

1. Shared Memory

Code:

Shared1

```
1 //shared1.cpp
2 /* After the headers the shared memory segment
3 (the size of our shared memory structure) is created with a call to shmget,
4 with the IPC_CREAT bit specified. It reads data from the shared memory. */
5 #include <unistd.h>
6 #include <stdlib.h>
7 #include <stdio.h>
8 #include <string.h>
9 #include <semaphore.h>
10 #include <sys/stat.h>
11 #include <fcntl.h>
12 #include <sys/types.h>
13 #include <sys/ipc.h>
14 #include <sys/shm.h>
15
16 #define TEXT_SZ 2048
17 struct shared_use_st {
18     int written_by_you;
19     char some_text[TEXT_SZ];
20 };
21
22 int main() {
23     char SEM_NAME[] = "foo";
24     sem_t * mutex;
25     mutex = sem_open(SEM_NAME, O_CREAT, 0644, 1);
26
27     if (mutex == SEM_FAILED) {
28         perror("Cannot create semaphore");
29         sem_unlink(SEM_NAME);
30         exit(-1);
31     }
32
33     int running = 1;
34     void *shared_memory = (void *)0;
35     struct shared_use_st *shared_stuff;
36     int shmid;
37     srand((unsigned int) getpid());
38     shmid = shmget((key_t)1234, sizeof(struct shared_use_st), 0666 | IPC_CREAT);
39
40     if (shmid == -1) {
41         fprintf(stderr, "shmget failed\n");
42         exit(EXIT_FAILURE);
43     }
44
45     /* We now make the shared memory accessible to the program. */
46     shared_memory = shmat(shmid, (void *)0, 0);
47     if (shared_memory == (void *)-1) {
48         fprintf(stderr, "shmat failed\n");
49         exit(EXIT_FAILURE);
50     }
51
52     printf("Memory attached at %X\n", (long)shared_memory);
```

```
51 |
52 | printf("Memory attached at %X\n", (long)shared_memory);
53 | /* The next portion of the program assigns the shared_memory segment to shared_stuff,
54 | which then prints out any text in written_by_you. The loop continues until end is found
55 | in written_by_you. The call to sleep forces the consumer to sit in its critical section,
56 | which makes the producer wait. */
57 | shared_stuff = (struct shared_use_st *)shared_memory;
58 | shared_stuff->written_by_you = 0;
59 |
60 | while(running) {
61 |     if (shared_stuff->written_by_you) {
62 |         sem_wait(mutex);
63 |         printf("You wrote: %s", shared_stuff->some_text);
64 |         sleep( rand() % 4 ); /* make the other process wait for us ! */
65 |         shared_stuff->written_by_you = 0;
66 |         sem_post(mutex);
67 |         if (strcmp(shared_stuff->some_text, "end", 3) == 0) {
68 |             running = 0;
69 |         }
70 |     }
71 | }
72 |
73 | /* Lastly, the shared memory is detached and then deleted. */
74 | if (shmdt(shared_memory) == -1) {
75 |     fprintf(stderr, "shmdt failed\n");
76 |     exit(EXIT_FAILURE);
77 | }
78 |
79 | if (shmctl(shmid, IPC_RMID, 0) == -1) {
80 |     fprintf(stderr, "shmctl(IPC_RMID) failed\n");
81 |     exit(EXIT_FAILURE);
82 | }
83 |
84 | sem_close(mutex);
85 | sem_unlink(SEM_NAME);
86 | exit(EXIT_SUCCESS);
87 | }
88 |
89 | }
```

Shared2:

```
1 /*
2 shared2.cpp: Similar to shared1.cpp except that it writes data to
3 the shared memory.
4 */
5 #include <unistd.h>
6 #include <stdlib.h>
7 #include <stdio.h>
8 #include <string.h>
9 #include <semaphore.h>
10 #include <sys/stat.h>
11 #include <fcntl.h>
12 #include <sys/types.h>
13 #include <sys/ipc.h>
14 #include <sys/shm.h>
15
16 #define TEXT_SZ 2048
17
18 struct shared_use_st {
19     int written_by_you;
20     char some_text[TEXT_SZ];
21 };
22
23 int main() {
24     char SEM_NAME[] = "foo";
25     sem_t * mutex;
26     mutex = sem_open(SEM_NAME, 0, 0644, 0);
27
28     if(mutex == SEM_FAILED) {
29         perror("Reader: Can't access semaphore");
30         sem_close(mutex);
31         exit(-1);
32     }
33
34     int running = 1;
35     void *shared_memory = (void *)0;
36     struct shared_use_st *shared_stuff;
37     char buffer[BUFSIZ];
38     int shmid;
39     shmid = shmget((key_t)1234, sizeof(struct shared_use_st), 0666 | IPC_CREAT);
40
41     if (shmid == -1) {
42         fprintf(stderr, "shmget failed\n");
43         exit(EXIT_FAILURE);
44     }
45
46     shared_memory = shmat(shmid, (void *)0, 0);
47     if (shared_memory == (void *)-1) {
48         fprintf(stderr, "shmat failed\n");
49         exit(EXIT_FAILURE);
50     }
51
52     printf("Memory attached at %X\n", (long)shared_memory);
```

```

51
52     printf("Memory attached at %X\n", (long)shared_memory);
53     shared_stuff = (struct shared_use_st *)shared_memory;
54
55     while(running) {
56         while(shared_stuff->written_by_you == 1) {
57             sleep(1);
58             printf("waiting for client...\n");
59         }
60         sem_wait(mutex);
61         printf("Enter some text: ");
62         fgets(buffer, BUFSIZ, stdin);
63         strncpy(shared_stuff->some_text, buffer, TEXT_SZ);
64         shared_stuff->written_by_you = 1;
65         sem_post(mutex);
66
67         if (strncmp(buffer, "end", 3) == 0) {
68             running = 0;
69         }
70     }
71
72     if (shmdt(shared_memory) == -1) {
73         fprintf(stderr, "shmdt failed\n");
74         exit(EXIT_FAILURE);
75     }
76
77     sem_close(mutex);
78     sem_unlink(SEM_NAME);
79     exit(EXIT_SUCCESS);
80 }

