EP 2 - Seção Crítica e Posix threads

- Gabriel Hoffman - 10783250
- Guilherme Balog - 11270649

Solução

Inicialmente tivemos dificuldades com a solução proposta pelos pseudocodigos, visto que estes estavam retornando uma solução que ficava alternando entre uma thread e outra sendo executada de forma completa e não com uma thread iniciando assim que outra libera a seção crítica. Porém após alguns testes e modificações conseguimos alcançar um resultado satisfatório que acreditamos estar correto.

```
while(true){
    while(turn != id);
    critical_section(id);
    turn = other;
    non_critical_section(id);
}
```

Condições

1. **Exclusão mútua.** Se o processo P_i está executando sua seção crítica, então nenhum outro processo pode executar sua seção crítica.
2. **Progresso.** Se nenhum processo está executando sua seção crítica e algum processo quer entrar em sua seção crítica, então somente aqueles processos que não estão executando suas seções remanescentes podem participar da decisão de qual entrará em sua seção crítica a seguir, e essa seleção não pode ser adiada indefinidamente.
3. **Espera limitada.** Há um limite, ou fronteira, para quantas vezes outros processos podem entrar em suas seções críticas após um processo ter feito uma solicitação para entrar em sua seção crítica e antes de essa solicitação ser atendida.

Justificativas

1. Satisfazemos esta condição quando colocamos a linha `while(turn != id)`, ou seja, um processo só pode entrar na sua seção crítica caso nenhum outro processo esteja em sua seção crítica, e isso acontece apenas quando é o seu **turn**.
2. Garantimos essa condição pois a seção crítica só é executada após o término da execução de uma seção não crítica, portanto somente os processos que já terminaram a execução das outras

seções participam da disputa. Além disso, a variável `turn` faz com que a ordem já esteja definida, sem a necessidade de disputa.

3. Atendemos a condição 3 quando após a saida de uma determinada thread de sua seção crítica o **turn** é alterado, ou seja, uma thread nunca entra em sua seção crítica 2 vezes seguidas.

Como saída tivemos o seguinte resultado:

```
[Inicio do programa]


[Fim do programa]


Thread 0 entrou na seção crítica
Count: 1
Thread 0 saiu na seção crítica
Thread 0 entrou na seção não crítica
Thread 0 saiu na seção não crítica

Thread 1 entrou na seção crítica
Count: 2
Thread 1 saiu na seção crítica
Thread 1 entrou na seção não crítica

Thread 0 entrou na seção crítica
Thread 1 saiu na seção não crítica
Count: 3
Thread 0 saiu na seção crítica
Thread 0 entrou na seção não crítica

Thread 1 entrou na seção crítica
Thread 0 saiu na seção não crítica
Count: 4
Thread 1 saiu na seção crítica
Thread 1 entrou na seção não crítica

Thread 0 entrou na seção crítica
Thread 1 saiu na seção não crítica
Count: 5
Thread 0 saiu na seção crítica
Thread 0 entrou na seção não crítica

Thread 1 entrou na seção crítica
Thread 0 saiu na seção não crítica
Count: 6
Thread 1 saiu na seção crítica
Thread 1 entrou na seção não crítica

Thread 0 entrou na seção crítica
Thread 1 saiu na seção não crítica
Count: 7
Thread 0 saiu na seção crítica
```

```
Thread 0 entrou na seção não crítica

Thread 1 entrou na seção crítica
Thread 0 saiu na seção não crítica
Count: 8
Thread 1 saiu na seção crítica
Thread 1 entrou na seção não crítica

Thread 0 entrou na seção crítica
Thread 1 saiu na seção não crítica
Count: 9
Thread 0 saiu na seção crítica
Thread 0 entrou na seção não crítica

Thread 1 entrou na seção crítica
Thread 0 saiu na seção não crítica
Count: 10
Thread 1 saiu na seção crítica
Thread 1 entrou na seção não crítica

E mais...
```

Alguns pontos interessantes podem ser observados:

- O programa exibe "Fim do programa" logo no início pois a função `main()` terminou de executar, mas enquanto isso as *threads* continuar executando
- Após a criação da *thread* 0 e da sua primeira execução, as duas *threads* começam a alternar, assim que uma acaba a seção crítica, a outra começa a executar.