```c
/**
 * @Author: Brandon Baars Mike Ford <brandon>
 * @Date:   Thursday, February 8th 2018, 5:03:24 pm
 * @Filename: reader.c
 * @Last modified by:   brandon
 * @Last modified time: Monday, February 12th 2018, 8:17:37 pm
 */

/*
* Mike Ford, Brandon Baars
* Reader Program with Shared Memory
*/

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

#define SIZE 4096

int main (int argc, char **argv) {

    // create our shared memory with unique key
    key_t key;
    char *path = "/tmp";
    int id = 'X';
    key = ftok(path, id);

    // shared memory segment id and pointers
    int shmId;
    char *shmPtr;
    char *s;

    // Create the shared segment
    if ((shmId = shmget(key, SIZE, IPC_CREAT | S_IRUSR | S_IWUSR)) < 0) {
        perror("Cannot create shared memory");
        exit(1);
    }

    // Attach the shared segment
    if ((shmPtr = shmat(shmId, 0, 0)) == (void *) - 1) {
        perror("I can't attach");
        exit(1);
    }

    *shmPtr = '/';
    while(*shmPtr != '~') {

        // WON'T WORK WITHOUT A PRINT STATEMENT
        printf("\n");

        while(*shmPtr != '#' && *shmPtr != '~')
            ;

        // if we received the quit command from the writer
        // break out of our loop
        if (*shmPtr == '~') {
            break;
        }

        // Read in everything in our shared memory
        for (s = ++shmPtr; *s != '%'; s++) {
            putchar(*s);
        }
```

```
67
68          *--shmPtr = '*';
69      }
70
71      return EXIT_SUCCESS;
72  }
```

```c
/**
 * @Author: Brandon Baars, Mike Ford <brandon>
 * @Date:   Thursday, February 8th 2018, 5:03:24 pm
 * @Filename: writer.c
 * @Last modified by:   brandon
 * @Last modified time: Monday, February 12th 2018, 8:12:21 pm
 */

/*
* Mike Ford, Brandon Baars
* Writer Program with Shared Memory
*/

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

#define SIZE 4096

int main (int argc, char **argv) {

    // create our shared memory with unique key
    key_t key;
    char *path = "/tmp";
    int id = 'X';

    key = ftok(path, id);

    // shared memory segment id and pointers
    int shmId;
    char *shmPtr, *shmCpy;
    char usrInpt = '/';
    char inputString[80];

    // Create the shared segment
    if ((shmId = shmget(key, SIZE, IPC_CREAT | S_IRUSR | S_IWUSR)) < 0) {
        perror("Cannot create shared memory");
        exit(1);
    }

    // Attach the shared segment
    if ((shmPtr = shmat(shmId, 0, 0)) == (void *) - 1) {
        perror("I can't attach");
        exit(1);
    }

    shmCpy = shmPtr;
    shmCpy++;

    while (usrInpt != '~') {

        fgets(inputString, 80, stdin);
        inputString[strlen(inputString) - 1] = '\0';

        if (!strcmp(inputString, "quit") || !strcmp(inputString, "Quit")) {
            *shmPtr = '~';
            break;
        }

        // Put what the user types into the shared memory segment
        // while((usrInpt = getchar()) != '\n') {
        //      *shmCpy++ = usrInpt;
```

```
67          // }
68          int c = 0;
69          for (c = 0; c < strlen(inputString); c++){
70              *shmCpy++ = inputString[c];
71          }
72
73          // put a character on the end of our input to let the reader know
74          // how many characters to read to.
75          *shmCpy = '%';
76
77          // change the pointer to a character to let the reader know theres
78          // data within our shared memory segment
79          *shmPtr = '#';
80
81          // Wait for the ack back from our client when they're done reading.
82          printf("Waiting for ack..\n");
83          while(*shmPtr != '*')
84              ;
85          memset(shmPtr, 0, 4096);
86          printf("Enter new message: ");
87      }
88
89      // Remove the shared memory
90      if (shmctl (shmId, IPC_RMID, 0) < 0) {
91                  perror ("can't deallocate\n");
92                  exit (1);
93      }
94
95      return EXIT_SUCCESS;
96  }
```