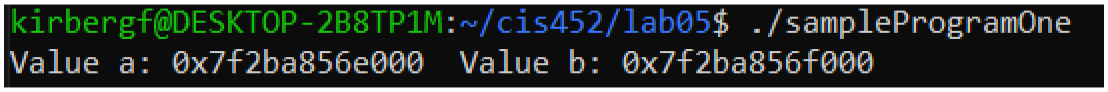
Sebastian Torrejon Alonzo
Fabian Kirberg

Lab 5

1. What is being output by sampleProgramOne (i.e., what is the meaning of the output values)?



```
kirbergf@DESKTOP-2B8TP1M:~/cis452/lab05$ ./sampleProgramOne
Value a: 0x7f2ba856e000  Value b: 0x7f2ba856f000
```

These values represent memory addresses. Value ‘a’ points to the start of the shared memory segment, and Value ‘b’ points to the end of the shared memory segment.

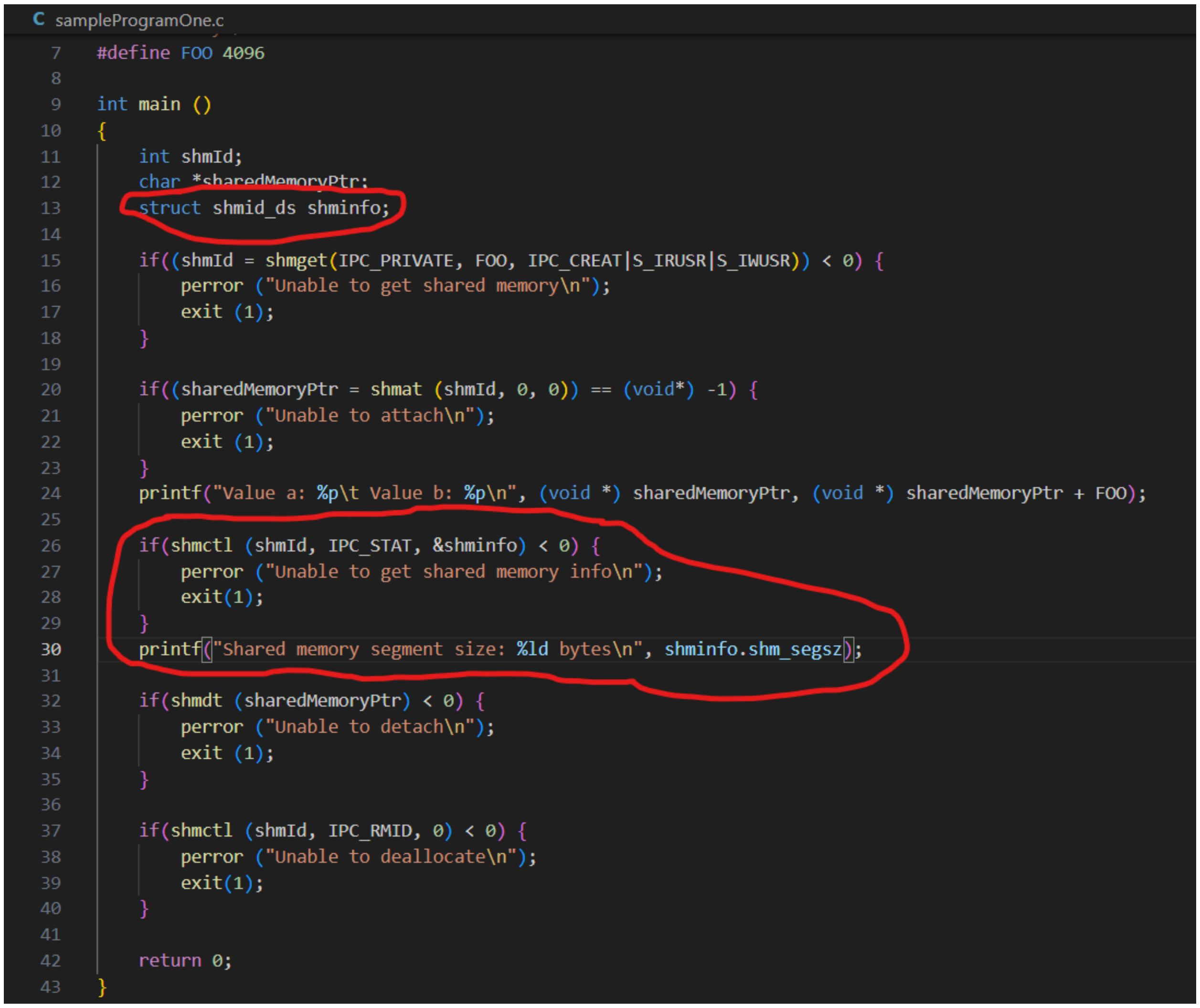
2. Read the man pages; then describe the meaning/purpose of each argument used by the shmget() function call.
IPC_PRIVATE: Create a new shared memory segment with a unique identifier.
FOO: Specifies the size of the shared memory segment in bytes. In this case, it is 4096 bytes.
IPC_CREAT | S_IRUSR | S_IWUSR: The flags that are used to specify the permissions for the shared memory segment. IPC_CREAT is used to create a new segment if it doesn't exist, and S_IRUSR | S_IWUSR sets the read and write permissions for the owner of the segment.

