

# Operating Systems Lab

CPE 435-01

## Lab 04: Shared Process Memory

By: David Thornton

Lab Date: 01 February 2021

Lab Due: 08 February 2021

Demonstration Due: 08 February 2021

# Introduction

The purpose of this lab is to give students an introduction to how processes share memory.

## Theory

### Topic 1: shmget()

- "... returns the identifier of the System V shared memory segment associated with the value of the argument key. It may be used either to obtain the identifier of a previously created shared memory segment (when shmflg is zero and key does not have the value IPC\_PRIVATE), or to create a new set." - [Source](#)

### Topic 2: shmat()

- "... accepts a shared memory ID, shm\_id, and attaches the indicated shared memory to the program's address space. The returned value is a pointer of type (void \*) to the attached shared memory. Thus, casting is usually necessary. If this call is unsuccessful, the return value is -1. Normally, the second parameter is NULL. If the flag is SHM\_RDONLY, this shared memory is attached as a read-only memory; otherwise, it is readable and writable." - [Source](#)

### Topic 3: shmctl()

- "... performs the control operation specified by cmd on the System V shared memory segment whose identifier is given in shmid." - [Source](#)

### Topic 4: shmdt()

- "... detaches the shared memory segment located at the address specified by shmaddr. from the address space of the calling process." - [Source](#)

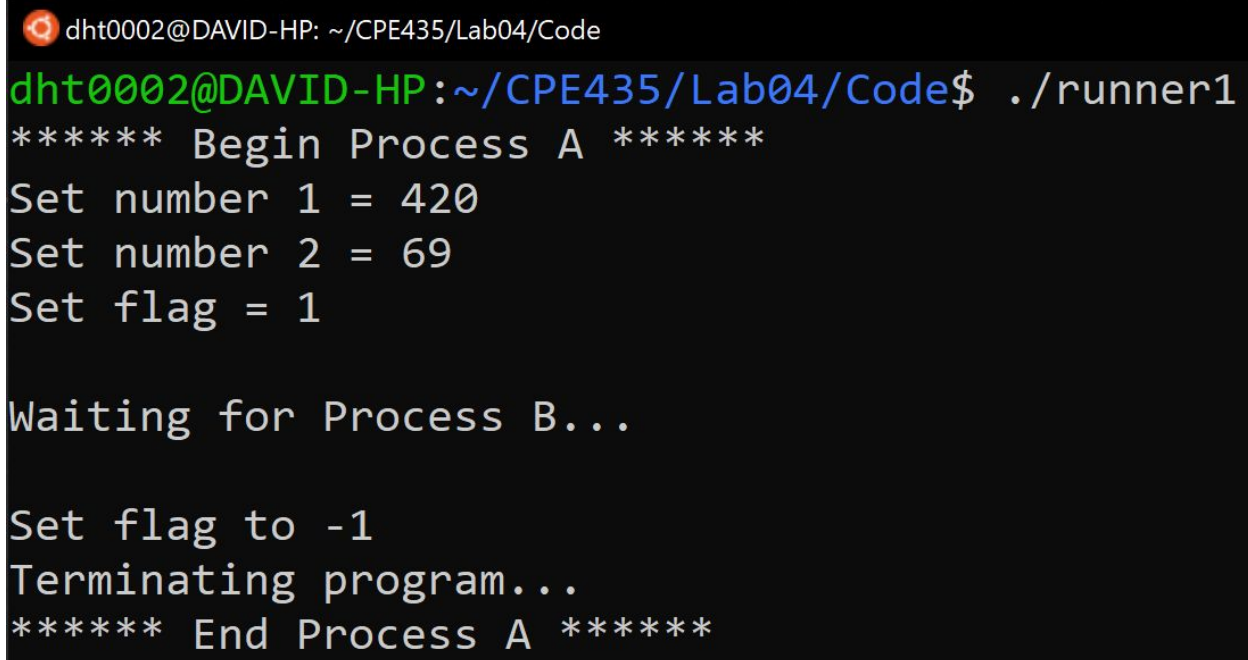
## Lab Assignment

Write the code for two distinct C processes that are not part of a parent-child relation. They both attach to a 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 change the flag to zero. Two processes continue running until the first process sets the value of flag to -1.

## Observations

All code performs as expected. Additional messages and error handling were implemented too.

