
OS/NW 과제 -6

단일 서버, 멀티 클라이언트 연결(fork) & 문자열 합치기

32184074 컴퓨터공학과 정민준 - 2022년 10월 31일



본문

<echo_client.c 코드>

```
lab04 완성본 echo_client.c
echo_server_fork.c x echo_client.c x
9
10 int main(int argc, char **argv)
11 {
12     struct sockaddr_in serveraddr;
13     int server_sockfd;
14     int client_len;
15     char buf[MAXLINE];
16
17     if ((server_sockfd = socket(AF_INET, SOCK_STREAM, 0)) == -1)
18     {
19         perror("error :");
20         return 1;
21     }
22
23     /* 연결요청할 서버의 주소와 포트번호, 프로토콜들을 지정한다. */
24     server_sockfd = socket(AF_INET, SOCK_STREAM, 0);
25     serveraddr.sin_family = AF_INET;
26     serveraddr.sin_addr.s_addr = inet_addr("127.0.0.1");
27     serveraddr.sin_port = htons(3600);
28
29     client_len = sizeof(serveraddr);
30
31     /* 서버에 연결을 시도한다. */
32     if (connect(server_sockfd, (struct sockaddr *)&serveraddr,
33         client_len) == -1)
34     {
35         perror("connect error :");
36         return 1;
37     }
38
39     memset(buf, 0x00, MAXLINE);
40     read(0, buf, MAXLINE); /* 키보드 입력을 기다린다. */
41
42     /* 입력 받은 데이터를 서버로 전송한다. */
43     if (write(server_sockfd, buf, nbyte: MAXLINE) <= 0)
44     {
45         perror("write error : ");
46         return 1;
47     }
48     memset(buf, 0x00, MAXLINE);
49     /* 서버로부터 데이터를 읽는다. */
50     if (read(server_sockfd, buf, MAXLINE) <= 0)
51     {
52         perror("read error : ");
53         return 1;
54     }
55     printf("read : %s", buf);
56     printf("\n");
57     close(server_sockfd);
58     return 0;
59 }
```

<echo_client.c>

<echo_server_fork.c 코드>

```
lab04 완성본 echo_server_fork.c
echo_server_fork.c x echo_client.c x
1 #include ...
10
11 #define MAXLINE 1024
12 #define PORTNUM 3600
13
14 int main(int argc, char **argv)
15 {
16     int listen_fd;
17     int client_fd[3]; //클라이언트가 담길 배열 3개
18     pid_t pid;
19     socklen_t addrlen;
20     int readn;
21     int p_readn;
22     char buf[MAXLINE] = {};
23     struct sockaddr_in client_addr, server_addr;
24
25     char arr[MAXLINE] = {}; //합쳐질 문자가 담길 문자열 초기화
26     int cnt = 0; //클라이언트의 순서를 나타냄
27
28     int pp[2]; //pipe 통신
29     pipe(pp);
30
31
32     if( (listen_fd = socket(AF_INET, SOCK_STREAM, 0)) < 0)
33     {
34         return 1;
35     }
36     memset((void *)&server_addr, 0x00, sizeof(server_addr));
37     server_addr.sin_family = AF_INET;
38     server_addr.sin_addr.s_addr = htonl(INADDR_ANY);
39     server_addr.sin_port = htons(PORTNUM);
40
41     if(bind(listen_fd, (struct sockaddr *)&server_addr, sizeof(server_addr))
42     {
43         perror("bind error");
44         return 1;
45     }
46     if(listen(listen_fd, 5) == -1)
47     {
48         perror("listen error");
49         return 1;
50     }
51
52     signal(SIGCHLD, SIG_IGN);
53     while(1)
54     {
55         addrlen = sizeof(client_addr);
56         client_fd[cnt] = accept(listen_fd,
57                                (struct sockaddr *)&client_addr, &addrlen);
58         if(client_fd[cnt] == -1)
59         {
60             continue;
61         }
62         pid = fork();
63         if(pid < 0)
64         {
65             perror("fork error");
66             continue;
67         }
68         if(pid == 0)
69         {
70             readn = 0;
71             while((readn = read(client_fd[cnt], buf, MAXLINE)) > 0)
72             {
73                 p_readn = readn;
74                 readn = 0;
75                 write(pp[1], buf, p_readn);
76                 readn = read(pp[0], arr, MAXLINE);
77                 if(readn < 0)
78                     break;
79                 write(client_fd[cnt], arr, readn);
80                 readn = 0;
81             }
82             _exit(0);
83         }
84         cnt++;
85     }
86 }
```

