Universidad De San Carlos De Guatemala
Facultad De Ingeniería
Escuela De Ciencias Y Sistemas
Sistemas Operativos 2 Sección A
Ing. Edgar René Ornelis Hoil
Aux. Derek Esquivel Díaz
Vacaciones Junio 2024

Practica 2

Multithreading

Objetivos

- Comprender como funcionan los hilos en Linux.
- Comprender los conceptos de concurrencia y paralelismo.
- Aplicar conceptos de sincronización de procesos.

Descripción

Las corporaciones utilizan computadoras de alto rendimiento llamados mainframes para aplicaciones que dependen de la escalabilidad y la confiabilidad. Por ejemplo, una institución bancaria podría utilizar un mainframe para albergar la base de datos de las cuentas de sus clientes, para las cuales se pueden enviar transacciones desde cualquiera de los miles de cajeros automáticos en todo el mundo.

La práctica consiste en realizar una aplicación en consola en C que permita almacenar los datos de usuario de un banco, así como poder realizar operaciones monetarias como depósitos, retiros y transacciones.

Especificaciones

Carga Masiva de Usuarios

Al iniciar la ejecución de la aplicación se podrá realizar una carga masiva de usuarios. Esta se realizará por medio de un archivo JSON proporcionado por el usuario por medio de la ruta del archivo, este tendrá la siguiente sintaxis:

```
[
    {
      "no_cuenta": 56913,
      "nombre": "Justen Fawkes",
      "saldo": 83474.3
    }, {
      "no_cuenta": 68304,
      "nombre": "Marilin Ferreira",
      "saldo": 25456.35
    }, {
      "no_cuenta": 20918,
      "nombre": "Peirce Oxtiby",
      "saldo": 65541.57
    }, {
      "no_cuenta": 53171,
      "nombre": "Daile Riddle",
      "saldo": 6129.11
    }
]
```

Donde:

- **no_cuenta:** Este es el numero de cuenta del usuario y deberá de ser único en el sistema.
- **nombre:** Nombre del usuario.
- **saldo:** Saldo actual que posee la cuenta.

La carga de estos datos deberá de realizarse en 3 hilos de forma **paralela**.

Durante la carga el sistema deberá de ser capaz de capturar los registros con error, omitirlos y listarlos en un reporte (ver sección de Reportes).

Operaciones Individuales

La aplicación poseerá un menú donde se le permitirá al usuario realizar una operación, estas operaciones serán:

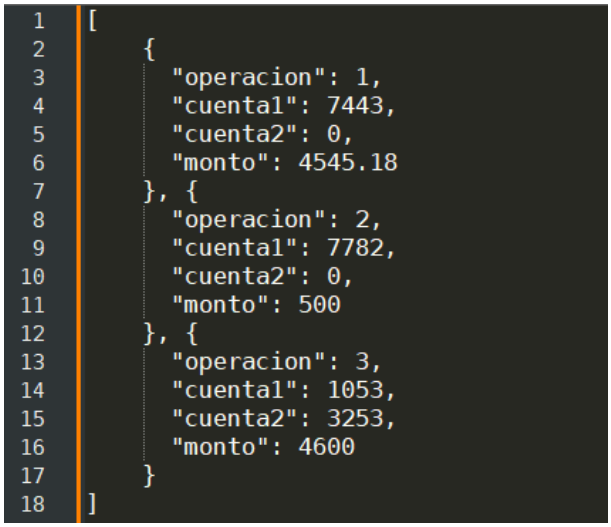
- **Deposito:** Se le solicitará al usuario el numero de cuenta y el monto a depositar.
- **Retiro:** Se le solicitará al usuario el numero de cuenta y el monto a retirar.
- **Transacción:** Se le solicitará al usuario el número de cuenta de la cuenta donde se retirará, el número de cuenta de la cuenta donde se depositará y el monto a transferir.
- **Consultar cuenta:** Se le solicitará al usuario el numero de cuenta y se deberá devolver la información de la cuenta.

El sistema deberá de mostrar un error en caso de que:

- No exista el número de cuenta
- La cuenta no tenga saldo suficiente para ejecutar la operación
- El monto indicado no es un numero o es menor a 0

Carga Masiva de Operaciones

Debido a que los mainframes no realizan solo una operación a la vez se realizara una carga masiva de operaciones que el sistema deberá de realizar de manera **concurrente** en 4 hilos. Esta carga se realizará por medio de un JSON con la siguiente sintaxis:



```
[
    {
      "operacion": 1,
      "cuenta1": 7443,
      "cuenta2": 0,
      "monto": 4545.18
    }, {
      "operacion": 2,
      "cuenta1": 7782,
      "cuenta2": 0,
      "monto": 500
    }, {
      "operacion": 3,
      "cuenta1": 1053,
      "cuenta2": 3253,
      "monto": 4600
    }
]
```

Donde:

- **operacion:** Esta será la operación que se realizará, estas pueden ser:
  - 1 -> Deposito
  - 2 -> Retiro
  - 3 -> Transferencia
- **cuenta1:**
  - Deposito: Si la operación es ‘deposito’ está será la cuenta donde se depositará el dinero.
  - Retiro: Si la operación es ‘retiro’ está será la cuenta de donde se retirará el dinero.
  - Transferencia: Si la operación es ‘transferencia’ esta es la cuenta de donde se retirará el monto a transferir.
- **cuenta2:**
  - Deposito: Si la operación es ‘deposito’ no se utiliza este campo.
  - Retiro: Si la operación es ‘retiro’ no se utiliza este campo.
  - Transferencia: Si la operación es ‘transferencia’ esta es la cuenta de donde se depositará el monto transferido.
- **monto:** Cantidad de dinero que será retirado, depositado o transferido.

