13장 소켓

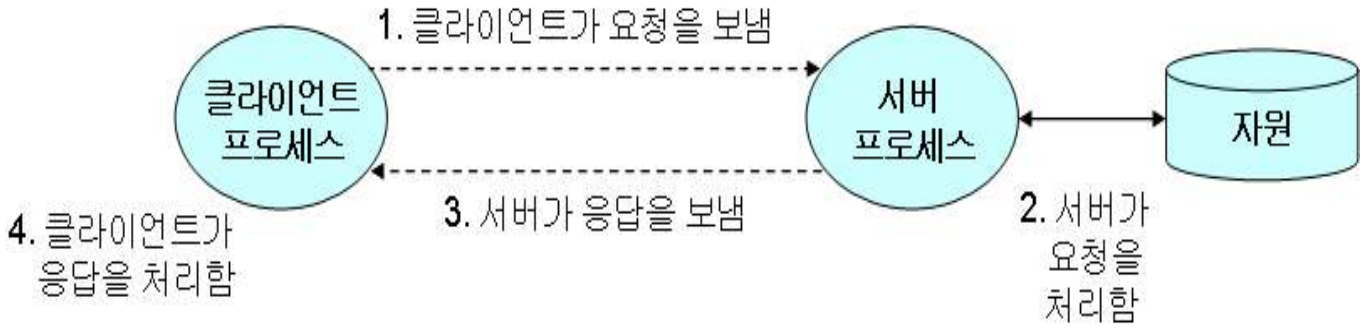
숙명여대 창병모

13.1 소켓

클라이언트-서버 모델

- 네트워크 응용 프로그램
  - 클리이언트-서버 모델을 기반으로 동작한다.

- 클라이언트-서버 모델
  - 하나의 서버 프로세스와 여러 개의 클라이언트로 구성된다.
  - 서버는 어떤 자원을 관리하고 클라이언트를 위해 자원 관련 서비스를 제공한다.

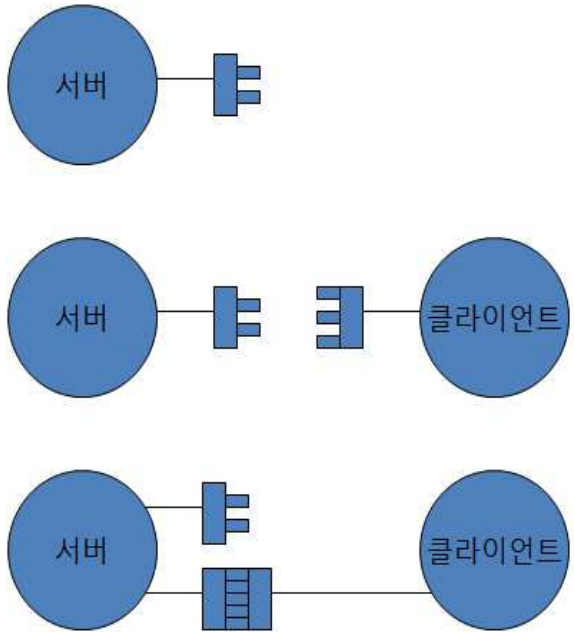


소켓의 종류

- 소켓
  - 네트워크에 대한 사용자 수준의 인터페이스를 제공
  - 소켓은 양방향 통신 방법으로 클라이언트-서버 모델을 기반으로 프로세스 사이의 통신에 매우 적합하다.
- 유닉스 소켓(AF_UNIX)
  - 같은 호스트 내의 프로세스 사이의 통신 방법
- 인터넷 소켓(AF_INET)
  - 인터넷에 연결된 서로 다른 호스트에 있는 프로세스 사이의 통신 방법

