

<b>Name</b>	Anurag Nair
<b>UID no.</b>	2022600035
<b>Experiment No.</b>	8

<b>AIM:</b>	Chat bot using Shared Memory
-------------	------------------------------

Program	
---------	--

<b>PROGRAM:</b>	<pre> Chat1.c  #include &lt;stdio.h&gt; #include &lt;stdlib.h&gt; #include &lt;unistd.h&gt; #include &lt;string.h&gt; #include &lt;semaphore.h&gt; #include &lt;fcntl.h&gt; #include &lt;sys/types.h&gt; #include &lt;sys/ipc.h&gt; #include &lt;sys/shm.h&gt; #include &lt;sys/wait.h&gt;  #define SHM_SIZE 1024  int main() {     char *shared_memory;     int shmid;      shmid = shmget(20, SHM_SIZE, IPC_CREAT   0666);     if (shmid == -1)     {         perror("Error in creating shared memory");         exit(1);     }     printf("Memory segment successfully created\n");      shared_memory = shmat(shmid, NULL, 0);     if (shared_memory == (char *)-1) </pre>
-----------------	--

```

{
    perror("Error in attaching memory");
    exit(1);
}

sem_t *sem_user1, *sem_user2;
int flag = 0;

sem_user1 = sem_open("/user1_semaphore", O_CREAT | O_EXCL,
0644, 0);
if (sem_user1 == SEM_FAILED)
{
    sem_unlink("/user1_semaphore");
    sem_user1 = sem_open("/user1_semaphore", O_CREAT | O_EXCL,
0644, 0);
    if (sem_user1 == SEM_FAILED)
    {
        perror("Semaphore creation failed");
        exit(EXIT_FAILURE);
    }
}

sem_user2 = sem_open("/user2_semaphore", O_CREAT | O_EXCL,
0644, 1);
if (sem_user2 == SEM_FAILED)
{
    sem_unlink("/user2_semaphore");
    sem_user2 = sem_open("/user2_semaphore", O_CREAT | O_EXCL,
0644, 1);
    if (sem_user2 == SEM_FAILED)
    {
        perror("Semaphore creation failed");
        exit(EXIT_FAILURE);
    }
}

while (1)
{
    sem_wait(sem_user2);
    if (flag == 1)

```

```

        printf("User 2 says: %s", shared_memory);
        flag = 1;
        printf("Reply to User 2: ");
        fgets(shared_memory, SHM_SIZE, stdin);
        sem_post(sem_user1);
    }
    return 0;
}

```

#### Chat2.c

```

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <semaphore.h>
#include <fcntl.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <sys/wait.h>

#define SHM_SIZE 1024

int main()
{
    char *shared_memory;
    int shmid;

    shmid = shmget(20, SHM_SIZE, 0666);
    if (shmid == -1)
    {
        perror("Error in creating shared memory");
        exit(1);
    }
    printf("Memory segment successfully created\n");

    shared_memory = shmat(shmid, NULL, 0);
    if (shared_memory == (char *)-1)
    {

```

```

    perror("Error in attaching memory");
    exit(1);
}

sem_t *sem_user1, *sem_user2;

sem_user1 = sem_open("/user1_semaphore", 0);
if (sem_user1 == SEM_FAILED)
{
    sem_unlink("/user1_semaphore");
    sem_user1 = sem_open("/user1_semaphore", O_CREAT | O_EXCL,
0644, 0);
    if (sem_user1 == SEM_FAILED)
    {
        perror("Semaphore creation failed");
        exit(EXIT_FAILURE);
    }
}

sem_user2 = sem_open("/user2_semaphore", 1);
if (sem_user2 == SEM_FAILED)
{
    sem_unlink("/user2_semaphore");
    sem_user2 = sem_open("/user2_semaphore", O_CREAT | O_EXCL,
0644, 1);
    if (sem_user2 == SEM_FAILED)
    {
        perror("Semaphore creation failed");
        exit(EXIT_FAILURE);
    }
}

while (1)
{
    sem_wait(sem_user1);
    printf("User 1 says: %s", shared_memory);
    printf("Reply to User 1: ");
    fgets(shared_memory, SHM_SIZE, stdin);
    sem_post(sem_user2);
}

```

```
return 0;
}
```

## OUTPUT:

```
anurag@anurag-VirtualBox:~/Desktop/bc35$ gcc file1.c
anurag@anurag-VirtualBox:~/Desktop/bc35$ ./a.out
Memory segment successfully created
User 1 says: hi
Reply to User 1: hello
User 1 says: how are you
Reply to User 1: good
User 1 says: takeoff via runway 07 exit at 4
Reply to User 1: roger captain
```

anurag@anurag-VirtualBox: ~/Desktop/bc35

```
anurag@anurag-VirtualBox:~/Desktop/bc35$ gcc file2.c
anurag@anurag-VirtualBox:~/Desktop/bc35$ ./a.out
Memory segment successfully created
Reply to User 2: hi
User 2 says: hello
Reply to User 2: how are you
User 2 says: good
Reply to User 2: takeoff via runway 07 exit at 4
User 2 says: roger captain
```

## CONCLUSION:

From this experiment, I learned how two programs can talk to each other using shared memory and semaphores in C. Shared memory lets them share data, while semaphores help them take turns and avoid messing up the shared data. It showed me the importance of making sure processes don't interfere with each other when sharing data.