Práctica 3: Resolución de problemas de sincronización mediante semáforos

Programación de Sistemas Concurrentes y Distribuidos

Dpto. de Informática e Ingeniería de Sistemas,
Grado de Ingeniería Informática
Escuela de Ingeniería y Arquitectura
Universidad de Zaragoza

1. Objetivos

En esta práctica se estudiará la resolución de problemas de sincronización mediante semáforos. En concreto, los objetivos de esta práctica son:

- comprender y profundizar en la sincronización de procesos,
- resolver problemas de sincronización de procesos utilizando semáforos,
- diseñar soluciones a problemas de sincronización basadas en instrucciones de tipo `await`,
- y profundizar en el modelo de concurrencia de C++.

2. Trabajo previo a la sesión en el laboratorio

Antes de la correspondiente sesión en el laboratorio, cada pareja de estudiantes deberá leer el enunciado, analizar los problemas que en él se proponen y realizar un diseño previo de las soluciones sobre las que va a trabajar. *Los resultados de su trabajo de análisis y diseño los tendrán que expresar en un documento que presentarán a los profesores antes del inicio de la sesión.* El documento debe contener como mínimo el nombre completo y el NIP de los dos estudiantes que integran la pareja y, para cada ejercicio,

- la descripción de los datos compartidos, enumeración de los procesos que los comparten, y una explicación de las restricciones de sincronización a resolver,

- y un diseño de "grano grueso" de la solución basado en instrucciones de tipo `await`.

El documento deberá llamarse `informe_P3_NIP1_NIP2.pdf` (donde `NIP1` **es el NIP menor** y `NIP2` **es el NIP mayor** de la pareja), y deberá entregarse antes del comienzo de la sesión de prácticas utilizando el comando `someter` en la máquina `hendrix.cps.unizar.es`

```
someter progc_22 informe_p3_NIP1_NIP2.pdf
```

Además, una copia en papel de este documento deberá entregarse a los profesores al inicio de la sesión. Su entrega es un pre-requisito para la realización de la práctica.

3. Semáforos en C++

C++ no ha integrado los semáforos hasta la versión 20. La versión que utilizamos, la 11, no los tiene. Por este motivo se suministra la clase `Semaphore_V4`, correspondiente a los ficheros `Semaphore_V4.hpp` y `Semaphore_V4.cpp`. Se proporciona también un ejemplo de uso `pruebaSemaforos.cpp`.

La clase implementada tiene un único constructor `Semaphore(const int n)`. El parámetro corresponde al número de permisos que se le asocia inicialmente, equivalente al usado en la notación algorítmica utilizada en clases de teoría.

La especificación de la clase semáforo ofrece dos versiones de `wait` y dos de `signal`. La diferencia entre ellas radica en el número de permisos que están involucrados en la acción. La versión sin parámetros (`sem.signal(), sem.wait()`) se corresponde con la explicada en clase (el valor del semáforo se incrementa/decrementa en una unidad respetando las reglas semánticas de la primitiva de sincronización). La semántica de la segunda versión (`sem.signal(3), sem.wait(2)`, por ejemplo), es una generalización de la anterior, que se explica en los comentarios del fichero `Semaphore_V4.hpp`.

4. Ejercicio

En una asignatura de este centro se ha decidido hacer un examen de laboratorio por parejas, que se forman conforme van llegando. En la puerta del laboratorio hay dos sillas. Cuando un alumno llega, se sienta en cualquier silla libre (si la hay disponible). Cuando haya dos alumnos sentados, el profesor le comunica a cada uno el NIP del otro, pasan al laboratorio y empiezan a trabajar cada uno en la parte del trabajo que le corresponde. Cuando el que tenga el NIP menor de la pareja acabe le comunica a su pareja que ha terminado, así como el resultado obtenido. El de NIP mayor, una vez terminada su parte y recibido el aviso de su pareja, también acaba, muestra la solución del trabajo hecho por la pareja, y termina. Por su parte, el profesor, una vez todas las parejas se han formado, debe esperar a que todas terminen el examen, para poder darlo por concluido.

