CI-0117 Laboratorio 5 Paralela y Concurrente
Rodrigo Vilchez Ulloa | B78292

---

2. PiPorSeries con Buzones

• Se modificó la clase Buzon, que fue creada para un laboratorio anterior, para que fuera capaz de recibir la información enviada por los procesos encargados de calcular una parte de la suma para calcular pi. Los procesos enviaban un struct que contenía una variable long (necesaria para el paso de mensajes) y una variable double que era el resultado de la suma parcial de pi que se había calculado en ese proceso. Mediante un objeto creado de la clase Buzon, el proceso utilizaba el método Enviar, el cual se encargaba de recibir el struct enviado por el proceso, y luego de hacer el llamado a msgsnd, donde ya el objeto tenía el id de la cola de mensajes, obtenido cuando este fue creado. Por último, el método main llama al método Recibir de la clase, el cual se encarga de hacer el llamado a msgrcv, para leer los struct enviados por los procesos y retornar una copia al main, donde se guarda esa información para posteriormente usar el valor del double calculado por los procesos para sumarlos a la variable de resultado. Utilizar el comando **make** para compilar el programa. Y correrlo con el comando **./PiPorSeries**

• Demostración

```
rigovil@rodrigo:~/Escritorio/UCR/I Semestre 2020/CI-0117/Practicas/Semana 5/PiPorSeries - Buzones$ make
g++ -c -g PiPorSeriesConMensajes.cc
g++ -c -g Buzon.cc
g++ -g PiPorSeriesConMensajes.o Buzon.o -o PiPorSeries
rigovil@rodrigo:~/Escritorio/UCR/I Semestre 2020/CI-0117/Practicas/Semana 5/PiPorSeries - Buzones$ ./PiPorSeries
Creando el proceso 29448: termino inicial 0, termino final 100000
Creando el proceso 29449: termino inicial 100000, termino final 200000
Creando el proceso 29450: termino inicial 200000, termino final 300000
Creando el proceso 29451: termino inicial 300000, termino final 400000
Creando el proceso 29452: termino inicial 400000, termino final 500000
Creando el proceso 29453: termino inicial 500000, termino final 600000
Creando el proceso 29454: termino inicial 600000, termino final 700000
Creando el proceso 29455: termino inicial 700000, termino final 800000
Creando el proceso 29456: termino inicial 800000, termino final 900000
Creando el proceso 29457: termino inicial 900000, termino final 1000000
Resultado parcial recibido     3.141582654
Resultado parcial recibido           5e-06
Resultado parcial recibido 1.666666667e-06
Resultado parcial recibido 8.333333333e-07
Resultado parcial recibido           5e-07
Resultado parcial recibido 3.333333333e-07
Resultado parcial recibido 2.380952381e-07
Resultado parcial recibido 1.785714286e-07
Resultado parcial recibido 1.388888889e-07
Resultado parcial recibido 1.111111111e-07
Valor calculado de Pi es      3.141591654  con 1000000 terminos
rigovil@rodrigo:~/Escritorio/UCR/I Semestre 2020/CI-0117/Practicas/Semana 5/PiPorSeries - Buzones$
```

3. PiPorSeries con hilos en Java

• En Java, lo que se hace es crear un vector tipo Thread con tamaño que depende de la cantidad de hilos que se quieran crear, y a cada celda se le asigna un "objeto tipo Thread", enviándosele como parámetro, en este caso, otro objeto que implementa la función run que es la que corre el hilo cuando comienza a operar cuando se le llama con el comando start. Por último, cada hilo finaliza con el llamado a join. La clase Suma está para almacenar el valor calculado por cada hilo. Se crea en el main un vector de objetos de esta clase según la cantidad de hilos que se vayan a crear, y se envía uno por referencia a cada objeto de la clase SeriePorPartes, que tiene como atributo a un objeto de la clase Suma. Cada vez que un hilo termine de calcular su parte de Pi, guardará el valor en su atributo de la clase Suma y por ende en el vector de objetos de la misma clase, que al final se utiliza para hacer una sumatoria con el resto de cálculos de los otros hilos.

Se pudo implementar una solución al problema de resolver piPorSeries utilizando un vector de objetos Doubles, se utilizó prácticamente el mismo programa con algunos cambios: se eliminó la clase Suma, por tanto el vector de objetos del main pasó a ser un vector de objetos Double, estos no se deben inicializar pues el valor almacenado en el objeto no puede ser modificado después, por lo que se debe crear el objeto una vez que el proceso haya calculado el resultado. Segundo, el constructor de la clase SeriePorPartes, recibe la referencia de ese vector de Doubles, así como del índice que indica cuál es la celda del vector que le corresponde a un proceso en particular, y una vez que el proceso haya calculado su parte de pi, este inicializará el objeto en la celda del vector correspondiente. Correr el programa con el comando **java PiPorSeriesThreadsDouble**.