```

Output:

```

004893625@csusb.edu@jib359-31:~
File Edit View Search Terminal Help
004893625@csusb.edu@jib359-31 ~]$ ls
SE201 CSE460 Diagram1.dia.autosave Music
SE202 CSE512 Documents perl5
SE320 CSE516 Downloads Pictur
SE330 CSE570 java.log.4598 Public
SE455 Desktop matlab_crash_dump.4598-1 salemv
004893625@csusb.edu@jib359-31 ~]$ cd CSE460/
004893625@csusb.edu@jib359-31 CSE460]$ ls
HW1 HW2 HW3 Lab1 Lab2 Lab3 Lab4 Lab5 Lab6
004893625@csusb.edu@jib359-31 CSE460]$ cd Lab7
004893625@csusb.edu@jib359-31 Lab7]$ ls
odclient.cpp proc.c shared1 shared2
odserver.cpp ps.c shared1.cpp shared2.cpp
004893625@csusb.edu@jib359-31 Lab7]$ ./shared1
Memory attached at C4799000
You wrote: Hello
You wrote: This is nice

004893625@csusb.edu@jib359-31:~/CSE460/Lab7
File Edit View Search Terminal Help
[004893625@csusb.edu@jib359-31 Lab7]$ ./shared2
Memory attached at CF0E3000
Enter some text: Hello
waiting for client...
Enter some text: This is nice
waiting for client...
Enter some text: 

```

2. POSIX Semaphores

Code:

