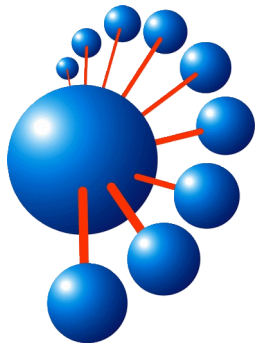

Proyecto 2 Sistemas Operativos

Baño Unisex

Profesora: Cecilia Hernández.
Alumnos: Diego Varas.
Jeremías Torres.

Mayo 16, 2017.

Introducción

En este trabajo afrontaremos y abordaremos un problema seleccionado al azar del libro “Little Book of Semaphores”.

El problema que tenemos fue el Baño Unisex.

Aquí estudiaremos, investigaremos e intentaremos solucionar este problema con el uso de semáforos, variables de condición/mutexes o monitores.

Definición problema

Problema 6.2 Little Book of Semaphores

El problema consiste en crear un baño unisex, éste se creó para la comodidad de un lugar donde el baño de mujeres estaba a 2 pisos de distancia, se aceptó la propuesta siempre y cuando se cumplieran ciertas reglas, que serás las restricciones del problema.

Las restricciones son las siguientes:

- No pueden haber hombres y mujeres al mismo tiempo en el baño.
- No pueden haber más de 3 personas en el baño (lo ve como que desperdicia el tiempo de la empresa tener tanto empleado usando el baño).

Algoritmos de Solución

En esta sección analizaremos los algoritmos que utilizamos para resolver el problema.
Algoritmo con Semaforos.

```
sem_wait(&empty);
sem_wait(&womanSemaphore);
m_count++;
if(m_count == 1){
        sem_wait(&menSemaphore);
}
printf("Mujer entra al baño\n");
sem_post(&full);
sem_post(&womanSemaphore);
bathroom();
sem_wait(&womanSemaphore);
sem_post(&empty);
printf("Mujer sale del baño\n");
m_count--;
if (m_count == 0){
        sem_post(&menSemaphore);
}
sem_post(&womanSemaphore);
sem_wait(&full);
pthread_exit(NULL);
```

Este algoritmo fue guiado por el que aparece en el libro guía para esta tarea:

```
femaleSwitch . lock ( empty )
        femaleMultiplex . wait ()
                # bathroom code here
        femaleMultiplex . signal ()
female Switch . unlock (empty)
```

con empty = semaphore (1)

Are there any problem with this solution? (¿Hay algún problema con esta solución?)

Esta última pregunta deja abierto a si la solución dada es correcta o no.

Aquí implementamos una solución al problema con mutex:

```
bano_u *bano = (bano_u *) datos;

//Bloqueamos la hebra para que nadie entre a modificar sus variables
      pthread_mutex_lock(&bano->mutex);

//usamos una variable de condición para evitar que una mujer entre cuando se encuentre un hombre (en el caso de hombre es viceversa)
 while (bano->hombres > 0 || bano->mujeres > 2) {
      printf("Mujer esperando....\n");
         pthread_cond_wait(&bano->lleno, &bano->mutex);
   }
   printf("Mujer entra al baño\n");
   bano->mujeres++;
   bano->total_mujeres++;
      pthread_mutex_unlock(&bano->mutex);
//Luego que fueron modificadas las variables desbloqueamos la hebra

   usar_bano(bano, 1000);
      pthread_mutex_lock(&bano->mutex);
   bano->mujeres--;
   printf("Mujer salio del baño\n");
      pthread_cond_broadcast(&bano->lleno);
      pthread_mutex_unlock(&bano->mutex);
   return NULL;
//pthread_cond_broadcast es para desbloquear la variable de condición antes mencionada
```

Para el código de los hombres es lo mismo.

Fue mucho más cómodo el saber donde estábamos equivocados y no tuvimos mayores problemas en el tema de los bloqueos y desbloqueos para proteger las variables.

Posibles problemas

Al utilizar los algoritmos del libro nos encontramos con algunos problemas en el manejo de las hebras, datos y variables, por ejemplo, la restricción estaba en que no podían haber más de 3 personas en el baño al mismo tiempo, pero aun así ingresaba un cuarto incluso un quinto empleado (depende de qué valores se les asignaba a hombre y mujer).

Al encontrarnos con este problema, implementamos un semáforo empty que tiene la cantidad de personas que puede tener simultáneamente el baño y otro semáforo “full” que iniciará en 0, éste se irá aumentando a medida que vayan ingresando al baño, contrariamente el semáforo empty irá decrementando. Esto nos permite tener control sobre la cantidad de personas que entran, ya que con la anterior solución que teníamos, solo usábamos un semáforo, el cual no garantizaba que aceptara la cantidad máxima del baño, sino que a veces podía entrar una persona de más.

Conclusión

Había que poner en práctica las habilidades de programación y poder aprender el uso de semáforos, en el transcurso de esta tarea nos dimos cuenta que nos cuesta un poco manejar con naturalidad las características de los semáforos, pero con las investigaciones para resolver el problema pudimos conocer y manejar un poco mejor los mutex, que nos dan también otras opciones para resolver otro tipo de problemas.