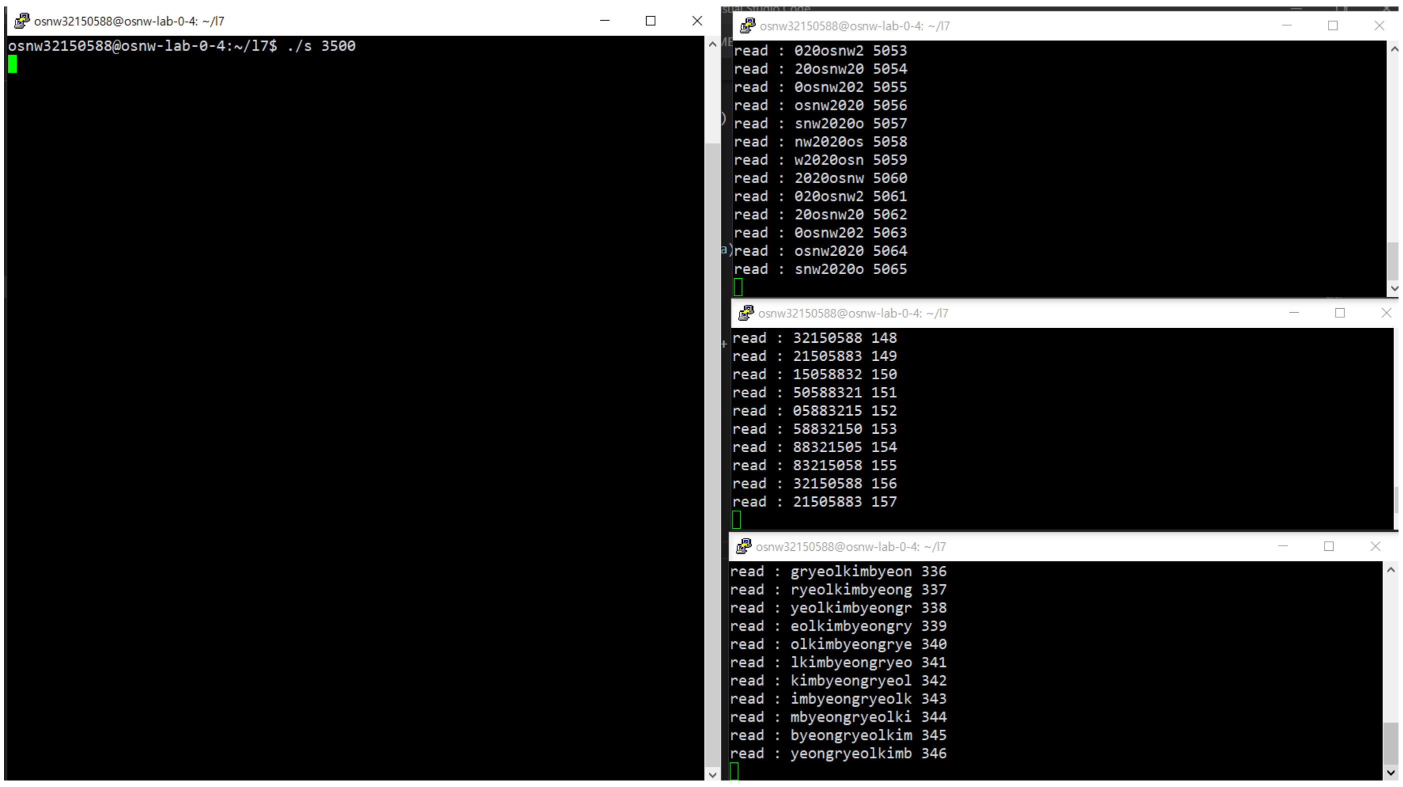
< 실행 화면>


osnw32150588@osnw-lab-0-4: ~/l7
osnw32150588@osnw-lab-0-4: ~/l7
osnw32150588@osnw-lab-0-4: ~/l7
osnw32150588@osnw-lab-0-4: ~/l7

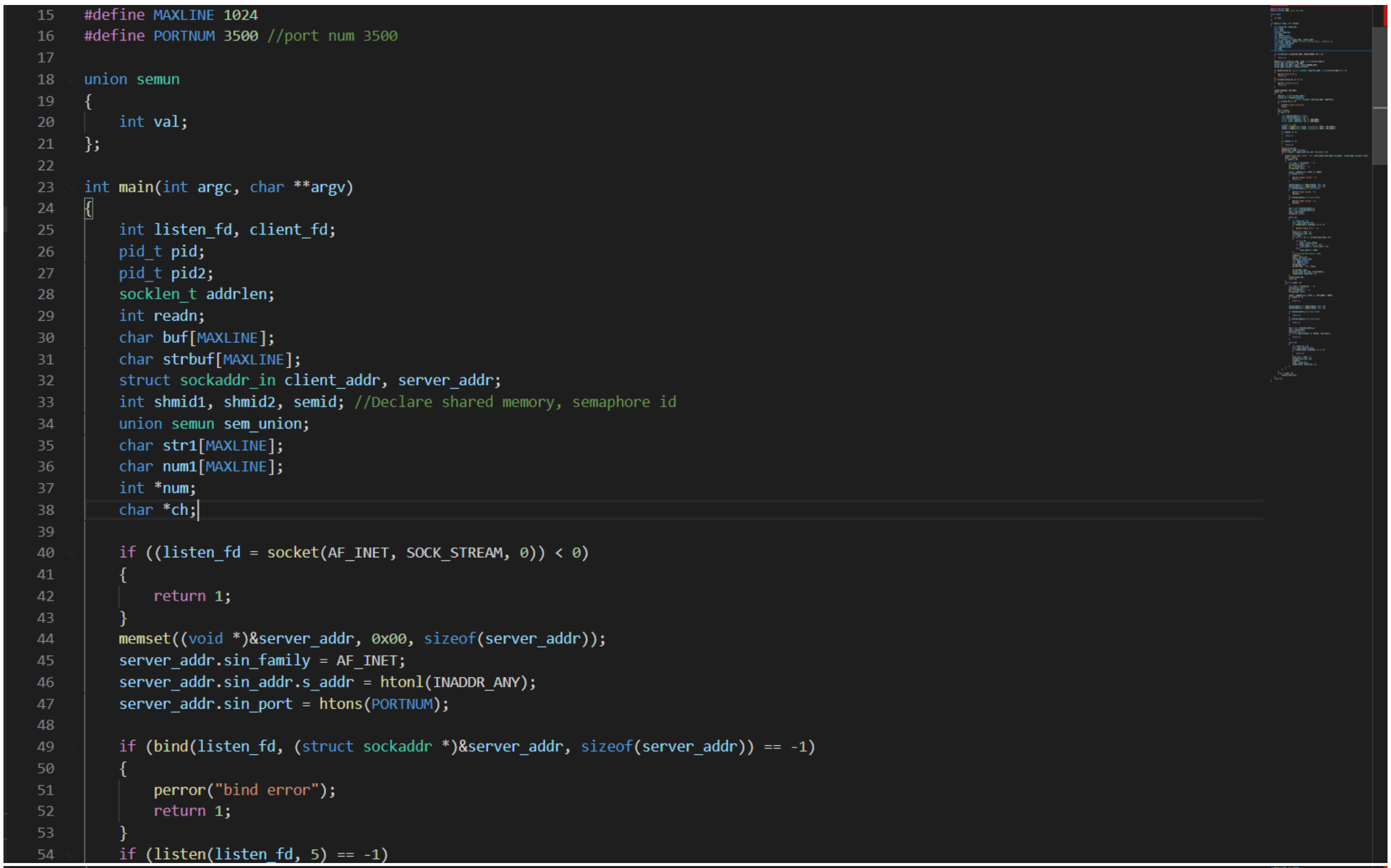
```
osnw32150588@osnw-lab-0-4:~/l7$ ./s 3500
```

```
read : 020osnw2 5053
read : 20osnw20 5054
read : 0osnw202 5055
read : osnw2020 5056
read : snw2020o 5057
read : nw2020os 5058
read : w2020osn 5059
read : 2020osnw 5060
read : 020osnw2 5061
read : 20osnw20 5062
read : 0osnw202 5063
read : osnw2020 5064
read : snw2020o 5065
```

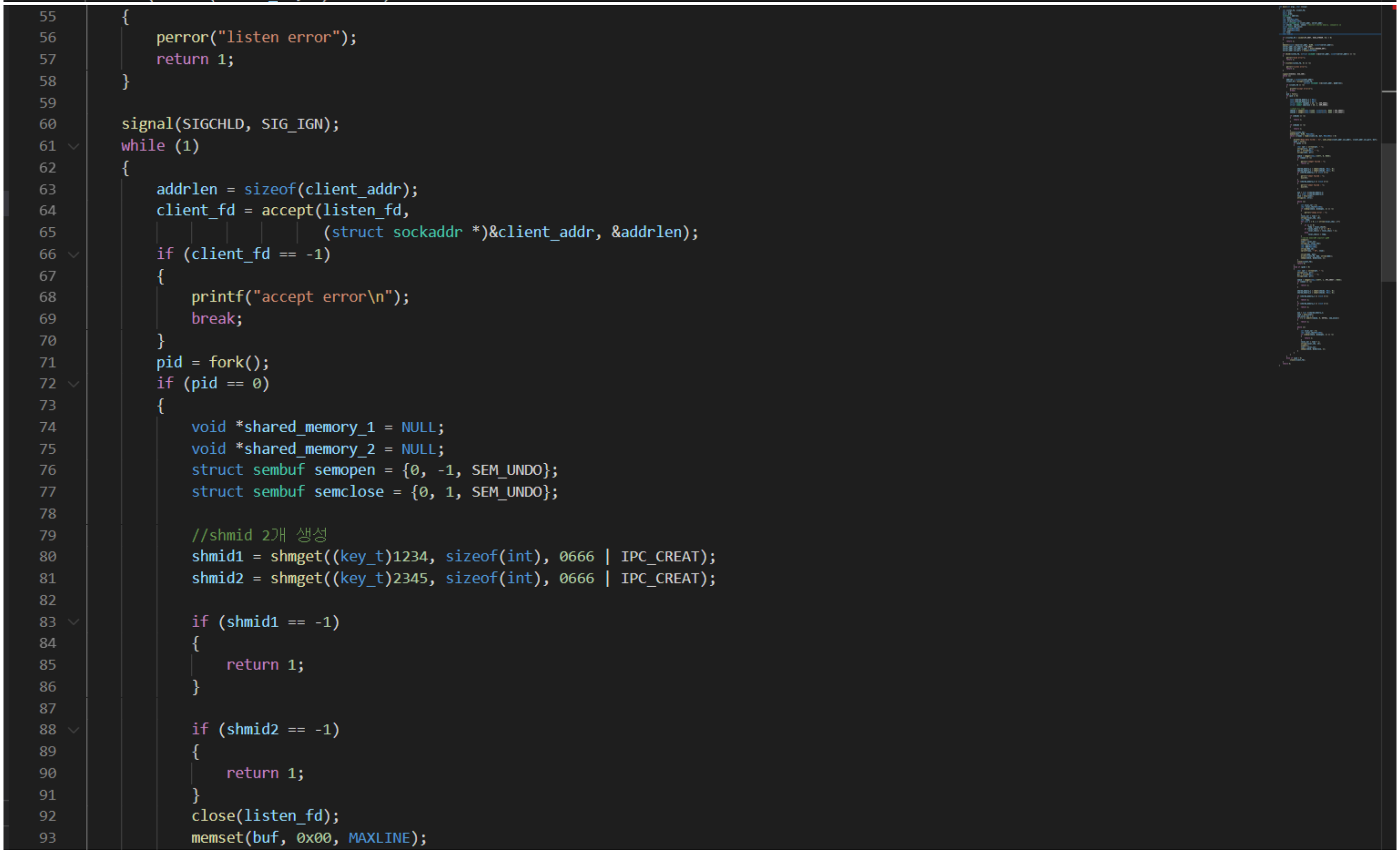
```
read : 32150588 148
read : 21505883 149
read : 15058832 150
read : 50588321 151
read : 05883215 152
read : 58832150 153
read : 88321505 154
read : 83215058 155
read : 32150588 156
read : 21505883 157
```

```
read : gryeolkimbyeon 336
read : ryeolkimbyeong 337
read : yeolkimbyeongr 338
read : eolkimbyeongry 339
read : olkimbyeongrye 340
