Anthony Thap & Brian Ackley

4 February 2014

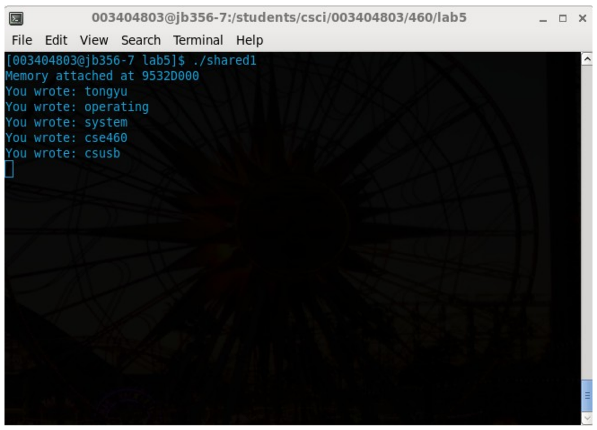
CSE460 Operating Systems

Tong Yu

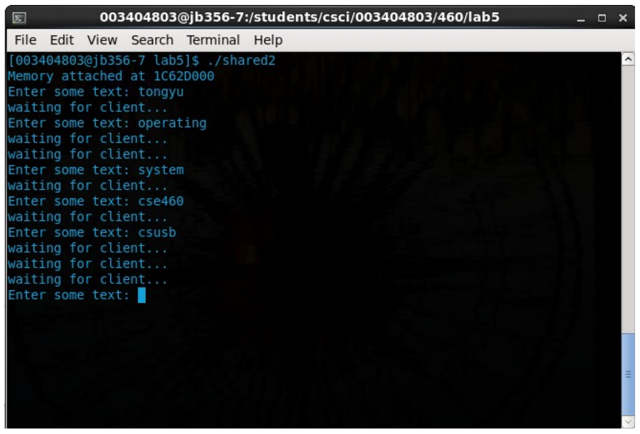
LAB #5

- Shared 1 & Shared 2, we saw when we ran both programs that once we responded as clients, the program would tell us our response that we sent and then waits again for another input.

```
/*  After the headers the shared memory segment
 (the size of our shared memory structure) is created with a call to shmget,
 with the IPC_CREAT bit specified. It reads data from the shared memory. */

#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>

#define TEXT_SZ 2048

struct shared_use_st {
    int written_by_you;
    char some_text[TEXT_SZ];
};

int main()
{
    int running = 1;
    void *shared_memory = (void *)0;
    struct shared_use_st *shared_stuff;
    int shmid;

    srand((unsigned int)getpid());

    shmid = shmget((key_t)1234, sizeof(struct shared_use_st), 0666 | IPC_CREAT);

    if (shmid == -1) {
        fprintf(stderr, "shmget failed\n");
        exit(EXIT_FAILURE);
    }

/* We now make the shared memory accessible to the program. */

    shared_memory = shmat(shmid, (void *)0, 0);
    if (shared_memory == (void *)-1) {
        fprintf(stderr, "shmat failed\n");
        exit(EXIT_FAILURE);
    }

    printf("Memory attached at %X\n", (long)shared_memory);

/* The next portion of the program assigns the shared_memory segment to shared_stuff,
 which then prints out any text in written_by_you. The loop continues until end is found
 in written_by_you. The call to sleep forces the consumer to sit in its critical section,
 which makes the producer wait. */

    shared_stuff = (struct shared_use_st *)shared_memory;
    shared_stuff->written_by_you = 0;
    while(running) {
        if (shared_stuff->written_by_you) {
            printf("You wrote: %s", shared_stuff->some_text);
            sleep( rand() % 4 ); /* make the other process wait for us ! */
            shared_stuff->written_by_you = 0;
            if (strncmp(shared_stuff->some_text, "end", 3) == 0) {
                running = 0;
            }
        }
    }

/* Lastly, the shared memory is detached and then deleted. */

    if (shmdt(shared_memory) == -1) {
        fprintf(stderr, "shmdt failed\n");
        exit(EXIT_FAILURE);
    }

    if (shmctl(shmid, IPC_RMID, 0) == -1) {
        fprintf(stderr, "shmctl(IPC_RMID) failed\n");
        exit(EXIT_FAILURE);
    }

    exit(EXIT_SUCCESS);
}


/*


  shared2.cpp: Similar to shared1.cpp except that it writes data to
  the shared memory.
*/
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>

#define TEXT_SZ 2048

struct shared_use_st {
    int written_by_you;
    char some_text[TEXT_SZ];
};

int main()
{
    int running = 1;
    void *shared_memory = (void *)0;
    struct shared_use_st *shared_stuff;
    char buffer[BUFSIZ];
    int shmid;

    shmid = shmget((key_t)1234, sizeof(struct shared_use_st), 0666 | IPC_CREAT);

    if (shmid == -1) {
        fprintf(stderr, "shmget failed\n");
        exit(EXIT_FAILURE);
    }

    shared_memory = shmat(shmid, (void *)0, 0);
    if (shared_memory == (void *)-1) {
        fprintf(stderr, "shmat failed\n");
        exit(EXIT_FAILURE);
    }

    printf("Memory attached at %X\n", (long)shared_memory);

    shared_stuff = (struct shared_use_st *)shared_memory;
    while(running) {
        while(shared_stuff->written_by_you == 1) {
            sleep(1);
            printf("waiting for client...\n");
        }
        printf("Enter some text: ");
        fgets(buffer, BUFSIZ, stdin);

        strncpy(shared_stuff->some_text, buffer, TEXT_SZ);
        shared_stuff->written_by_you = 1;

        if (strncmp(buffer, "end", 3) == 0) {
                running = 0;
        }
    }

    if (shmdt(shared_memory) == -1) {
        fprintf(stderr, "shmdt failed\n");
        exit(EXIT_FAILURE);
    }
    exit(EXIT_SUCCESS);
}
```

