آزمایشگاه سیستم عامل

گزارش آزمایش شماره ۴

نام استاد: مهندس حمیدرضا کیخا

تاریخ: ۵ / ۹ / ۱۴۰۰

نام و نام‌خانوادگی: نگار کرمی

شماره دانشجویی: ۹۷۲۲۰۳۹

نام و نام‌خانوادگی: مجید نامی

شماره دانشجویی: ۹۷۲۸۰۹۵

دانشگاه صنعتی امیرکبیر

(پلی تکنیک تهران)

سوال۱:

با استفاده از دستورکار یک shared memory ایجاد می‌کنیم و بعد یک argument از terminal می‌گیریم و در shared memory آن را می‌نویسیم و بعد یک پردازه‌ی فرزند آن عبارتی را که در حافظه مشترک نوشته شده بود می‌خواند و آن را چاپ می‌کند.

کد:

```c
#include <stdio.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <unistd.h>

int main (int argc, char *argv[])
{
    int seg_id;
    char* shared_memory;
    int size = 50;

    struct shmid_ds shm_buffer;
    int seg_size;

    seg_id = shmget(IPC_PRIVATE, size, S_IRUSR | S_IWUSR);

    shared_memory = (char*) shmat (seg_id, NULL, 0);
    printf("shared memory address %p\n", shared_memory);

    shmctl(seg_id, IPC_STAT, &shm_buffer);
    seg_size = shm_buffer.shm_segsz;
    printf("segment size: %d\n", seg_size);

    sprintf(shared_memory, "%s", argv[1]);
    printf("Added to the shared memory: %s\n\n", argv[1]);

    shmdt(shared_memory);

    if(fork() == 0)
    {
        char *str = (char*) shmat(seg_id, NULL, 0);

        printf("Data read from memory: %s\n",str);

        shmdt(shared_memory);

        shmctl(seg_id, IPC_RMID, NULL);

        return 0;
    }

    return 0;
}
```

خروجی:

```
negar@ubuntu:~$ cd Desktop
negar@ubuntu:~/Desktop$ gcc shared_memory.c -o output
negar@ubuntu:~/Desktop$ ./output Hello
shared memory address 0x7f34a6ddb000
segment size: 50
Added to the shared memory: Hello

negar@ubuntu:~/Desktop$ Data read from memory: Hello
```

عبارت Hello که در حافظه مشترک نوشته شده بود، خوانده و چاپ شد.

سوال ۲:

Client:

Socket را با تابع socket() تعریف می‌کنیم. اگر مقدار بازگردانده شده از این عبارت نامنفی بود، یعنی ارتباط درست برقرار شده است. آرگومان‌های آن را نیز بر مبنای عبارات زیر مقدار دهی می‌کنیم.

AF_INET: مشخص می‌کند که Address Family این socket چیست. در اینجا AF_INET به معنی IPv4 است و اگر از AF_INET6 استفاده میکردیم، به معنی IPv6 بود.

SOCKET_STREAM: نوع پرتکل transport layer این socket را به عنوان TCP تنظیم می‌کند. اگر از متغیر از پیش تعریف شده‌ی SOCKET_DGRAM استفاده میکردیم، سوکت با پروتکل UDP با transport layer شبکه ارتباط برقرار می‌کرد.

Protocol num را ۰ می‌گذاریم تا خود سیستم شماره‌ی پروتکل را تعیین کند.

