



**UNIVERSIDADE FEDERAL DE SÃO PAULO
INSTITUTO DE CIÊNCIA E TECNOLOGIA
BACHARELADO EM CIÊNCIA E TECNOLOGIA**

**Laboratório 03
Seção Crítica por espera ocupada**

Nomes: Ana Júlia de Oliveira Bellini
Willian Dihanster Gomes de Oliveira

RA: 111774
RA: 112269

**SÃO JOSÉ DOS CAMPOS
2018**

Introdução

Em programação paralela e/ou concorrente, é comum a existência de seções do código que necessitam de uma maior atenção. Em alguns casos, por exemplo, pode ser que dois *threads* executam um incremento sobre uma variável global ao mesmo tempo, o que pode gerar resultados indesejáveis.

Para contornar a situações, diversos algoritmos de controle de seção crítica foram criados. A ideia é definir as regiões críticas e implementar procedimentos como exclusão mútua, que vai garantir que apenas um thread faça uma dada operação por vez.

Um desses algoritmos é o Algoritmo de Manna-Pnueli (a versão utilizada é a *Manna-Pnueli central server algorithm*), e nessa versão, a metodologia é a de cliente-servidor. Há duas variáveis globais (*request* e *respond*), laços infinitos para os processos, para o cliente, as requisições sendo feitas até que sua requisição seja respondida e para o servidor, espera até que haja uma requisição e responde. [1]

Metodologia

Para este trabalho, foi feita uma implementação do algoritmo de Manna-Pnueli, utilizando PThreads em linguagem C. Foram criados dois tipos de threads, servidor e cliente, conforme a Figura 1, abaixo.

```
void *cliente(void *tid) {
    int thid = *(int*)tid;
    while (1) {
        while (resposta != thid) {
            solicitar = thid;
        }
        cont = cont + 1;
        printf("Thread Cliente ID: %d, Contador = %d\n", thid, cont);
        sleep(3);
        resposta = -1;
    }
}

void *servidor(void *tid) {
    long thid = *(long*)tid;
    printf("Thread Servidor ID: %ld\n", thid);
    while (1) {
        while (solicitar == -1);
        resposta = solicitar;
        while (resposta != -1);
        solicitar = -1;
    }
}
```

Figura 1: Funções “cliente” e “servidor” em linguagem C.

Para comprovar o funcionamento do algoritmo, foram feitos dois testes, utilizando 2 e 4 threads clientes, respectivamente. Além disso, foi utilizada outra versão do algoritmo, mas sem o controle de entrada para a seção crítica.

Especificações

Para os experimentos realizados, as especificações da máquina utilizada são descritas a seguir:

Processador: Intel (R) Core (TM) i5-7200U CPU @ 2.50GHz

Núcleos físicos: 2

Memória Cache: 3Mb L3

Memória RAM: 8 GB

Threads: 4

Hyperthreading: Sim

Sistema Operacional: Ubuntu 18.04.1 64 bits

Compilador: gcc 7.3.0

Resultados

Para 2 threads, com o controle da seção crítica, o algoritmo apresentou um comportamento regular. Sendo assim, a variável *Contador* tem seu valor incrementado como o esperado.

```
Thread Cliente ID: 1, Contador = 1000588266
Thread Cliente ID: 1, Contador = 1000588267
Thread Cliente ID: 2, Contador = 1000588268
Thread Cliente ID: 1, Contador = 1000588269
Thread Cliente ID: 2, Contador = 1000588270
Thread Cliente ID: 1, Contador = 1000588271
Thread Cliente ID: 2, Contador = 1000588272
Thread Cliente ID: 2, Contador = 1000588273
Thread Cliente ID: 1, Contador = 1000588274
Thread Cliente ID: 2, Contador = 1000588275
Thread Cliente ID: 1, Contador = 1000588276
Thread Cliente ID: 2, Contador = 1000588277
Thread Cliente ID: 1, Contador = 1000588278
Thread Cliente ID: 1, Contador = 1000588279
Thread Cliente ID: 1, Contador = 1000588280
Thread Cliente ID: 1, Contador = 1000588281
Thread Cliente ID: 1, Contador = 1000588282
Thread Cliente ID: 1, Contador = 1000588283
Thread Cliente ID: 2, Contador = 1000588284
Thread Cliente ID: 1, Contador = 1000588285
Thread Cliente ID: 2, Contador = 1000588286
Thread Cliente ID: 1, Contador = 1000588287
Thread Cliente ID: 1, Contador = 1000588288
Thread Cliente ID: 2, Contador = 1000588289
Thread Cliente ID: 1, Contador = 1000588290
```

Figura 2: Execução do algoritmo de Manna-Pnueli, com 2 threads.

