CSE344 SYSTEM PROGRAMING

HW-4

REFİK ORKUN ARSLAN
151044063

main:

Initiliaze Semaphore

```
key_t key = ftok(argv[0], 1);
if(    (semid = semget(key, 2, S_IRUSR | S_IWUSR |
IPC_CREAT)) == -1)
        {
                perror("semget\n");
        }
void sem_init(int semid, int semaphore_number, int initial_value)
{
        union semun argument;
        argument.val = initial_value;
        if(semctl(semid, semaphore_number, SETVAL, argument) == -1)
        {
                perror("semctl\n");
        }
}
```

Set semaphore iniliaze

Detached Thread Initiliaze

```
pthread_t thr;
        pthread_attr_t attr;
        int s;
        s = pthread_attr_init(&attr);
        if (s != 0)
                perror("pthread_attr_init");
        s = pthread_attr_setdetachstate(&attr,
PTHREAD_CREATE_DETACHED);
        if (s != 0)
                perror("pthread_attr_setdetachstate");
        s = pthread_create(&thr, &attr, producerTask, (void *) pt);
        if (s != 0)
                perror("pthread_create");
        s = pthread_attr_destroy(&attr);
        if (s != 0)
        perror("pthread_attr_destroy");
```

Finally Sıpplier Thread Wait Consumer Thread

```
for (int i = 0; i < NCONSUMERS; i++)
	{
		if(!pthread_equal(pthread_self(),ct[i].conId))
		{
			pthread_join(consthread[i],NULL);
		}
	}
```

producerTask:

info → ptime=number of consumer * N*2

if (ch=='1')

If the value read is 1, the semaphore associated with 1 is increased.

```
fprintf(stdout, "%sSupplier: read from input a ‘1’. Current amounts: %d x ‘1’, %d x ‘2’.\n",asctime(timeinfo),value1,value2 );
sem_action(semid, ONE, 1, &sem[0]);
```

The value read was made before the semaphore was incremented. Then the deliver task was printed.

if(value1>value2)

If the number 1 was ahead, it was printed accordingly.

When 2 is read, the same operations are performed.

ConsumerTask:

for (i = 0; i < info->iter; ++i)

each thread should return N

```
sem[0].sem_op = -1;
sem[0].sem_num = 0;
sem[0].sem_flg = 0;

sem[1].sem_op = -1;
sem[1].sem_num = 1;
sem[1].sem_flg = 0;
semop(semid,sem, 2);
```

1 and 2 have to be at the same time so that the semaphore can consume. That's why semop(semid,sem, 2) is used.

Then the printing operations were performed.

OUTPUT

```
(null)Supplier: read from input a ‘2’. Current amounts: 0 x ‘1’, 0 x ‘2’.
Sat May 14 04:52:15 2022
Supplier: delivered a ‘2’. Post-delivery amounts: 0 x ‘1’, 1 x ‘2’.

Sat May 14 04:52:15 2022
Supplier: read from input a ‘2’. Current amounts: 0 x ‘1’, 1 x ‘2’.
Sat May 14 04:52:15 2022
Supplier: delivered a ‘2’. Post-delivery amounts: 0 x ‘1’, 2 x ‘2’.

Sat May 14 04:52:15 2022
Supplier: read from input a ‘2’. Current amounts: 0 x ‘1’, 2 x ‘2’.
Sat May 14 04:52:15 2022
Supplier: delivered a ‘2’. Post-delivery amounts: 0 x ‘1’, 3 x ‘2’.

Sat May 14 04:52:15 2022
Supplier: read from input a ‘2’. Current amounts: 0 x ‘1’, 3 x ‘2’.
Sat May 14 04:52:15 2022
Supplier: delivered a ‘2’. Post-delivery amounts: 0 x ‘1’, 4 x ‘2’.

Sat May 14 04:52:15 2022
Supplier: read from input a ‘1’. Current amounts: 0 x ‘1’, 4 x ‘2’.
Sat May 14 04:52:15 2022
Supplier: delivered a ‘1’. Post-delivery amounts: 1 x ‘1’, 4 x ‘2’.

Sat May 14 04:52:15 2022
Consumer-0 at iteration 0 (waiting). Current amounts: 1 x ‘1’, 4 x ‘2’.
Sat May 14 04:52:15 2022
Consumer-0 at iteration 0 (consumed). Post-consumption amounts: 0 x ‘1’, 3 x ‘2’.

Sat May 14 04:52:15 2022
Supplier: read from input a ‘1’. Current amounts: 0 x ‘1’, 3 x ‘2’.
Sat May 14 04:52:15 2022
Supplier: delivered a ‘1’. Post-delivery amounts: 1 x ‘1’, 3 x ‘2’.

Sat May 14 04:52:15 2022
Supplier: read from input a ‘1’. Current amounts: 0 x ‘1’, 2 x ‘2’.
Sat May 14 04:52:15 2022
Supplier: delivered a ‘1’. Post-delivery amounts: 1 x ‘1’, 2 x ‘2’.

Sat May 14 04:52:15 2022
Consumer-0 at iteration 1 (waiting). Current amounts: 1 x ‘1’, 2 x ‘2’.
Sat May 14 04:52:15 2022
Consumer-0 at iteration 1 (consumed). Post-consumption amounts: 0 x ‘1’, 1 x ‘2’.

Sat May 14 04:52:15 2022
Consumer-1 at iteration 0 (waiting). Current amounts: 1 x ‘1’, 2 x ‘2’.
Sat May 14 04:52:15 2022
Consumer-1 at iteration 0 (consumed). Post-consumption amounts: 0 x ‘1’, 1 x ‘2’.

Sat May 14 04:52:15 2022
Supplier: read from input a ‘1’. Current amounts: 0 x ‘1’, 1 x ‘2’.
Sat May 14 04:52:15 2022
Supplier: delivered a ‘1’. Post-delivery amounts: 1 x ‘1’, 1 x ‘2’.

Sat May 14 04:52:15 2022
Consumer-1 at iteration 1 (waiting). Current amounts: 1 x ‘1’, 1 x ‘2’.
Sat May 14 04:52:15 2022
Supplier: read from input a ‘1’. Current amounts: 0 x ‘1’, 0 x ‘2’.
Sat May 14 04:52:15 2022
Consumer-1 at iteration 1 (consumed). Post-consumption amounts: 0 x ‘1’, 0 x ‘2’.
```

I tried many scenarios but got no errors in the output.