در ادامه یک struct sockaddr_in ایجاد می‌کنیم و آن را serv_addr می‌گذاریم که نشان دهنده‌ی آدرس سرور است.

```c
int sockfd;
int n;
struct sockaddr_in serv_addr;
char buffer[MAXDATALEN];
char buf[10];


void *chat_read(int sockfd)
{
    while (1)
    {
        n = recv(sockfd, buffer, MAXDATALEN - 1, 0);
        if (n == 0)
        {
            printf("\nSERVER IS OFF :(\n");
            exit(0);
        }

        if (n > 0)
        {
            printf("-> %s", buffer);
            bzero(buffer, MAXDATALEN);
        }
    }
}

void *chat_write(int sockfd)
{
    while (1)
    {
        fgets(buffer, MAXDATALEN - 1, stdin);

        n = send(sockfd, buffer, strlen(buffer), 0);

        if (strncmp(buffer, "/quit", 5) == 0)
            exit(0);

        bzero(buffer, MAXDATALEN);
    }
}
```

```c
int main(int argc, char *argv[])
{

    pthread_t thr1, thr2;

    if (argc != 4)
    {
        printf("Enter server ip, port and your username in order :(.\n");
        exit(1);
    }

    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd == -1)
        printf("client socket error\n");
    else
        printf("socket created\n");

    serv_addr.sin_family = AF_INET;
    serv_addr.sin_port = htons(PORT);
    serv_addr.sin_addr.s_addr = inet_addr(argv[1]);


    strcpy(buf, argv[3]);

    if (connect(sockfd, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) == -1)
    {
        printf("client connect error\n");
        exit(0);
    }
    else
        printf("\rYOU JOINED AS %s\n", buf);

    send(sockfd, buf, strlen(buf), 0);

    pthread_create(&thr2, NULL, (void *)chat_write, (void *) (intptr_t) sockfd);
    pthread_create(&thr1, NULL, (void *)chat_read, (void *) (intptr_t) sockfd);

    pthread_join(thr2, NULL);
    pthread_join(thr1, NULL);
}
```

Server:

بعد از آنکه تنظیمات port و Address سرور انجام شد، سرور بر روی port تنظیم شده (۸۰۷۸) گوش می‌کند و منتظر اتصال یک کاربر می‌ماند. در زمان اتصال کاربر و اجرای دستورات، پیام‌های مناسب در برنامه چاپ می‌شوند.

```c
typedef struct
{
    int port;
    char username[15];
} User;

char username[15];
User users[50] = {0};
int userLast = 0;
User groups[50][50] = {0};
int groupLast[50] = {0};
char buffer[MAXDATALEN];

int search(int port, User *list, int last)
{
    for (int i = 0; i < last; i++)
    {
        if (list[i].port == port)
            return i;
    }
    return -1;
}


void delete_list(int port, User *list, int *last)
{
    int ptr = search(port, list, *last);
    if (ptr == -1)
    {
        return;
    }

    for (int i = ptr; i < *last - 1; i++)
    {
        list[i] = list[i + 1];
    }
    (*last)--;
}

void display_list(const User *list, int last)
{
    printf("Current online users:\n");
    if (last== 0)
    {
        printf("No one is online\n");
        return;
    }
```

```
    for (int i = 0; i < last; i++)
    {
        printf("%s\t",list[i].username);
    }
    printf("\n\n");
}


void *server(void *arguments)
{
    User *args = arguments;

    char buffer[MAXDATALEN], uname[10];
    char *msg = (char *)malloc(MAXDATALEN);
    int my_port, x, y;
    int msglen;

    my_port = args->port;
    strcpy(uname, args->username);

    while (true)
    {
        bzero(buffer, 256);
        y = recv(my_port, buffer, MAXDATALEN, 0);

        /* Client quits */
        if (!y || strncmp(buffer, "/quit", 5) == 0)
        {
            printf("!!! %s left the chat !!!\n\n",uname);

            delete_list(my_port, users, &userLast);
            for (int i = 0; i < 50; i++)
            {
                delete_list(my_port, groups[i], &groupLast[i]);
            }

            display_list(users, userLast);

            close(my_port);
            free(msg);

            break;
        }
```

```
    else if (strncmp(buffer, "join", 4) == 0)
    {
        char *group_id_str = malloc(sizeof(MAXDATALEN));
        strcpy(group_id_str, buffer + 6);
        int group_id = atoi(group_id_str);
        printf("!!! %d: %s joined group number %d !!!\n\n", my_port, uname, group_id);
        User *temp;
        temp = malloc(sizeof(User));
        temp->port = my_port;
        strcpy(temp->username, uname);
        groups[group_id][groupLast[group_id]++] = *temp;

    }

    else if (strncmp(buffer, "send", 4) == 0)
    {
        int nextSpace;

        for(int k = 5; k < strlen(buffer); k++)
            if(buffer[k] == ' ')
            {
                nextSpace = k;
                break;
            }
        char *group_id_str = malloc(sizeof(MAXDATALEN));
        strncpy(group_id_str, buffer + 6, nextSpace);
        int group_id = atoi(group_id_str);

        if (search(my_port, groups[group_id], groupLast[group_id]) == -1)
        {
            continue;
        }

        printf("%s %s\n", uname, buffer);

        strcpy(msg, uname);
        strcat(msg, " ");
        strcat(msg, buffer + nextSpace);
        msglen = strlen(msg);

        for (int i = 0; i < groupLast[group_id]; i++)
        {
            if (groups[group_id][i].port == my_port)
                continue;
            send(groups[group_id][i].port, msg, msglen, 0);
        }
```

