Server

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

#define SIZE 400
#define DATA_NOT_FILLED -1
#define DATA_FILLED 0
#define DATA_READ_CLIENT 1

typedef struct data {
   int status;
   char buff[100];
} datal;

int main() {
   key_t key;
   int shmid, i = 0;
   char ch;
   datal *shm_ptr;

   // Use a fixed, absolute path for consistent key
   key = ftok("/tmp", 'A');
```

```
if (key == -1) {
   perror("ftok");
   exit(1);
}

// Create shared memory segment
shmid = shmget(key, SIZE, IPC_CREAT | 0666);
if (shmid < 0) {
   perror("shmget");
   exit(1);
}

printf("Shared memory created with id: %d\n", shmid);

// Attach shared memory
shm_ptr = (datal *)shmat(shmid, NULL, 0);
if (shm_ptr == (void *)-1) {
   perror("shmat");
   exit(1);
}

shm_ptr->status = DATA_NOT_FILLED;

printf("Enter text ending with '#': ");
fflush(stdout);

// Read input until '#' is entered
i = 0;
```

```
while ((ch = getchar()) != '#' && i < (int)sizeof(shm_ptr->buff) - 1) {
   shm_ptr->buff[i++] = ch;
}
shm_ptr->buff[i] = '\0';

// Write to shared memory
shm_ptr->status = DATA_FILLED;
printf("Data written to shared memory: \"%s\"\n", shm_ptr->buff);

// Wait for client to read the data
int waitCount = 0;
while (shm_ptr->status != DATA_READ_CLIENT && waitCount < 20) {
   printf("Waiting for client to read the data...\n");
   fflush(stdout);
   sleep(1);
   waitCount++;
}

if (shm_ptr->status == DATA_READ_CLIENT) {
   printf("\n Client has read the data.\n");
} else {
   printf("\n Timeout: Client did not read data within 20 seconds.\n");
}

// Detach shared memory
if (shmdt(shm_ptr) == -1) {
   perror("shmdt");
} else {
```

```
        printf("Shared memory detached successfully.\n");
    }

    // Remove shared memory
    if (shmctl(shmid, IPC_RMID, NULL) == -1) {
        perror("shmctl");
    } else {
        printf("Shared memory removed successfully.\n");
    }

    printf("Server exiting.\n");
    return 0;
}
```

Receiver

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

#define SIZE 400
#define DATA_NOT_FILLED -1
```

```
#define DATA_FILLED 0
#define DATA_READ_CLIENT 1

typedef struct data {
   int status;
   char buff[100];
} datal;

int main() {
   key_t key;
   int shmid;
   datal *shm_ptr;

   // Use same key path as server
   key = ftok("/tmp", 'A');
   if (key == -1) {
      perror("ftok");
      exit(1);
   }

   // Access the existing shared memory
   shmid = shmget(key, SIZE, 0666);
   if (shmid < 0) {
      perror("shmget");
      exit(1);
   }

   // Attach shared memory
```

```
shm_ptr = (datal *)shmat(shmid, NULL, 0);
if (shm_ptr == (void *)-1) {
   perror("shmat");
   exit(1);
}

printf("Client started, waiting for server to write data...\n");

// Wait for server to write data (with timeout)
int waitCount = 0;
while (shm_ptr->status != DATA_FILLED && waitCount < 20) {
   printf("Waiting for server to fill data...\n");
   fflush(stdout);
   sleep(1);
   waitCount++;
}

if (shm_ptr->status == DATA_FILLED) {
   printf("\n✅ Data read from shared memory: \"%s\"\n", shm_ptr->buff);

   // Mark as read
   shm_ptr->status = DATA_READ_CLIENT;
   printf("Client marked data as read.\n");
} else {
   printf("\n⚠ Timeout: Server did not write data within 20 seconds.\n");
}

// Detach shared memory
```

```
    if (shmdt(shm_ptr) == -1) {
        perror("shmdt");
    } else {
        printf("Shared memory detached by client successfully.\n");
    }

    printf("Client exiting.\n");
    return 0;
}
```