SIENA*college*
Computer Science

Computer Science 330
Operating Systems
Siena College
Fall 2022

Lab 5: System V Shared Memory

Due: 11:59 PM, Wednesday, November 2, 2022

In this lab, you will see how to use System V shared memory with Unix processes.

You may work alone or with a partner or two on this lab.

Learning goals:

1. To learn about System V shared memory
2. To use System V shared memory to implement communication between Unix processes
3. To gain experience using pointers

Getting Set Up

In Canvas, you will find link to follow to set up your GitHub repository, which will be named `shmem-lab-yourgitname`, for this lab. Only one member of the group should follow the link to set up the repository on GitHub, then others should request a link to be granted write access.

`fork()` Practice

The *Syracuse Sequence* starting at a given positive integer n is defined by repeated application of the function

$$f(n) = \begin{cases} \frac{n}{2} & \text{if } n \text{ is even} \\ 3n+1 & \text{if } n \text{ is odd} \end{cases}$$

The sequence ends when the value reaches 1. It is not known if all values of n eventually lead to 1, but no one has found a value for which it doesn't or proven that one must exist.

Practice Program: Write a program `forksyr.c` that computes and prints the Syracuse sequence within a child process created by `fork`. Include a `Makefile` that compiles this program into an executable called `forksyr`. (10 points)

Notes:

- Do some error checking. If your program is run without a command-line argument, print a "Usage" line. If your program is given a string that cannot be successfully converted to an integer, or if the conversion results in a negative starting value for the sequence, print an appropriate error message.

- If you store the values in the sequence with `int` values, you will notice they could overflow the storage capabilities of the data type. You can delay this somewhat by using `long` values. Even then, you could overflow the values with certain starting values. In this case, also print an error message.

Adding Shared Memory

Practice Program: Now, create a second version of your program that uses System V shared memory. The child process will be responsible for computing the entire sequence, but the parent process will be responsible for printing it. Call your program `forksyrshared.c` and include a `Makefile` that compiles this program into `forksyrshared`. You may assume that the sequence is no longer than 10,000 numbers and use that to determine the size of the shared memory segment you allocate. (25 points)

Once your program is working, add a call to `sleep(2)` (recall that "2" here refers to the manual section, not the argument to pass – you'll want to pass a larger number of seconds) to your program before you detach and free the shared memory segment. In a separate window, use the `ipcs` command to see that your shared memory segment is listed, and that it goes away when your program terminates. Save this output and include it in a file `ipcs.txt` to be included in your submission (5 points).

Submission

Commit and push!

Grading

This assignment will be graded out of 40 points.

Feature	Value	Score
`forksyr.c`	10	
`forksyrshared.c`	25	
`ipcs.txt`	5	
Total	40	