Execute | Beautify | Share | Source Code | Help

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

#define SHM_SIZE 1024  // Define the size of shared memory

int main() {
    key_t key = 1234;  // Key for shared memory segment
    int shm_id;
    char *shm_ptr;

    // Create the shared memory segment
    shm_id = shmget(key, SHM_SIZE, IPC_CREAT | 0666);
    if (shm_id == -1) {
        perror("shmget");
        exit(1);
    }

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

```

Terminal

```
Writer process: Enter a message to write to shared memory: hii
Writer process: Data written to shared memory: hii
Reader process: Data read from shared memory: hii
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#include <unistd.h>
#define MAX_TEXT 512
struct message {
    long msg_type;
    char text[MAX_TEXT];
};
int main() {
    key_t key = ftok("progfile", 65); // Generate unique key
    int msgid = msgget(key, 0666 | IPC_CREAT); // Create message queue
    struct message msg;
    msg.msg_type = 1; // Set message type
    printf("Write Message: ");
    fgets(msg.text, MAX_TEXT, stdin);
    msgsnd(msgid, &msg, sizeof(msg), 0);
    msgrcv(msgid, &msg, sizeof(msg), 1, 0); // Receive message
    printf("Data Received: %s", msg.text);
    msgctl(msgid, IPC_RMID, NULL);
    return 0;
}
```

```
Write Message: ooop
sending message=ooop

Data Received: ooop
```