Modserver

```
1 #include <sys/types.h>
2 #include <sys/ipc.h>
3 #include <sys/shm.h>
4 #include <stdio.h>
5 #include <semaphore.h>
6 #include <sys/types.h>
7 #include <sys/stat.h>
8 #include <fcntl.h>
9 #include <stdlib.h>
10 #include <string.h>
11 #include <time.h>
12 #include <iostream>
13
14 #define SHMSZ 27
15 #define SIZE 1024
16 using namespace std; char SEM_NAME[] = "SEM";
17 char modif_buffer[SIZE];
18 int main() {
19     int running = 1;
20     char ch;
21     int shmid;
22     key_t key;
23     char *shm, *s;
24     sem_t *mutex;
25     key = 1000;
26     //names the shared memory segment
27     //creates and initializes semaphore
28     mutex = sem_open(SEM_NAME, 0, 0644, 0);
29     if(mutex == SEM_FAILED) {
30         perror("Unable to execute semaphore");
31         sem_close(mutex);
32         exit(-1);
33     }
34     //creates the shared memory segment with this key
35     shmid = shmget(key, SHMSZ, 0666);
36     if(shmid < 0) {
37         perror("Failure in shmget");
38         exit(-1);
39     }
40     shm = (char*) shmat(shmid, NULL, 0);
41     //attach to virtual mem
42     int size;
43     while (running) {
44         sem_wait(mutex);
45         for (s = shm; *s != 0; s++) {
46             modif_buffer[size++] = *s;
47             printf("%s\n", modif_buffer);
48             if (modif_buffer == "quit") {
49                 running = 0;
50                 break;
51             }
52         }
53     }
```



```
45 for (s = shm; *s != 0; s++) {
46     modif_buffer[size++] = *s;
47     printf("%s\n", modif_buffer);
48     if (modif_buffer == "quit") {
49         running = 0;
50         break;
51     }
52 }
53 sem_post(mutex);
54 }
55 *shm = '*';
56 sem_close(mutex);
57 shmctl(shmid, IPC_RMID, 0);
58 exit(0);
59 }
```

```
1#include <sys/types.h>
2#include <sys/ipc.h>
3#include <sys/shm.h>
4#include <stdio.h>
5#include <semaphore.h>
6#include <sys/types.h>
7#include <sys/stat.h>
8#include <fcntl.h>
9#include <stdlib.h>
10#include <string.h>
11#include <time.h>
12#include <iostream>
13
14#define SHMSZ 27
15#define TEXT_SZ 2048
16char SEM_NAME[] = "SEM";
17struct shared_use_st {
18    int written_by_you;
19    char some_text[TEXT_SZ];
20};
21int main() {
22    int running = 1;
23    void *shared_memory = (void *)0;
24    struct shared_use_st *shared_stuff;
25    char buffer[BUFSIZ];
26    int shmid;
27    key_t key;
28    char *shm,*s;
29    sem_t *mutex;
30    key = 1000;
31    //creates and initializes semaphore
32    mutex = sem_open(SEM_NAME, O_CREAT, 0644, 1);
33    if (mutex == SEM_FAILED) {
34        perror("Unable to create semaphore");
35        sem_unlink(SEM_NAME);
36        exit(-1);
37    }
38    shm = (char*) shmat(shmid, NULL, 0);
39    //attach to shared mem
40    shmid = shmget(key, SHMSZ, IPC_CREAT | 0666);
41    if(shmid < 0) {
42        perror("Failure in shmget");
43        exit(-1);
44    }
45    shmid = shmget((key_t)1234, sizeof(struct shared_use_st), 0666 |
46    IPC_CREAT);if (shmid == -1) {
47        fprintf(stderr, "shmget failed\n");
48        exit(EXIT_FAILURE);
49    }
50    shared_memory = shmat(shmid, (void *)0, 0);
51    if (shared_memory == (void *)-1) {
52        fprintf(stderr, "Failure\n");
```

```

52 fprintf(stderr, "Failure\n");
53 exit(EXIT_FAILURE);
54 }
55 shared_stuff = (struct shared_use_st *)shared_memory;
56 while(running) {
57 while(shared_stuff->written_by_you == 1) {
58 sleep(1);
59 }
60 sem_wait(mutex);
61 printf("Enter some text: ");
62 fgets(buffer, BUFSIZ, stdin);
63 strncpy(shared_stuff->some_text, buffer, TEXT_SZ);
64 shared_stuff->written_by_you = 1;
65 sem_post(mutex);
66 if (strcmp(buffer, "quit", 3) == 0) {
67 running = 0;
68 }
69 }
70 if (shmdt(shared_memory) == -1) {
71 fprintf(stderr, "shmdt failed\n");
72 exit(EXIT_FAILURE);
73 }
74 sem_close(mutex);
75 sem_unlink(SEM_NAME);
76 shmctl(shmid, IPC_RMID, 0);
77 exit(0);
78 }

```

Output:

```

004893625@csusb.edu@jb359-31:~
File Edit View Search Terminal Help
004893625@csusb.edu@jb359-31 ~]$ ls
SE201 CSE460 Diagram1.dia.autosave Music
SE202 CSE512 Documents perl5
SE320 CSE516 Downloads Pictu
SE330 CSE570 java.log.4598 Public
SE455 Desktop matlab_crash_dump.4598-1 salemv
004893625@csusb.edu@jb359-31 ~]$ cd CSE460/
004893625@csusb.edu@jb359-31 CSE460]$ ls
w1 HW2 HW3 Lab1 Lab2 Lab3 Lab4 Lab5 Lab6
004893625@csusb.edu@jb359-31 CSE460]$ cd Lab7
004893625@csusb.edu@jb359-31 Lab7]$ ls
odclient.cpp proc.c shared1 shared2
odserver.cpp ps.c shared1.cpp shared2.cpp
004893625@csusb.edu@jb359-31 Lab7]$ ./shared1
Memory attached at C4799000
You wrote: Hello
You wrote: This is nice

```

```

004893625@csusb.edu@jb359-31:~/CSE460/Lab7
File Edit View Search Terminal Help
[004893625@csusb.edu@jb359-31 Lab7]$ ./shared2
Memory attached at CF0E3000
Enter some text: Hello
waiting for client...
Enter some text: This is nice
waiting for client...
Enter some text: 

```


3. XV6 System Calls

```
.          1 1 512
..         1 1 512
README    2 2 2290
cat        2 3 13700
echo       2 4 12708
forktest   Process ls with pid 3 running
2 5 8152
grep       2 6 15576
init       2 7 13296
kill       2 8 12760
ln         2 9 12664
Process ls with pid 3 running
ls         2 10 14848
mkdir      2 11 12840
rm         2 12 12824
sh         2 13 23308
stressfs   2 14 13488
usertests  2 Process ls with pid 3 running
15 56424
wc         2 16 14240
foo        2 17 13428
zombie     2 18 12488
console    3 19 0
Process sh with pid 2 running
$ ps
Process sh with pid 2 running
Process sh with pid 2 running
Process sh with pid 4 running
exec: fail
exec ps failed
Process sh with pid 2 running
$ quit
Process sh with pid 2 running
Process sh with pid 5 running
exec: fail
exec quit failed
Process sh with pid 2 running
+ 
```

Everything i completed for this assignment therefore the fulll score we expect is a **20/20 on this lab assignment.**