Name: K.Saikrishna

Reg-No: 192311106

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 one process writes data to a shared memory segment, and another process reads it.

Algorithm

1. Create a shared memory segment using shmget.
2. Attach the shared memory segment to the address space of the process using shmat.
3. In one process:
    - Write data to the shared memory segment.
4. In another process:
    - Attach to the same shared memory.
    - Read the data from the shared memory segment.
5. Detach and delete the shared memory segment using shmdt and shmctl.

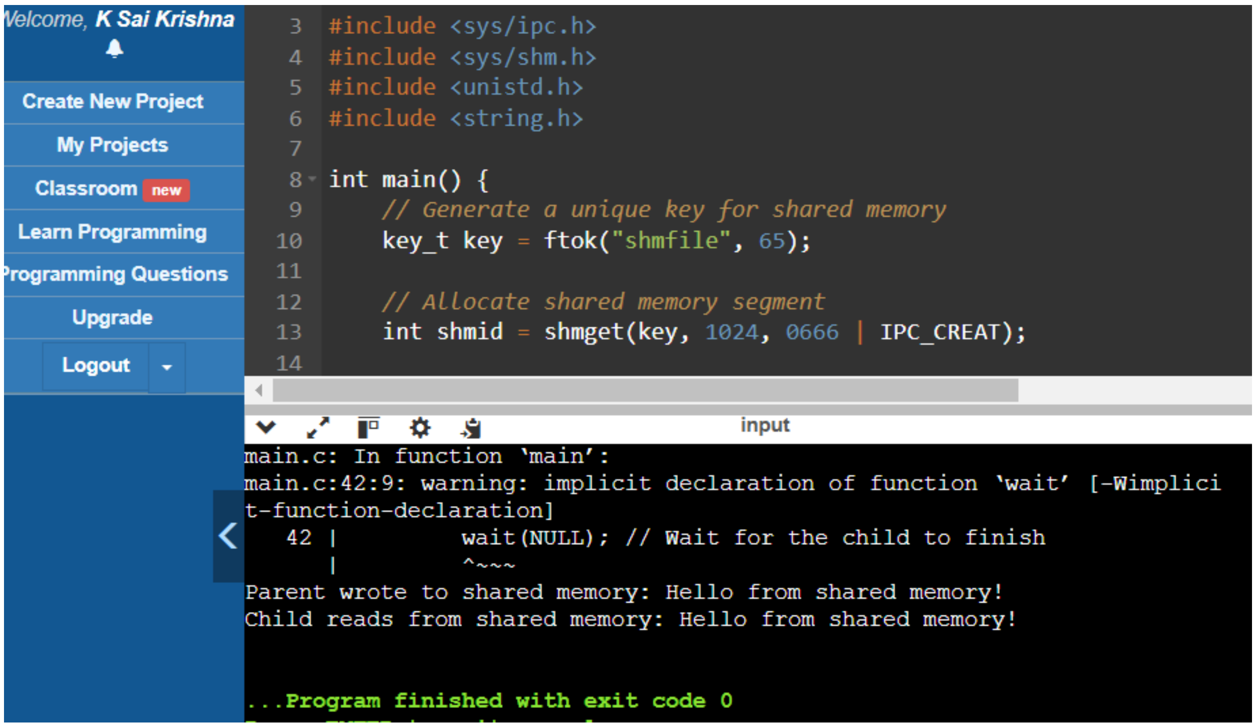
Procedure

1. Use fork() to create a parent and child process.
2. Parent writes data to the shared memory segment.
3. Child reads the data from the same shared memory segment.
4. Detach and clean up the shared memory after communication.

Code

```
include <sys/shm.h>
#include <sys/types.h>
#include <unistd.h>
#include <string.h>
#include <sys/wait.h>

#define SHM_SIZE 1024

int main() {
    key_t key = ftok("shmfile", 65);
    int shmid = shmget(key, SHM_SIZE, 0666 | IPC_CREAT);
    if (shmid == -1) {
        perror("shmget failed");
        return 1;
    }

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

    pid_t pid = fork();

    if (pid == 0) {
        sleep(1);
        printf("Child read: %s\n", shared_memory);
        shmdt(shared_memory);
    } else {
        strcpy(shared_memory, "Hello from parent!");
        wait(NULL);
        shmctl(shmid, IPC_RMID, NULL);
    }

    return 0;
}
```

Output:



```c
#include <sys/ipc.h>
#include <sys/shm.h>
#include <unistd.h>
#include <string.h>

int main() {
    // Generate a unique key for shared memory
    key_t key = ftok("shmfile", 65);

    // Allocate shared memory segment
    int shmid = shmget(key, 1024, 0666 | IPC_CREAT);

```

```
main.c: In function ‘main’:
main.c:42:9: warning: implicit declaration of function ‘wait’ [-Wimplici
t-function-declaration]
   42 |         wait(NULL); // Wait for the child to finish
      |         ^~~~
Parent wrote to shared memory: Hello from shared memory!
Child reads from shared memory: Hello from shared memory!


...Program finished with exit code 0
```

Result

- **Parent Process** writes the message "Hello from parent!" to the shared memory.
- **Child Process** reads the message from the shared memory and prints: Child read: Hello from parent!