The Department of Electrical and Computer Engineering

The University of Alabama in Huntsville

CPE 435 Lab-04

IPC Using Shared Memory

Introduction

Exercise

Write the code for two distinct C processes that are not part of a parent-child relation. They both attach to shared memory area containing a new type structure with four data types (first number, second-number, Sum and Flag). The first process must set the values for first and second numbers, then it updates the value of flag to 1 (which means there is new data). The second process is to find the sum of these two numbers and display the sum of the two values and changes the flag to zero. Two processes continue running until the first process sets the value of flag to -1.

Hint

Define the shared structure like:

```
struct info {
float value1, value2;
float sum;
int flag;
};
# define KEY ((key_t)(1234))
# define SEGSIZE sizeof (struct info)
```

Topics for theory:

shmget(): *int shmget(key_t key, size_t size, int shmflg);*

It returns the identifier of the shared memory segment associated with the value of *key*. Based on a value of *shmflg*, either a new segment is created or identifier of already created segment

shmat(): *void *shmat(int shmid, const void *shmaddr, int shmflg);*

It attaches the shared memory segment identified by *shmid* and returns the pointer in the calling process address space.

shmctl(): *int shmctl(int shmid, int cmd, struct shmid_ds *buf);*

It performs the control operation in the shared memory segment identified by *shmid*. The operation is as defined by value *shmid*

shmdt(): *int shmdt(const void *shmaddr);*

This function detaches the shared memory segment identified by *shmaddr*

Please visit Study_Lab05andLab04.pptx file page 1-10 for demo examples.

Deliverables

Lab Report

The following material in each section is expected:

1. Cover page with your name, lab number, course name, and dates
2. Theory/Background (Material or methods relevant to the lab, a few sentences on each)
    a. shmget()
    b. shmat()
    c. shmctl()
    d. shmdt()
3. Observations (Show output demonstrating the two processes can use the shared memory correctly.)
    a. Process 1 can set value1, value2, and flag
        i. Setting flag to -1 by process 1 should cause both processes to terminate
    b. The second process sets sum to the sum of value1 and value2. It should display the result.
4. Conclusion (Did your program work as expected, what can you take away from the lab?)
5. Appendix (for source code, submit the text in a table)

The report should be submitted as a single pdf document with the source code for your program within it.

Recorded Demonstration

The following material in each section is expected:

1. Introduce yourself and give the name of the lab
2. Show and discuss how your program works
3. Compile the program
4. Show that the two processes interact as required
5. Recordings should be around 5 to 7 minutes on average

The recorded demonstration should be submitted as an mp4 file.