read : lkimbyeongryeo 341
read : kimbyeongryeol 342
read : imbyeongryeolk 343
read : mbyeongryeolki 344
read : byeongryeolkim 345
read : yeongryeolkimb 346
```

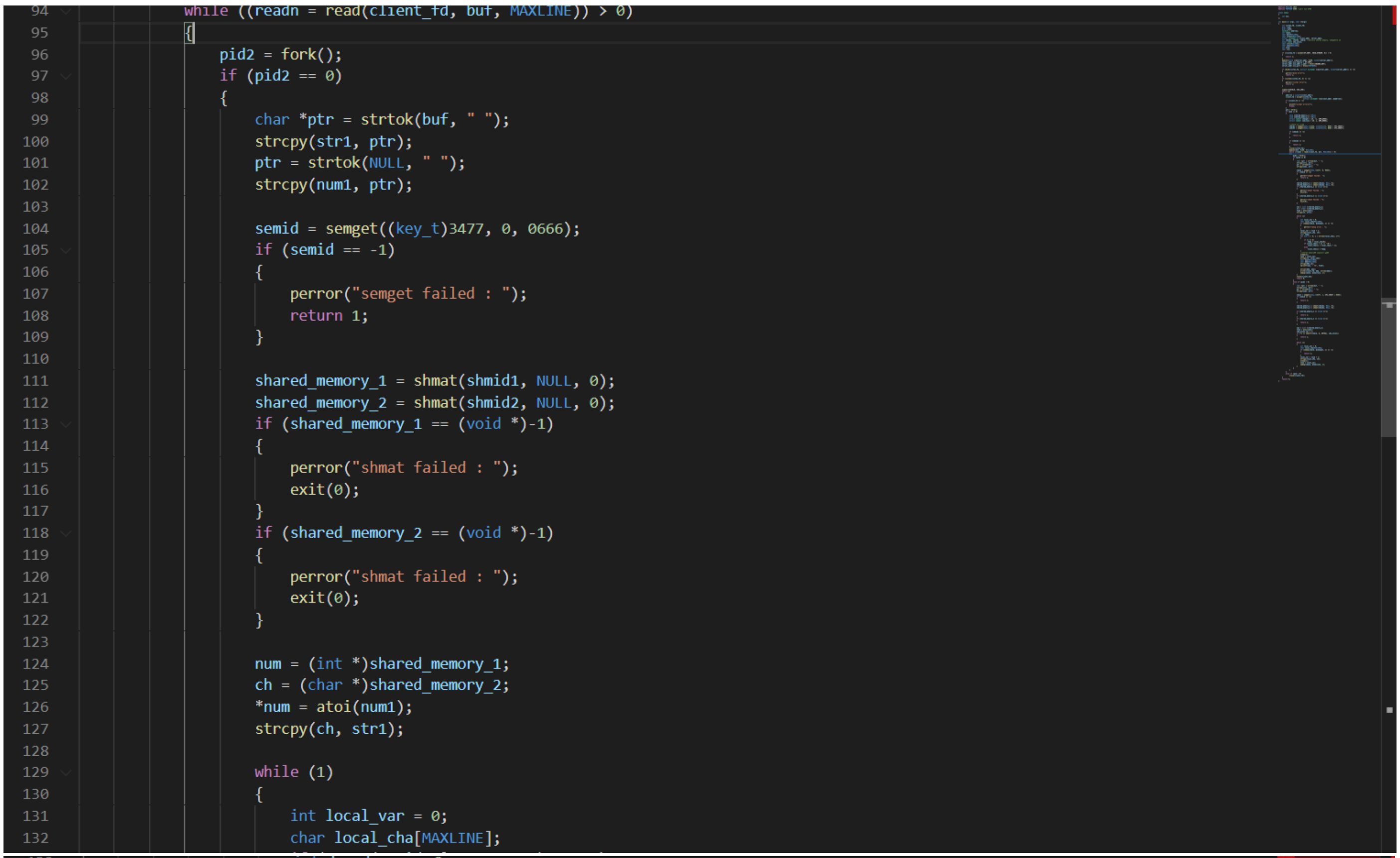
<소스 코드>



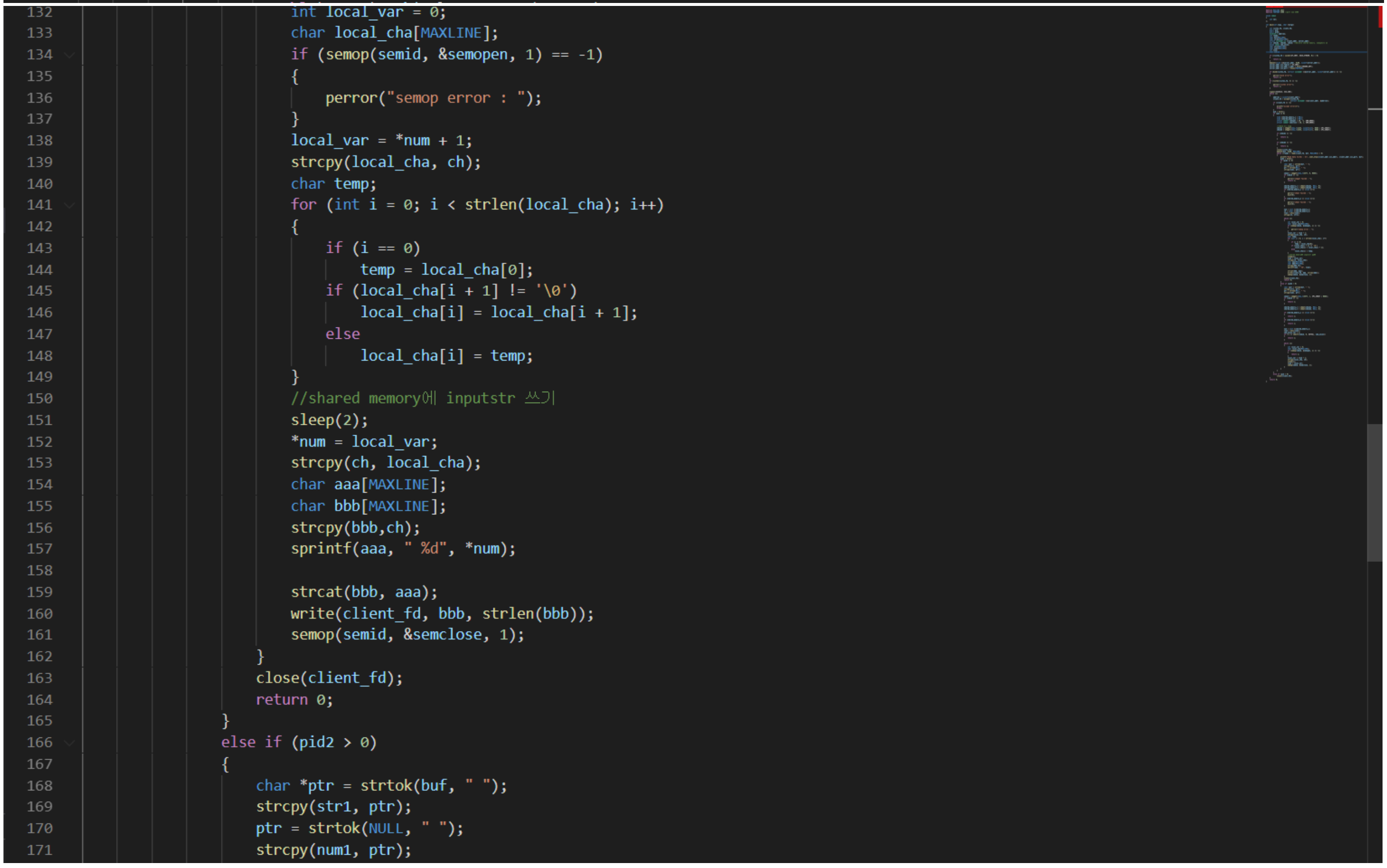
```c
#define MAXLINE 1024
#define PORTNUM 3500 //port num 3500

union semun
{
    int val;
};

int main(int argc, char **argv)
{
    int listen_fd, client_fd;
    pid_t pid;
    pid_t pid2;
    socklen_t addrlen;
    int readn;
    char buf[MAXLINE];
    char strbuf[MAXLINE];
    struct sockaddr_in client_addr, server_addr;
    int shmid1, shmid2, semid; //Declare shared memory, semaphore id
    union semun sem_union;
    char str1[MAXLINE];
    char num1[MAXLINE];
    int *num;
    char *ch;

    if ((listen_fd = socket(AF_INET, SOCK_STREAM, 0)) < 0)
    {
        return 1;
    }
    memset((void *)&server_addr, 0x00, sizeof(server_addr));
    server_addr.sin_family = AF_INET;