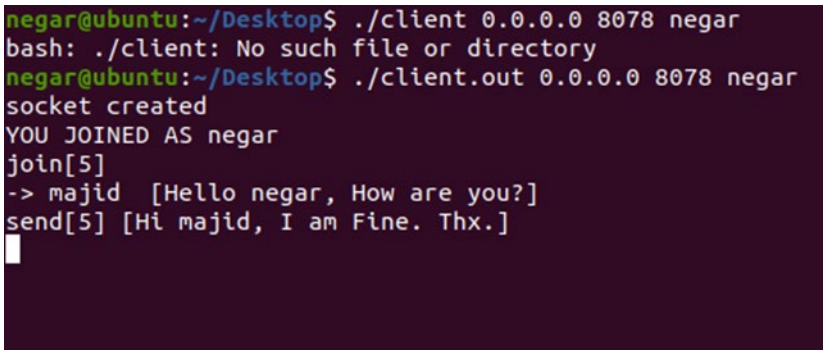
```
148                 for (int i = 0; i < groupLast[group_id]; i++)
149                 {
150                     if (groups[group_id][i].port == my_port)
151                         continue;
152                     send(groups[group_id][i].port, msg, msglen, 0);
153                 }
154 
155                 bzero(msg, MAXDATALEN);
156             }
157 
158             else if (strncmp(buffer, "leave", 5) == 0)
159             {
160                 char *group_id_str = malloc(sizeof(MAXDATALEN));
161                 strcpy(group_id_str, buffer + 7);
162                 printf("%s\n", group_id_str);
163                 int group_id = atoi(group_id_str);
164                 printf("!!! %s left group number %d !!!\n\n", uname, group_id);
165 
166                 delete_list(my_port, groups[group_id], &groupLast[group_id]);
167             }
168             display_list(users, userLast);
169         }
170         return 0;
171     }
172 
173 
174     int main(int argc, char *argv[])
175     {
176 
177         int sockfd, new_fd = 0;
178         struct sockaddr_in server_addr;
179         struct sockaddr_in client_addr;
180         int z;
181         pthread_t thr;
182         int yes = 1;
183 
184         printf("Starting Server\n");
185 
186         int temp = PORT;
187 
188 
189         server_addr.sin_family = AF_INET;
190         server_addr.sin_addr.s_addr = htonl(INADDR_ANY);
191         server_addr.sin_port = htons(temp);
192         int addlen = sizeof(struct sockaddr_in);
```

```
194         sockfd = socket(AF_INET, SOCK_STREAM, 0);
195         if (sockfd == -1)
196         {
197             printf("error");
198             exit(1);
199         }
200 
201         if (setsockopt(sockfd, SOL_SOCKET, SO_REUSEADDR, &yes, sizeof(int)) == -1)
202         {
203             printf("error");
204             exit(1);
205         }
206 
207         if (bind(sockfd, (struct sockaddr *)&server_addr, sizeof(struct sockaddr)) == -1)
208         {
209             printf("failed\n");
210             exit(1);
211         }
212 
213         listen(sockfd, BACKLOG);
214         printf("waiting for clients to join...\n");
215 
216 
217         while (true)
218         {
219             new_fd = accept(sockfd, (struct sockaddr *)&client_addr, &addlen);
220 
221             bzero(username, 10);
222             recv(new_fd, username, sizeof(username), 0);
223 
224             printf("%s JOINED CHAT \n\n",username);
225             User *temp;
226             temp = malloc(sizeof(User));
227             temp->port = new_fd;
228             strcpy(temp->username, username);
229             users[userLast++] = *temp;
230 
231             User args;
232             args.port = new_fd;
233             strcpy(args.username, username);
234 
235             pthread_create(&thr, NULL, server, (void *)&args);
236             pthread_detach(thr);
237         }
238 
239         close(sockfd);
240     }
```

