Práctica 3: **Comunicación entre procesos**

Abril 2018

Índice

1. Introducción

La práctica 3 se centra en aprender a utilizar los métodos más comunes de comunicación entre procesos. Hay múltiples métodos de comunicación entre procesos entre los cuales podemos mencionar las tuberías, los ficheros, la comunicación vía red o también las señales (signal).

En esta práctica utilizaremos los métodos de comunicación entre procesos para desarrollar un productor-consumidor. El problema del productor-consumidor es un ejemplo clásico de problema de sincronización de procesos. De forma general, en este paradigma de programación hay varios procesos, los productores, que "producen" información y se la "entregan" a los consumidores para que realicen alguna operación sobre ellos. El problema de sincronización en este paradigma de programación consiste en que el consumidor no intente tomar un producto si el buffer está vacío y consuma los productos del buffer una vez añadidos.

Esta práctica se basa en este paradigma, aunque se modifica (un poco) el esquema de funcionamiento para facilitar la sincronización de los productores y los consumidores.

2. La práctica

En esta práctica sólo habrá un productor y varios consumidores: el productor obtendrá la información de un fichero externo. El productor filtrará los datos a procesar y se los entregará a los consumidores, los cuales cogerán los datos filtrados y los procesaran. La comunicación entre el productor y los consumidores se realizará mediante señales y ficheros. En particular, las señales se utilizaran para sincronizar el productor y los consumidores, mientras que los ficheros se utilizaran como buffer para que el productor pueda entregar los datos a los consumidores.

2.1. Los ficheros

El productor lee los datos del fichero `data.csv`. Se trata de un fichero de texto que se ha obtenido de http://publish.illinois.edu/dbwork/open-data. El fichero original tiene 14,7 millones líneas de texto y aquí se ha recortado a 100000 líneas (99999 líneas de datos más la cabecera). El fichero contiene datos sobre viajes en taxi de la cuidad de Nueva York. Cada línea del fichero contiene los datos de un viaje y los datos éstos está separados entre sí por comas (Comman Separated Values).

El productor filtrará de este fichero de texto las columnas 8 y 9 (en concreto, el passenger_count y trip_time_in_secs) que nos indican cuantos pasajeros iba en el coche durante cada viaje y cuanto ha durado cada viaje. El objetivo final es calcular la media de pasajeros y de duración de viaje. Pero el productor no calculará esta media, sino que "delegará" este cálculo a los consumidores. De esta forma el productor únicamente se dedica a filtrar los datos mientras que los consumidores se dedican a calcular las medias.

El productor tiene un buffer de comunicación para cada consumidor que haya. El buffer de comunicación es un simple fichero de texto en el que el productor coloca los datos que el productor ha filtrado. El productor genera, para cada consumidor, bloques de N datos (cada dato se corresponda a la columna 8 y 9 de una línea el fichero de entrada) . En la sección 3 se dan más detalles al respecto, tanto respecto el nombre del fichero que se genera para cada consumidor como el "tamaño" del buffer que se genera.

Dado que puede haber múltiples consumidores, cada uno sólo calculará una media parcial de la media total de pasajeros así como de la duración del viaje. Es por ello que, cuando el productor

haya acabado de leer, filtrar y entregar todos los datos a los consumidores, estos deben ”devolver” la media parcial al productor, el cual calculará la media final.

2.2. Señales para la comunicación entre procesos

Es necesario sincronizar correctamente el productor y los consumidores para que el consumidor lea los datos cuando el productor los haya colocado. Cada vez que el productor haya enviado N datos a un consumidor, deberá notificar que los datos han sido enviados. De forma similar, cuando el productor ya no tenga más datos a enviar a los consumidores, lo deberá notificar a los consumidores. De esta forma, cada consumidor podrá calcular la media parcial de los datos que ha recibido para notificarlos al productor, el cual podrá calcular la media final.

2.3. El algoritmo a implementar

El algoritmo a implementar es éste. Supongamos que hay 1 productor y C consumidores.

1. El productor lee los datos del fichero de texto y filtra las columnas 8 y 9 de este fichero, ver sección 2.1. El productor tiene, para cada consumidor, un buffer de comunicación para enviarle los datos. Este buffer de comunicación es un fichero de disco[1].

2. El productor envía, a través del buffer, los datos a cada consumidor en bloques de N datos (cada dato se corresponde al valor de la columna 8 y 9 de una línia). Cada vez que el productor ”llena” un buffer, le envía al consumidor correspondiente la señal SIGUSR1. De esta forma se le notifica al consumidor que puede leer los datos que hay en el buffer.

3. El productor envia los datos a los consumidores de forma cíclica: comienza por enviar N datos al primer consumidor, después envia N datos al segundo, y así sucesivamente hasta llegar al último consumidor. Entonces vuelve a comenzar por el primer consumidor y así sigue el procedimiento de forma cíclica.