    server_addr.sin_addr.s_addr = htonl(INADDR_ANY);
    server_addr.sin_port = htons(PORTNUM);

    if (bind(listen_fd, (struct sockaddr *)&server_addr, sizeof(server_addr)) == -1)
    {
        perror("bind error");
        return 1;
    }
    if (listen(listen_fd, 5) == -1)
```



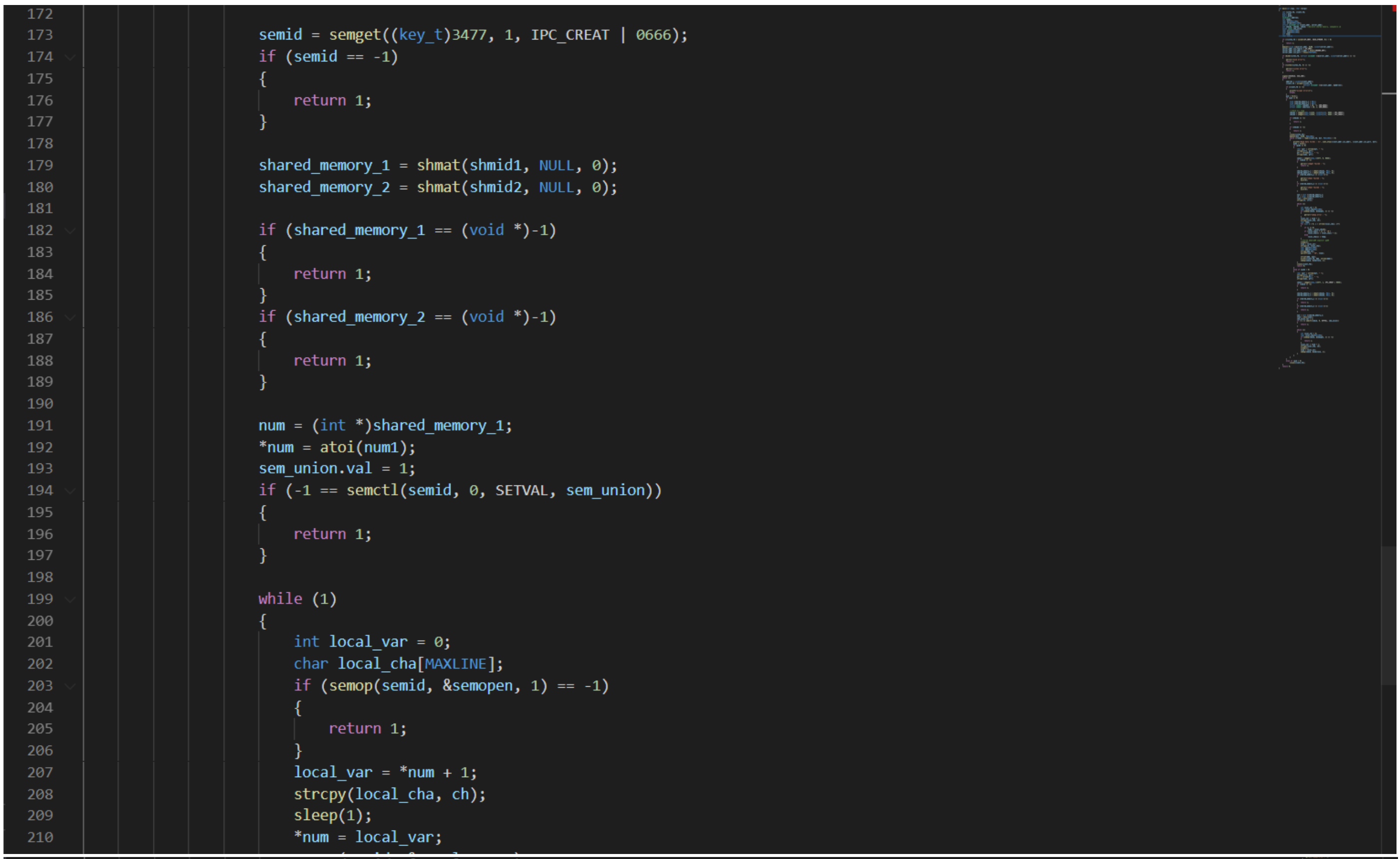
```c
    {
        perror("listen error");
        return 1;
    }

    signal(SIGCHLD, SIG_IGN);
    while (1)
    {
        addrlen = sizeof(client_addr);
        client_fd = accept(listen_fd,
                           (struct sockaddr *)&client_addr, &addrlen);
        if (client_fd == -1)
        {
            printf("accept error\n");
            break;
        }
        pid = fork();
        if (pid == 0)
        {
            void *shared_memory_1 = NULL;
            void *shared_memory_2 = NULL;
            struct sembuf semopen = {0, -1, SEM_UNDO};
            struct sembuf semclose = {0, 1, SEM_UNDO};

