



SEMINARIO - Prazo de Entrega: 17/04/2023

ATENÇÃO: Descrever as soluções com o máximo de detalhes possível, no caso de programas, inclusive a forma como os testes foram feitos. Todos os artefatos (slides, código fonte de programas, e outros) gerados para este trabalho devem ser adicionados em um repositório no site `github.com`, com o seguinte formato:

Aluno1Aluno2_ws_OS_RR_2023.

OBSERVAÇÃO: Os temas abaixo já foram escolhidos em sala de aula. Todos os softwares desenvolvidos devem ser implementados na linguagem de programação C/C++ ou em RUST. Na apresentação do seminário a equipe deverá apresentar um slide com os resultados do tema definido. Cada equipe terá 8 minutos para efetuar sua apresentação.

[TEMA - 1] Jogo da Velha

O projeto a ser desenvolvido, consiste na criação do jogo da velha. Neste caso, o jogo será feito por dois usuários artificiais (ou seja duas threads) no mesmo jogo da velha. Cada usuário deve operar o jogo no mesmo tabuleiro do jogo da velha 3 por 3. Assim, o recurso compartilhado será tabuleiro do jogo da velha, logo os jogadores devem esperar a sua vez de jogar e obedecer a regras do jogo.

Sugestão de consulta:

- <http://wccclab.cs.nchu.edu.tw/www/images/105-2Python/python%206.pdf>

[TEMA - 2] Barbeiro Sonolento

O projeto a ser desenvolvido, consiste no seguinte problema: Na barbearia há um barbeiro, uma cadeira de barbeiro e n cadeiras para eventuais clientes esperarem a vez. Quando não há clientes, o barbeiro senta-se na cadeira de barbeiro e cai no sono. Quando chega um cliente, ele precisa acordar o barbeiro. Se outros clientes chegarem enquanto o barbeiro estiver cortando o cabelo de um cliente, eles se sentarão (se houver cadeiras vazias) ou sairão da barbearia (se todas as cadeiras estiverem ocupadas). Logo, a equipe deve implementar um software que simule o problema apresentado e destacar as possíveis situações de disputa pelos recursos compartilhados.

Sugestão de consulta:

- <http://www.inf.ufsc.br/~bosco.sobral/ensino/ine5645/Lab3.pdf>

[TEMA - 3] Números Primos entre 0 e 1.000.000

O projeto a ser desenvolvido, consiste em desenvolver um software que imprima os números primos existentes entre 0 e 1.000.000. Utilize threads para cada faixa de 1000 valores crie uma thread e dispare o processo para cada uma delas.

Sugestão de consulta:

- <https://www.geeksforgeeks.org/how-to-find-prime-and-palindrome-numbers-using-multi-threading-in-java/>



[TEMA - 4] Problema do Jantar dos Filósofos

O projeto a ser desenvolvido, consiste no seguinte problema: Cinco filósofos estão sentados em uma mesa redonda para jantar. Cada filósofo tem um prato com espaguete à sua frente. Cada prato possui um garfo para pegar o espaguete. O espaguete está muito escorregadio e, para que um filósofo consiga comer, será necessário utilizar dois garfos. Lembre-se que é apenas uma analogia. Nesse sentido, cada filósofo alterna entre duas tarefas: comer ou pensar. Quando um filósofo fica com fome, ele tenta pegar os garfos à sua esquerda e à sua direita; um de cada vez, independente da ordem. Caso ele consiga pegar dois garfos, ele come durante um determinado tempo e depois recoloca os garfos na mesa. Em seguida ele volta a pensar. Logo, a equipe deve implementar um software que simule o problema apresentado e destacar as possíveis situações de disputa pelos recursos compartilhados.

Sugestão de consulta:

- <https://blog.pantuza.com/artigos/o-jantar-dos-filosophos-problema-de-sincronizacao-em-sistemas-operacionais>

[TEMA - 5] Identificação de deadlock usando algoritmo do banqueiro

O projeto a ser desenvolvido, consiste em implementar um software para identificar deadlock utilizando o algoritmo do banqueiro que considera cada requisição no momento em que ela ocorre, verificando se essa requisição leva a um estado seguro; Se sim, a requisição é atendida, se não o atendimento é adiado para um outro momento. Premissas adotadas por um banqueiro (SO) para garantir ou não crédito (recursos) para seus clientes (processos); e Nem todos os clientes (processos) precisam de toda a linha de crédito (recursos) disponível para eles.