4. Una vez el productor ha leído todos los datos del fichero de entrada y ha colocado los datos filtrados en los buffers que correspondan, éste envía un SIGTERM a todos los consumidores. De esta forma se notifica a todos los consumidores que no habrá más datos a procesar. Los consumidores podrán calcular la media parcial de los datos que han recibido. Los datos parciales se colocaran en a través del mismo fichero que comunicación que antes se ha utilizado para transferir datos del productor al consumidor[2].

5. Una vez finalicen todos los consumidores, el productor recogerá los resultados parciales de cada consumidor y calculará la media final.

3. Implementación

Junto con esta práctica se entrega una plantilla a partir de la cual se puede comenzar a trabajar. Se recomienda utilizar esta plantilla puesto que de esta forma únicamente habrá que concentrarse

[1] El fichero se abre en modo ”append” (añadir). De esta forma el productor puede ”añadir” datos al buffer aunque el consumidor no haya tenido tiempo de leer los datos del bloque anterior. Gracias a esto se facilita mucho la sincronización de productor y consumidor.

[2] Para realizar este proceso se ”borrará” el fichero original y el consumidor generará uno nuevo en el que únicamente colocará los resultados parciales

en los puntos primordiales y básicos de esta práctica: la sincronización de procesos mediante señales y la comunicación entre procesos mediante un buffer de disco. Utilizar esta plantilla como base para vuestro programa.

Se recomienda (encarecidamente!) que los datos se transfieran en formato binario utilizando las llamadas a sistema write y read. También Se recomienda comenzar la implementación con un productor y un consumidor. Una vez funcione, se puede ampliar a varios consumidores.

3.1. Ejecución de la plantilla

En el código plantilla que se entrega sólo se crea un consumidor y un productor. El código acepta tres parámetros por línea de comandos: el nombre del fichero de texto a procesar, el numero de consumidores y el tamaño N del buffer de comunicación.

Observar que el PID del consumidor es el nombre el fichero que se utiliza como buffer de comunicación que se utiliza entre el productor y el consumidor.

Podéis compilar la práctica y comprobar el funcionamiento utilizando 'gcc':

```
$ gcc practica3.c -o practica3
$ ./practica3 datos.csv 1 10000
```

En este ejemplo el valor de N es 10000. Veréis una salida en pantalla similar a la siguiente:

```
Productor pid: 7105
Consumidor pid: 7106
Productor ha leido 10000 lineas
Productor ha leido 10000 lineas
Productor ha leido 10000 lineas
Productor ha leido 10000 lineas
Productor ha leido 10000 lineas
Productor ha leido 10000 lineas
Productor ha leido 10000 lineas
Productor ha leido 10000 lineas
Productor ha leido 10000 lineas
Productor ha leido 9999 lineas
TOTAL de lineas leidas: 0
Media de pasajeros: 0.000000 - Media de tiempo de viaje: 0.000000
```

Observar que no siempre se envían 10000 líneas al consumidor. En concreto, el último consumidor recibirá un bloque de tamaño menor. Este es un detalle que hay que tener en cuenta para la implementación del código.

3.2. Ejecución con la solución

Junto con la plantilla se entrega también el código compilado de la solución. De esta forma se puede comprobar el buen funcinamiento de la implementación. Aquí se muestra el resultado de la ejecución con un consumidor y un valor de N de 10000 líneas.

```
$ ./solucion data.csv 1 10000
```

```
El productor ha enviado 10000 lineas al consumidor 9716
El productor ha enviado 10000 lineas al consumidor 9716
El consumidor 9716 ha leido 10000 lineas
El productor ha enviado 10000 lineas al consumidor 9716
El productor ha enviado 10000 lineas al consumidor 9716
El consumidor 9716 ha leido 10000 lineas
El productor ha enviado 10000 lineas al consumidor 9716
El consumidor 9716 ha leido 10000 lineas
El productor ha enviado 10000 lineas al consumidor 9716
El productor ha enviado 10000 lineas al consumidor 9716
El consumidor 9716 ha leido 10000 lineas
El productor ha enviado 10000 lineas al consumidor 9716
El consumidor 9716 ha leido 10000 lineas
El productor ha enviado 10000 lineas al consumidor 9716
El productor ha enviado 9999 lineas al consumidor 9716
El consumidor 9716 ha leido 10000 lineas
El consumidor 9716 ha leido 10000 lineas
El consumidor 9716 ha leido 10000 lineas
El consumidor 9716 ha leido 10000 lineas
El consumidor 9716 ha leido 9999 lineas

El consumidor 9716 ha leido 99999 lineas en total
Media parcial de pasajeros: 1.891539 - Media parcial del tiempo de viaje: 772.211731

Productor: en total se han leido 99999 lineas
Media de pasajeros: 1.891539 - Media del tiempo de viaje: 772.211731
```

4. Entrega y cualificación

Cada grupo de prácticas tiene que entregar un fichero que contenga todos los ficheros necesarios para compilar el código. El nombre del fichero debe indicar los componentes de la pareja, por ejemplo, `GarridoLluis_PetruzziPatricio.zip`.

En caso que se realize la implementación con un sólo consumidor, la cualificación máxima que se puede obtener es un 8. Si la implementación permite múltiples consumidores, será un 10.