main.c

Share Run

```c
#include <sys/wait.h>

int main() {

    key_t key = 1234;
    int shmid = shmget(key, 1024, 0666 | IPC_CREAT);
    if (shmid == -1) {
        perror("shmget failed");
        return 1;
    }

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

    if (fork() == 0) {
        sleep(1);
        printf("Child reads: %s\n", shared_mem);
        shmdt(shared_mem);
    } else {
        strcpy(shared_mem, "Hello IPC via Shared Memory");
        printf("Parent writes: %s\n", shared_mem);
        wait(NULL);
        shmdt(shared_mem);
        shmctl(shmid, IPC_RMID, NULL);
    }

    return 0;
}

```

Output

Clear

```
Parent writes: Hello IPC via Shared Memory
Child reads: Hello IPC via Shared Memory


=== Code Execution Successful ===
```

main.c

Share Run

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

struct msg_buffer {
    long msg_type;
    char msg_text[100];
} message;

int main() {
    key_t key = 1234;
    int msgid = msgget(key, 0666 | IPC_CREAT);

    if (fork() == 0) {
        sleep(1);
        msgrcv(msgid, &message, sizeof(message), 1, 0);
        printf("Receiver got: %s\n", message.msg_text);
        msgctl(msgid, IPC_RMID, NULL);
    } else {
        message.msg_type = 1;
        strcpy(message.msg_text, "Hello from Sender");
        msgsnd(msgid, &message, sizeof(message), 0);
        printf("Sender sent: %s\n", message.msg_text);
    }
    return 0;
}

```

Output

Clear

```
Sender sent: Hello from Sender


=== Code Execution Successful ===
```