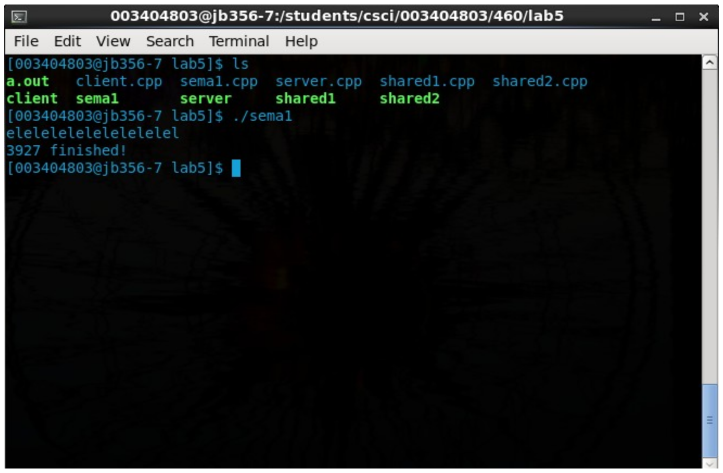

003404803@jb356-7:/students/csci/003404803/460/lab5
File Edit View Search Terminal Help

```
[003404803@jb356-7 lab5]$ ./shared1
Memory attached at 9532D000
You wrote: tongyu
You wrote: operating
You wrote: system
You wrote: cse460
You wrote: csusb
```


003404803@jb356-7:/students/csci/003404803/460/lab5
File Edit View Search Terminal Help

```
[003404803@jb356-7 lab5]$ ./shared2
Memory attached at 1C62D000
Enter some text: tongyu
waiting for client...
Enter some text: operating
waiting for client...
waiting for client...
Enter some text: system
waiting for client...
Enter some text: cse460
waiting for client...
Enter some text: csusb
waiting for client...
waiting for client...
waiting for client...
Enter some text: 
```

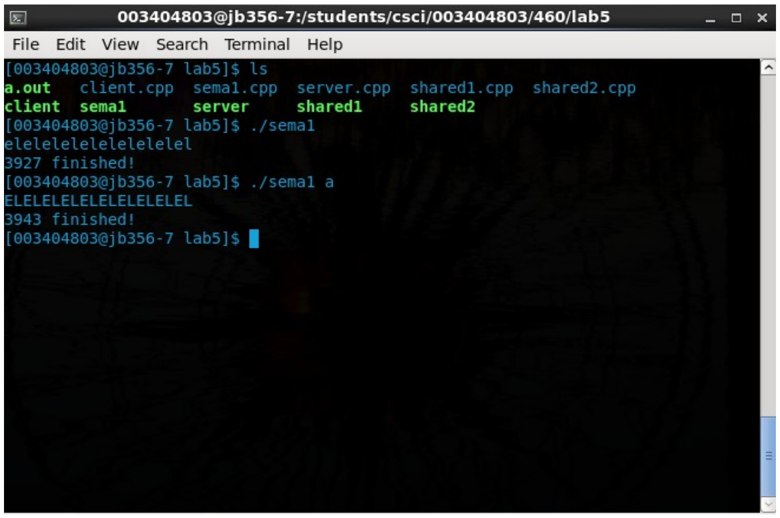
- Semaphores – It prints out e & l because it is entering the process in this case, it is only one process.



003404803@jb356-7:/students/csci/003404803/460/lab5
File Edit View Search Terminal Help

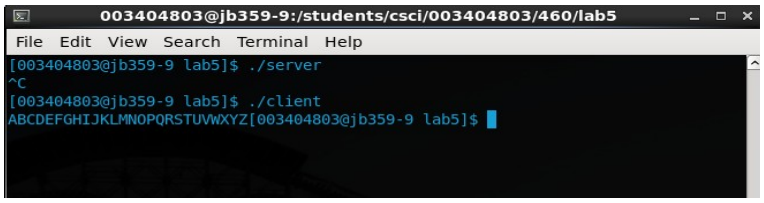
```
[003404803@jb356-7 lab5]$ ls
a.out   client.cpp  sema1.cpp  server.cpp  shared1.cpp  shared2.cpp
client  sema1       server     shared1     shared2
[003404803@jb356-7 lab5]$ ./sema1
elelelelelelelelel
3927 finished!
[003404803@jb356-7 lab5]$
```

- Semaphores – The letters E & L get capitalized because the “a” meets the argument for being greater than 1.



003404803@jb356-7:/students/csci/003404803/460/lab5
File Edit View Search Terminal Help

```
[003404803@jb356-7 lab5]$ ls
a.out   client.cpp  sema1.cpp  server.cpp  shared1.cpp  shared2.cpp
client  sema1       server     shared1     shared2
[003404803@jb356-7 lab5]$ ./sema1
elelelelelelelelel
3927 finished!
[003404803@jb356-7 lab5]$ ./sema1 a
ELELELELELELELELEL
3943 finished!
[003404803@jb356-7 lab5]$
```



003404803@jb359-9:/students/csci/003404803/460/lab5
File Edit View Search Terminal Help

```
[003404803@jb359-9 lab5]$ ./server
^C
[003404803@jb359-9 lab5]$ ./client
ABCDEFGHIJKLMNOPQRSTUVWXYZ[003404803@jb359-9 lab5]$
```

- We put all the assigned tasks in this page, we will be giving ourselves full credit.