**Lab 7**

**Semaphores**

**Edgar Cobos**

**Problem Description**

Semaphores are best used to synchronize operations for multiple processes that access a critical section. They are also used to avoid process starvation and deadlocks. A semaphore with a positive value indicates that the specific resource deemed a critical section, is available. If the value is zero, then that resource is currently in use, so other processes that wish to use that specific critical section resource will need to wait until it becomes available. The system guarantees that the test and increment operation (P and V) on the semaphore are atomic. This lab focuses on the simple creation of a semaphore with the *IPC\_EXCL* condition, which forces the semaphore creation to fail if one already exists. We will also use system commands to view and remove semaphores. Additionally, we will use the *semctl* function to manipulate a semaphore’s data.

**Problems Encountered**

**Issue:** Running into this error when compiling the provided code:

semcontrol.c:8:7: error: redefinition of 'semun'

union semun {

^

/usr/include/sys/sem.h:176:7: note: previous definition is here

union semun {

^

**Solution:** As it turns out, it looks like the union is already defined by the system because we imported the <sys/sem.h> header. In this case, I will just comment out the definition from the code provided

ecobos:/Users/ecobos/git/OperatingSystems/lab7

[11:45:38]: sed -n '176,180p' /usr/include/sys/sem.h

union semun {

int val; /\* value for SETVAL \*/

struct semid\_ds \*buf; /\* buffer for IPC\_STAT & IPC\_SET \*/

unsigned short \*array; /\* array for GETALL & SETALL \*/

};

**Issue:** The provided instructions, “use system command *ipcrm sem id* to remove the semaphore set”, don’t provide much detail regarding how to actually use the command.

**Solution:** Looking up the *ipcrm* system command with *man,* I was able to figure out how to remove the semaphore. I used the following command:

>> ipcrm –s <semaphore ID>

The –s flag is needed to be able to specify the *semid* for the semaphore we wish to remove.

**C Code**

# include <stdio.h>

# include <sys/types.h>

# include <sys/ipc.h>

# include <sys/sem.h>

# include <unistd.h>

# include <stdlib.h>

# define NS 3

//time.h is included to be able to use the 'ctime' function

# include <time.h>

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Already defined in the system files (usr/include/sys/sem.h). Commenting out.

union semun {

int val;

struct semid\_ds \*buf;

ushort \*array;

};

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int main(void) {

int sem\_id, sem\_value, i;

key\_t ipc\_key; //used to setore the unique key for the semaphore

struct semid\_ds sem\_buf;

static ushort sem\_array[NS] = {3, 1, 4}; //define the semaphore set

union semun arg;

ipc\_key = ftok(".", 'S'); //generate the unique key for the semaphore

// Create semaphore and add IPC\_EXCL creation condition to the semflag parameter

// IPC\_EXCL forces semget to fail if the semaphore already exists

if ((sem\_id = semget(ipc\_key, NS, IPC\_CREAT | 0666 | IPC\_EXCL)) == -1) {

//should only execute below code if there is an issue with semget

perror ("semget: IPC | 0666");

exit(1);

}

printf ("Semaphore identifier %d\n", sem\_id);

/\* Set arg (the union) to the address of the storage location for \*/

/\* returned semid\_ds value \*/

arg.buf = &sem\_buf;

// Fetch the semaphore set's struct semid\_ds, storing it in the memory pointed to by arg.buf

if (semctl(sem\_id, 0, IPC\_STAT, arg) == -1) {

perror ("semctl: IPC\_STAT"); //should only execute if there is an issue with semctl

exit(2);

}

printf ("Create %s", ctime(&sem\_buf.sem\_ctime));

/\* Set arg (the union) to the address of the initializing vector \*/

arg.array = sem\_array;

//Set values of all semaphores in the set to the values in the array pointed to by arg.array

if (semctl(sem\_id, 0, SETALL, arg) == -1) {

perror("semctl: SETALL");

exit(3);

}

//Itereate through and return the value of semaphore at semnum i

for (i=0; i<NS; ++i) {

if ((sem\_value = semctl(sem\_id, i, GETVAL, 0)) == -1) {

perror("semctl : GETVAL");

exit(4);

}

printf ("Semaphore %d has value of %d\n",i, sem\_value);

}

/\*\*\* remove semaphore \*\*\*

if (semctl(sem\_id, 0, IPC\_RMID, 0) == -1) {

perror ("semctl: IPC\_RMID");

exit(5);

}

\*/

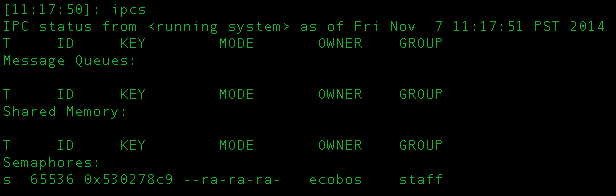
}

**Output**

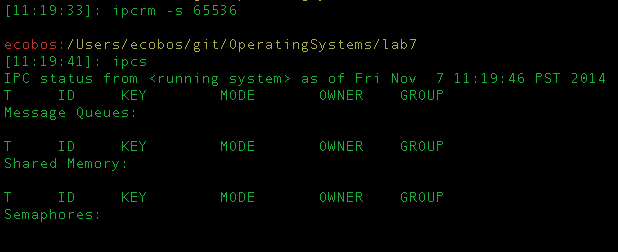
Running the *semcontrol* program for the first time with the semaphore remove code commented out, as specified.



Using the *ipcs* system command to list the existing sempahores



Using the *ipcrm* system command to remove the semaphore with *semid* 65536



Running the *semcontrol* program that uses the *IPC\_EXCL* condition with *semget*. Therefore, since the semaphore already exists, the *semget* command is forced to fail because of the condition specified.