            //shmid 2개 생성
            shmid1 = shmget((key_t)1234, sizeof(int), 0666 | IPC_CREAT);
            shmid2 = shmget((key_t)2345, sizeof(int), 0666 | IPC_CREAT);

            if (shmid1 == -1)
            {
                return 1;
            }

            if (shmid2 == -1)
            {
                return 1;
            }
            close(listen_fd);
            memset(buf, 0x00, MAXLINE);
```



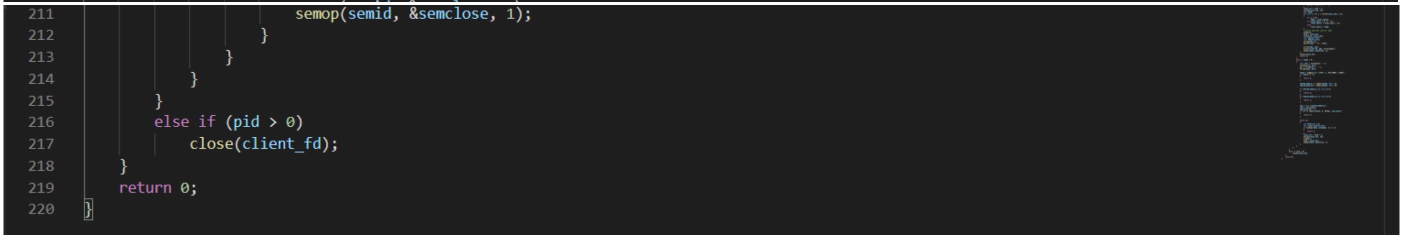
```c
        while ((readn = read(client_fd, buf, MAXLINE)) > 0)
        {
            pid2 = fork();
            if (pid2 == 0)
            {
                char *ptr = strtok(buf, " ");
                strcpy(str1, ptr);
                ptr = strtok(NULL, " ");
                strcpy(num1, ptr);

                semid = semget((key_t)3477, 0, 0666);
                if (semid == -1)
                {
                    perror("semget failed : ");
                    return 1;
                }

