INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO
SISTEMAS OPERATIVOS
GRUPO: 2CM8

ALUMNO:
NAVA VIVAS ANA PAOLA
MENDOZA FLORES ERICK GABRIEL

# SEMÁFOROS

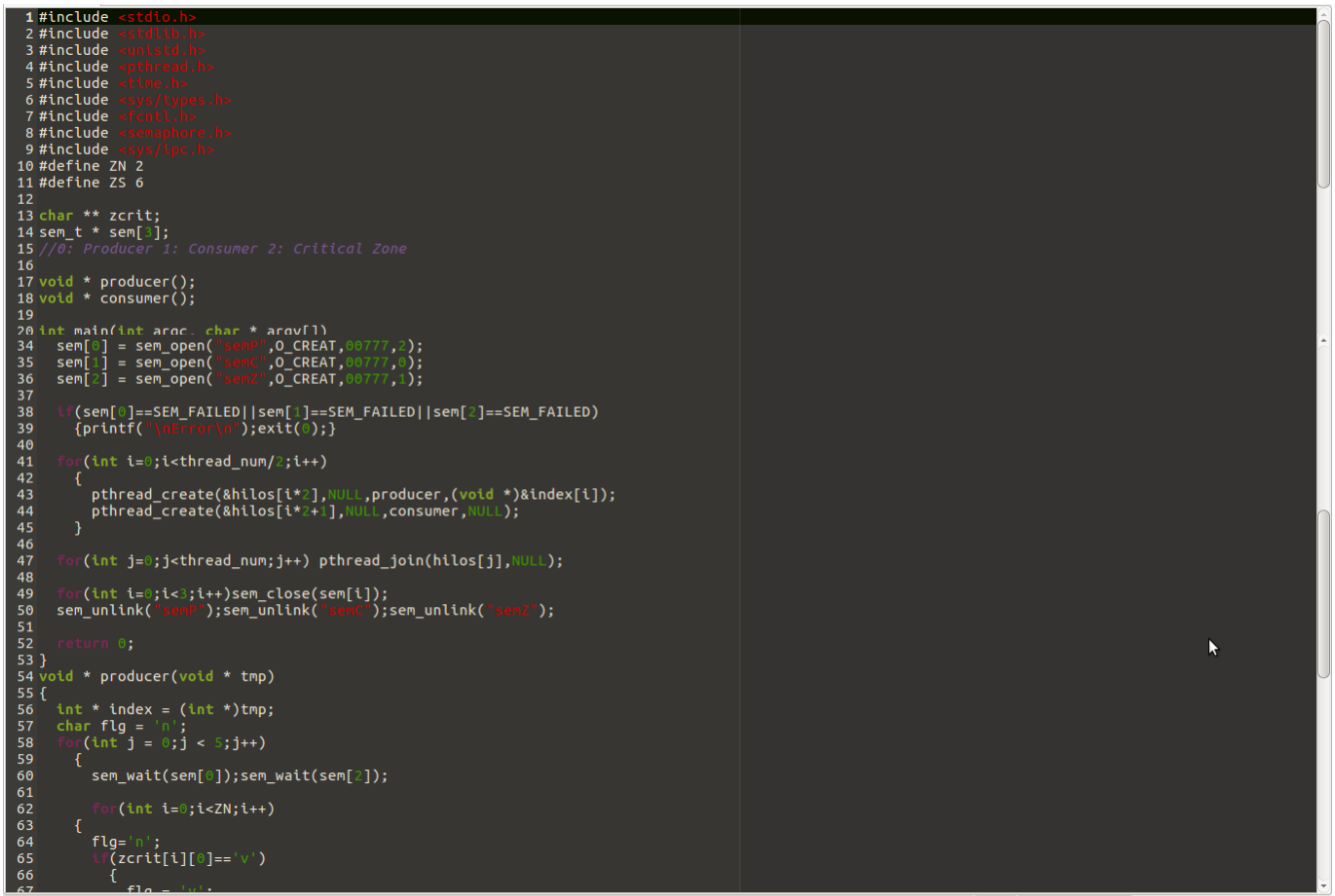
## Introducción:

Lo semáforos posix ertenecen al estándar POSIX.1b y son relativamente recientes (1993).