3. Describe two specific uses of the shmctl() function call.

1. shmctl() can be used with IPC_RMID to deallocate the memory of the shared memory segments, this is done in sampleProgramOne.
2. shmctl() can be used with the IPC_STAT command to retrieve information about a shared memory segment into a struct shmid_ds structure.

Sebastian Torrejon Alonzo
Fabian Kirberg

4. Read the man pages, then use shmctl() to modify sampleProgramOne so that it prints out the size of the shared memory segment. What changes/lines do you have to add to the program?



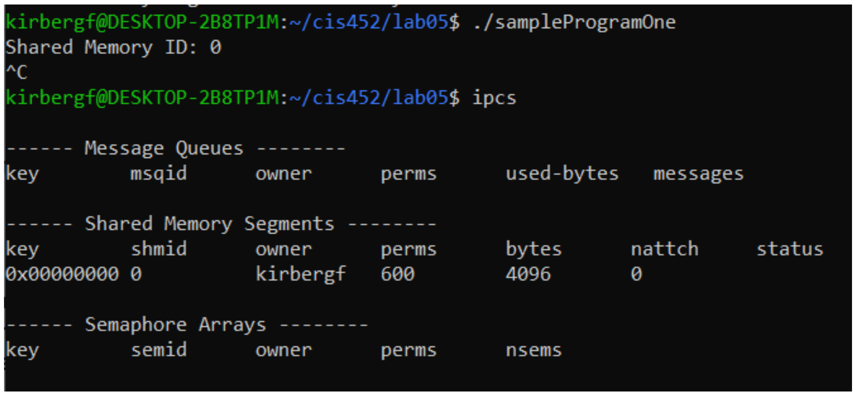
sampleProgramOne.c

```c
#define FOO 4096

int main ()
{
    int shmId;
    char *sharedMemoryPtr;
    struct shmid_ds shminfo;

    if((shmId = shmget(IPC_PRIVATE, FOO, IPC_CREAT|S_IRUSR|S_IWUSR)) < 0) {
        perror ("Unable to get shared memory\n");
        exit (1);
    }

    if((sharedMemoryPtr = shmat (shmId, 0, 0)) == (void*) -1) {
        perror ("Unable to attach\n");
        exit (1);
    }
    printf("Value a: %p\t Value b: %p\n", (void *) sharedMemoryPtr, (void *) sharedMemoryPtr + FOO);

    if(shmctl (shmId, IPC_STAT, &shminfo) < 0) {
        perror ("Unable to get shared memory info\n");
        exit(1);
    }
    printf("Shared memory segment size: %ld bytes\n", shminfo.shm_segsz);

    if(shmdt (sharedMemoryPtr) < 0) {
        perror ("Unable to detach\n");
        exit (1);
    }

    if(shmctl (shmId, IPC_RMID, 0) < 0) {
        perror ("Unable to deallocate\n");
        exit(1);
    }

    return 0;
}
```

