**XINU Problems**

**Problem 1: Semaphore**

**#include** <xinu.h>

int *n* = 0;

void *producer*(sid32 *prod*,sid32 *cons*)

{

**while** (*n* <= 5) {

*wait*(*cons*);

*n*++;

*signal*(*prod*);

}

}

void *consumer*(sid32 *prod*,sid32 *cons*)

{

**while** (*n* <= 5) {

*wait*(*prod*);

*printf*("*%d*\n", *n*);

*signal*(*cons*);

}

}

int *hello1*()

{

sid32 *prod* = *semcreate*(1);

sid32 *cons* = *semcreate*(0);

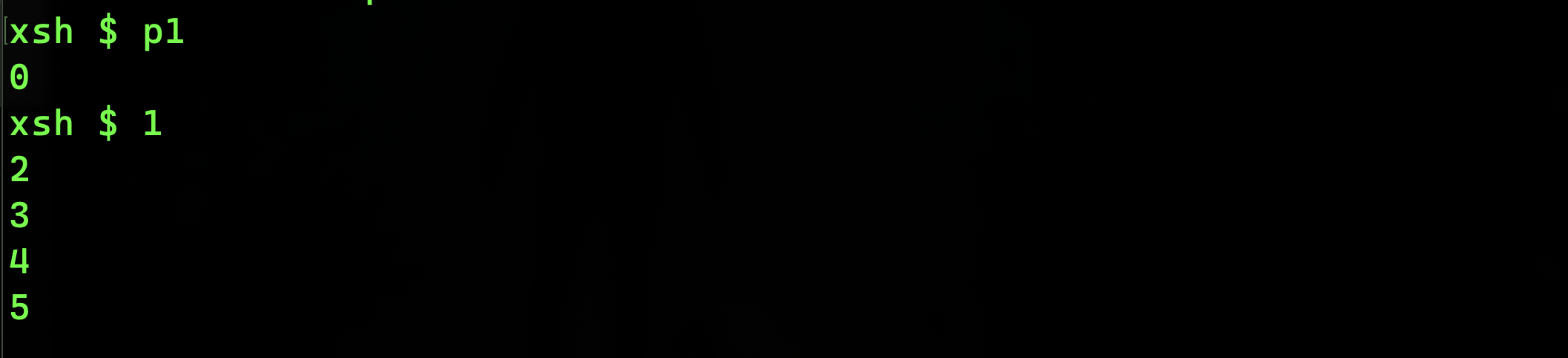
*resume*(*create*(*producer*, 1024, 20, "producer", 2, *prod*, *cons*));

*resume*(*create*(*consumer*, 1024, 20, "consumer", 2, *prod*, *cons*));

**return** 0;

}

**Terminal Output:**

****

**Problem 2: Semaphore**

#include <xinu.h>

int n = 5;

void producer(*sid32* prod,*sid32* cons)

{

while (n >= 0) {

wait(cons);

printf("prod:%d\n",n--);

signal(prod);

}

printf("Done - produced\n");

}

void consumer(*sid32* prod,*sid32* cons)

{

while (n >= 0) {

wait(prod);

printf("cons:%d\n",n--);

signal(cons);

}

printf("Done - consumed\n");

}

int hello1()

{

*sid32* prod = semcreate(1);

*sid32* cons = semcreate(0);

resume(create(producer, 1024, 20, "producer", 2, prod, cons));

resume(create(consumer, 1024, 20, "consumer", 2, prod, cons));

return 0;

}

**Terminal Output:**

****

**Problem 3: Semaphore**

#include <xinu.h>

int n = 5;

void producer(*sid32* prod,*sid32* cons)

{

while (n >= 0) {

wait(cons);

printf("prod:%d\n",n--);

signal(prod);

}

printf("Done - produced\n");

}

void consumer(*sid32* prod,*sid32* cons)

{

while (n >= 0) {

wait(prod);

printf("cons:%d\n",n--);

signal(cons);

}

printf("Done - consumed\n");

}

int hello1()

{

*sid32* prod = semcreate(1);

*sid32* cons = semcreate(0);