<echo_server_fork.c - 1>

```
lab04 완성본 echo_server_fork.c lab04_ | Debug
echo_server_fork.c x echo_client.c x
51
52 signal(SIGCHLD, SIG_IGN);
53 while(1)
54 {
55     addrlen = sizeof(client_addr);
56     client_fd[cnt] = accept(listen_fd,
57         (struct sockaddr *)&client_addr, &addrlen);
58     if(client_fd[cnt] == -1)
59     {
60         printf("accept error\n");
61         break;
62     }
63     pid = fork();
64
65     //자식 프로세스
66     if(pid == 0)
67     {
68         close(listen_fd);
69         memset(buf, 0x00, MAXLINE);
70
71         //클라이언트로 부터 문자열(buf) 수신
72         while((readn = read(client_fd[cnt], buf, MAXLINE)) > 0)
73         {
74             printf("Read Data %s(%d) : %s",
75                 inet_ntoa(client_addr.sin_addr),
76                 client_addr.sin_port,
77                 buf);
78
79             //부모 프로세스에 문자열(buf) 전송
80             write(fd: pp[1], buf, nbytes: strlen(s: buf));
81             memset(buf, 0x00, MAXLINE);
82         }
83         return 0;
84     }
85
86     //부모 프로세스
87     else if(pid > 0)
88     {
89         //자식 프로세스로 부터 전송받은 값 수신
90         p_readn = read(pp[0], buf, MAXLINE);
91
92         //개행문자 제거
93         for(int i = 0 ; buf[i] != 0 ; i++){
94             if(buf[i] == '\n')
95             {
96                 buf[i] = 0;
97                 break;
98             }
99         }
100
101     }
```

<echo_server_fork.c - 2>

```
lab04 완성본 / echo_server_fork.c
echo_server_fork.c x echo_client.c x

71 //클라이언트로 부터 문자열(buf) 수신
72 while((readn = read(client_fd[cnt], buf, MAXLINE)) > 0)
73 {
74     printf("Read Data %s(%d) : %s",
75           inet_ntoa(client_addr.sin_addr),
76           client_addr.sin_port,
77           buf);
78
79     //부모 프로세스에 문자열(buf) 전송
80     write(fd: pp[1], buf, nbyte: strlen(s: buf));
81     memset(buf, 0x00, MAXLINE);
82 }
83 return 0;
84 }
85
86 //부모 프로세스
87 else if( pid > 0)
88 {
89     //자식 프로세스로 부터 전송받은 값 수신
90     p_readn = read(pp[0], buf, MAXLINE);
91
92     //개행문자 제거
93     for(int i = 0 ; buf[i] != 0 ; i++){
94         if(buf[i] == '\n')
95         {
96             buf[i] = 0;
97             break;
98         }
99     }
100
101 }
102
103 //띄어쓰기 삽입 & 문자열 합치기
104 strcat(arr, " ");
105 strcat(arr, buf);
106
107 //클라이언트에게 전송
108 write(fd: client_fd[cnt], buf: arr, nbyte: sizeof(arr));
109 close(client_fd[cnt]);
110
111 //다음 클라이언트로 순서로 넘어가기 위해 cnt값 증가
112 cnt++;
113 }
114 close(listen_fd);
115 return 0;
116 }
```

<echo_server_fork.c - 3>

<결과 화면>

The image displays four terminal windows arranged in a 2x2 grid, showing the execution of a server and three clients. Each window has a title bar with the user 'minjunjung' and the host 'ce4074@osnw-stud: ~/lab04'.

- 1. echo_server_fork**: The terminal shows the command `./echo_server_fork` being executed. It then displays three lines of data received from clients: `Read Data 127.0.0.1(61128) : osnw`, `Read Data 127.0.0.1(61640) : 32184074`, and `Read Data 127.0.0.1(62152) : hello`.
- 2. echo_client(1)**: The terminal shows the command `./echo_client` being executed. It then displays the input `osnw` and the output `read : osnw`.
- 3. echo_client(2)**: The terminal shows the command `./echo_client` being executed. It then displays the input `32184074` and the output `read : osnw 32184074`.
- 4. echo_client(3)**: The terminal shows the command `./echo_client` being executed. It then displays the input `hello` and the output `read : osnw 32184074 hello`.

<결과 출력>

입력 받은 문자열값을 순서대로 연결, 각 클라이언트에 순서대로 반환 해주고 있습니다.

1. 3명의 클라이언트를 받아야 하기 때문에 정수형 `client_fd` 배열을 크기 3으로 선언합니다.
2. `fork()`를 사용해서 자식 프로세스를 생성해서 클라이언트로 부터 입력을 받습니다.
3. 클라이언트로 부터 입력받은 값을 자식 프로세스는 pipe 통신으로 부모 프로세스로 전달합니다.
4. 부모 프로세스는 입력받은 값을 개행문자를 제거하고 `strcat`을 통해서 빈 문자열 `arr`을 이어 붙이면서 새로운 문자열을 만듭니다.
5. 클라이언트에 새롭게 만들어진 문자열 `arr`을 보내줍니다.