|  |
| --- |
| Lab – 11 |
| Solution |
| Capt Shorahbeel, Capt Raza |

Task – 1 we ran this program multiple times without semaphores(by commenting out wait(s),signal(s) lines). This resulted in unexpected values of variable sum. Most of the times, value ranged from 12-18 whereas the output should have been 20. This behavior was due to un-coordination between two processes. Two process were accessing a shared variable sum.

/\* Add 10 to a variable in each of two processes. \*/

int sum = 0;

binarysem s = 1;

void add10() {

int i;

int local;

for (i = 1; i <= 10;i++) {

wait(s);

local = sum;

sum = local + 1;

signal(s);

}

}

void main() {

cobegin {

add10(); add10();

}

cout << "Sum = " << sum << endl;

}

Once semaphores were used, the output was correct. It was20

Task – 2 In this program, two different processes were using a shared variable n. the output of the program was from 14-16 most of the times. Two implement coordinated access to the shared variable, I used semaphores. A variable s of binarysem type was declared with initial value as 1. Wait(s) was inserted before the critical section in both function codes and signal(s) was added after the critical section.

After modification, the output is correct.

int n = 0;

binarysem s=1;

void p() {

int temp, i;

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

wait(s);

temp = n;

n = temp + 1;

signal(s);

}

}

void q() {

int temp, i;

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

wait(s);

temp = n;

n = temp + 1;

signal(s);

}

}

void main() {

cobegin { p(); q(); }

cout << "The value of n is " << n << "\n";

}

Task – 3 For integer non-negative values, semaphore type is used. It can be initialized by using initialsem(semaphore s,int n) function. I initialized semaphore with 1 and ran the code, the output is 20.

The output is same in both the cases i.e binarysem and semaphore(integer)

int sum = 0;

semaphore s;

void add10() {

int i;

int local;

for (i = 1; i <= 10;i++) {

wait(s);

local = sum;

sum = local + 1;

signal(s);

}

}

void main() {

initialsem(s,1);

cobegin {

add10(); add10();

}

cout << "Sum = " << sum << endl;

}

Task – 4 in this program, signal(s) call was missing in function p() and wait(s) was missing in function q(). This resulted into varying output. The output value for variable count ranged from 7-10. I implemented complete semaphores in both the functions, this time the output is correct.

int count=0;

binarysem s=1;

int total = 5;

int i=0;

int j=0;

void p()

{

for(;i<total;i++){

wait(s);

count = count+1;

cout<<"Count Inside p="<<count<<endl;

signal(s);

}

}

void q()

{

for(;j<total;j++){

wait(s);

count = count+1;

cout<<"Count Inside q="<<count<<endl;

signal(s);

}

}

main(){

cobegin{

p();q();

}

}