                shared_memory_1 = shmat(shmid1, NULL, 0);
                shared_memory_2 = shmat(shmid2, NULL, 0);
                if (shared_memory_1 == (void *)-1)
                {
                    perror("shmat failed : ");
                    exit(0);
                }
                if (shared_memory_2 == (void *)-1)
                {
                    perror("shmat failed : ");
                    exit(0);
                }

                num = (int *)shared_memory_1;
                ch = (char *)shared_memory_2;
                *num = atoi(num1);
                strcpy(ch, str1);

                while (1)
                {
                    int local_var = 0;
                    char local_cha[MAXLINE];
```



```c
                    int local_var = 0;
                    char local_cha[MAXLINE];
                    if (semop(semid, &semopen, 1) == -1)
                    {
                        perror("semop error : ");
                    }
                    local_var = *num + 1;
                    strcpy(local_cha, ch);
                    char temp;
                    for (int i = 0; i < strlen(local_cha); i++)
                    {
                        if (i == 0)
                            temp = local_cha[0];
                        if (local_cha[i + 1] != '\0')
                            local_cha[i] = local_cha[i + 1];
                        else
                            local_cha[i] = temp;
                    }
                    //shared memory에 inputstr 쓰기
                    sleep(2);
                    *num = local_var;
                    strcpy(ch, local_cha);
                    char aaa[MAXLINE];
                    char bbb[MAXLINE];
                    strcpy(bbb,ch);
                    sprintf(aaa, " %d", *num);

                    strcat(bbb, aaa);
                    write(client_fd, bbb, strlen(bbb));
