Clique em A tutorial on modern multithreading and concurrency in C++ para abrir o recurso

Gettimeofday: benchmarking - Execution time of C program - Stack Overflow

Time command in Linux: How to use time command on Linux - Linux Tutorials - Learn Linux Configuration

Clique em Mutex in C++ para abrir o recurso.

Clique em C++11 Threads, Locks and Condition Variables para abrir o recurso

Experimentação de conta corrente

Neste experimento, construa alguns programas usando std::thread (C++) que reproduzam o problema clássico de condição de corrida no contexto de uma conta corrente e uma respectiva solução.

Garanta que os resultados entre uma solução com condição de corrida seja diferente de uma solução sem condição de corrida.

Preparação:

Construa um programa que seja seu *baseline* para as próximas fases do experimento. Esse programa deve se assemelhar a:

```
main() {

  ....

  saldo = 1000.00;

  depositos(); // realiza uma "infinidade" de depositos, e.g. 2147483000 depositos de 5.0 unidades monetárias

  saques(); // realiza uma "infinidade" de saques, e.g. 2147483000 saques de 2.0 unidades monetárias
```

```
  printf("Saldo final: %.2lf\n", saldo);

  ....

}
```

Fase 1

Construa uma outra versão da solução anterior usando std::thread (C++) e que tenha condição de corrida no saque/deposito da conta corrente. Nesta solução, o resultado do saldo final não será igual a versão *baseline*. Exemplo:

```
main() {

  ....

  saldo = 1000.00;

  std::thread(..., depositos, ...); // a mesma funcao depositos() anterior

  std::thread(..., saques, ...); // a mesma funcao saques() anterior

  ....

  printf("Saldo final: %.2lf\n", saldo);

  ....

}
```

PS. se o saldo final **for igual** ao resultado do experimento da preparação, a Fase 1 de seu experimento não foi feita corretamente: troca os valores usados como saque/depósito; e/ou aumente a quantidade de "infinidade" de saques/depósitos

Fase 2

Construa uma nova solução da versão anterior resolvendo o problema da condição de corrida com seção crítica (mutex). Garanta que o saldo final apresentado nesta fase do experimento seja igual ao *baseline* e, consequentemente, diferente do resultado apresentado na Fase 1.

Construa uma versão **inteligente** desta solução - raciocine bastante antes de construir sua solução.

Entregas

Entregue as 3 versões desenvolvidas e as respectivas evidências de execuções (eg. print de tela da execução).