소켓 연결

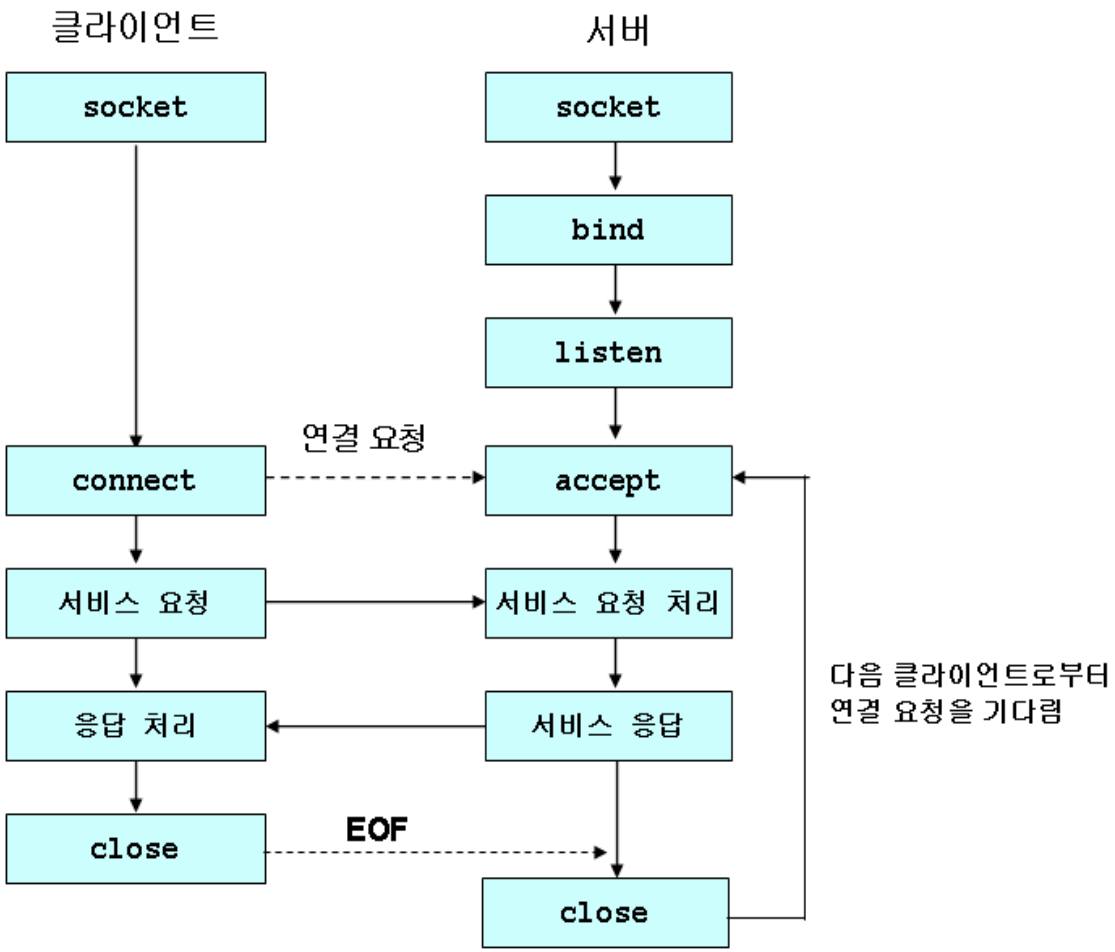
1. 서버가 소켓을 만든다.
2. 클라이언트가 소켓을 만든 후 서버에 연결 요청을 한다.
3. 서버가 클라이언트의 연결 요청을 수락하여 소켓 연결이 이루어진다.



소켓 연결 과정

- 서버
  1. socket() 호출을 이용하여 소켓을 만들고 이름을 붙인다.
  2. listen() 호출을 이용하여 대기 큐를 만든다.
  3. 클라이언트로부터 연결 요청을 accept() 호출을 이용하여 수락
  4. 소켓 연결이 이루어지면
     - 자식 프로세스를 생성하여
     - 클라이언트로부터 요청 처리
     - 클라이언트에게 응답한다.
- 클라이언트
  1. socket() 호출을 이용하여 소켓을 만든다.
  2. connection() 호출을 이용하여 서버에 연결 요청을 한다.
  3. 서버가 연결 요청을 수락하면 소켓 연결이 만들어진다.
  4. 서버에 서비스를 요청하고 서버로부터 응답을 받아 처리