Son una variable de tipo sem_t sobre la cual se pueden realizar las funciones clásicas de los semáforos.

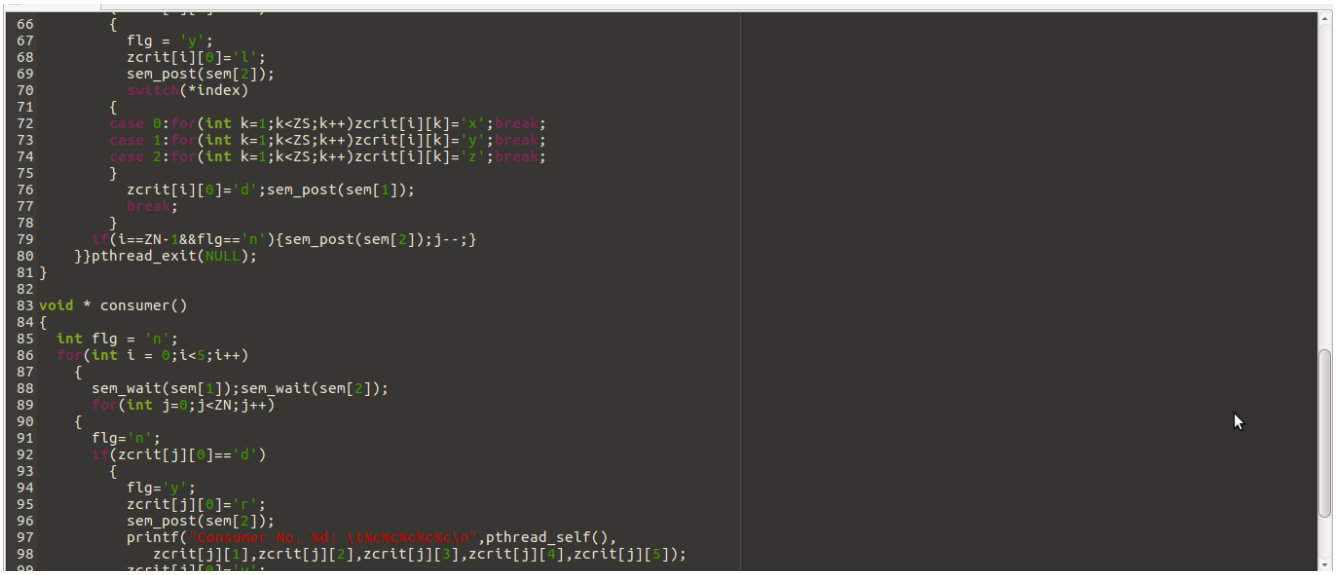
## Desarrollo:

El programa consistía en sincronizar unos semáforos para controlar la información que entraba y salía de dos zonas críticas. Se tenían 3 hilos productores y 3 hilos consumidores, el objetivo era que los hilos productores escribieran xxxxx, yyyyy, zzzzz 5 veces respectivamente y que los hilos consumidores entraran a las zonas críticas y leyeran esa información de tal manera que pudieran capturarla antes de que algún productor sobreescribiera. Para resolver dicho problema se usaron semáforos Posix.