Los alumnos y el profesor van a ser representados, respectivamente, por los procesos *alumno* y *profesor* esquematizados en el anexo 6. El trabajo que hay que resolver, entre todos, es encontrar y mostrar información sobre los datos de una matriz (estos datos se encuentran en el fichero `datos.txt`). Cada pareja deberá trabajar con una fila distinta

de la matriz, y cada componente tiene una tarea distinta a desarrollar sobre su fila, como se esquematiza en el Anexo.

La práctica pide diseñar una solución al problema descrito basada en instrucciones de tipo `await` y, posteriormente, codificar el correspondiente programa concurrente en C++ aplicando la técnica de paso de testigo (véase la Lección 6 de la asignatura).

5. Entrega de la práctica

Una vez la práctica terminada, los dos componentes de la pareja deben entregar, cada uno desde su cuenta, el mismo fichero comprimido `practica_3_NIP1_NIP2.zip` (donde `NIP1` **es el NIP menor** y `NIP2` **es el NIP mayor** de la pareja) con el siguiente contenido:

1. El fichero `practica_3.cpp` con el main de programa
2. El fichero `disegno_final.pdf` con el diseño final mediante `awaits` que realmente se ha implementado. Seguramente no coincidirá con el diseño del trabajo previo, ya que es bastante probable que durante la implementación se hayan cambiado bastantes cosas.
3. El directorio `librerias` que se suministra (que, a su vez, contiene las librerías de semáforos y para generar ficheros de log)
4. El fichero `Makefile_p3` que compila el fuente, generando el ejecutable `practica_3`
5. Todos los demás ficheros requeridos para que la ejecución de `make -f Makefile_p3` genere el ejecutable pedido: `practica_3`

Generación del fichero `.zip` a entregar

Con el objetivo de homogeneizar los contenidos del fichero `.zip` vamos a proceder como sigue:

1. Creamos un directorio `practica_3_NIP1_NIP2` que contenga los ficheros que hay que entregar. Es importante tener presente que **se ha de hacer exactamente de esta manera**.
2. Con el botón derecho del ratón sobre la carpeta seleccionamos la opción "Compress..." y le damos en nombre requerido, `practica_3_NIP1_NIP2.zip`
3. Alternativamente lo podemos hacer desde la terminal como sigue. Una vez creado el directorio `practica_3_NIP1_NIP2` con los ficheros pedidos ejecutamos lo siguiente desde la terminal:

```
zip -r practica_3_NIP1_NIP2.zip practica_3_NIP1_NIP2
```

Con el fin de comprobar que el `zip` contiene todos los ficheros que debe, y organizados adecuadamente, podéis ejecutar el script `pract_3_entrega_correcta.bash`. Leed la cabecera del fichero, que explica cómo utilizarlo.

Entrega del fichero en `hendrix`

Para la entrega del fichero `.zip` se utilizará el comando `someter` en la máquina `hendrix.cps.unizar.es`

```
someter progc_22 practica_3_NIP1_NIP2.zip
```

Fechas de entrega de la práctica

La fecha de entrega depende de la fecha en que se haya tenido la sesión de prácticas:

- Las sesiones del 26 de octubre del 2022 deben entregar no más tarde del 5 de noviembre del 2022, a las 20:00
- Las sesiones del 27 de octubre del 2022 deben entregar no más tarde del 6 de noviembre del 2022, a las 20:00
- Las sesiones del 2 de noviembre del 2022 deben entregar no más tarde del 12 de noviembre del 2022, a las 20:00
- Las sesiones del 3 de noviembre del 2022 deben entregar no más tarde del 13 de noviembre del 2022, a las 20:00

Hay que asegurarse de que la práctica funciona correctamente en los ordenadores del laboratorio (hay que vigilar aspectos como los permisos de ejecución, juego de caracteres utilizado en los ficheros, etc.). También es importante someter código limpio (donde se ha evitado introducir mensajes de depuración que no proporcionan información al usuario). El tratamiento de errores debe ser adecuado, de forma que si se producen debería informarse al usuario del tipo de error producido. Además se considerarán otros aspectos importantes como calidad del diseño del programa, adecuada documentación de los fuentes, correcto formateado de los fuentes, etc.