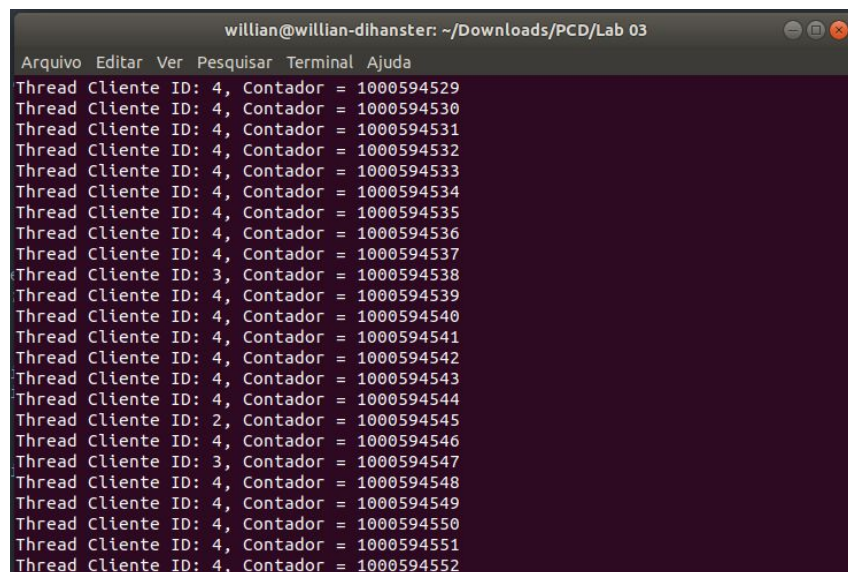
Na versão sem o controle da seção crítica, ambos os threads exibem seus resultados ao mesmo tempo. Inicialmente, os resultados eram mostrados de acordo com a

ordem em que os threads foram criados, mas após algum tempo, a ordem foi sendo mudada, e foram exibidos resultados incorretos, como mostrado na Figura 3.

```
Thread Cliente ID: 2, Contador = 1000466324
Thread Cliente ID: 1, Contador = 1000466325
Thread Cliente ID: 2, Contador = 1000466327
Thread Cliente ID: 1, Contador = 1000466326
Thread Cliente ID: 2, Contador = 1000466328
Thread Cliente ID: 1, Contador = 1000466329
Thread Cliente ID: 1, Contador = 1000466330
Thread Cliente ID: 2, Contador = 1000466331
Thread Cliente ID: 2, Contador = 1000466332
Thread Cliente ID: 1, Contador = 1000466333
```

Figura 3: Execução do algoritmo de Manna-Pnueli, com 2 threads, sem o controle da seção crítica.