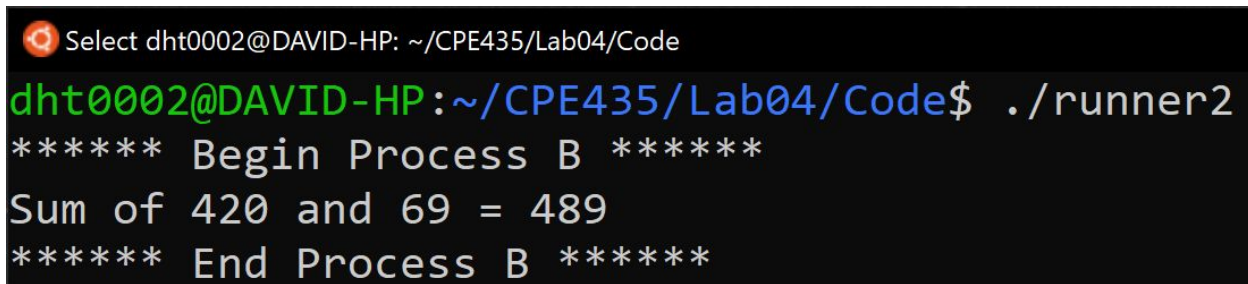


```
dht0002@DAVID-HP: ~/CPE435/Lab04/Code
dht0002@DAVID-HP:~/CPE435/Lab04/Code$ ./runner1
***** Begin Process A *****
Set number 1 = 420
Set number 2 = 69
Set flag = 1

Waiting for Process B...

Set flag to -1
Terminating program...
***** End Process A *****
```

Figure 1: Normal execution of processA.c



```
Select dht0002@DAVID-HP: ~/CPE435/Lab04/Code
dht0002@DAVID-HP:~/CPE435/Lab04/Code$ ./runner2
***** Begin Process B *****
Sum of 420 and 69 = 489
***** End Process B *****
```

Figure 2: Normal execution of processB.c

## Conclusion

This lab was successful in introducing me to the concept of shared process memory.

[Demo link](#)

# Appendix

## Appendix 1: processA.c

```
// *****  
// Program Title: Lab 04  
// Project File: processA.c  
// Name: David Thornton  
// Course Section: CPE-435, SP 2021  
// Due Date: 02/08/2021  
// *****  
  
#include <stdio.h>  
#include <stdlib.h>  
#include <sys/types.h>  
#include <sys/ipc.h>  
#include <sys/shm.h>  
#include "header.h"  
  
int main(void)  
{  
    int id;  
    struct info* ctrl;  
  
    if ((id = shmget(KEY, SEGSIZE, IPC_CREAT | 0666)) < 0)  
    {  
        printf("Error! Exiting program...\n");  
        exit(1);  
    }  
  
    ctrl = (struct info*) shmat(id, 0, 0);  
  
    if (ctrl <= (struct info*)(0))  
    {  
        printf("Error! Exiting program...\n");  
        exit(2);  
    }  
  
    ctrl->num1 = 420;  
    ctrl->num2 = 69;  
    ctrl->flag = 1;
```

```

printf("***** Begin Process A *****\n");
printf("Set Number 1 = %i\n", ctrl->num1);
printf("Set Number 2 = %i\n", ctrl->num2);
printf("Set Flag = %i\n\n", ctrl->flag);
fflush(stdout);

while(ctrl->flag != 0); // wait for flag to get set by process B

ctrl->flag = -1;
printf("Set flag to %i\n", ctrl->flag);
printf("Terminating program...\n");
fflush(stdout);
shmdt(ctrl);
shmctl(id, IPC_RMID, NULL);
exit(0);
}

```

#### Appendix 2: processB.c

```

// *****
// Program Title: Lab 04
// Project File: processB.c
// Name: David Thornton
// Course Section: CPE-435, SP 2021
// Due Date: 02/08/2021
// *****

#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include "header.h"

int main(void)
{
    int id;
    struct info* ctrl;

    if ((id = shmget(KEY, SEGSIZE, 0)) < 0)
    {
        printf("Error! Exiting program...\n");
    }
}

```

```

        exit(1);
    }

    ctrl = (struct info*) shmat(id, 0, 0);

    if (ctrl <= (struct info*)(0))
    {
        printf("Error! Exiting program...\n");
        exit(2);
    }

    while(ctrl->flag == 0);
    ctrl->sum = ctrl->num1 + ctrl->num2; // calculate and store sum
    printf("***** Begin Process B *****\n");
    printf("Sum of %i and %i = %i\n\n", ctrl->num1, ctrl->num2,
ctrl->sum);

    ctrl->flag = 0; // set the flag for process A
    while(ctrl->flag != -1); // wait for process A to set flag to -1
    shmdt(ctrl);
    exit(0);
}

```

### Appendix 3: header.h

```

// *****
// Program Title: Lab 04
// Project File: header.h
// Name: David Thornton
// Course Section: CPE-435, SP 2021
// Due Date: 02/08/2021
// *****

struct info
{
    int num1, num2, sum, flag;
};
# define KEY ((key_t)(1234))
# define SEGSIZE sizeof(struct info)

```