

```

1  /**
2   * @Author: Brandon Baars Mike Ford <brandon>
3   * @Date:   Thursday, February 8th 2018, 5:03:24 pm
4   * @Filename: reader.c
5   * @Last modified by:   brandon
6   * @Last modified time: Monday, February 12th 2018, 8:17:37 pm
7   */
8
9  /*
10 * Mike Ford, Brandon Baars
11 * Reader Program with Shared Memory
12 */
13
14 #include <stdio.h>
15 #include <stdlib.h>
16 #include <sys/types.h>
17 #include <sys/stat.h>
18 #include <sys/ipc.h>
19 #include <sys/shm.h>
20
21 #define SIZE 4096
22
23 int main (int argc, char **argv) {
24
25     // create our shared memory with unique key
26     key_t key;
27     char *path = "/tmp";
28     int id = 'X';
29     key = ftok(path, id);
30
31     // shared memory segment id and pointers
32     int shmId;
33     char *shmPtr;
34     char *s;
35
36     // Create the shared segment
37     if ((shmId = shmget(key, SIZE, IPC_CREAT | S_IRUSR | S_IWUSR)) < 0) {
38         perror("Cannot create shared memory");
39         exit(1);
40     }
41
42     // Attach the shared segment
43     if ((shmPtr = shmat(shmId, 0, 0)) == (void *) - 1) {
44         perror("I can't attach");
45         exit(1);
46     }
47
48     *shmPtr = '/';
49     while(*shmPtr != '~') {
50
51         // WON'T WORK WITHOUT A PRINT STATEMENT
52         printf("\n");
53
54         while(*shmPtr != '#' && *shmPtr != '~')
55             ;
56
57         // if we received the quit command from the writer
58         // break out of our loop
59         if (*shmPtr == '~') {
60             break;
61         }
62
63         // Read in everything in our shared memory
64         for (s = ++shmPtr; *s != '%'; s++) {
65             putchar(*s);
66         }

```

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

```

1  /**
2   * @Author: Brandon Baars, Mike Ford <brandon>
3   * @Date:   Thursday, February 8th 2018, 5:03:24 pm
4   * @Filename: writer.c
5   * @Last modified by:   brandon
6   * @Last modified time: Monday, February 12th 2018, 8:12:21 pm
7   */
8
9  /*
10   * Mike Ford, Brandon Baars
11   * Writer Program with Shared Memory
12   */
13
14  #include <stdio.h>
15  #include <stdlib.h>
16  #include <string.h>
17  #include <sys/types.h>
18  #include <sys/stat.h>
19  #include <sys/ipc.h>
20  #include <sys/shm.h>
21
22  #define SIZE 4096
23
24  int main (int argc, char **argv) {
25
26      // create our shared memory with unique key
27      key_t key;
28      char *path = "/tmp";
29      int id = 'X';
30
31      key = ftok(path, id);
32
33      // shared memory segment id and pointers
34      int shmId;
35      char *shmPtr, *shmCpy;
36      char usrInpt = '/';
37      char inputString[80];
38
39      // Create the shared segment
40      if ((shmId = shmget(key, SIZE, IPC_CREAT | S_IRUSR | S_IWUSR)) < 0) {
41          perror("Cannot create shared memory");
42          exit(1);
43      }
44
45      // Attach the shared segment
46      if ((shmPtr = shmat(shmId, 0, 0)) == (void *) - 1) {
47          perror("I can't attach");
48          exit(1);
49      }
50
51      shmCpy = shmPtr;
52      shmCpy++;
53
54      while (usrInpt != '~') {
55
56          fgets(inputString, 80, stdin);
57          inputString[strlen(inputString) - 1] = '\0';
58
59          if (!strcmp(inputString, "quit") || !strcmp(inputString, "Quit")) {
60              *shmPtr = '~';
61              break;
62          }
63
64          // Put what the user types into the shared memory segment
65          // while((usrInpt = getchar()) != '\n') {
66              *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 }

```