

main.c

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4 #include <sys/ipc.h>
5 #include <sys/shm.h>
6 #include <sys/types.h>
7 #include <unistd.h>
8 #include <sys/wait.h>
9 #define SHM_KEY 12345
10 #define SHM_SIZE 1024
11 void writer() {
12     int shm_id;
13     char *shm_ptr;
14     shm_id = shmget(SHM_KEY, SHM_SIZE, IPC_CREAT | 0666);
15     if (shm_id < 0) {
16         perror("shmget failed");
17         exit(1);}
18     shm_ptr = (char *)shmat(shm_id, NULL, 0);
19     if (shm_ptr == (char *)(-1)) {
20         perror("shmat failed");
21         exit(1);}
22     printf("Writer: Enter some text to share: ");
23     fgets(shm_ptr, SHM_SIZE, stdin);
24     if (shmdt(shm_ptr) < 0) {
25         perror("shmdt failed");
26         exit(1);}}
27 void reader() {
28     int shm_id;
29     char *shm_ptr;
30     shm_id = shmget(SHM_KEY, SHM_SIZE, 0666);
31     if (shm_id < 0) {

```

Output

Writer: Enter some text to share: Reader: Received data from shared memory:

```
31+ if (shm_id < 0) {
32     perror("shmget failed");
33     exit(1);}
34 shm_ptr = (char *)shmat(shm_id, NULL, 0);
35+ if (shm_ptr == (char *)(-1)) {
36     perror("shmat failed");
37     exit(1);}
38 printf("Reader: Received data from shared memory: %s", shm_ptr);
39+ if (shmdt(shm_ptr) < 0) {
40     perror("shmdt failed");
41     exit(1);}}
42+ int main() {
43     pid_t pid;
44     pid = fork();
45+ if (pid == -1) {
46     perror("fork failed");
47     exit(1);}
48+ if (pid == 0) {
49     sleep(1);
50     reader();}
51+ else {
52     writer();
53     wait(NULL);}
54 shmctl(shmget(SHM_KEY, SHM_SIZE, 0666), IPC_RMID, NULL);
55 return 0;}
```

main.c

Share

Run

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4 #include <sys/ipc.h>
5 #include <sys/msg.h>
6 #include <sys/types.h>
7 #include <unistd.h>
8 #include <sys/wait.h>
9 #define MSG_SIZE 1024
10 struct msg_buffer {
11     long msg_type;
12     char msg_text[MSG_SIZE];
13 }
14 #define MSG_KEY 12345
15 void writer() {
16     int msgid;
17     struct msg_buffer message;
18     msgid = msgget(MSG_KEY, IPC_CREAT | 0666);
19     if (msgid == -1) {
20         perror("msgget failed");
21         exit(1);
22     }
23     message.msg_type = 1; // Message type is 1
24     printf("Writer: Enter some text to send: ");
25     fgets(message.msg_text, MSG_SIZE, stdin);
26     if (msgsnd(msgid, &message, sizeof(message.msg_text), 0) == -1) {
27         perror("msgsnd failed");
28         exit(1);
29     }
30     printf("Writer: Message sent to queue\n");
31 }
32 void reader() {
33     int msgid;
34     struct msg_buffer message;
35     msgid = msgget(MSG_KEY, 0666);
```

Output

Clear

Writer: Enter some text to send: hii  
Writer: Message sent to queue  
Reader: Received message: hii  
  
=== Code Execution Successful ===

```
32· if (msgid == -1) {
33·     perror("msgget failed");
34·     exit(1);
35· if (msgrcv(msgid, &message, sizeof(message.msg_text), 1, 0) == -1) {
36·     perror("msgrcv failed");
37·     exit(1);
38· printf("Reader: Received message: %s", message.msg_text);}
39· int main() {
40·     pid_t pid;
41·     pid = fork();
42·     if (pid == -1) {
43·         perror("fork failed");
44·         exit(1);
45·     if (pid == 0) {
46·         sleep(1);
47·         reader();
48·     } else {
49·         writer();
50·         wait(NULL);}
51· msgctl(msgget(MSG_KEY, 0666), IPC_RMID, NULL);
52· return 0;}
```