Michael Smith

CSE 460 – Operating Systems

Lab 6

2/13/18

Total points: 20

1. **Pthread Code:**

#include <pthread.h>

#include <stdio.h>

using namespace std;

void \*runner ( void \*data ){

char \*tname = ( char \* )data;

printf("I am %s\n", tname );

pthread\_exit ( 0 );

}

int main () {

pthread\_t id1, id2, id3;

pthread\_attr\_t attr1, attr2, attr3;

char \*tnames[3] = { "Thread 1", "Thread 2", "Thread 3" };

pthread\_attr\_init ( &attr1 );

pthread\_attr\_init ( &attr2 );

pthread\_attr\_init ( &attr3 );

pthread\_create ( &id1, &attr1, runner, tnames[0] );

pthread\_create ( &id2, &attr2, runner, tnames[1] );

pthread\_create ( &id3, &attr3, runner, tnames[2] );

pthread\_join ( id1, NULL );

pthread\_join ( id2, NULL );

pthread\_join ( id3, NULL );

return 0;

}

**Pthread Output:**

mike@DESKTOP-SEEUKNP:~/cse460/lab6$ ./pthreads\_demo

I am Thread 1

I am Thread 2

I am Thread 3

mike@DESKTOP-SEEUKNP:~/cse460/lab6$

**SDL Thread Code:**

#include <SDL/SDL.h>

#include <SDL/SDL\_thread.h>

#include <stdio.h>

using namespace std;

int runner ( void \*data ){

char \*tname = ( char \* )data;

printf("I am %s\n", tname );

return 0;

}

int main (){

SDL\_Thread \*id1, \*id2, \*id3;

char \*tnames[3] = { "Thread 1", "Thread 2", "Thread 3" };

id1 = SDL\_CreateThread ( runner, tnames[0] );

id2 = SDL\_CreateThread ( runner, tnames[1] );

id3 = SDL\_CreateThread ( runner, tnames[2] );

SDL\_WaitThread ( id1, NULL );

SDL\_WaitThread ( id2, NULL );

SDL\_WaitThread ( id3, NULL );

return 0;

}

**SDL Thread Output:**

mike@DESKTOP-SEEUKNP:~/cse460/lab6$ ./sdlthread\_demo

I am Thread 1

I am Thread 2

I am Thread 3

mike@DESKTOP-SEEUKNP:~/cse460/lab6$

1. **Using Semaphores:**
   1. When the program is executed as a background process it will continue to print out the 'I' and 'e'. This is due to the fact that the value is less than 1 when committed to the background. Output:

mike@DESKTOP-SEEUKNP:~/cse460/lab6$ ./sema1 &

[2] 12220

[1] Done ./sema1

mike@DESKTOP-SEEUKNP:~/cse460/lab6$ elelelelelelelelelel

12220 finished!

* 1. When the program is executed with the –a command, the value is greater then 0 which created the output of uppercase 'E' and 'L'. Output:

mike@DESKTOP-SEEUKNP:~/cse460/lab6$ ./sema1 a

ELELELELELELELELELEL

12221 finished!

mike@DESKTOP-SEEUKNP:~/cse460/lab6$

* 1. I modified the code to take in a command of 1 or 0 as instructed

**Code:**

//sema1.cpp

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

int i, pause\_time;

char ce = 'e', cl = 'l';

srand ( ( unsigned int ) getpid() );

sem\_id = semget ( (key\_t) 1234, 1, 0666 | IPC\_CREAT );

**//Changed the below code**

int check;

check = atoi(argv[1]);

if ( check > 0 ) {

if ( !set\_semvalue( 1 ) ) {

cout << "Semaphore initialized failed!" << endl;

exit ( EXIT\_FAILURE );

}

if ( check == 1 ) {

ce = 'E';

cl = 'L';

}

sleep ( 1 );

} else {

if ( !set\_semvalue( 0 ) ) {

cout << "Semaphore initialized failed!" << endl;

exit ( EXIT\_FAILURE );

}

sleep ( 1 );

}

for ( i = 0; i < 10; i++ ){

if ( !SEM\_DOWN () ) exit ( EXIT\_FAILURE );

cout << ce; fflush ( stdout );

pause\_time = rand() % 3;

sleep ( pause\_time );

cout << cl; fflush ( stdout );

if ( !SEM\_UP() ) exit ( EXIT\_FAILURE );

pause\_time = rand() % 2;

sleep ( pause\_time );

}

cout << endl << getpid() << " finished!" << endl;

if ( argc > 0 ) {

sleep ( 2 );

del\_semvalue ();

}

exit ( EXIT\_SUCCESS );

}

**Output:**

mike@DESKTOP-SEEUKNP:~/cse460/lab6$ ./sema1 1

ELELELELELELELELELEL

12244 finished!

mike@DESKTOP-SEEUKNP:~/cse460/lab6$ ./sema1 0

^C

mike@DESKTOP-SEEUKNP:~/cse460/lab6$

1. **XV6 Scheduling:**
   1. The modified proc.c output

Process initcode with the pid 1 running

Process initcode with the pid 1 running

Process initcode with the pid 1 running

Process initcode with the pid 1 running

sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 58

Process initcode with the pid 1 running

Process initcode with the pid 1 running

Process initcode with the pid 1 running

Process initcode with the pid 1 running

Process initcode with the pid 1 running

Process initcode with the pid 1 running

Process initcode with the pid 1 running

Process init with the pid 1 running

init: starting sh

Process init with the pid 1 running

Process init with the pid 2 running

Process init with the pid 2 running

Process init with the pid 2 running

Process init with the pid 2 running

Process init with the pid 2 running

Process init with the pid 2 running

Process sh with the pid 2 running

$

* 1. After implementing the foo.c program the output looks like:

ParenProcess foo with the pid 4 running

Process foo with the pid 4 running

Process foo with the pid 3 running

Process foo with the pid 3 running

Process foo with the pid 3 running

t 3 creating child 4

Process foo with the pid 4 running

Process foo with the pid 4 running

Child 4 created

Process foo with the pid 3 running

Process foo with the pid 3 running

Process foo with the pid 4 running

Process foo with the pid 4 running

Process foo with the pid 4 running

Process foo with the pid 3 running

Process foo with the pid 3 running

Process foo with the pid 4 running

And the pattern is continued until PID 6 and then the parent creates another child, 7, and continues till 7 is done then goes back through 6. It is assumed that the parent doesn’t wait to check if there are any other children and just exits.

* 1. The modified proc.h code

struct proc {

uint sz; pde\_t\* pgdir;

char \*kstack;

enum procstate state;

int pid;

struct proc \*parent;

struct trapframe \*tf;

struct context \*context;

void \*chan;

int killed;

struct file \*ofile[NOFILE];

struct inode \*cwd;

char name[16];

uint createTime;

int sleepTime;

int readyTime;

int runTime;

int priority;

int tickcounter;

char dum[8];

};

The modified proc.c portion

static struct proc\*

allocproc(void)

{

struct proc \*p;

char \*sp;

acquire(&ptable.lock);

for(p = ptable.proc; p < &ptable.proc[NPROC]; p++)

if(p->state == UNUSED)

goto found;

release(&ptable.lock);

return 0;

found:

p->state = EMBRYO;

p->pid = nextpid++;

p->createTime = ticks;

p->readyTime = 0;

p->runTime = 0;

p->sleepTime = 0;

release(&ptable.lock);

if((p->kstack = kalloc()) == 0){

p->state = UNUSED;

return 0;

}

Modified Code Output

cpu1: starting

cpu0: starting

Process initcode with pid 1 running with createTime 0

Process initcode with pid 1 running with createTime 0

sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 58

Process initcode with pid 1 running with createTime 0

Process initcode with pid 1 running with createTime 0

Process initcode with pid 1 running with createTime 0

Process initcode with pid 1 running with createTime 0

,Process initcode with pid 1 running with createTime 0

,Process initcode with pid 1 running with createTime 0

,Process initcode with pid 1 running with createTime 0

,Process init with pid 1 running with createTime 0

,init: starting sh

Process init with pid 1 running with createTime 0

,Process init with pid 2 running with createTime 75

,Process init with pid 2 running with createTime 75

,Process init with pid 2 running with createTime 75

$ foo 4

Process sh with pid 2 running with createTime 75

,Process sh with pid 2 running with createTime 75

,Process sh with pid 5 running with createTime 15061

,Process sh with pid 5 running with createTime 15061

,Process foo with pid 5 running with createTime 15061

,Parent 5 creating child 6

,Process foo with pid 6 running with createTime 15062

,Child 6 created

Process foo with pid 6 running with createTime 15062

,Process foo with pid 6 running with createTime 15062

,Process foo with pid 6 running with createTime 15062

,Process foo with pid 6 running with createTime 15062

,Process foo with pid 5 running with createTime 15061

,Parent 5 creating child 7

Process foo with pid 7 running with createTime 15129

,Process foo with pid 7 running with createTime 15129

,Process foo with pid 7 running with createTime 15129

,Process foo with pid 7 running with createTime 15129

,Process foo with pid 9 running with createTime 15269

,Process foo with pid 9 running with createTime 15269

,Process foo with pid 9 running with createTime 15269

,Process foo with pid 5 running with createTime 15061

,Process foo with pid 5 running with createTime 15061

,Process foo with pid 5 running with createTime 15061

,Process foo with pid 5 running with createTime 15061

,Process foo with pid 5 running with createTime 15061

,Process foo with pid 5 running with createTime 15061

,Process foo with pid 5 running with createTime 15061

,Process foo with pid 5 running with createTime 15061

,Process sh with pid 2 running with createTime 75

I was able to complete each section of the lab. I was able to produce correct outputs and added scripts of the outputs. I believe I have earned a full 20/20 points on this lab report.