소켓 연결 과정


클라이언트
서버
socket
socket
bind
listen
연결 요청
connect
accept
서비스 요청
서비스 요청 처리
응답 처리
서비스 응답
다음 클라이언트로부터
연결 요청을 기다림
close
EOF
close

소켓 만들기

int socket(int domain, int type, int protocol)

소켓을 생성하고 소켓을 위한 파일 디스크립터를 리턴, 실패하면 -1을 리턴

- 인터넷 소켓

  fd = socket(AF_INET, SOCK_STREAM, DEFAULT _PROTOCOL);

- 유닉스 소켓

  fd = socket(AF_UNIX, SOCK_STREAM, DEFAULT _PROTOCOL);

소켓에 이름(주소) 주기

int bind(int fd, struct sockaddr* address, int addressLen)

소켓에 대한 이름 바인딩이 성공하면 0을 실패하면 -1을 리턴한다.

- 유닉스 소켓 이름

```
struct sockaddr_un {
    unsigned short sun_family;      // AF_UNIX
    char sun_path[108];             // 소켓 이름
}
```

- 인터넷 소켓 이름

```
struct sockaddr_in {
    unsigned short sin_family;      // AF_INET
    unsigned short sin_port;        // 인터넷 소켓의 포트 번호
    struct in_addr sin_addr;        // 32-bit IP 주소
    char sin_zero[8];               // 사용 안 함
}
```

소켓 큐 생성

- listen() 시스템 호출
  - 클라이언트로부터의 연결 요청을 기다린다.
  - 연결 요청 대기 큐의 길이를 정한다.

int listen(int fd, int queueLength)

소켓 fd에 대한 연결 요청을 기다린다. 성공하면 0을 실패하면 -1을 리턴

- listen(serverFd, 5);

소켓에 연결 요청

- connect() 시스템 호출
  - fd가 나타내는 클라이언트 소켓과 address가 나타내는 서버 소켓과의 연결을 요청한다.
  - 성공하면 fd를 서버 소켓과의 통신에 사용할 수 있다.

int connect(int *fd*, struct sockaddr* *address*, int *addressLen*)

성공하면 0을 실패하면 -1를 리턴한다.

연결 요청 수락

int accept(int fd, struct sockaddr* address, int* addressLen)

성공하면 새로 만들어진 복사본 소켓의 파일 디스크립터, 실패하면 -1을 리턴

- 서버가 클라이언트로부터의 연결요청을 수락하는 내부 과정
  1. 서버는 fd가 나타내는 서버 소켓을 경청하고
  2. 클라이언트의 연결 요청이 올 때까지 기다린다.
  3. 클라이언트로부터 연결 요청이 오면
     원래 서버 소켓과 같은 복사본 소켓을 만들어
     이 복사본 소켓과 클라이언트 소켓을 연결
  4. 연결이 이루어지면
     address는 클라이언트 소켓의 주소로 세팅되고
     addressLen는 그 크기로 세팅
  5. 새로 만들어진 복사본 소켓의 파일 디스크립터를 리턴

대문자 변환 서버

- 이 프로그램
  - 입력받은 문자열을 소문자를 대문자로 변환한다.
  - 서버와 클라이언트로 구성된다.

- 서버
  - 소켓을 통해 클라이언트로부터 받은 문자열을 소문자를 대문자로 변환하여 소켓을 통해 클라이언트에 다시 보낸다.

