Programación Concurrente y Tiempo Real

Práctica nº 2

Semáforos y Memoria Compartida

(Curso 2012/2013)

Escuela Superior de Informática

Universidad de Castilla-La Mancha

Índice

Objetivos

Objetivo general

Resolver una serie de problemas de concurrencia de **manera incremental**, empleando las **primitivas POSIX** específicas de gestión de semáforos y memoria compartida.

Objetivos específicos

El alumno deberá adquirir las habilidades para la resolución de problemas de concurrencia utilizando los **mecanismos de sincronización y comunicación entre procesos** basados en memoria compartida y semáforos.

Implementar la solución teórica obtenida mediante los mecanismos que proporciona el estándar POSIX. Estos son los llamados IPC y más concretamente la implementación de segmentos de memoria compartida y semáforos nombrados. El alumno deberá adquirir las habilidades necesarias para la creación y espera de procesos así como el correcto tratamiento de las señales básicas relativas a la finalización de la aplicación.

Especificación

Apartado A (Control concurrente de sumadores/restadores)

Realizar un control adecuado del acceso concurrente al fichero de la aplicación de los sumadores/restadores discutida en la práctica anterior, evitando condiciones de carrera. Utilice un semáforo binario para ello.

Apartado B (Integrando variables de memoria compartida)

A partir de la solución obtenida en el apartado A, sustituir el fichero de la aplicación de los sumadores/restadores, el cual contenía el valor entero que se iba actualizando de manera concurrente, por una variable de memoria compartida que gestione dicho valor.

A semejanza de la versión con el fichero, suponga que la variable compartida es un array de caracteres con un tamaño de 12 caracteres.

Apartado C (El barbero dormilón – Versión 1)

Llevar a cabo una implementación que simule el **problema clásico del barbero dormilón** haciendo uso de semáforos y variables de memoria compartida de POSIX. El enunciado de este problema clásico se expone a continuación:

La barbería tiene una sala de espera con N sillas y la habitación del barbero tiene un único sillón en el que un cliente se sienta para que el barbero trabaje. Si no hay clientes, entonces el barbero se duerme a la espera de que algún cliente lo despierte.

Cuando el barbero es despertado, entonces le corta el pelo al cliente durante un tiempo fijo S, el cual espera a que el barbero le notifique que ha terminado para poder marcharse.

Si un cliente entra en la barbería y todas las sillas están ocupadas, es decir, tanto el sillón del barbero como las sillas de la sala de espera, entonces el cliente se marcha. Si el barbero está ocupado pero hay sillas disponibles, entonces el cliente se sienta en una de ellas.

Simule la evolución de los procesos cliente y barbero, indicando en la salida estándar las acciones realizadas por cada proceso. En la última sección de este documento tiene disponible un fragmento de la salida de la ejecución de la versión 5, solicitada en esta práctica. No obstante, en **Campus Virtual está disponible un archivo de texto** con la salida completa de una posible ejecución de la versión 5, la cual se discute más adelante.

El tiempo de corte de pelo debe generarse de manera aleatoria entre 1 segundo y S segundos. Utilice la función *sleep()* para simular dicha acción.

Apartado D (El barbero dormilón – Versión 2)

En esta ocasión, la barbería tiene una **sala de espera** con N sillas y una capacidad máxima de M clientes que pueden estar en el interior de la barbería (sentados en las N sillas o esperando de pie en la sala). Si la barbería está llena (M clientes en total) y llega un cliente nuevo, este cliente no se esperará y abandonará la barbería.

Únicamente los clientes que estén sentados en las sillas podrán pasar a ocupar el sillón para que el barbero les corte el pelo. De esta forma, cuando un cliente se levante de su silla para ocupar el sillón, alguno de los clientes de los que está de pie ocupará su sitio en la silla libre.

Apartado E (El barbero dormilón – Versión 3)

La barbería cuenta, además, con una **sala donde se atiende a los clientes**. En esta sala hay tres sillones y tres barberos que cortarán el pelo a los clientes. Debe plantear una solución con el mayor nivel de paralelismo posible.

