UNIVERSIDAD DE SANTIAGO DE CHILE
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA INFORMÁTICA

LABORATORIO Nº2
SISTEMAS OPERATIVOS
Sincronización de hebras con pthreads

Profesor: Cristobal Acosta
Fernando Rannou
Ayudantes: Luis Loyola
Gabriel Godoy

Fecha de entrega: 18/01/2016 hasta las 23:55
luis.loyola@usach.cl
gabriel.godoy@usach.cl

ENUNCIADO

En “Hebralandia” grupos de hebras han tenido problemas para determinar cuál grupo es más eficiente. Para esto un señor proceso el cual tiene problemas para interceptar listas a propuesto hacer una competencia. El señor proceso ha propuesto que el grupo que logre determinar la intersección de un grupo de listas en menor tiempo será el ganador.

Se tendrán N listas en un archivo de entrada pasado como argumento al programa con el siguiente formato:

N1 n_1_1 n_1_2 n_1_3 … n_1_N1

N2 n_2_1 n_2_2 n_2_3 … n_2_N2

…

Con:

- Ni: número de términos de la lista. Los términos en cada lista se encuentran ordenados numéricamente.
- n_i_j: elemento j de la lista i

Ejemplo:

```
4 2 6 9 20
8 1 3 5 9 12 17 19 20
3 7 9 20
```

La finalidad de esta competencia es encontrar al grupo de hebras que ejecute la intersección de las N listas en menor tiempo posible. Para esto fue elaborado el siguiente algoritmo:

1. Todos los grupos deben leer el archivo “listas.dat”. (Se debe esperar que todos los grupos lean el archivo).

2. Sea “S” la lista más corta del ordenamiento.

3. Mientras queden listas hacer:

    a. Sea K la siguiente lista del ordenamiento.

    b. Realizar la intersección entre S y K paralelizando la rutina. Es aquí donde las hebras deben participar en conjunto. La lista S debe ser compartida por todas las hebras y el acceso a los distintos elementos de S debe ser exclusivo. Sea S de largo |S| y K de largo |K|, entonces cada hebra tiene acceso a |S| elementos de S (la lista más pequeña), y a |K|/P de K (lista más larga). Sea P el número de hebras del equipo. El número de elementos de K que se reparten debe ser consecutivo para cada hebra, es decir, los primeros |K|/P elementos de K a la hebra 0, los segundos |K|/P elementos a la hebra 1, etc. El algoritmo a aplicar como equipo es el siguiente:

        i. Sea S’ una lista vacia.

        ii. Ordenar las listas locales k (sólo lo que cada hebra puede visualizar) por número de documentos en orden creciente.

        iii. Por cada elemento de S (recorrido secuencial O(S)), hacer búsqueda binaria en los elementos de K que pueda visualizar la hebra (búsqueda binaria O ((K/P) log (K/P)).

        iv. Si se encontró el documento revisado en la lista K, entonces agregar el

elemento a S’ (bloqueando la lista S’ para acceso exclusivo).

c. Liberar la memoria usada por S y ahora S=S’

d. Volver a 3.

El resultado de la prueba será almacenada en el archivo “resultado.txt” y tendrá el siguiente formato:

“Número del equipo que obtuvo el primer lugar:” Ni “Tiempo del equipo que obtuvo el primer lugar:” Ti “Número del equipo que obtuvo el segundo lugar:” Nk “Tiempo del equipo que obtuvo el segundo lugar:” Ti “Número del equipo que obtuvo el tercer lugar:” Nj “Tiempo del equipo que obtuvo el tercer lugar:” Ti “Hebra más eficiente:” ni “Hebra más eficiente en promedio:” nk “Intersección de las listas:” $S_1 S_2 S_3 \ldots S_p$

Donde:

Ni: Equipo que determine la intersección de listas en menor tiempo.

Ti: Tiempo que demoro en obtener la intersección de listas el equipo ‘i’.

ni: Hebra que procese su parte de la lista K más rápido.

nk: Hebra que en promedio de todos sus procesamientos de la lista K demore menor tiempo.

ASPECTOS A CONSIDERAR

- El programa debe ejecutarse de la siguiente forma:

```
./competencia -g Ei -h Hi -i inputfile
```

Donde:

- Ei: Representa la cantidad de equipos de threads de la competencia (siempre después de

la opción -g).

- Hi: Representa la cantidad threads por equipo (siempre después de la opción -h).
- inputfile: Representa la ruta relativa o absoluta al archivo de entrada (siempre después de la opción -i).
- Los parámetros pueden estar en cualquier orden (buscar cómo utilizar la función *getopt()*).

- Las sincronizaciones deben ser realizadas mediante semáforos binarios y variables de condición.
- El bloqueo de la lista S (la lista más corta) se realizara solo para leer un elemento, y posteriormente se desbloqueara la lista, en caso contrario se limitaría la concurrencia.
- Los grupos de hebras deben comenzar al mismo tiempo.
- Considerar las listas extensas, incluso la más pequeña.
- En cualquier paso del algoritmo la intersección S puede ser vacía por lo que el algoritmo de intersección debe detenerse.
- Las listas no son necesariamente múltiplos de P.
- La cantidad de listas también es extensa.

ENTREGABLES

- Código fuente: Identado y comentado, en caso de utilizar C++, por favor utilizar realmente C++, es decir orientado a objetos.
- Makefile: Con automatización de la compilación, el makefile **no debe ejecutar el programa sólo debe compilar**.
- README: Con las instrucciones para compilar, ejecutar, nombres de autores, ciertos problemas detectados en su solución.

Debe comprimir todos los archivos indicados anteriormente en formato tar, zip o rar.

Debe indicar en el nombre del archivo a subir el RUT de los integrantes.

La fecha de entrega es la indicada en la primera plana del enunciado, después de eso se aplicará descuento de 1 punto por día de atraso. En caso de no poder subir su trabajo, enviar por mail con el asunto "Sistope Lab2 - 2015.2".