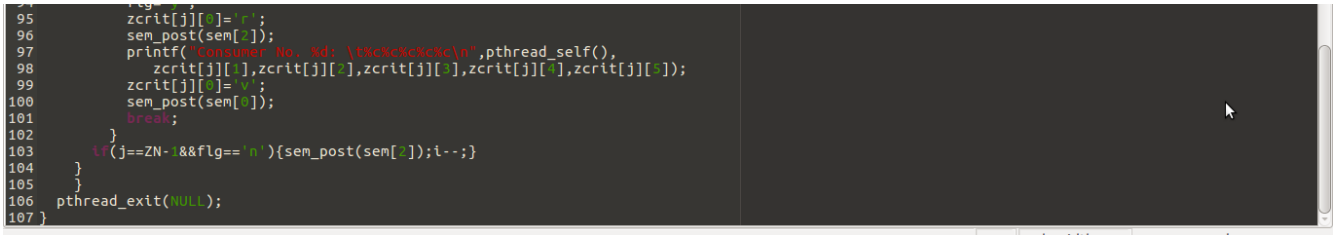


```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <pthread.h>
#include <time.h>
#include <sys/types.h>
#include <fcntl.h>
#include <semaphore.h>
#include <sys/ipc.h>
#define ZN 2
#define ZS 6

char ** zcrit;
sem_t * sem[3];
//0: Producer 1: Consumer 2: Critical Zone

void * producer();
void * consumer();

int main(int argc, char * argv[])
  sem[0] = sem_open("semP",O_CREAT,00777,2);
  sem[1] = sem_open("semC",O_CREAT,00777,0);
  sem[2] = sem_open("semZ",O_CREAT,00777,1);

  if(sem[0]==SEM_FAILED||sem[1]==SEM_FAILED||sem[2]==SEM_FAILED)
    {printf("\nError\n");exit(0);}

  for(int i=0;i<thread_num/2;i++)
    {
      pthread_create(&hilos[i*2],NULL,producer,(void *)&index[i]);
      pthread_create(&hilos[i*2+1],NULL,consumer,NULL);
    }

  for(int j=0;j<thread_num;j++) pthread_join(hilos[j],NULL);

  for(int i=0;i<3;i++)sem_close(sem[i]);
  sem_unlink("semP");sem_unlink("semC");sem_unlink("semZ");

  return 0;
}
void * producer(void * tmp)
{
  int * index = (int *)tmp;
  char flg = 'n';
  for(int j = 0;j < 5;j++)
    {
      sem_wait(sem[0]);sem_wait(sem[2]);

      for(int i=0;i<ZN;i++)
    {
      flg='n';
      if(zcrit[i][0]=='v')
        {
```

El algoritmo revisa cada una de la áreas de la zona crítica y ve si puede escribir en ella preguntando al semáforo su valor. El algoritmo no es perfecto ya que algunos de los hilos se pueden estancar ahí. Sin embargo es de las mejores opciones que existe ya que los semáforos POSIX lo hacen muy eficientemente.



```
66          {
67            flg = 'y';
68            zcrit[i][0]='l';
69            sem_post(sem[2]);
70            switch(*index)
71          {
72          case 0:for(int k=1;k<ZS;k++)zcrit[i][k]='x';break;
73          case 1:for(int k=1;k<ZS;k++)zcrit[i][k]='y';break;
74          case 2:for(int k=1;k<ZS;k++)zcrit[i][k]='z';break;
75          }
76            zcrit[i][0]='d';sem_post(sem[1]);
77            break;
78          }
79        if(i==ZN-1&&flg=='n'){sem_post(sem[2]);j--;}
80      }}pthread_exit(NULL);
81  }
82
83  void * consumer()
84  {
85    int flg = 'n';
86    for(int i = 0;i<5;i++)
87      {
88        sem_wait(sem[1]);sem_wait(sem[2]);
89        for(int j=0;j<ZN;j++)
90      {
91        flg='n';
92        if(zcrit[j][0]=='d')
93          {
94            flg='y';
95            zcrit[j][0]='r';
96            sem_post(sem[2]);
97            printf("Consumer No. %d: \t%c%c%c%c%c\n",pthread_self(),
98               zcrit[j][1],zcrit[j][2],zcrit[j][3],zcrit[j][4],zcrit[j][5]);
```



```
95            zcrit[j][0]='r';
96            sem_post(sem[2]);
97            printf("Consumer No. %d: \t%c%c%c%c%c\n",pthread_self(),
98               zcrit[j][1],zcrit[j][2],zcrit[j][3],zcrit[j][4],zcrit[j][5]);
99            zcrit[j][0]='v';
100           sem_post(sem[0]);
101           break;
102         }
103       if(j==ZN-1&&flg=='n'){sem_post(sem[2]);i--;}
104     }
105     }
106   pthread_exit(NULL);
107 }
```

El consumidor lo único que hace es leer de las zonas, recurriendo a la misma táctica que su contraparte productora. Solo resta mostrar en pantalla el resultado y listo. Ambos procesos terminan y cierrar sus espacios de memoria compartida, en este caso los semáforos.

## Funcionamiento: