Universidade Federal de Roraima
Departamento de Ciência da Computação
Sistemas Operacionais – DCC403

SEMINARIO - Prazo de Entrega: 03/06/2025 e 05/06/2025

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_2025`.

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 seminario 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 obdecer 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 Filosofos

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-filosofos-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 profundida 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 serie 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: Fn = Fn - 1 + Fn – 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 . (n – 1). (n – 2). (n – 3) ... 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, simultaneamento 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

[TEMA - 11] Resolução Paralela de Sudoku com Multithreading

Desenvolver uma aplicação que utiliza *multithreading* para acelerar o processo de resolução ou verificação de um tabuleiro de Sudoku. A ideia é dividir o problema em partes independentes que possam ser executadas simultaneamente, utilizando threads para explorar o paralelismo disponível nos processadores modernos. O programa recebe um tabuleiro preenchido e verifica se ele é válido, utilizando threads separadas para verificar:

- Cada uma das 9 linhas
- Cada uma das 9 colunas
- Cada um dos 9 blocos 3x3
  Totalizando até 27 threads trabalhando em paralelo.

Sugestão de consulta:

- https://www.secs.oakland.edu/~llamocca/Courses/ECE4772/F23/FinalProject/Group7_sudoku.pdf
- https://cse.buffalo.edu/faculty/miller/Courses/CSE633/Sankar-Spring-2014-CSE633.pdf

[TEMA - 12] Processamento Paralelo de Imagens com Multithreading

Implementar uma aplicação que aplica filtros a uma imagem (como escala de cinza, inversão de cores, blur simples, entre outros) utilizando *multithreading* para acelerar o processamento. A ideia é dividir a imagem em blocos (por linhas, colunas ou quadrantes), atribuindo cada parte a uma *thread* que executa a transformação desejada. Ao final, o resultado deve ser reconstruído em uma única imagem de saída.

Exemplos de filtros possíveis:

- Escala de cinza (grayscale)
- Inversão de cores
- Blur simples (média de pixels vizinhos)
- Detecção de bordas (filtro Sobel simples)

Sugestão de consulta:

- https://datacarpentry.github.io/image-processing/instructor/02-image-basics.html
- https://realpython-com.translate.goog/image-processing-with-the-python-pillow-library

[TEMA - 13] Busca Paralela em Arquivos com Multithreading
Desenvolver uma aplicação que realiza a busca por uma ou mais palavras-chave em múltiplos arquivos de texto (por exemplo, arquivos `.txt` em um diretório). A tarefa deve ser dividida entre várias *threads*, onde cada thread processa um ou mais arquivos em paralelo. A aplicação deve exibir:

- O nome de cada arquivo onde a palavra foi encontrada
- A(s) linha(s) onde ela aparece
- O tempo total de execução da busca

Sugestão de consulta:
- https://docs.python.org/3/library/concurrent.futures.html
- https://pynative.com/python-search-for-a-string-in-text-files/

[TEMA - 14] Simulação de Elevadores com Multithreading
Implementar um sistema onde cada elevador é representado por uma *thread*, e cada elevador responde a solicitações de passageiros (requisições para subir ou descer). As requisições podem ser geradas automaticamente ou inseridas pelo usuário.

A lógica do sistema pode incluir:

- Fila de requisições (compartilhada ou individual por elevador)
- Lógica de despacho simples (por proximidade ou distribuição balanceada)
- Simulação de movimento (subir/descer com *sleep* para tempo de deslocamento)
- Impressão do estado atual dos elevadores (andar atual, direção, destino)

Sugestão de consulta:
- https://realpython.com/intro-to-python-threading/#using-a-queue
- https://people.ece.cornell.edu/land/courses/ece4760/FinalProjects/s2007/aoc6_dah64/aoc6_dah64/index.html

[TEMA - 15] Cálculo Paralelo de Matrizes (Multiplicação de Matrizes com Multithreading)
Desenvolver um programa que multiplica duas matrizes de forma paralela, distribuindo o trabalho entre múltiplas *threads*. Cada thread será responsável por calcular uma ou mais linhas (ou células) da matriz resultante.

Exemplo:

- Dada `A (m x n)` e `B (n x p)`, calcular `C = A × B`, onde cada thread pode calcular:
  - Uma linha de `C`
  - Um bloco de linhas
  - Cada elemento individual

Cenário sugerido:

- Matrizes grandes geradas aleatoriamente (ex: 500×500)
- Comparação entre a versão sequencial e paralela (tempo de execução)

Sugestão de consulta:
- https://www.geeksforgeeks.org/multiplication-of-matrix-using-threads/