- 클라이언트
  - 표준입력으로부터 문자열을 입력받아
  - 이를 소켓을 통해 서버에 보낸 후에
  - 대문자로 변환된 문자열을 다시 받아 표준출력에 출력

cserver.c

```
1 #include <stdio.h>
2 #include <signal.h>
3 #include <sys/types.h>
4 #include <sys/socket.h>
5 #include <sys/un.h>
6 #define DEFAULT_PROTOCOL 0
7 #define MAXLINE 100
9 /* 소문자를 대문자로 변환하는 서버 프로그램 */
10 int main ()
11 {
12    int listenfd, connfd, clientlen;
13    char inmsg[MAXLINE], outmsg[MAXLINE];
14    struct sockaddr_un serverUNIXaddr, clientUNIXaddr;
15
16    signal(SIGCHLD, SIG_IGN);
17    clientlen = sizeof(clientUNIXaddr);
18
19    listenfd = socket(AF_UNIX, SOCK_STREAM, DEFAULT_PROTOCOL);
20    serverUNIXaddr.sun_family = AF_UNIX;
```

cserver.c

```
21    strcpy(serverUNIXaddr.sun_path, "convert");
22    unlink("convert");
23    bind(listenfd, &serverUNIXaddr, sizeof(serverUNIXaddr));
24
25    listen(listenfd, 5);
26
27    while (1) { /* 소켓 연결 요청 수락 */
28       connfd = accept(listenfd, &clientUNIXaddr, &clientlen);
29       if (fork ( ) == 0) {
30          /* 소켓으로부터 한 줄을 읽어 대문자로 변환하여 보냄 */
31          readLine(connfd, inmsg);
32          toUpper(inmsg, outmsg);
33          write(connfd, outmsg, strlen(outmsg)+1);
34          close(connfd);
35          exit (0);
36       } else close(connfd);
37    }
38 }
```

cserver.c

```
40 /* 소문자를 대문자로 변환 */
41 toUpper(char* in, char* out)
42 {
43    int i;
44    for (i = 0; i < strlen(in); i++)
45        if (islower(in[i]))
46            out[i] = toupper(in[i]);
47        else out[i] = in[i];
48        out[i] = NULL;
49 }
50
51 /* 한 줄 읽기 */
52 readLine(int fd, char* str)
53 {
54    int n;
55    do {
56        n = read(fd, str, 1);
57     } while(n > 0 && *str++ != NULL);
58     return(n > 0);
59 }
```

cclient.c

```
1 #include <stdio.h>
2 #include <signal.h>
3 #include <sys/types.h>
4 #include <sys/socket.h>
5 #include <sys/un.h>
6 #define DEFAULT_PROTOCOL 0
7 #define MAXLINE 100
8
9 /* 소문자-대문자 변환: 클라이언트 프로그램 */
10 int main ( )
11 {
12    int clientfd, result;
13    char inmsg[MAXLINE], outmsg[MAXLINE];
14    struct sockaddr_un serverUNIXaddr;
15
16    clientfd = socket(AF_UNIX, SOCK_STREAM, DEFAULT_PROTOCOL);
17    serverUNIXaddr.sun_family = AF_UNIX;
18    strcpy(serverUNIXaddr.sun_path, "convert");
```

cclient.c

```
20   do { /* 연결 요청 */
21       result = connect(clientfd, &serverUNIXaddr, sizeof(serverUNIXaddr));
22       if (result == -1) sleep(1);
23   } while (result == -1);
24
25   printf("변환할 문자열 입력:₩n");
26   fgets(inmsg, MAXLINE, stdin);
27   write(clientfd,inmsg,strlen(inmsg)+1); // 변환할 문자열 보내기
28
29   /* 소켓으로부터 변환된 문자열을 한 줄 읽어서 프린트 */
30   readLine(clientfd,outmsg);
31   printf("%s --> ₩n%s", inmsg, outmsg);
32   close(clientfd);
33   exit(0);
34 }
```

13.2 인터넷 소켓

인터넷 상의 호스트

- 인터넷 상의 호스트는 32 비트 IP 주소를 갖는다
  - 예: 203.252.201.8
- IP 주소는 대응하는 도메인 이름을 갖는다.
  - 예: 203.252.201.8 --> www.sookmyung.ac.kr
- 32 비트 IP 주소는 저장

```
/* 인터넷 주소 구조체 */
struct in_addr {
    unsigned int s_addr; // 네트워크 바이트 순서(big-endian)
};
```

