CIS 452 02 – Lab 5 – Gabrielle Munson and Gabe Baksa

Sample Program 1

1. **What is being output by sampleProgramOne (i.e., what is the meaning of the output values) ?**
   The first value printed is the starting address of the shared memory segment, and the second value printed is one byte past the end address of the shared memory segment since it is the starting address + FOO.
2. **Read the man pages; then describe the meaning/purpose of each argument used by the shmget() function call.**
   According to the man pages, the syntax for shmget() is:
   int shmget(key_t key, size_t size, int shmflg);
   The first argument, key, is used to specify the shared memory segment.
   The second argument, size, is used to specify the size of the shared memory segment in what seems to be bytes.
   The third argument, shmflg, contains flags that control the behaviors and permissions of the shared memory segment.
3. **Describe two specific uses of the shmctl() function call.**
   One specific use of the shmctl() function call is using IPC_SET to allow you to modify some members of the segment. This would be used if a new process needed some kind of permission to the shared memory segment that it doesn't currently have. Another use of the function call would be using IPC_STAT which copies the information from the kernel data structure. This would be useful for gathering information about the segment to see who has certain permissions and such.
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?**
   On line 14 we added a struct to hold the shared memory segment information that is returned from shmctl(). `struct shmid_ds shmInfo;`
   On line 27 we added an if statement to ensure the shmctl() function call accessed the shared memory information properly. We enter the id of the segment, what command we would like to run, followed by the struct the information should go into.

```
if (shmctl(shmId, IPC_STAT, &shmInfo) < 0)
{
    perror("Unable to retrieve shared memory info\n");
    exit(1);
}
```

   On line 33 we then added a print statement to print out the size of the segment from the shm_sedsz field of the shmInfo struct.

```
printf("Size of the shared memory segment: %zu bytes\n", shmInfo.shm_segsz);
```

IPCS Screenshots

After pause() is inserted, program is ran and compiled, Ctrl+C is pressed, and ipcs is ran:

```
gmunson@gabbis-laptop:/mnt/c/Users/munso/OneDrive/Desktop/OS/cis452/labs/lab5$ ipcs

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

------ Shared Memory Segments --------
key        shmid      owner      perms      bytes      nattch     status
0x00000000 7          gmunson    600        4096       0

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

After ipcrm (ipcrm -m 7) is ran followed by ipcs:

```
gmunson@gabbis-laptop:/mnt/c/Users/munso/OneDrive/Desktop/OS/cis452/labs/lab5$ 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
```

Programming Assignment Screenshots

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  COMMENTS

Enter a string: again
Enter a string: check
Enter a string: check
Enter a string: check
Enter a string: it seems to be working fine
Enter a string: ok
Enter a string: checking output
Enter a string: again
Enter a string: again
Enter a string: again
Enter a string: check
Enter a string: again
Enter a string: double checking this message
Enter a string: quit

Writer shutting down gracefully...
root@gb_laptop:/mnt/c/Users/gbaks/Downloads/cis452/labs/lab5#
```

```
Reader received message: checking output
Reader 2 is now ready.
Reader received message: again
Reader 1 is now ready.
Reader received message: again
Reader 1 is now ready.
Reader received message: again
Reader 1 is now ready.
Reader received message: check
Reader 1 is now ready.
Reader received message: again
Reader 1 is now ready.
Reader received message: double checking this message
Reader 2 is now ready.

Reader shutting down gracefully...
root@gb_laptop:/mnt/c/Users/gbaks/Downloads/cis452/labs/lab5#
```

```
Reader received message: checking output
Reader 1 is now ready.
Reader received message: again
Reader 1 is now ready.
Reader received message: again
Reader 2 is now ready.
Reader received message: again
Reader 2 is now ready.
Reader received message: check
Reader 1 is now ready.
Reader received message: again
Reader 2 is now ready.
Reader received message: double checking this message
Reader 1 is now ready.

Reader shutting down gracefully...
root@gb_laptop:/mnt/c/Users/gbaks/Downloads/cis452/labs/lab5#
```