Assignment 7B

server.cpp

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

#define SHM_SIZE 2  // size of shared memory in bytes
#define SHM_KEY 1234   // unique key for shared memory

using namespace std;

int main() {
    int shm_id;
    char *shm_ptr;

    // 1. Create a shared memory segment
    shm_id = shmget(SHM_KEY, SHM_SIZE, IPC_CREAT | 0666);
    if (shm_id < 0) {
        perror("shmget failed");
        exit(1);
    }
    cout << "Server: Shared memory created with ID \n" << shm_id << "\n";

    // 2. Attach the shared memory segment to server's address space
    shm_ptr = (char *) shmat(shm_id, NULL, 0);
    if (shm_ptr == (char *) -1) {
        perror("shmat failed");
        exit(1);
    }

    cout << "Server: Writing message to shared memory...\n";

    // 3. Write a message to shared memory
    char message[SHM_SIZE];
    cin.getline(message, SHM_SIZE);
    strncpy(shm_ptr, message, SHM_SIZE);

    cout << "Server: Message written.\n";

    // 4. Detach the shared memory
    shmdt(shm_ptr);
    return 0;
}
```

---

receiver.cpp

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

#define SHM_SIZE 2
#define SHM_KEY 1234

using namespace std;

int main() {
    int shm_id;
    char *shm_ptr;

    // 1. Access the shared memory segment created by server
    shm_id = shmget(SHM_KEY, SHM_SIZE, 0666);
    if (shm_id < 0) {
        perror("shmget failed");
        exit(1);
    }

    // 2. Attach to shared memory
    shm_ptr = (char *) shmat(shm_id, NULL, 0);
    if (shm_ptr == (char *) -1) {
        perror("shmat failed");
        exit(1);
    }

    // 3. Read and display the message
    cout << "Client: Message from shared memory:\n" << shm_ptr << "\n";

    // 4. Detach from shared memory
    shmdt(shm_ptr);

    // 5. Remove shared memory (optional: usually server does this)
    shmctl(shm_id, IPC_RMID, NULL);

    return 0;
}
```

Output

Server Side Execution

```
$ ./server
Server: Shared memory created with ID
512
Server: Writing message to shared memory...
Hi
Server: Message written.
```

Client Side Execution

```
$ ./receiver
Client: Message from shared memory:
Hi
```