6. Anexo con la estructura de la solución

```
...
#include <Semaphore_V4.hpp>

using namespace std;

//-------------------------------------------------------
const int N_EST = 60;       //# de estudiantes
const int N_FIL = N_EST/2; //# de filas en la matriz
const int N_COL = 1000;     //# de columnas

...

//-------------------------------------------------------
//Pre:  <fila> es un índice de fila de <D>
//Post: devuelve el máximo de la fila <fila>
int maxFila(int D[N_FIL][N_COL],int fila) {
    ...
    return max;
}

//Pre:  <fila> es un índice de fila de <D>
//Post: devuelve la suma de los els. de la fila <fila>
int sumaFila(int D[N_FIL][N_COL],int fila) {
    ...
    return sum;
}
//-------------------------------------------------------
void Estudiante(int nip,...) {

    //esperar por una silla libre
    //esperar me sea asignada pareja y fila
    if nip<miPareja {
        //calcular máx de mi fila
        //hacérselo llegar a mi pareja
    }
    else {
        //calcular la suma de mi fila
        //coger info de max (de mi pareja)
        //mostrar resultados
        //comunicar finalización
    }
}
//-------------------------------------------------------
void Profesor(...) {
    for(int i=0; i<N_FIL; i++) {
        //esperar a que haya dos
        //comunicar a cada uno su pareja, y la fila que les toca
    }

    //esperar que todos hayan terminado
}
```

```
//-----------------------------------------------------
int main() {
    int D[N_FIL][N_COL];        //para almacenar los datos
    int fila = 0;               //cada pareja cogerá una
    int pareja[N_EST];          //pareja[i] será la pareja asignada

    //cargar "datos.txt" en "D"
    ...
    cout << "Prueba finalizada\n";
    return 0;
}
```

7. Anexo con un ejemplo de la salida de una ejecución

Cada fila corresponde al esquema (`fila,pareja,max de fila,suma de fila`):

```
28  | 35-54 | 54066 | 27864374
16  | 38-58 | 67287 | 33319797
14  | 8-20  | 97345 | 48254193
15  | 13-18 | 20239 | 10079275
24  | 7-52  | 48451 | 24160829
22  | 43-44 | 29781 | 14766143
9   | 11-50 | 16174 | 8104553
2   | 4-6   | 18498 | 9428152
21  | 25-40 | 98883 | 48687570
19  | 51-57 | 811   | 394499
27  | 46-47 | 70126 | 34347937
25  | 30-37 | 24954 | 12404136
29  | 48-56 | 42221 | 21137208
4   | 23-27 | 20820 | 10673588
8   | 5-15  | 220   | 109701
13  | 16-28 | 72722 | 36232008
6   | 29-59 | 29714 | 14719067
20  | 31-45 | 16699 | 8659894
23  | 26-41 | 69414 | 35134500
0   | 0-1   | 91073 | 44221927
1   | 2-3   | 79802 | 39218017
3   | 9-10  | 90274 | 45389636
10  | 12-39 | 79401 | 39907873
17  | 21-42 | 75084 | 36773601
5   | 49-55 | 54631 | 27765634
18  | 33-34 | 53688 | 27227814
7   | 14-36 | 61052 | 30852484
11  | 22-32 | 58987 | 29700486
12  | 17-19 | 56658 | 28755390
26  | 24-53 | 50244 | 24998551


Prueba finalizada
```