DNS(Domain Name System)

- 인터넷은 IP 주소와 도메인 이름 사이의 맵핑을 DNS라 부르는 전세계적인 분산 데이터베이스에 유지한다.

```
/* DNS 호스트 엔트리 구조체 */
struct hostent {
    char *h_name;          // 호스트의 공식 도메인 이름
    char **h_aliases;      // null로 끝나는 도메인 이름의 배열
    int h_addrtype;        // 호스트 주소 타입(AF_INET)
    int h_length;          // 주소의 길이
    char **h_addr_list;    // null로 끝나는 in_addr 구조체의 배열
};
```

DNS 관련 함수

- 호스트의 IP 주소 혹은 도메인 이름을 이용하여 DNS로부터 호스트 엔트리를 검색할 수 있다.

```
struct hostent *gethostbyaddr(const char* addr, int len, int type);
```

길이가 len이고 주소 타입 type인 호스트 주소 addr에 해당하는 hostent 구조체를 리턴한다.

```
struct hostent* gethostbyname(char* name);
```

도메인 이름에 대응하는 hostent 구조체에 대한 포인터를 리턴한다.

DNS 관련 함수

- in_addr 구조체 형식으로 된 IP 주소를 프린트 할 수 있는 스트링으로 변환

```
char* inet_ntoa(struct in_addr address);
```

IP 주소 address에 대응하는 A.B.C.D 포맷의 스트링을 리턴한다.

```
unsigned long inet_addr(char* string);
```

A.B.C.D 포맷의 IP 주소를 네트워크 바이트 순서로 된 이진 데이터로 변환하여 리턴한다.

인터넷 소켓

- 인터넷 소켓
  - 서로 다른 호스트에서 실행되는 클라이언트-서버 사이의 통신
  - 양방향(2-way) 통신
  - 소켓을 식별하기 위해 호스트의 IP 주소와 포트 번호를 사용

