The Department of Electrical and Computer Engineering

The University of Alabama in Huntsville

CPE 435 Lab-04

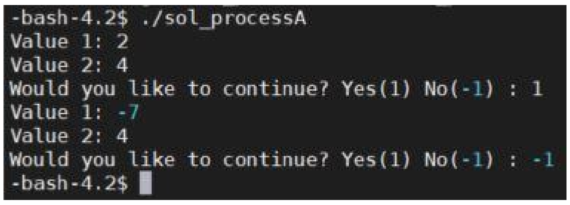
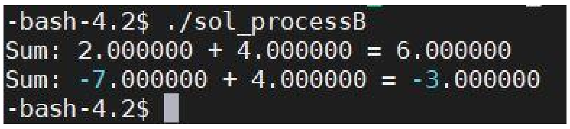
IPC Using Shared Memory

Introduction

Exercise

Write the code for two distinct C processes that are not part of a parent-child relation. These will be separate programs from separate source files. They will both attach to a shared memory region 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. It should then change the flag to zero (which means the addition was completed). The two processes will continue running this method of exchange until the first process sets the value of flag to -1 by user input.

Example Output

First Process	Second Process
	

```
-bash-4.2$ ./sol_processA
Value 1: 2
Value 2: 4
Would you like to continue? Yes(1) No(-1) : 1
Value 1: -7
Value 2: 4
Would you like to continue? Yes(1) No(-1) : -1
-bash-4.2$
```

```
-bash-4.2$ ./sol_processB
Sum: 2.000000 + 4.000000 = 6.000000
Sum: -7.000000 + 4.000000 = -3.000000
-bash-4.2$
```

Note that the above sample programs will continue running until the user enters a -1 value for the flag. This means that both programs will loop until the flag is set to that value. Otherwise the flag will be used to help synchronize access to the shared memory.

Hint

Define the shared structure as follows:

```
struct info {
 float value1, value2;
 float sum;
 int flag;
};
#define KEY ((key_t)(1234))
#define  MSIZ  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 IPC_SharedMemory.pptx file page 7-12 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 and will only terminate upon setting the flag to -1.)
    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.

Demonstration

Demonstration may either be in person in the lab or submitted alongside the report as an mp4 file (a recording). The following material in each section is expected:

1. Introduce yourself and give the name of the lab
2. Compile the necessary programs
3. Run and show that the programs behave as specified, discussing as necessary the behavior of the programs observed