Es importante que para la realización de estas operaciones se utilicen técnicas de sincronización para evitar la corrupción de los valores y así asegurar la veracidad de los datos.

Durante la carga el sistema deberá de ser capaz de capturar los registros con error, omitirlos y listarlos en un reporte (ver sección de Reportes).

Reportes

Para una empresa como un banco es importante tener reportes sobre la ejecución para que se puedan asegurar del correcto funcionamiento del sistema, los reportes que se solicitaran son:

Estado de cuentas

Desde el menú, un operador podrá generar un reporte que mostrara la información de los usuarios, este será escrito en un JSON y deberá mostrar la siguiente información:

```json
[
    {
      "no_cuenta": 20589,
      "nombre": "Falito Byron",
      "saldo": 64100.22
    }, {
      "no_cuenta": 42264,
      "nombre": "Dell Chalker",
      "saldo": 61142.2
    }, {
      "no_cuenta": 82571,
      "nombre": "Stafford Greenier",
      "saldo": 29018.35
    }
]
```

Donde:

- **no_cuenta:** Este es el número de cuenta del usuario y deberá de ser único en el sistema.
- **nombre:** Nombre del usuario.
- **saldo:** Saldo actual que posee la cuenta.

Reporte de carga de usuarios

El reporte de carga se realizará de manera **automática** al terminar la carga masiva de usuarios en un archivo llamado "*carga_ yyyy_MM_dd-HH_mm_ss .log",* teniendo la fecha en la que se generó el reporte. El contenido del reporte es:

```
---------- Carga de usuarios ----------

Fecha: yyyy-MM-dd HH:mm:ss

Usuarios Cargados:
Hilo #1: 17
Hilo #2: 20
Hilo #3: 12
Total: 49

Errores:
    - Linea #4: Numero de cuenta duplicado
    - Linea #12: Saldo no puede ser menor a 0
    ...
```

Primero se deberá de escribir un desglose de cuantos usuarios cargo en el sistema cada hilo, así como el total de usuarios cargados.

Luego se deberán listar los errores en los datos, estos errores pueden ser:

- Números de cuenta duplicados
- Si número de cuenta no es un numero entero positivo
- Si el saldo introducido no es un número real positivo

Reporte de carga de operaciones masivas

Cuando se realice una carga masiva de operaciones se realizará de manera **automática** un reporte de las operaciones realizadas en un archivo llamado *"operaciones_ yyyy_MM_dd-HH_mm_ss .log"*, teniendo la fecha en la que se generó el reporte. El contenido del reporte es:

```
---------- Resumen de operaciones ----------

Fecha: yyyy-MM-dd HH:mm:ss

Operaciones realizadas:
Retiros: 50
Depositos: 33
Transferencias: 67
Total: 150

Operaciones por hilo:
Hilo #1: 55
Hilo #2: 35
Hilo #3: 20
Hilo #4: 40
Total: 150

Errores:
    - Linea #8: Saldo insuficiente
    - Linea #56: Número de cuenta no existe
    - Linea #99: Operación no valida
    ...
```

Primero se deberá de escribir un desglose de las operaciones realizadas, así como el total de operaciones realizadas por cada hilo.

Luego se deberán listar los errores en las operaciones, estos errores pueden ser:

- Números de cuenta no existe
- Si el monto no es un numero o es un número menor a 0
- El identificador de la operación no existe
- Si la cuenta no tiene el saldo suficiente para ejecutar la operación

Observaciones

- La práctica se realizará en parejas.
- El lenguaje de programación a utilizar será C.
- Para la realización de multithreading se utilizará la librería pthreads.
- Cualquier copia parcial o total será reportada a la Escuela de Ciencias y Sistemas para que proceda como indica el reglamento.

Entregables

- Código fuente.
- Manual técnico con explicación de lo realizado en la práctica. Este debe ser un archivo .md (Markdown) y será colocado como el README de la carpeta.

Forma de entrega

- Utilizar un repositorio de GitHub, el cual debe ser privado con el nombre: **SO2_GRUPO<<no. grupo>>.**
- Crear la carpeta "**Practica 2**"
- Agregar al auxiliar al repositorio: **Desquivel501**
- La entrega se realizará por medio de UEDI en el apartado correspondiente, donde el estudiante subirá un archivo de texto con el link a su repositorio.

La entrega se debe realizar el 16 de junio de 2024 antes de las 23:59.