Sugestão de consulta:

- <https://alexcoletta.eng.br/artigos/deadlock-em-sistemas-operacionais/>

[TEMA - 6] Servidor de Jogos para match

O projeto a ser desenvolvido, consiste em desenvolver um Multi-threaded Server Game. Quando você pensa em um servidor de jogo, pode se surpreender ao descobrir que muitos são de single thread. Alguns chegam a ponto de desenvolver um modelo de *Single Threaded Apartment*. Compreensivelmente, esse design leva a gargalos de desempenho. Faz muito mais sentido usar vários threads para que os pacotes dos jogadores sejam processados em paralelo. O principal problema que a maioria dos desenvolvedores enfrenta ao trabalhar com um servidor de jogo multi-thread é o acesso do servidor aos recursos. Os jogadores costumam jogar juntos em um MMORPG, o que pode significar o compartilhamento de mapas, itens, missões e assim por diante. Além disso, o servidor também deve controlar monstros que atacam os jogadores e soltam itens. São muitos comportamentos de estado para trabalhar, e manipular esses estados em vários threads pode levar a uma série de problemas. Logo, a equipe deve implementar um software que simule Multi-threaded Server Game para um jogo a ser definido.

Sugestão de consulta:

- <https://www.geeksforgeeks.org/socket-programming-multi-threading-python/>
- <https://doc.rust-lang.org/book/ch20-02-multithreaded.html>



[TEMA - 7] Busca em profundidade usando multithreading

O projeto a ser desenvolvido, consiste em implementar o algoritmo de busca em profundidade em grafos usando multithreading. O algoritmo de busca em profundidade (ou Depth-First Search – DFS) consiste em: dado um grafo, o algoritmo visita todos os vértices e todos os arcos do grafo numa determinada ordem e atribui um número a cada vértice: o k-ésimo vértice descoberto recebe o número k.

Sugestão de consulta:

- https://www.ime.usp.br/~pf/algoritmos_para_grafos/aulas/dfs.html
- <https://play.rust-lang.org/?gist=9a2b519c12000e5db8ffcc328a8c00ac&version=stable&backtrace=0>
- <https://www.daniweb.com/programming/software-development/threads/456242/parallel-dfs>

[TEMA - 8] Fibonacci usando multithreading

O projeto a ser desenvolvido, consiste em gerar a sequência de números da série de Fibonacci até o número 1.000.000. Utilize threads para cada faixa de 1000 valores, crie uma thread e dispare o processo para cada uma delas. A sequência de Fibonacci é composta por uma sucessão de números descrita pelo famoso matemático italiano Leonardo de Pisa (1170-1250), mais conhecido como Fibonacci, no final do século 12. O matemático percebeu uma regularidade matemática a partir de um problema criado por ele mesmo. Assim, a fórmula a seguir define a sequência: $F_n = F_{n-1} + F_{n-2}$.

Sugestão de consulta:

- <https://www.coc.com.br/blog/soualuno/matematica/o-que-e-a-sequencia-de-fibonacci>
- <https://benjaminbrandt.com/fibonacci-in-rust/>
- <https://stackoverflow.com/questions/42121296/multithreaded-fibonacci>

[TEMA - 9] Fatorial usando multithreading

O projeto a ser desenvolvido, consiste em calcular o fatorial de todos os números até 1.000.000. Utilize threads para cada faixa de 1000 valores, crie uma thread e dispare o processo para cada uma delas. O fatorial de um número inteiro e positivo “n”, representado por “n!” é obtido a partir da multiplicação de todos os seus antecessores até o número um, cuja expressão genérica é $n! = n \cdot (n-1) \cdot (n-2) \cdot (n-3) \dots 2,1$.

Sugestão de consulta:

- <https://programming-idioms.org/idiom/31/recursive-factorial-simple/450/rust>
- <https://users.rust-lang.org/t/parallel-product-for-factorial-surprised-by-the-results/30776/14>
- <https://dev.to/dandyvica/using-threads-on-rust-part-3-2bpf>



[TEMA - 10] Editor de Texto

O projeto a ser desenvolvido, consiste em implementar um simples editor de texto que execute a seguinte funcionalidade: Dado um arquivo de texto, quando este arquivo é alterado, simultaneamente os seguintes arquivos de backup devem ser criados e alterados paralelamente: 1) salvar somente números; 2) salvar somente letras; 3) salvar somente caracteres especiais; e 4) salvar o número de linhas já escritos. Utilize threads para identificar os itens a serem salvos e crie 4 threads para alterar cada arquivo simultaneamente.

Sugestão de consulta:

- https://doc.rust-lang.org/rust-by-example/std_misc/file/create.html
- <https://www.programming-idioms.org/idiom/62/find-substring-position/498/rust>
- <https://doc.rust-lang.org/book/ch16-01-threads.html>