Apartado F (El barbero dormilón – Versión 4)

Cuando acaben de cortar el pelo, cada cliente pagará al barbero que le cortó el pelo una cantidad de dinero fija D, más una propia (aleatoria) entre 0 y P Euros. La barbería cuenta con una **caja compartida** para todos los barberos y clientes. El cliente depositará el dinero total en una caja, de la que el barbero extraerá únicamente la propina, dejando la cantidad fija D como beneficio de la barbería.

Al finalizar la simulación, deberá mostrar la cantidad de dinero almacenada en la caja.

Apartado G (El barbero dormilón – Versión 5)

Los tres barberos se clasifican en **lento, medio y rápido**, en función del tiempo que tardan en cortar el pelo. Se hará la siguiente distinción en el tiempo que emplea cada barbero en cortar el pelo. El primer barbero tardará exactamente *S* segundos, el segundo exactamente *2*S* segundos y el tercero exactamente *3*S* segundos. Idealmente, un cliente intentará elegir al barbero más rápido de los disponibles.

En la última sección de este documento tiene disponible un fragmento de la salida de la ejecución de esta versión. No obstante, en **Campus Virtual está disponible un archivo de texto** con la salida completa de una posible ejecución de esta versión.

Consideraciones relevantes

El alumno debe plantear y realizar una solución correcta, con el mayor grado posible de paralelismo, que coordine los barberos y sus clientes de forma adecuada utilizando semáforos y memoria compartida. Se debe definir el código de programa que debe ejecutar cada proceso cliente y los procesos barbero, de forma que puedan evolucionar, sin ningún tipo de interbloqueo.

Implementar una simulación del problema planteado mediante el uso de las primitivas que ofrece el IPC de POSIX y con los siguientes **requerimientos específicos**:

1) La simulación debe coordinar a los clientes y a los barberos correctamente.
2) Cada cliente será un proceso Unix (generado a partir de un único programa).
3) Cada barbero será otro proceso Unix.
4) Debe haber un proceso que realice las inicializaciones necesarias y lance los procesos que intervienen en la resolución del problema. Éste se encargará de lanzar inicialmente los barberos y posteriormente, en tiempos aleatorios, a los clientes.
5) El proceso que se encarga de las inicializaciones deberá pasar todas las constantes necesarias para la simulación por línea de órdenes a los procesos barbero y cliente, de forma que no será necesario su recompilación si se cambiara el nombre de algún recurso compartido o constante.
6) Cuando finalice la ejecución (porque han finalizado todos los clientes o se ha forzado la finalización mediante Control+C) el proceso principal deberá liberar todos aquellos recursos inicialmente creados.
7) Tras lanzar los procesos necesarios, se ha de mostrar información sobre qué está haciendo cada uno de ellos y sobre su estado. Para ello, consulte el fichero de texto de salida resultante de una posible ejecución de la versión 5 del problema planteado.

Sugerencias

1. **Utilice un único fichero Makefile** que albergue distintas reglas, una para cada versión de la práctica, y que automatice la compilación de las distintas versiones.
2. **Utilice, cuando sea posible, la biblioteca proporcionada** para abstraer el uso de los IPC de POSIX y que permita utilizar las operaciones *wait* y *signal* directamente, así como, articular mecanismos de alto nivel para gestionar los segmentos de memoria compartida.
3. **Consulte el fichero de salida de una posible ejecución de la versión 5** del problema de la barbería, disponible en Campus Virtual, para utilizar un formato similar a la hora de enviar mensajes a la salida estándar.
4. Como **mínimo se debe realizar hasta el apartado C**, siendo muy conveniente implementar todos los apartados.

Entrega y valoración de la práctica

- Esta práctica es **opcional**.
- **No es necesario generar documentación** de la práctica.
- Es **muy recomendable mostrar al profesor de prácticas** correspondiente la evolución de las distintas versiones de la práctica al final de las sesiones de laboratorio y en tutorías. Esta interacción podrá considerarse como **evaluación positiva en la valoración de la participación en clase**.
- La entrega del código fuente se realizará mediante una tarea en el espacio virtual de la asignatura en *Campus Virtual*. La fecha límite para subir dicho código es el 18 de Marzo.