                    semop(semid, &semclose, 1);
                }
                close(client_fd);
                return 0;
            }
            else if (pid2 > 0)
            {
                char *ptr = strtok(buf, " ");
                strcpy(str1, ptr);
                ptr = strtok(NULL, " ");
                strcpy(num1, ptr);
```



```c

                    semid = semget((key_t)3477, 1, IPC_CREAT | 0666);
                    if (semid == -1)
                    {
                        return 1;
                    }

                    shared_memory_1 = shmat(shmid1, NULL, 0);
                    shared_memory_2 = shmat(shmid2, NULL, 0);

                    if (shared_memory_1 == (void *)-1)
                    {
                        return 1;
                    }
                    if (shared_memory_2 == (void *)-1)
                    {
                        return 1;
                    }

                    num = (int *)shared_memory_1;
                    *num = atoi(num1);
                    sem_union.val = 1;
                    if (-1 == semctl(semid, 0, SETVAL, sem_union))
                    {
                        return 1;
                    }

                    while (1)
                    {
                        int local_var = 0;
                        char local_cha[MAXLINE];
                        if (semop(semid, &semopen, 1) == -1)
                        {
                            return 1;
                        }
                        local_var = *num + 1;
                        strcpy(local_cha, ch);
                        sleep(1);
                        *num = local_var;
```



```c
                        semop(semid, &semclose, 1);
                    }
                }
            }
        }
        else if (pid > 0)
            close(client_fd);
    }
    return 0;
}
```