

B.BHANUTEJA REDDY-192325016

9. Illustrate the concept of inter-process communication using shared memory with a C program.

Aim:

To demonstrate inter-process communication (IPC) using shared memory in C, where two or more processes can communicate by accessing a common memory space.

Concept:

Inter-process communication (IPC) allows processes to exchange data and synchronize their actions. Shared memory is one of the most efficient IPC mechanisms. It allows multiple processes to access the same region of memory. One process writes to this shared memory, and other processes can read from it.

Procedure:

1. Create a shared memory segment using `shmget()`.
2. Attach the shared memory to the process address space using `shmat()`.
3. Write data into the shared memory by the writer process.
4. Read data from the shared memory by the reader process.
5. Detach the shared memory using `shmdt()`.
6. Destroy the shared memory segment using `shmctl()` once communication is done.

CODE:

```
#include <stdio.h>

#include <sys/ipc.h>

#include <sys/shm.h>

#include <string.h>

#define SHM_SIZE 1024

int main() {

    key_t key = 1234;
```

```
int shmid;

char *shm;

shmid = shmget(key, SHM_SIZE, 0666 | IPC_CREAT);
if (shmid == -1) {
    perror("shmget failed");
    return 1;
}

shm = shmat(shmid, NULL, 0);
if (shm == (char *) -1) {
    perror("shmat failed");
    return 1;
}

printf("Enter a message to write to shared memory: ");
fgets(shm, SHM_SIZE, stdin);

shmdt(shm);

printf("Message written to shared memory.\n");

return 0;
}

#include <stdio.h>

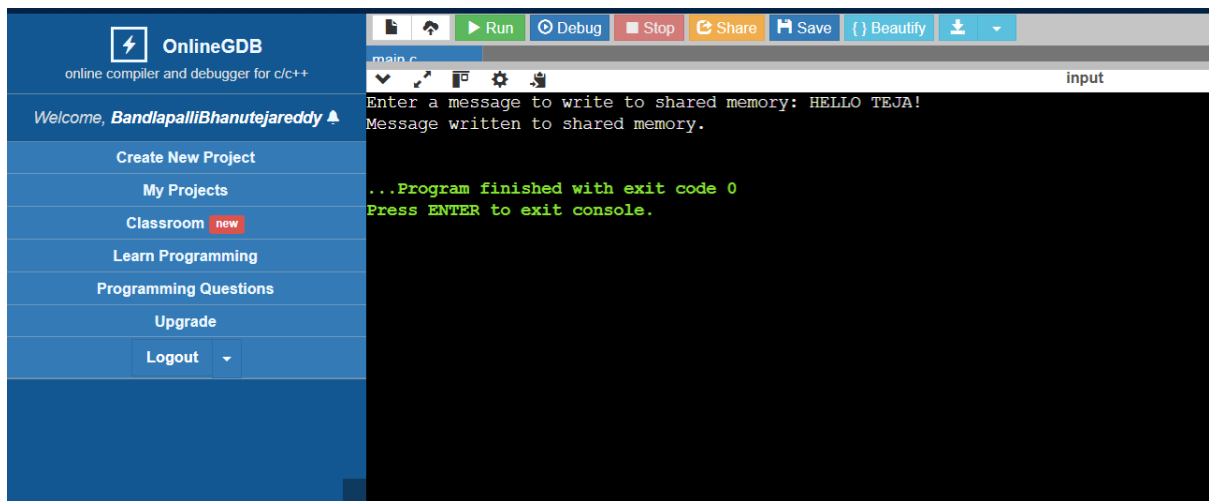
#include <sys/ipc.h>

#include <sys/shm.h>

#define SHM_SIZE 1024
```

```
int main() {  
    key_t key = 1234;  
  
    int shmid;  
  
    char *shm;  
  
    shmid = shmget(key, SHM_SIZE, 0666);  
    if (shmid == -1) {  
        perror("shmget failed");  
        return 1;  
    }  
  
    shm = shmat(shmid, NULL, 0);  
    if (shm == (char *) -1) {  
        perror("shmat failed");  
        return 1;  
    }  
  
    printf("Message read from shared memory: %s\n", shm);  
  
    shmdt(shm);  
    return 0;  
}
```

OUTPUT:



The screenshot displays the OnlineGDB web interface. On the left is a blue sidebar with the OnlineGDB logo and navigation links: 'Welcome, BandlapalliBhanutejareddy', 'Create New Project', 'My Projects', 'Classroom' (with a 'new' badge), 'Learn Programming', 'Programming Questions', 'Upgrade', and 'Logout'. The main area features a top toolbar with icons for file operations and buttons for 'Run', 'Debug', 'Stop', 'Share', 'Save', 'Beautify', and a download menu. Below the toolbar, a file explorer shows 'main.c' selected. The console output area at the bottom shows the program's execution: 'Enter a message to write to shared memory: HELLO TEJA!', 'Message written to shared memory.', and '...Program finished with exit code 0'. A prompt 'Press ENTER to exit console.' is visible at the bottom of the console.

```
main.c
input
Enter a message to write to shared memory: HELLO TEJA!
Message written to shared memory.
...Program finished with exit code 0
Press ENTER to exit console.
```