Sistemas Operativos 2/2024
Laboratorio 3

Profesores:
Cristóbal Acosta (cristobal.acosta@usach.cl)
Fernando Rannou (fernando.rannou@usach.cl)
Miguel Cárcamo (miguel.carcamo@usach.cl)
Ayudantes:
Ricardo Hasbun (ricardo.hasbun@usach.cl)
Daniel Calderón (daniel.calderon.r@usach.cl)

December 5, 2024

1 Objetivos Generales

Este laboratorio tiene como objetivo aplicar técnicas de programación imperativa mediante lenguaje C, como la recepción de parámetros mediante `getopt` y compilación mediante `Makefile` sobre sistema operativo Linux. Además de aplicar conocimiento adquiridos en cátedra sobre Concurrencia, Multihebrado, y llamados al SO Linux de manera exitosa.

2 Objetivos Específicos

1. Usar las funcionalidades de `getopt()` como método de recepción de parámetros de entradas.
2. Uso del Makefile para compilación por partes de programas.
3. Utilizar recursos de pthread para sincronizar hebras y proteger recursos compartidos.
4. Implementar soluciones de exclusión mutua por software.
5. Construir funciones de lectura y escritura de archivos.
6. Practicar técnicas de documentación de programas.

3 Enunciado

Encontrar valores máximos o mínimos en arreglos de gran dimensión es crucial en diversas áreas debido a su importancia para la toma de decisiones, la optimización de recursos y el análisis de datos. Estos valores extremos son fundamentales para identificar puntos de referencia clave, como el mejor rendimiento, el costo más bajo o los niveles críticos en fenómenos naturales, médicos o financieros. Además, en entornos de gran escala, el desafío radica en diseñar algoritmos eficientes que minimicen el uso de tiempo y memoria, empleando estrategias como paralelización. Esta operación es esencial para resolver problemas complejos, generar insights valiosos y garantizar decisiones informadas en ciencia, industria y tecnología.

En el contexto de este laboratorio, se explorará el uso de hebras, secciones críticas y mecanismos de sincronización, con el fin de resolver un problema de comparación de valores dentro de un arreglo, donde se busca encontrar el máximo valor.

3.1 Descripción del problema

Dado un arreglo de números enteros, de largo N, existe el problema de conocer cuál es el máximo valor que posee dicha colección de valores. Un enfoque de solución está dado por la utilización de hebras que permitan paralelizar el proceso de comparación de los valores del arreglo en cuestión. De esta forma, se debe crear una cantidad de hebras igual a la cantidad de posiciones que tenga el arreglo. En otras palabras, si el arreglo tiene 10 posiciones, entonces se deben crear 10 hebras.

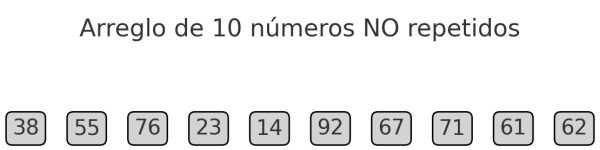

Figure 1: Arreglo con 10 números. Donde el arreglo es circular.

Para esta aplicación, los números en el arreglo serán únicos. Es decir, no hay repeticiones.

4 El programa

4.1 Lógica de la solución

1. El programa recibe, por línea de comando, el nombre del archivo que contiene los valores a estudiar, con el fin de crear un arreglo A de tamaño igual a la cantidad de números que tenga el archivo.

2. A continuación, se crea una cantidad de hebras igual al largo del arreglo A.

3. Cada hebra i almacenará en una variable local `myval`, el valor inicial `A[i]`.

4. Cada hebra i solo puede acceder a las posiciones i e $i-1$, y el arreglo se trata en forma circular.

5. Repetidamente, cada hebra i, compara `A[i-1]` con `myval`. Si `A[i-1]` es mayor que `myval`, entonces copia `A[i-1]` en `A[i]`.

6. Este proceso continúa, asíncronamente y concurrentemente, hasta que el valor máximo se haya propagado por todo el arreglo. Una vez suceda esto, el programa finaliza. Usted debe diseñar una estrategia eficiente de cómo detectar el fin de la iteración.

7. Al finalizar, el máximo estará en todas las posiciones del arreglo.

8. Sería muy ineficiente, en cada momento preguntar si el número es el mismo en todas las posiciones.

Linea de comando del programa:

```
$ ./lab3 -i input.txt -o output.txt -D
```

Donde cada uno de los argumentos de la linea de comandos es:

- `i`: archivo de entrada con los valores.
- `o`: archivo de salida.
- `D`: flag para mostrar por stdout el arreglo al finalizar.

4.2 Requerimientos

Como requerimientos no funcionales, se exige lo siguiente:

- Debe funcionar en sistemas operativos con kernel Linux.
- Debe ser implementado en lenguaje de programación `C`.
- Se debe utilizar un archivo `Makefile` para compilar los distintos targets.
- Realizar el programa utilizando buenas prácticas, dado que este laboratorio no contiene manual de usuario ni informe, es necesario que todo esté debidamente comentado.
- Los programas se encuentren desacoplados, es decir, que se desarrollen las funciones correspondientes en otro archivo .c para mayor entendimiento de la ejecución.

5 Entregables

El laboratorio es en parejas. Si se elije una pareja, esta no podrá ser cambiada durante el semestre. Se descontará 1 punto (de nota) por día de atraso con un máximo de tres días, a contar del cuarto se evaluará con nota mínima. Debe subir en un archivo comprimido ZIP (una carpeta) a USACH virtual con, al menos, los siguientes entregables:

- `Makefile`: Archivo para compilar los programas.
- `lab3.c`: Archivo principal del laboratorio, contiene el main. Se concentra en obtener y validar los datos entregados como argumentos del programa. Si los datos son validos, debe abrir el archivo de entrada, crear y esperar por las hebras.
- `funciones.c y funciones.h` : Archivo con funciones, en el .c el desarrollo de la función y en el . h las cabeceras. Recuerde que todas las funciones deben estar comentadas, explicadas de forma entendible especificando sus entradas, funcionamiento y salida. Si una función no esta explicada se bajara puntaje.

Recuerde que todas las funciones deben estar comentadas, explicadas de forma entendible especificando sus entradas, funcionamiento y salida. Si una función no está explicada se bajará puntaje. Se deben comentar todas las funciones de la siguiente forma:

```
// Entradas: explicar qué se recibe
// Salidas: explicar qué se retorna
// Descripción: explicar qué hace
```

El archivo comprimido (al igual que la carpeta) debe llamarse:
RUTESTUDIANTE1_RUTESTUDIANTE2.zip
Ejemplo 1: 19689333k_186593220.zip

- **NOTA 1**: El archivo debe ser subido a uvirtual en el apartado "Entrega Laboratorio N°3".

- **NOTA 2**: Cualquier diferencia en el formato del laboratorio que es entregado en este documento, significará un descuento de puntos.
- **NOTA 3**: SOLO UN ESTUDIANTE DEBE SUBIR EL LABORATORIO.
- **NOTA 4**: En caso de solicitar corrección del laboratorio, esta será en los computadores del Diinf, es decir, si funciona en esos computadores no hay problema.
- **NOTA 5**: Cualquier comprimido que no siga el ejemplo 1, significará un descuento de 1 punto de nota.

6 Fecha de entrega

JUEVES 19 DE DICIEMBRE DEL 2024.