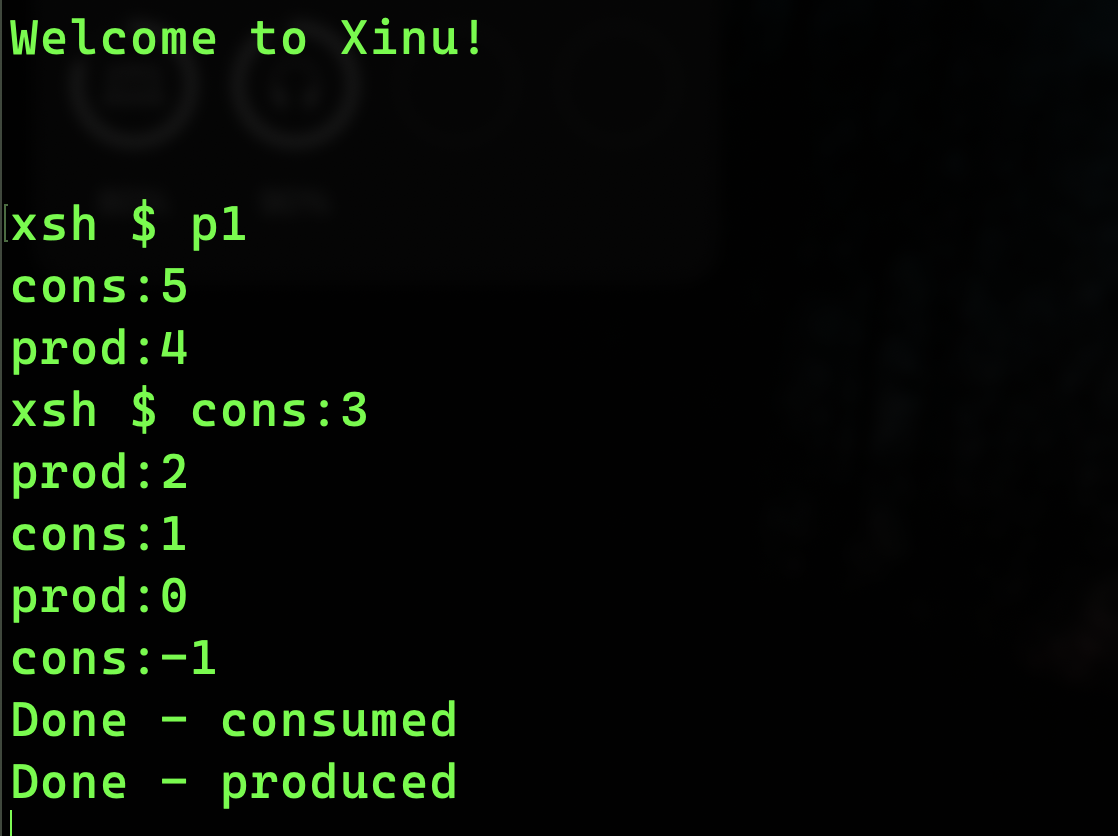
resume(create(consumer, 1024, 20, "consumer", 2, prod, cons));

resume(create(producer, 1024, 20, "producer", 2, prod, cons));

return 0;

}

**Terminal Output:**

****

**Problem 4: Semaphore**

**Solution of the error of Problem 3**

#include <xinu.h>

int n = 5;

void producer(*sid32* prod,*sid32* cons)

{

while (n >= 0) {

wait(cons);

printf("prod:%d\n",n--);

signal(prod);

}

printf("Done - produced\n");

}

void consumer(*sid32* prod,*sid32* cons)

{

while (n >= 0) {

wait(prod);

if(n<0){

signal(cons);

break;

}

printf("cons:%d\n",n--);

signal(cons);

}

printf("Done - consumed\n");

}

int hello1()

{

*sid32* prod = semcreate(1);

*sid32* cons = semcreate(0);

resume(create(consumer, 1024, 20, "consumer", 2, prod, cons));

resume(create(producer, 1024, 20, "producer", 2, prod, cons));

return 0;

}

**Terminal Output:**

**A screen shot of a computer

AI-generated content may be incorrect.**

**Problem 5: ReadyList**

#include <xinu.h>

void proc3()

{

printf("Process 1\n");

}

void proc4()

{

printf("Process 2\n");

}

void PrintReady()

{

*qid16* head = queuehead(readylist);

*qid16* tail = head + 1;

while (head != tail)

{

struct *qentry* curr = queuetab[head];

printf("id=%d,key=%d,prev=%d,next=%d\n", head, curr.qkey, curr.qprev, curr.qnext);

head = curr.qnext;

}

}

int hello1()

{

resume(create(proc3, 1024, 20, "proc3", 0));

resume(create(proc4, 1024, 20, "proc4", 0));

return 0;

}

**Terminal Output:**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Problem 6: Ready Process Info from Proctab**

#include <xinu.h>

void proc3()

{

printf("Proces 3 - %d\n",getpid());

while(1){

}

}

void proc4()

{

printf("Process 4 - %d\n",getpid());

while (1)

{

}

}

void PrintReady()

{

*qid16* head = queuehead(readylist);

*qid16* tail = head + 1;

printf("Head=%d, Tail=%d\n", head, tail);

while (head != tail)

