CIS–4020 Lab
Shared Memory

Last Revised: October 24, 2014

1 Introduction

In this lab you will experiment with the System V shared memory API. Unlike previous labs there is no kernel programming in this lab; all programing is entirely in user mode. The purpose of this lab is to expose you to System V IPC (interprocess communication), and to give you some experience with the concept of shared memory. The kernel implements shared memory by mapping the same physical page frames into the virtual address spaces of the sharing processes. You will explore some of the implications of this approach in this lab.

2 Shared Memory API

The System V API is a little unusual in a couple of respects. First, the objects you create using it (shared memory segments in our case) persist until they are explicitly destroyed or until the system is rebooted. Even if all processes that are using a shared memory segment terminate, the shared memory segment will continue to exist. A future process could still attach to it and use it, assuming it had permission to do so.

Shared memory segments are identified with a "key." This key is usually generated from `ftok` using a mutually agreed upon file name as a base. The file name must refer to an existing, accessible file and would typically be a file of significance to the system using the shared memory. In addition `ftok` uses a project identifier that allows different software systems to create distinct shared memory segments off the same file name. In this lab use the file name corresponding to this document and a project identifier of 1.

With the key in hand you can use `shmget` to create a shared memory segment based on that key. This function returns a segment identifier that you can use in the later calls. The key is only needed for getting the segment identifier.

Be aware that when you create a shared memory segment using the `IPC_CREAT` flag, you should also provide appropriate permissions to apply to the new segment. For example `0666` permissions (the leading zero is necessary to indicate that 666 is in octal) allow all users to read and write the segment. If you forget to specifiy permissions, no permissions will be applied and you will end up with a segment you can't manipulate. Furthermore the `ipcs` tool, mentioned below, will not display any segments for which you lack read access. Thus a segment with no permissions will be invisible in `ipcs`.

To attach to the shared memory segment use `shmat`. This function returns a pointer to the segment. You may now use the segment like any other memory region with the understanding that some other process may also be using it at the same time. It is permitted for multiple processes to attach to the same segment (indeed, that is why it is a *shared* memory segment).

When you no longer need the segment you should call `shmdt` to detach from it. The segment is not removed

even if all processes detach from it unless an appropriate flag has been given to the `shmctl` function.

Please review the manual pages for all the functions mentioned above to familiarize yourself with their operation.

3 Programs

For this lab you are to write two programs. The first program should create a shared memory segment (the exact size is not important), copy the string "Hello, World" into the segment and then pause. The second program should attach to the segment, display the string that it finds in the segment (as left by the first program), detach from the segment and then terminate. When the first program resumes, it should also detach from the segment. Do the following:

1. Run the two programs above and verify that they behave as expected.
2. Modify the programs so that they print the address of the shared memory segment (use the %p format specifier with printf). Take note of the addresses the programs see.
3. What happens if you run the first program to completion and then run the second program after the first one has terminated?

Use the `ipcs` command to view the shared memory segments and `ipcrm` to remove the left over segment created by your test programs. What happens when you run the program that uses the shared memory segment if the segment does not exist?

4 Report

Write a report for this lab following the lab report template provided by your instructor. Include the significant parts of your code, some comments about how it works, and what you observed.