Atividade Acadêmica: Análise e Aplicação de Sistemas Operacionais

Professor: Marcos Ricardo Kich

DESAFIOS

(Módulos 2, 3 e 4)

M2 | Tarefa - Gerenciamento de Processos na Prática

Para exercitarmos os conceitos estudados até o momento, realizaremos uma atividade prática. Nesta atividade, você deverá:

- Desenvolver um programa que utilize as primitivas **fork** e **exec** juntas. Este programa implementará atividades diferentes para os processos pai e filho.
- O **processo pai** ficará responsável por apresentar a transposta de uma matriz de tamanho 10x10 previamente inicializada (você deve popular a matriz original). Dica: https://en.wikipedia.org/wiki/Transpose
- O **processo filho** ficará responsável pela leitura de uma página de notícias de grande circulação para encontrar uma determinada palavra. Dica: utilize os comandos: **curl** para baixar a página, **egrep** para encontrar a palavra usando uma expressão regular; e **wc** para contar a quantidade de vezes que a palavra desejada aparece na página escolhida. A primitiva **exec** será utilizada no trecho de código executado pelo processo filho.

M3 | Tarefa - Tratamento de Sinais na Prática

Para exercitarmos os conceitos estudados até o momento, realizaremos uma atividade prática. Nesta atividade, você deverá:

- Desenvolver um programa que, ao receber o sinal **SIGUSR2**, execute uma função responsável por abrir o navegador Firefox. A URL deve ser digitada pelo usuário previamente à chamada. Para a implementação do sinal, utilize a primitiva **sigaction**.

Para desenvolver esta atividade, você deverá utilizar o ambiente Linux. Para isso, utilize o mesmo ambiente configurado para o Desafio do Módulo II.

M4 | Tarefa - Threads na Prática

Para exercitarmos os conceitos estudados até o momento, realizaremos uma atividade prática. Nesta atividade, você deverá:

- Desenvolver um programa usando **threads** que calcule e mostre a quantidade de números primos existentes entre 1 (um) e 5.000.000 (cinco milhões). Para isso, você deve criar **4 threads**. As threads deverão acessar uma variável global que indica qual é o próximo número para **verificar se é primo ou não**. Dica: qual é o nome dado ao trecho de código acessado pelas threads de forma concorrente? Como gerenciar o acesso de modo a garantir exclusão mútua?

Para desenvolver esta atividade, você deverá utilizar o ambiente Linux. Para isso, utilize o mesmo ambiente configurado para os Desafios dos Módulos II e III.

Será avaliado se programa executa corretamente e se faz uso de forma adequada de **mutex** ou semáforos para garantir a exclusão mútua. Por exemplo, é muito simples colocar um **pthread_mutex_lock** na primeira linha da função que implementa a thread e um **pthread_mutex_unlock** na última linha, mas isso só vai fazer com que durante toda execução daquela thread as outras fiquem esperando e dessa forma não ocorre execução paralela de fato. **Mutex ou semaforo deve ser utilizado apenas quando necessário no código.**