Universidade Federal do Rio de Janeiro 2020.1(REMOTO)
Computação concorrente - Silvana Rossetto

Aluno: Lucas Tatsuya Tanaka 118058149

Questão 1:

- **A)**A principal característica de um programa concorrente seria a mudança de paradigma de programação, na qual ocorre a troca do ideário da utilização apenas de uma Thread para a troca do uso de múltiplas Threads no programa visando o aproveitamento da arquitetura moderna de processadores.

- **B)**A aceleração máxima será de 2,5 caso sejam utilizados 4 Threads para o processamento da tarefa, caso seja utilizado mais Threads o tempo proposto de aceleração será maior de acordo com a lei de Amdahl.

- **C)**Seção crítica seria o trecho do código concorrente na qual deve ser impedido o surgimento de resultados indesejáveis devido ao acesso de múltiplas Threads a mesma parte do código.

- **D)**A sincronização por exclusão mútua seria um dos diversos mecanismos de programação concorrente na qual tem por objetivo controlar a ordem de execução sob a cessão crítica do código.

Questão 2:

Para -3 não é possível pois mesmo com a falta de sincronização por exclusão mútua os valores das Threads não conseguiram chegar nesses valores pois seria necessário que as 3 operações de x-- foram realizados uma em seguida da outra porem isso nao é possivel fazendo sempre o menor resultado ser -2

Para -1 é possível através das operações descritas
* T2(1) o valor de x se torna -1
* T2(2) o valor se torna -2
* T2(3) O if aceita o valor de x

* T1(1) o valor de x se torna -1
* T2(4) o valor printado será -1

Para 1 é possível através das operações descritas
* T3(1) o valor se torna 1
* T3(2) o valor se torna 2
* T3(3) o if aceita o valor de x
* T2(1) o valor x se torna 1
* T3(4) o valor printado será 1

Para 3 é possível através das operações descritas
* T3(1) o valor se torna 1
* T3(2) o valor se torna 2
* T3(3) o if aceita o valor de x
* T1(1) o valor x se torna 3
* T3(4) o valor printado será 3

Questão 3:

- **A)**A seguinte implementação não garante exclusão mútua pois pode ocorrer de ser executado da seguinte maneira:

* T1(1) a aplicação é rejeitada de entrar
* T0(1) se adentra no secao critica
* T1(2)
* T1(3) se troca o valor de TURN para 0
* T0(2) se executa a seção crítica do programa mesmo com as variáveis TURN não sendo consistente com o loop While

Logo com o término do processamento se identifica que a garantia de exclusão mútua não existe.

- **B)** Não essa solução não atende os requisitos de implementação propostos, pois acaba por não apresentar a seção de entrada e a seção de saída proposta.

Questão 4:

```
int x=0, y=0; //variaveis globais

//-----------------------------Thread 1
void *T1(void *id){
 int a=0;
 while(a<2) {
  mutex_lock
 if(x==0)
 {printf("x=%d\n",x);}
 a++;
 printf("a=%d\n", a);
        }
    mutex unlock
    }

//-----------------------------Thread 2
 void *T2(void *id){
 int a=2;
 while(a>0) {
   mutex_lock
 if(x==0)
 {printf("x=%d\n",x);}
 a--;
 fprintf(file,"a=%d\n", a)
   mutex_unclock
  }  }

//-----------------------------Thread 3

void *T3(void *id){
  mutex_lock
y++;
  mutex_unlock
}
```

Foram retirados os x++ e x-- pois em sua execução
havia uma redundância, pois caso seja usado exclusão mútua
as operações de sobre o x resultam sempre em x=0. Já
os locks foram posicionados entre a seção crítica com
o objetivo de gerar a exclusão mútua.

Questão 5:

- **A)**As palavras escritas pelos programas no arquivo bar seriam "Ola mundo!" e "foo nao existe".

- **B)**Não, pois mesmo com o uso da biblioteca pthread.h no código, apenas foi utilizado uma Thread para realizar o processamento da tarefa, gerando assim nenhuma disputa pelo mesma variável durante a realização da função tarefa no programa.