror("socket");
        exit(1);
```

```
}

addr.sin_family = AF_INET;
addr.sin_addr.s_addr = htonl(inet_addr("211.223.201.30"));
addr.sin_port = htons(5010);

if(connect(sd, (struct sockaddr *)&addr, sizeof(addr)) == -1) {
        perror("connect");
        exit(1);
}
```

24.

[클라이언트 프로그램]

```
#include<sys/types.h>
#include<sys/socket.h>

#include<netinet/in.h>

#define TIME_SERVER "211.223.201.30"
#define TIME_PORT 5010

main()
{
        int sock;
        struct sockaddr_in server;
        char buf[256];

        if ((sock = socket(AF_INET, SOCK_STREAM, 0)) == -1)
                exit(1);

        server.sin_family = AF_INET;
        server.sin_addr.s_addr = htonl(inet_addr(TIME_SERVER));
        server.sin_port = htons(TIME_PORT);

        if (connect(sock, (struct sockaddr *)&server, sizeof(server)))
                exit(1);
```

```
        if (recv(sock, buf, sizeof(buf), 0) == -1)
                exit(1);
        printf("Time information from server is %s", buf);
        close(sock);
}
```

[서버 프로그램]

```
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#define TIME_SERVER "211.223.201.30"
#define TIME_PORT 5010

main()
{
        int sock;
        struct sockaddr_in server, client;
        int server_len = sizeof(server);
        char buf[256];
        int buf_len;

        server.sin_family = AF_INET;
        server.sin_addr.s_addr = htonl(inet_addr(TIME_SERVER));
        server.sin_port = htons(TIME_PORT);

        if ((sock = socket(AF_INET, SOCK_DGRAM, 0)) == -1)
                exit(1);
        client.sin_family = AF_INET;
        client.sin_addr.s_addr = htonl(INADDR_ANY);
        client.sin_port = htons(0);
        if (bind(sock, (struct sockaddr *)&client, sizeof(client)) < 0)
                exit(1);

        buf[0] = '?'; buf[1] = '₩0';
        buf_len = sendto(sock, buf, strlen(buf) + 1, 0,
```

```
                (struct sockaddr *)&server, server_len);
        if (buf_len < 0)
                exit(1);

        buf_len = recvfrom(sock, buf, 256, 0, (struct sockaddr *)0, (int *)0);
        if (buf_len < 0)
                exit(1);
        printf("Time information from server is %s", buf);
        close(sock);
}
```

25.

[클라이언트 프로그램]

```
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#define TIME_SERVER "211.223.201.30"
#define TIME_PORT 5010

main()
{
        int sock;
        struct sockaddr_in server, client;
        int server_len = sizeof(server);
        char buf[256];
        int buf_len;

        server.sin_family = AF_INET;
        server.sin_addr.s_addr = htonl(inet_addr(TIME_SERVER));
        server.sin_port = htons(TIME_PORT);

        if ((sock = socket(AF_INET, SOCK_DGRAM, 0)) == -1)
                exit(1);
        client.sin_family = AF_INET;
        client.sin_addr.s_addr = htonl(INADDR_ANY);
```

```
        client.sin_port = htons(0);
        if (bind(sock, (struct sockaddr *)&client, sizeof(client)) < 0)
                exit(1);

        buf[0] = '?'; buf[1] = '₩0';
        buf_len = sendto(sock, buf, strlen(buf) + 1, 0,
                (struct sockaddr *)&server, server_len);
        if (buf_len < 0)
                exit(1);

        buf_len = recvfrom(sock, buf, 256, 0, (struct sockaddr *)0, (int *)0);
        if (buf_len < 0)
                exit(1);
        printf("Time information from server is %s", buf);
        close(sock);
}
```

[서버 프로그램]

```
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<time.h>

# define TIME_PORT 5010

main()
{
        int sock;
        struct sockaddr_in server, client;
        int server_len;
        int client_len = sizeof(client_len);
        char buf[256];
        int buf_len;
        time_t today;

        if ((sock = socket(AF_INET, SOCK_DGRAM, 0)) == -1)
```

```
                exit(1);
        server.sin_family = AF_INET;
        server.sin_addr.s_addr = htonl(INADDR_ANY);
        server.sin_port = htons(TIME_PORT);

        if (bind(sock, (struct sockaddr *)&server, sizeof(server)))
                exit(1);

        while(1)
        {
                buf_len = recvfrom(sock, buf, 256, 0,
                (struct sockaddr *)&client, &client_len);
                if (buf_len < 0)
                        exit(1);
                printf("Server: Got %s₩n", buf);

                time(&today);
                strcpy(buf, ctime(&today));
                sendto(sock, buf, strlen(buf) + 1, 0,
                (struct sockaddr *)&client, client_len);
        }
}
```