Programación de Sistemas (PrSist)

Departamento de Informática de Sistemas y Computadoras (DISCA)
Universitat Politècnica de València

Trabajo asignatura

Contenido

1 Objetivos

El objetivo final del trabajo de la asignatura es ofrecer un esquema general de uso de variables compartidas, en las cuales se garantiza el acceso correcto a la misma en un escenario concurrente por el uso de las construcciones adecuadas proporcionadas por el estándar POSIX.

2 Acceso al reloj para medida del tiempo

Es bastante habitual en las aplicaciones (y todavía más en aquellas vinculadas al control de procesos), tener que acceder a un reloj de sistema para conocer el instante actual, e incluso ser capaces de medir el tiempo transcurrido. Veamos en el siguiente ejemplo como hacerlo

```
/* ejemplo uso reloj: calculo de los primos*/
#include <stdio.h>      /* printf */
#include <time.h>       /* clock_t, clock, CLOCKS_PER_SEC */
#include <math.h>       /* sqrt */

int frequency_of_primes (int n) {
    int i,j;
    int freq=n-1;
    for (i=2; i<=n; ++i) for (j=sqrt(i);j>1;--j) if (i%j==0) {--freq;
break;}
    return freq;
}

int main ()
{
    clock_t t;
    int f;
    t = clock();
    printf ("calculando...\n");
    f = frequency_of_primes (99999);
    printf ("El numero de primos menor que 100.000 es: %d\n",f);
    t = clock() - t;
    printf ("Me ha costado %d clicks (%f
segundos).\n",(int)t,((float)t)/CLOCKS_PER_SEC);
```

```
    return 0;
}
```

Este ejemplo nos sirve para medir el tiempo que ha empleado en ejecutar unas instrucciones, sin embargo, no es adecuado para medir el tiempo transcurrido en el caso en que tengamos que ejecutar suspensiones de los threads, este tiempo no se cintabiliza.
En ese caso es necesario acceder a un reloj de tiempo real para medir el paso del tiempo desde una marca inicio absoluta. Para estos caso podemos usar la siguiente función de la librería time.h

```
long getCurrentMicroseconds()
{
      struct timespec currentTime;
      clock_gettime(CLOCK_MONOTONIC, &currentTime);
      return (currentTime.tv_sec)*1000000 + (currentTime.tv_nsec) / 1000;
}
```

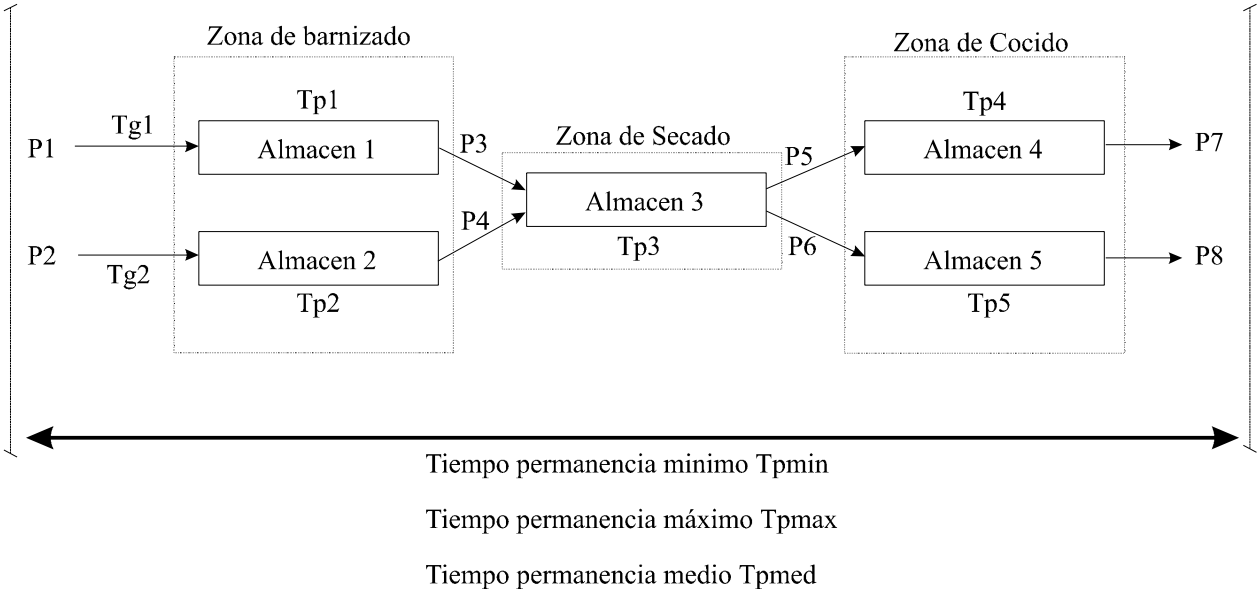
con una invocación

```
long current_time = getCurrentMicroseconds();
```