We first declare a struct shmid_ds variable called shminfo to store information about the shared memory segment. Then, we use shmctl() with the IPC_STAT command to fill this structure with information about the segment. Finally, we print out the size of the shared memory segment using shminfo.shm_segsz.

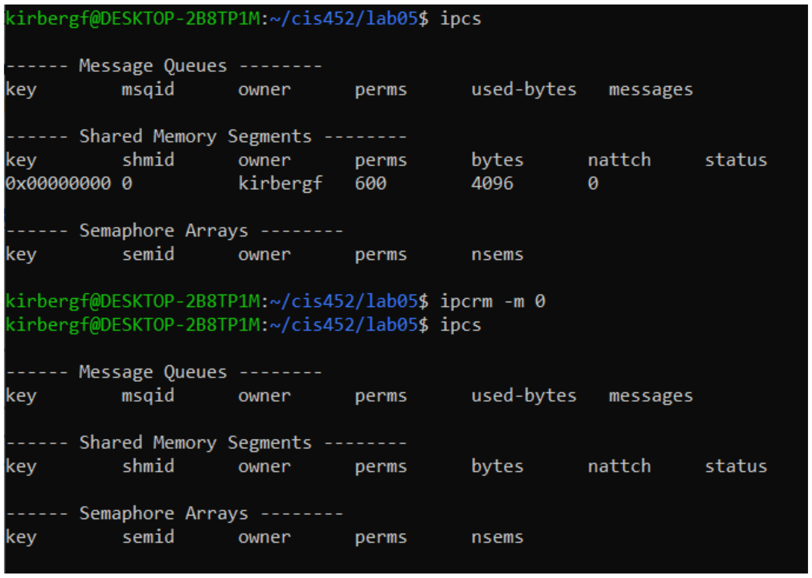
Sebastian Torrejon Alonzo
Fabian Kirberg

Perform the following operations:
• Modify the print statement in sampleProgramOne to determine the ID of the shared memory segment
• Insert a pause() after the print statement, recompile and run
• Terminate the Sample Program (^C) and run the ipcs utility
• Take a screenshot



```
kirbergf@DESKTOP-2B8TP1M:~/cis452/lab05$ ./sampleProgramOne
Shared Memory ID: 0
^C
kirbergf@DESKTOP-2B8TP1M:~/cis452/lab05$ ipcs

------ Message Queues --------
key        msqid      owner      perms      used-bytes   messages

------ Shared Memory Segments --------
key        shmid      owner      perms      bytes      nattch     status
0x00000000 0          kirbergf   600        4096       0

------ Semaphore Arrays --------
key        semid      owner      perms      nsems
```

• Use the ipcrm utility to remove the shared memory segment
• Re-run the ipcs utility to verify that it worked
• Take a screenshot



```
kirbergf@DESKTOP-2B8TP1M:~/cis452/lab05$ ipcs

------ Message Queues --------
key        msqid      owner      perms      used-bytes   messages

------ Shared Memory Segments --------
key        shmid      owner      perms      bytes      nattch     status
0x00000000 0          kirbergf   600        4096       0

------ Semaphore Arrays --------
key        semid      owner      perms      nsems

kirbergf@DESKTOP-2B8TP1M:~/cis452/lab05$ ipcrm -m 0
kirbergf@DESKTOP-2B8TP1M:~/cis452/lab05$ ipcs

------ Message Queues --------
key        msqid      owner      perms      used-bytes   messages

------ Shared Memory Segments --------
key        shmid      owner      perms      bytes      nattch     status

------ Semaphore Arrays --------
key        semid      owner      perms      nsems
```