{

struct *qentry* curr = queuetab[head];

struct *procent* process = proctab[head];

printf("id=%d,prio=%d,stklen=%d,name=%s, semid=%d,parent=%d\n", head, process.prprio, process.prstklen, process.prname, process.prsem, process.prparent);

head = curr.qnext;

}

}

int LF()

{

resume(create(proc3, 1024, 20, "proc3", 0));

resume(create(proc4, 1024, 20, "proc4", 0));

resume(create(PrintReady, 1024, 20, "readyPrinter", 0));

return 0;

}

**Terminal Output:**

**A computer screen shot of a code

AI-generated content may be incorrect.**

**Problem 7: WaitList**

#include <xinu.h>

void proc3(*sid32* sem)

{

printf("Proces 3 - %d\n", getpid());

wait(sem);

while (1)

{

}

}

void proc4(*sid32* sem)

{

printf("Process 4 - %d\n", getpid());

wait(sem);

while (1)

{

}

}

void printer()

{

for (int i = 0; i < NPROC; i++)

{

struct *procent* curr = proctab[i];

if (curr.prstate == PR\_WAIT)

{

printf("pid-%d\n",i);

}

}

}

int LF()

{

*sid32* sem = semcreate(0);

resume(create(proc3, 1024, 20, "proc3", 1,sem));

resume(create(proc4, 1024, 20, "proc4", 1,sem));

resume(create(printer, 1024, 20, "readyPrinter", 0));

return 0;

}

**Terminal Output:**

**A blue background with white text

AI-generated content may be incorrect.**

**Problem 8,9: Rescheduling process and Printing the Prio, Resumed process ID, and Resumed By process ID**

**Terminal Output:**

**A screen shot of a computer

AI-generated content may be incorrect.**

**A screen shot of a computer

AI-generated content may be incorrect.**

**Problem 10: Accessing semtab**

#include<xinu.h>

void proc1(){

printf("Process 1 : %d\n", getpid());

while (1){

}

}

void proc2(){

printf("Process 2 : %d\n", getpid());

while (1){

}

}

void Printer(){

struct *sentry* curr;

for(int i = 0; i < NSEM; i++){

curr = semtab[i];

if(curr.sstate == S\_USED){

printf("Semid = %d, count = %d\n", i, curr.scount);

}

}

}

int hello1(){

*sid32* sem1 = semcreate(0);

*sid32* sem2 = semcreate(1);

*sid32* sem3 = semcreate(2);

*sid32* sem4 = semcreate(3);

printf("Newly created semaphore ids - %d, %d, %d, %d\n", sem1, sem2, sem3, sem4);

resume(create(proc1, 1024, 20, "proc1", 0));