Realizando la diferencia entre dos invocaciones a la función en dos puntos del código, obtendremos el tiempo absoluto en microsegundos que ha transcurrido entre la ejecución de esas instrucciones, se cual sea los estados por los que haya pasado del thread o proceso que los invoque.

3 Enunciado

Sea el sistema de fabricación según el esquema mostrado en la figura:



donde tenemos un conjunto de threads P1 .. P8 y un conjunto de almacenes de proceso de 1 a 5.
Los procesos P1 y P2 son generadores de piezas. Estas se crean de forma cíclica cada Tg1 y Tg2 respectivamente. Una vez se ha generado las piezas del proceso 1 pasan al almacén 1 donde tendrán que permanecer durante un tiempo Tp1 seg. Lo mismo ocurre con las piezas de P2. Una vez han permanecido el tiempo necesario en sus almacenes pasan a una zona común que es el almacén 3 donde deben permanecer Tp3 seg. Esto lo gestionan los threads P3 y P4, los cuales están realizando una consulta periódica (se entiende que el valor del pooling será menor de Tp3 para evitar que se deterioren las piezas) para ver si ya ha transcurrido el tiempo de mover una pieza. Una vez han terminado esta fase existen dos threads que los extraen de dicho almacén y los pasan cada uno a la siguiente fase según se muestra en la figura, siguiendo el mismo esquema de permanencia y extracción guiado por Tp4 y Tp5. Los threads P1 y P2 son meros generadores de piezas (productores) Los threads P3,P4,P5,P6

se encargan de sacar las piezas de los almacenes y pasarlos a otros una vez cumplido el tiempo de permanencia en los mismos (son consumidores en un almacén y productores en otro).Finalmente los procesos P7 y P8 se encargarán de sacar piezas cumplido el tiempo (son consumidores puros).

4 Apartados.

1. Definir las estructuras de datos necesarias en C, que permita modelar la información relativa a cada una de las piezas durante su trasiego por la fabrica, así como de los almacenes (fases) por las que van pasando. Es importante para cumplir con este apartado hacer el desarrollo de forma que se permita la facilidad de modificación de los parámetros relevantes del sistema (tamaño de los almacenes, tiempos de permanencia, así como la organización del código en ficheros (2p).
2. Diseñar e implementar un programa en C que mediante la ejecución de los hilos que se requiera, controle el sistema propuesto de acuerdo a la lógica descrita, mostrando en cada momento el estado de los almacenes (número de piezas que contiene) (4p).
3. Controlar los tiempos de permanencia de cada pieza en cada almacén, considerando que una pieza será defectuosa si sobrepasa el tiempo de cada zona en 2 segundos para la sección de barnizado, 4 la de secado y 6 la de cocido. Para ello utilizar la llamada a la función clock() incluida en time.h la cual permite medir el paso del tiempo, así como definir variables del tipo time_t para almacenar puntos temporales (3p).
4. Se propone controlar el paso de las piezas por los almacenes en función de quien las ha generadoi. De esta forma las piezas generadas por P1 realizarían el siguiente recorrido: Almacen1->Almacen3->Almacen4. Y las piezas generadas por P2 harían el recorrido Almacen2->Almacen3->Almacen5. (1p)