¿Es posible hacer el cálculo con solo un objecto "Double"? La respuesta en un principio es no, pues los objetos Double, una vez que se les inicializa con un valor, va a conservarlo y por tanto es inmutable, por lo que no se podrá actualiazar su valor para sumar cada valor proporcionado por los procesos, a no ser que se cree el objeto Double una vez que se logre obtener el resultado final de alguna forma.

• Demostración

```
rigovil@rodrigo:~/Escritorio/UCR/I Semestre 2020/CI-0117/Practicas/Semana 5/PiPorSeries Java$ javac PiPorSeriesThreadsDouble
Note: PiPorSeriesThreadsDouble.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
rigovil@rodrigo:~/Escritorio/UCR/I Semestre 2020/CI-0117/Practicas/Semana 5/PiPorSeries Java$ java PiPorSeriesThreadsDouble
Calculo de PiPorSeries utilizando un vector de objetos Double

Valor calculado de PI: 3.1415826535897846
rigovil@rodrigo:~/Escritorio/UCR/I Semestre 2020/CI-0117/Practicas/Semana 5/PiPorSeries Java$
```

Figure 1: PiPorSeries utilizando vector de Doubles.

4. SumarUno

• Para medir el tiempo, lo que se hace es utilizar la función clock que lo que hace es contar la cantidad de tics del reloj que han pasado desde un cierto momento y luego se divide entre la constante CLOCKS PER SEC para encontrar la cantidad de segundos que tardó el CPU en ese programa o en ciertos momentos del programa. El programa SumaUno efectivamente tiene problemas de race condition, pues varios procesos intentan leer y modificar al mismo tiempo la variable donde se guarda el resultado final, lo que deja que ciertas sumas, por más que se hayan realizado, no se reflejan en el resultado final. Por lo que, para solucionarlo, se implementó el uso de semáforos en el otro programa, y cada vez que un proceso va a entrar a la función que suma uno mil veces al resultado en la variable de memoria compartida, tiene que esperar a que no haya otro proceso ejecutando la misma función y a que este "libere" el semáforo para que otro proceso pueda acceder, así hasta que todos los procesos hayan ejecutado esa función.

Por último, se utilizó la clase semáforo utilizada en trabajos anteriores para implementar un programa similar que sume uno a una variable mil veces por proceso. Se utilizaron hilos y una variable global que almacena el resultado, cuando un proceso va a añadir uno mil veces, debe esperar a que el semáforo se lo permita. Compilar el programa con **make** y correrlo con **./SumaUnoSem**.

• Demostración

```
rigovil@rodrigo:~/Escritorio/UCR/I Semestre 2020/CI-0117/Practicas/Semana 5/SumaUno$ ./a.out
Serial version:      Valor acumulado es  100000  con 100 procesos in 0.000509 seconds
Fork, Race Cond.:    Valor acumulado es  97541  con 100 procesos in 0.009573 seconds
```

Figure 2: Corriendo el programa dándose race condition.

```
rigovil@rodrigo:~/Escritorio/UCR/I Semestre 2020/CI-0117/Practicas/Semana 5/SumaUno$ ./a.out
Serial version:      Valor acumulado es  100000  con 100 procesos in 0.000510 seconds
Fork, Race Cond.:    Valor acumulado es  99316  con 100 procesos in 0.006353 seconds
Fork, No Race Cond.: Valor acumulado es  100000  con 100 procesos in 0.005242 seconds
```

Figure 3: Corriendo el programa sin darse el race condition.

```
rigovil@rodrigo:~/Escritorio/UCR/I Semestre 2020/CI-0117/Practicas/Semana 5/SumaUno$ make
g++ -c -g SumaUnoSem.cc -lpthread
g++ -c -g Semaphore.cc
g++ -g SumaUnoSem.o Semaphore.o -o SumaUnoSem -lpthread
rigovil@rodrigo:~/Escritorio/UCR/I Semestre 2020/CI-0117/Practicas/Semana 5/SumaUno$ ./SumaUnoSem
Resultado: 100000rigovil@rodrigo:~/Escritorio/UCR/I Semestre 2020/CI-0117/Practicas/Semana 5/SumaUno$
```

Figure 4: Corriendo el programa con la clase Semáforo.