Presentación onLine

Únicamente se subirá un fichero comprimido con tar y gzip cuyo nombre coincidirá con el dni del estudiante (ejemplo: 5987654.tar.gz). Al descomprimir el fichero, se deberá obtener la siguiente estructura de directorios:

/include	Contendrá los ficheros de cabecera .h
/src	Contendrá los ficheros fuente .c
/obj	Contendrá los ficheros objeto (se deberán crear mediante make).
/exec	Contendrá los ejectuables (se deberán crear mediante make).
/data	Contendrá los ficheros para ejecutar las pruebas.
/	El directorio raíz contendrá únicamente dos ficheros; un archivo **makefile** que deberá permitir compilar todos los programas relacionados con la práctica mediante la llamada directa a *make* y un fichero ascii (**author.txt**) que contendrá nombre y apellidos del autor.

Bibliografía básica

- [KERN91] Kernighan B. W., Ritchie D. M. - El lenguaje de programación C - Ed. Prentice Hall 1991.
- [ROCK04] Rochkind, M.J. - Advanced Unix Programming (2nd Edition) - Ed. Addison-Wesley 2004.
- [ROBB97] Robbins, K.A., Robbins, S – UNIX Programación Práctica – Ed. Prentice Hall 1997.
- [UNIX] Páginas del manual electrónico de UNIX.

Ejemplo de simulación por salida estándar

```
[Barbero 8199] Barbero <lento> durmiendo zZzZzZz
[Cliente 8200] entra en la barberia.
[Cliente 8200] ocupa sitio en la barberia (quedan 7 huecos).
[Cliente 8200] ocupa una silla libre.
[Cliente 8200] elige al barbero <rapido>.
[Cliente 8200] libera silla.
[Cliente 8200] despierta al barbero <rapido>.
[Barbero 8197] Barbero <rapido> durmiendo zZzZzZz
[Barbero 8197] Barbero <rapido> cortando el pelo
[Barbero 8198] Barbero <medio> durmiendo zZzZzZz
[Cliente 8201] entra en la barberia.
[Cliente 8201] ocupa sitio en la barberia (quedan 6 huecos).
[Cliente 8201] ocupa una silla libre.
[Cliente 8201] elige al barbero <medio>.
[Cliente 8201] libera silla.
[Cliente 8201] despierta al barbero <medio>.
[Barbero 8198] Barbero <medio> cortando el pelo
[Barbero 8197] Barbero <rapido> fin de corte
[Cliente 8200] da una propina 1 Euros a barbero <rapido>.
[Barbero 8197] Barbero <rapido> agradece la propina de 1 Euros!
[Barbero 8197] Barbero <rapido> durmiendo zZzZzZz
[Cliente 8200] se va de la barberia (quedan 7 huecos).
[Cliente 8203] entra en la barberia.
[Cliente 8203] ocupa sitio en la barberia (quedan 6 huecos).
[Cliente 8204] entra en la barberia.
[Cliente 8204] ocupa sitio en la barberia (quedan 5 huecos).
[Cliente 8204] ocupa una silla libre.
[Cliente 8203] ocupa una silla libre.
[Cliente 8204] elige al barbero <rapido>.
[Cliente 8204] libera silla.
[Cliente 8204] despierta al barbero <rapido>.
[Barbero 8197] Barbero <rapido> cortando el pelo
[Cliente 8203] elige al barbero <lento>.

***** RESTO DE SIMULACIÓN HASTA QUE TODOS LOS CLIENTES TERMINAN *****

-------------- Finalizacion de procesos -----------
Finalizando proceso [8197]... <Ok>
Finalizando proceso [8198]... <Ok>
Finalizando proceso [8199]... <Ok>
[Barbero 8197] Finalizado (SIGINT)

-------------- Liberando recursos -----------------
Recursos generales...
[Barbero 8198] Finalizado (SIGINT)
Recursos relativos a los 3 barberos...

-------------- Ganancias Finales ------------------
La barbería ha ganado 100 Euros
[Barbero 8199] Finalizado (SIGINT)
```