CECS 326 Assignment 3

Due: April 12, 2018

Cooperating processes need to communicate between them. Another way Linux supports interprocess communication is shared memory. A System V shared memory segment can be created and controlled using system calls **shmget**, **shmctl**, **shmat**, **shmdt**, and a C library function **memcpy**. Please consult the man pages of these system calls for details. This assignment is designed to illustrate a critical problem with processes executing concurrently that try to access shared data.

For this assignment you need to copy the following two C++ programs (named *shmp1.cpp* and *shmc1.cpp* with a header file *registration.h*) into your Linux directory, compile them into *shmp1*, and *shmc1*, respectively. Then run shmp1 and observe what happens. Run shmp1 at least 5 times and observe and report the results.

The program must run successfully on a Linux machine.

Do the following for this assignment:

1. Compile *shmp1.cpp* and *shmc1.cpp* into *shmp1* and *shmc1*, respectively, and run *shmp1*. Try to understand what the programs do
2. Annotate the programs with adequate amount of comments throughout the program to explain what the program does.
3. Run shmp1 at least 5 times and observe the results. Explain the problems you have observed in these runss on the cover page.
4. Submit on BeachBoard and a hard copy of the annotated *shmp1.cpp* and *shmc1.cpp*, with a cover page that provide your name, your student ID, course # and section, assignment #, due date, submission date, a description of these programs and any problem you observe.
5. The programs must be properly formatted and adequately commented to enhance readability and understanding.
6. Demonstrate your program on a Linux machine in the Lab and explain details of the program logic and execution results.

/\* Header file to be used with

\* shmp1.c and shmc1.c

\*/

struct CLASS {

char class\_number[6];

char date[7];

char title[50];

int seats\_left;

};

/\* shmp1.cpp \*/

#include "registration.h"

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/shm.h>

#include <sys/wait.h>

#include <unistd.h>

#include <stdlib.h>

#include <iostream>

#include <stdio.h>

#include <memory.h>

using namespace std;

CLASS myclass = { "1001", "120186", "Operating Systems", 15 };

#define NCHILD 3

int shm\_init( void \* );

void wait\_and\_wrap\_up( int [], void \*, int );

void rpterror( char \*, char \* );

main(int argc, char \*argv[])

{

int child[NCHILD], i, shmid, semid;

void \*shm\_ptr;

char ascshmid[10], ascsemid[10], pname[14];

strcpy (pname, argv[0]);

shmid = shm\_init(shm\_ptr);

sprintf (ascshmid, "%d", shmid);

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

child[i] = fork();

switch (child[i]) {

case -1:

rpterror ("fork failure", pname);

exit(1);

case 0:

sprintf (pname, "shmc%d", i+1);

execl("shmc1", pname, ascshmid, (char \*)0);

perror ("execl failed");

exit (2);

}

}

wait\_and\_wrap\_up (child, shm\_ptr, shmid);}

int shm\_init(void \*shm\_ptr)

{

int shmid;

shmid = shmget(ftok(".",'u'), sizeof(CLASS), 0600 | IPC\_CREAT);

if (shmid == -1) {

perror ("shmget failed");

exit(3);

}

shm\_ptr = shmat(shmid, (void \* ) 0, 0);

if (shm\_ptr == (void \*) -1) {

perror ("shmat failed");

exit(4);

}

memcpy (shm\_ptr, (void \*) &myclass, sizeof(CLASS) );

return (shmid);

}

void wait\_and\_wrap\_up(int child[], void \*shm\_ptr, int shmid)

{

int wait\_rtn, w, ch\_active = NCHILD;

while (ch\_active > 0) {

wait\_rtn = wait( (int \*)0 );

for (w = 0; w < NCHILD; w++)

if (child[w] == wait\_rtn) {

ch\_active--;

break;

}

}

cout << "Parent removing shm" << endl;

shmdt (shm\_ptr);

shmctl (shmid, IPC\_RMID, (struct shmid\_ds \*) 0);

exit (0);

}

void rpterror(char \*string, char \*pname)

{

char errline[50];

sprintf (errline, "%s %s", string, pname);

perror (errline);

}

/\* shmc1.cpp \*/

#include "registration.h"

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/sem.h>

#include <sys/shm.h>

#include <sys/wait.h>

#include <unistd.h>

#include <stdlib.h>

#include <iostream>

#include <stdio.h>

#include <memory.h>

using namespace std;

CLASS \*class\_ptr;

void \*memptr;

char \*pname;

int shmid, ret;

void rpterror(char \*), srand(), perror(), sleep();

void sell\_seats();

main(int argc, char\* argv[])

{

if (argc < 2) {

fprintf (stderr, "Usage:, %s shmid\n", argv[0]);

exit(1);

}

pname = argv[0];

sscanf (argv[1], "%d", &shmid);

memptr = shmat (shmid, (void \*)0, 0);

if (memptr == (char \*)-1 ) {

rpterror ("shmat failed");

exit(2);

}

class\_ptr = (struct CLASS \*)memptr;

sell\_seats();

ret = shmdt(memptr);

exit(0);

}

void sell\_seats()

{

int all\_out = 0;

srand ( (unsigned) getpid() );

while ( !all\_out) { /\* loop to sell all seats \*/

if (class\_ptr->seats\_left > 0) {

sleep ( (unsigned)rand()%5 + 1);

class\_ptr->seats\_left--;

sleep ( (unsigned)rand()%5 + 1);

cout << pname << " SOLD SEAT -- "

<< class\_ptr->seats\_left << " left" << endl;

}

else {

all\_out++;

cout << pname << " sees no seats left" << endl;

}

sleep ( (unsigned)rand()%10 + 1);

}

}

void rpterror(char\* string)

{

char errline[50];

sprintf (errline, "%s %s", string, pname);

perror (errline);

}