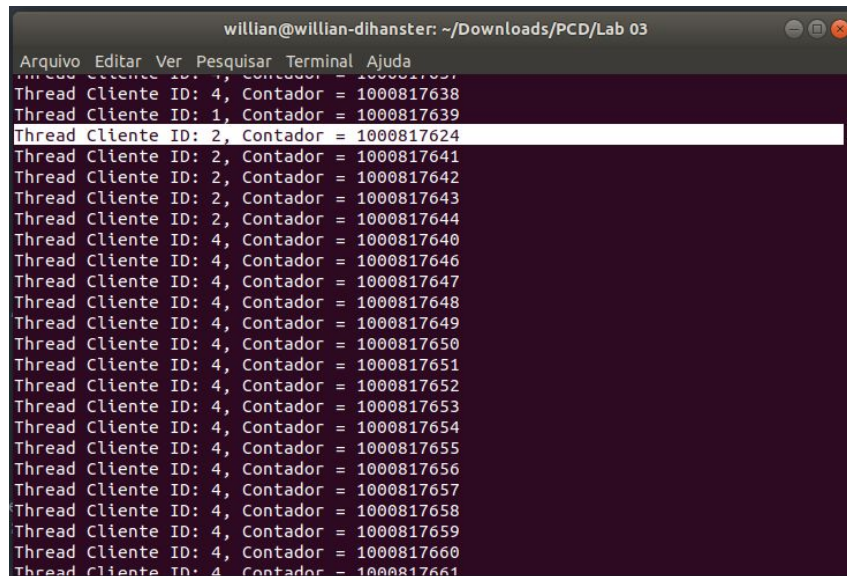
Para o caso de 4 threads clientes, nota-se que os threads 1 e 2 foram executados menos vezes, enquanto os threads 3 e 4 foram responsáveis pela maioria das execuções, mas sem inconsistência no valor da variável global, como o esperado.



```
willian@willian-dihanster: ~/Downloads/PCD/Lab 03
Arquivo Editar Ver Pesquisar Terminal Ajuda
Thread Cliente ID: 4, Contador = 1000594529
Thread Cliente ID: 4, Contador = 1000594530
Thread Cliente ID: 4, Contador = 1000594531
Thread Cliente ID: 4, Contador = 1000594532
Thread Cliente ID: 4, Contador = 1000594533
Thread Cliente ID: 4, Contador = 1000594534
Thread Cliente ID: 4, Contador = 1000594535
Thread Cliente ID: 4, Contador = 1000594536
Thread Cliente ID: 4, Contador = 1000594537
Thread Cliente ID: 3, Contador = 1000594538
Thread Cliente ID: 4, Contador = 1000594539
Thread Cliente ID: 4, Contador = 1000594540
Thread Cliente ID: 4, Contador = 1000594541
Thread Cliente ID: 4, Contador = 1000594542
Thread Cliente ID: 4, Contador = 1000594543
Thread Cliente ID: 4, Contador = 1000594544
Thread Cliente ID: 2, Contador = 1000594545
Thread Cliente ID: 4, Contador = 1000594546
Thread Cliente ID: 3, Contador = 1000594547
Thread Cliente ID: 4, Contador = 1000594548
Thread Cliente ID: 4, Contador = 1000594549
Thread Cliente ID: 4, Contador = 1000594550
Thread Cliente ID: 4, Contador = 1000594551
Thread Cliente ID: 4, Contador = 1000594552
```

Figura 4: Execução do algoritmo de Manna-Pnueli, com 4 threads

Ao executar 4 threads sem um controle de acesso à seção crítica, todos os threads exibem seus resultados simultaneamente, e houve inconsistência no valor da variável global, pois sem o controle, dois threads podem ter tentado alterar a variável global ao mesmo tempo, como mostra a Figura 5.



```
willian@willian-dihanster: ~/Downloads/PCD/Lab 03
Arquivo Editar Ver Pesquisar Terminal Ajuda
Thread Cliente ID: 1, Contador = 1000817637
Thread Cliente ID: 4, Contador = 1000817638
Thread Cliente ID: 1, Contador = 1000817639
Thread Cliente ID: 2, Contador = 1000817624
Thread Cliente ID: 2, Contador = 1000817641
Thread Cliente ID: 2, Contador = 1000817642
Thread Cliente ID: 2, Contador = 1000817643
Thread Cliente ID: 2, Contador = 1000817644
Thread Cliente ID: 4, Contador = 1000817640
Thread Cliente ID: 4, Contador = 1000817646
Thread Cliente ID: 4, Contador = 1000817647
Thread Cliente ID: 4, Contador = 1000817648
Thread Cliente ID: 4, Contador = 1000817649
Thread Cliente ID: 4, Contador = 1000817650
Thread Cliente ID: 4, Contador = 1000817651
Thread Cliente ID: 4, Contador = 1000817652
Thread Cliente ID: 4, Contador = 1000817653
Thread Cliente ID: 4, Contador = 1000817654
Thread Cliente ID: 4, Contador = 1000817655
Thread Cliente ID: 4, Contador = 1000817656
Thread Cliente ID: 4, Contador = 1000817657
Thread Cliente ID: 4, Contador = 1000817658
Thread Cliente ID: 4, Contador = 1000817659
Thread Cliente ID: 4, Contador = 1000817660
Thread Cliente ID: 4, Contador = 1000817661
```

Figura 5: Execução do algoritmo de Manna-Pnueli, com 4 threads, sem o controle da seção crítica.

Conclusão

Como o esperado, com a implementação do algoritmo de exclusão mútua, Manna-Pnueli, não houveram problemas com o incremento da variável compartilhada global, como esperado. Já sem o controle da seção crítica, houveram resultados indesejados para a variável Contador. Com isso, reforça-se a importância do uso de algoritmos para controle da seção crítica que implementem a exclusão mútua, quando necessário.

Referências

[1] Ben-Ari, Mordechai. *Principles of concurrent and distributed programming*. Vol. 2. Addison-Wesley, 2012.