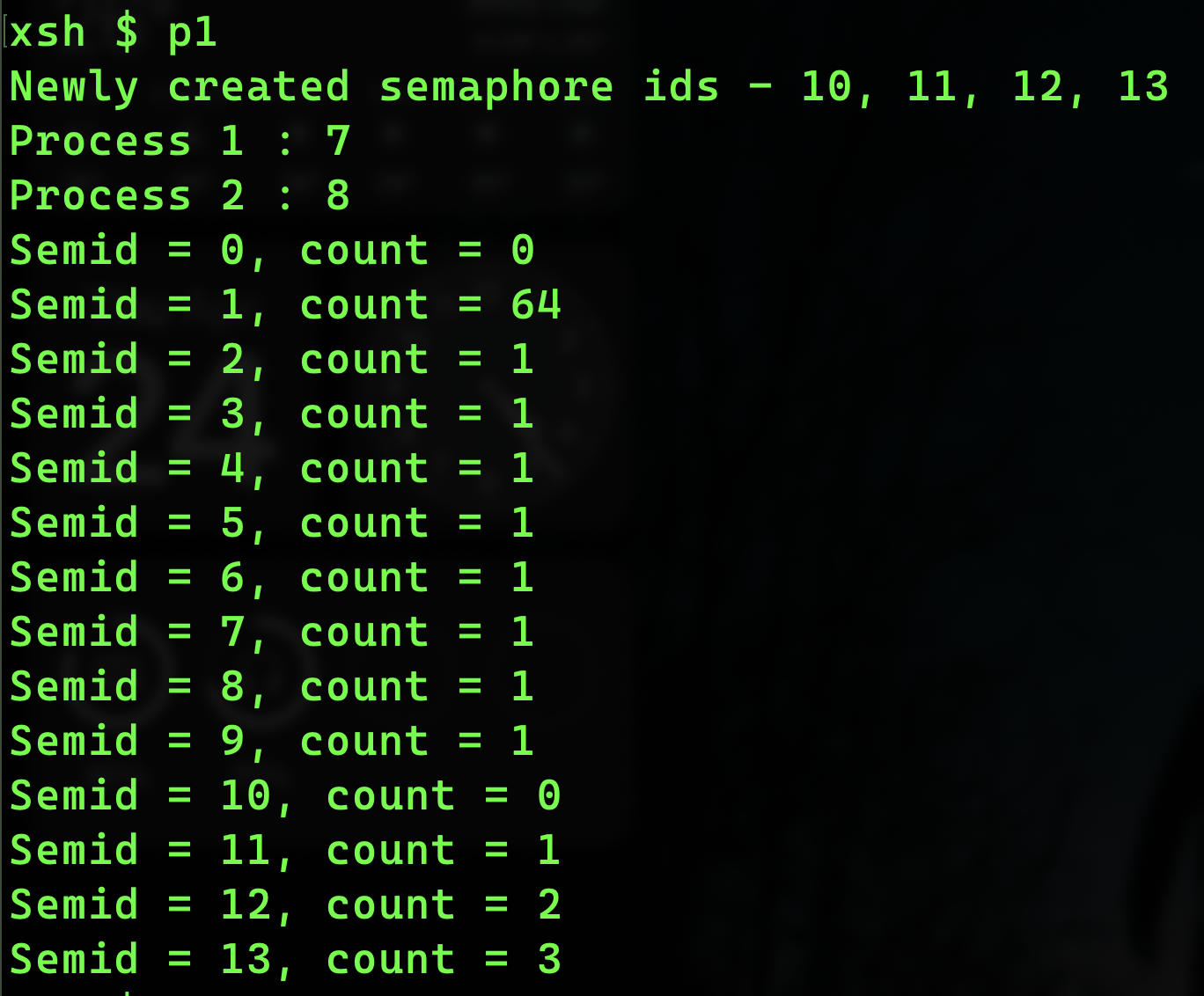
resume(create(proc2, 1024, 20, "proc2", 0));

resume(create(Printer, 1024, 20, "Printer", 0));

return 0;

}

**Terminal Output:**

****

**Problem 11: semdelete**

#include<xinu.h>

void proc1(){

printf("Process 1 : %d\n", getpid());

while (1){

}

}

void proc2(){

printf("Process 2 : %d\n", getpid());

while (1){

}

}

void Printer(){

struct *sentry* curr;

for(int i = 0; i < NSEM; i++){

curr = semtab[i];

if(curr.sstate == S\_USED){

printf("Semid = %d, count = %d\n", i, curr.scount);

}

}

}

int hello1(){

*sid32* sem1 = semcreate(0);

*sid32* sem2 = semcreate(1);

*sid32* sem3 = semcreate(2);

*sid32* sem4 = semcreate(3);

printf("Newly created semaphore ids - %d, %d, %d, %d\n", sem1, sem2, sem3, sem4);

resume(create(proc1, 1024, 20, "proc1", 0));

resume(create(proc2, 1024, 20, "proc2", 0));

resume(create(Printer, 1024, 20, "Printer", 0));

semdelete(sem3);

semdelete(sem4);

printf("After deleting sem3 and sem4\n");

resume(create(Printer, 1024, 20, "Printer", 0));

return 0;

}

**Terminal Output:**

**A computer screen shot of a number

AI-generated content may be incorrect.**

**Problem 12: semreset**

#include <xinu.h>

void proc1()

{

printf("Process 1 : %d\n", getpid());

while (1)

{

}

}

void proc2()

{

printf("Process 2 : %d\n", getpid());

while (1)

{

}

}

void Printer()

{

struct *sentry* curr;

for (int i = 0; i < NSEM; i++)

{

curr = semtab[i];

if (curr.sstate == S\_USED)

{

printf("Semid = %d, count = %d\n", i, curr.scount);

}

}

}

int hello1()

{

*sid32* sem1 = semcreate(0);

*sid32* sem2 = semcreate(1);

*sid32* sem3 = semcreate(2);

*sid32* sem4 = semcreate(3);

printf("Newly created semaphore ids - %d, %d, %d, %d\n", sem1, sem2, sem3, sem4);

resume(create(proc1, 1024, 20, "proc1", 0));

resume(create(proc2, 1024, 20, "proc2", 0));

resume(create(Printer, 1024, 20, "Printer", 0));

semreset(sem3, 0);

semreset(sem4, 0);

printf("After reseting sem3 and sem4\n");

resume(create(Printer, 1024, 20, "Printer", 0));

return 0;

}

**Terminal Output:**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**System Programming**

**Problem 6: Semaphore**

**Terminal Output:**