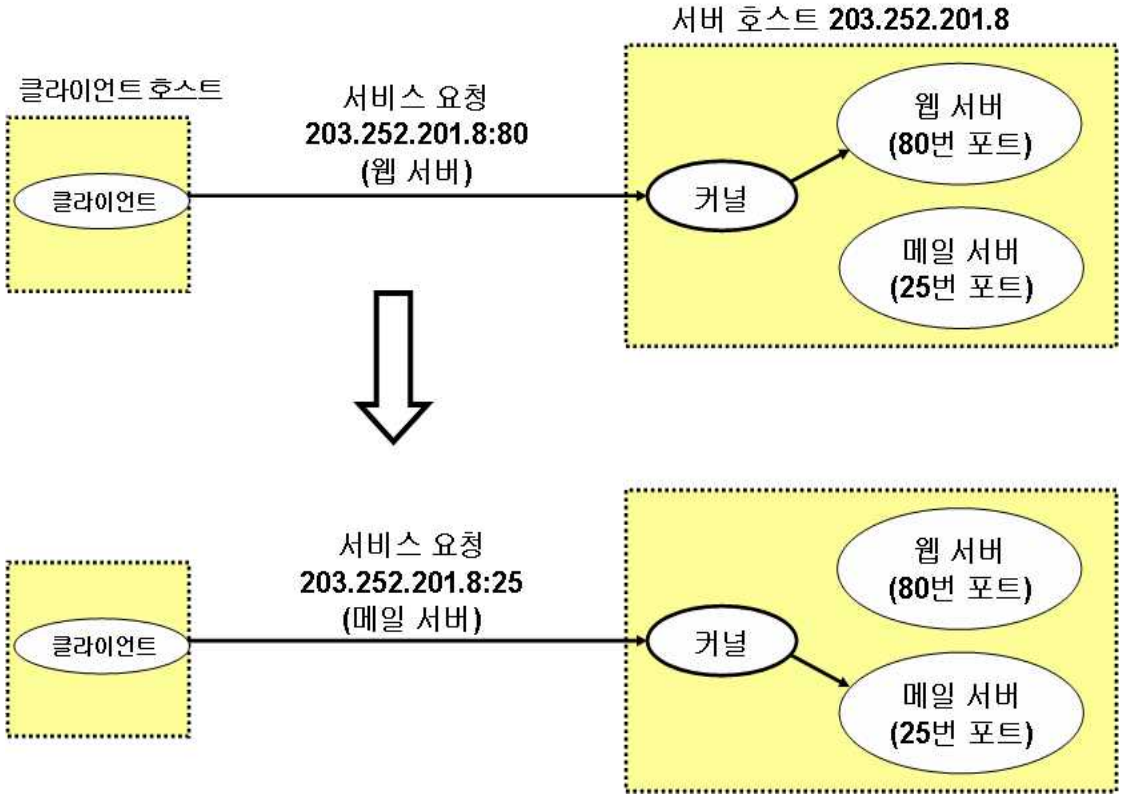
- 인터넷 소켓 연결 예

인터넷 소켓

- 인터넷 소켓 통신을 사용하는 SW
  - 웹 브라우저
  - ftp
  - telnet
  - ssh

- 잘 알려진 서비스의 포트 번호
  - 시간 서버: 13번 포트
  - ftp 서버: 20,21번 포트
  - 텔넷 서버: 23번 포트
  - 메일 서버: 25번 포트
  - 웹 서버: 80번 포트

클라이언트-서버 인터넷 소켓 연결 과정



인터넷 소켓 이름(주소)

- 인터넷 소켓 이름(주소)

```
struct sockaddr_in {
    unsigned short sin_family;  // AF_INET
    unsigned short sin_port;    // 인터넷 소켓의 포트 번호
    struct in_addr sin_addr;    // 32-bit IP 주소
    char sin_zero[8];           // 사용 안 함
}
```

- 소켓 이름을 위한 포괄적 구조체

```
struct sockaddr {
    unsigned short sa_family;  // 프로토콜 패밀리
    char sa_data[14];          // 주소 데이터
};
```

파일 서버-클라이언트

- 서버
  - 파일 이름을 받아 해당 파일을 찾아 그 내용을 보내주는 서비스
  - 명령줄 인수로 포트 번호를 받아 해당 소켓을 만든다.
  - 이 소켓을 통해 클라이언트로부터 파일 이름을 받아
  - 해당 파일을 열고 그 내용을 이 소켓을 통해 클라이언트에게 보낸다.

- 클라이언트
  - 명령줄 인수로 연결할 서버의 이름과 포트 번호를 받아 해당 서버에 소켓 연결을 한다.
  - 이 연결을 통해 서버에 원하는 파일 이름을 보낸 후
  - 서버로부터 해당 파일 내용을 받아 사용자에게 출력한다.

fserver.c

```
1 #include <stdio.h>
2 #include <signal.h>
3 #include <sys/types.h>
4 #include <sys/socket.h>
5 #include <netinet/in.h>
6 #include <arpa/inet.h>
7 #include <netdb.h>
8 #define DEFAULT_PROTOCOL 0
9 #define MAXLINE 100
10
11 /* 파일 서버 프로그램 */
12 int main (int argc, char* argv[])
13 {
14    int listenfd, connfd, port, clientlen;
15    FILE *fp;
16    char inmsg[MAXLINE], outmsg[MAXLINE];
17    struct sockaddr_in serveraddr, clientaddr;
18    struct hostent *hp;
19    char *haddrp;
```

fserver.c

```
21   signal(SIGCHLD, SIG_IGN);
22
23   if (argc != 2) {
24      fprintf(stderr, "사용법: %s <port>₩n", argv[0]);
25      exit(0);
26   }
27   port = atoi(argv[1]);
28
29   listenfd = socket(AF_INET, SOCK_STREAM, DEFAULT_PROTOCOL);
30
31   bzero((char *) &serveraddr, sizeof(serveraddr));
32   serveraddr.sin_family = AF_INET;
33   serveraddr.sin_addr.s_addr = htonl(INADDR_ANY);
34   serveraddr.sin_port = htons((unsigned short)port);
35   bind(listenfd, &serveraddr, sizeof(serveraddr));
36   listen(listenfd, 5);
```