خروجی:

```
negar@ubuntu:~/Desktop$ make
negar@ubuntu:~/Desktop$ ls
client.c  client.out  lab  Makefile  server.c  server.out
negar@ubuntu:~/Desktop$ ./server.out
Starting Server
waiting for clients to join...
negar JOINED CHAT

majid JOINED CHAT

!!! 5: majid joined group number 0 !!!

Current online users:
negar   majid

!!! 4: negar joined group number 0 !!!

Current online users:
negar   majid

majid send[5] [Hello negar, How are you?]

Current online users:
negar   majid

negar send[5] [Hi majid, I am Fine. Thx.]

Current online users:
negar   majid

]

!!! majid left group number 0 !!!

Current online users:
negar   majid

Current online users:
negar   majid

!!! majid left the chat !!!

Current online users:
negar
```

```
negar@ubuntu:~$ cd Desktop
negar@ubuntu:~/Desktop$ ./client.out 0.0.0.0 8078 majid
socket created
YOU JOINED AS majid
join[5]
send[5] [Hello negar, How are you?]
-> negar  [Hi majid, I am Fine. Thx.]
leave[5]

/quit
negar@ubuntu:~/Desktop$
```



در سه تصویر بالا تصویر اول مربوط به سرور است و تصویر دوم و سوم مربوط به دو کاربر هستند.

در این قسمت اینگونه عمل کرده‌ایم که دو کاربر به سرور متصل می‌شوند، سپس در یک گروه به نام ۵ عضو شده‌اند و در ادامه هر کدام برای دیگری یک پیام فرستاده‌اند سپس کاربر اول از گروه ۵ خارج می‌شود و همچنین آفلاین نیز می‌شود.

سوال ۳:

با توجه به دستور کار، جمله گفته شده را پردازه پدر ارسال می‌کند و پردازه فرزند، حروف بزرگ آن جمله را به کوچک و حروف کوچک را به بزرگ تبدیل می‌کند و آن را به پردازه پدر باز می‌گرداند.

کد:

```c
#include <sys/types.h>
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <ctype.h>

#define BUFFER_SIZE 25
#define READ_END 0
#define WRITE_END 1

int main(void)
{
    char write_msg[BUFFER_SIZE] = "This is First Proccess";
    char read_msg[BUFFER_SIZE];
    int fd[2];
    pid_t pid;

    if(pipe(fd) == -1)
    {
        fprintf(stderr, "pipe failed\n");
        return 1;
    }
    pid = fork();

    if(pid < 0)
    {
        fprintf(stderr, "fork failed\n");
        return 1;
    }
    if(pid > 0)
    {
        close(fd[READ_END]);
        write(fd[WRITE_END], write_msg, strlen(write_msg));
        close(fd[WRITE_END]);
    }
    else
    {
        close(fd[WRITE_END]);
        read(fd[READ_END], read_msg, BUFFER_SIZE);
        for(int i = 0 ;i < strlen(read_msg); i++)
        {
            if(islower(read_msg[i]))
                read_msg[i] = toupper(read_msg[i]);

          else
                read_msg[i] = tolower(read_msg[i]);

        }
        printf("%s\n", read_msg);
        close(fd[READ_END]);
    }
    return 0;
}
```

خروجی:

```
negar@ubuntu:~/Desktop$ gcc pipe.c -o output
negar@ubuntu:~/Desktop$ ./output
tHIS IS fIRST pROCCESS
negar@ubuntu:~/Desktop$
```