fserver.c

```
38   while (1) {
39       clientlen = sizeof(clientaddr);
40       connfd = accept(listenfd, &clientaddr, &clientlen);
41
42       /* 클라이언트의 도메인 이름과 IP 주소 결정 */
43      hp = gethostbyaddr((char *)&clientaddr.sin_addr.s_addr,
44                                   sizeof(clientaddr.sin_addr.s_addr), AF_INET);
45      haddrp = inet_ntoa(clientaddr.sin_addr);
46      printf("서버: %s (%s) %d에 연결됨₩n",
47              hp->h_name, haddrp, clientaddr.sin_port);
48
```

fserver.c

```
49      if (fork ( ) == 0) {
50          readLine(connfd, inmsg);     /* 소켓에서 파일 이름을 읽는다 */
51          fp = fopen(inmsg, "r");
52          if (fp == NULL) {
53              write(connfd, "해당 파일 없음", 10);
54          } else { /* 파일에서 한 줄씩 읽어 소켓을 통해 보낸다 */
55              while(fgets(outmsg, MAXLINE, fp) != NULL)
56                  write(connfd, outmsg, strlen(outmsg)+1);
57          }
58          close(connfd);
59          exit (0);
60      } else close(connfd);
61  }  // while
62 }  // main
```

fclient.c

```
1 #include <stdio.h>
2 #include <signal.h>
3 #include <sys/types.h>
4 #include <sys/socket.h>
5 #include <netinet/in.h>
6 #include <arpa/inet.h>
7 #include <netdb.h>
8 #define DEFAULT_PROTOCOL 0
9 #define MAXLINE 100
10
11 /* 파일 클라이언트 프로그램 */
12 int main (int argc, char* argv[])
13 {
14    int clientFd, port, result;
15    char *host, inmsg[MAXLINE], outmsg[MAXLINE];
16    struct sockaddr_in serverAddr;
17    struct hostent *hp;
```

fclient.c

```
19   if (argc != 3) {
20      fprintf(stderr, "사용법 : %s <host> <port>₩n", argv[0]);
21      exit(0);
22   }
23
24   host = argv[1];
25   port = atoi(argv[2]);
27   clientFd = socket(AF_INET, SOCK_STREAM, DEFAULT_PROTOCOL);
28
29   /* 서버의 IP 주소와 포트 번호를 채운다. */
30   if ((hp = gethostbyname(host)) == NULL)
31      perror("gethostbyname error"); // 호스트 찾기 오류
32   bzero((char *) &serverAddr, sizeof(serverAddr));
33   serverAddr.sin_family = AF_INET;
34   bcopy((char *)hp->h_addr_list[0],
35            (char *)&serverAddr.sin_addr.s_addr, hp->h_length);
36   serverAddr.sin_port = htons(port);
```

fclient.c

```
38   do { /* 연결 요청 */
39      result = connect(clientFd, &serverAddr,  sizeof(serverAddr));
40      if (result == -1) sleep(1);
41   } while (result == -1);
42
43   printf("파일 이름 입력:");
44   scanf("%s", inmsg);
45   write(clientFd,inmsg,strlen(inmsg)+1);
46
47   /* 소켓으로부터 파일 내용 읽어서 프린트 */
48   while (readLine(clientFd,outmsg))
49      printf("%s", outmsg);
50   close(clientFd);
51   exit(0);
52 }
```

핵심 개념

- 소켓은 양방향 통신 방법으로 클라이언트-서버 모델을 기반으로 프로세스 사이의 통신에 매우 적합하다.

- 소켓에는 같은 호스트 내의 프로세스 사이의 통신을 위한 유닉스 소켓과 다른 호스트에 있는 프로세스 사이의 통신을 위한 인터넷 소켓이 있다.