# Universidade Federal do Rio Grande do Norte - UFRN
# Centro de Ensino Superior do Seridó - CERES
# Departamento de Computação e Tecnologia - DCT

**Curso:** Bacharelado em Sistemas de Informação
**Disciplina:** DCT2101 – Sistemas Operacionais
**Professor:** João Borges
**Data:** 26 de julho de 2024

## Atividade em Dupla
## Unidade 2 - Tarefa 2
*Algoritmo do Banqueiro*

ATENÇÃO 1: Só serão aceitos trabalhos **Individuais** ou em **Dupla**, mais do que isso invalidará o trabalho;
ATENÇÃO 2: Não serão permitidos plágios entre os trabalhos, sendo punidos, ambos os componentes que tiverem seus trabalhos iguais, com nota 0 (zero).

- Esta atividade consiste na implementação do algoritmo do banqueiro, para a evitar a ocorrência de impasses.

  1. Para que haja a necessidade de utilização do algoritmo do banqueiro, é necessário considerar:
     - Os $m$ tipos de recursos disponíveis no sistema, com suas quantidades, e
     - Uma quantidade $n$ de processos que querem acessá-los, processos estes que são executado em *threads* separadas.
  2. Observe que, por acessar recursos compartilhados, é importante que se utilizem semáforos para controlar a concorrência.

- No SIGAA está uma versão do algoritmo do banqueiro (*banqueiro.c*), que servirá como "esqueleto" inicial para a implementação necessária.

  1. Para executar o código, é preciso informar o número de clientes (threads) e o número de classes (tipos) de recursos disponíveis no sistema.

     ```
     Uso: ./banqueiro num_clientes num_recursos
     ```

  2. Neste exemplo, cada cliente é definida por uma *thread*, que será criada por meio da função **cliente**.
     (a) Cada cliente fica em looping (*while(executa)*), gerando requisições, até conseguir todos os recursos que precisa, quando conseguir todos os recursos, irá finalizar.

3. A quantidade de recursos disponíveis para cada um dos tipos de recurso informados (`num_recursos`), que os clientes (threads), informados em (`num_clientes`), precisam, são gerados aleatoriamente pelo simulador.
4. As requisições de cada cliente também são geradas aleatoriamente.
   (a) A função que gera as requisições é a `gera_requisicao`.
   (b) Ela retorna um vetor contendo a requisição gerada.
5. Sempre que um cliente gera uma requisição, o sistema executa o algoritmo do banqueiro, para verificar se a requisição pode ser atendida.
   (a) Essa verificação é realizada na função `requisicao`.
   (b) O código desta requisição deverá ser implementada, conforme é definido no algoritmo do banqueiro.
      – Esta função deverá retornar:
        1 - Caso a requisição possa ser atendida
        0 - Caso contrário
   (c) Para a requisição ser verificada, também é necessário implementar o algoritmo de segurança, como também é definida pelo algoritmo do banqueiro.
   (d) O código do algoritmo de segurança deverá ser implementado na função `seguranca`
      – Esta função deverá retornar:
        1 - Caso o sistema esteja em estado seguro
        0 - Caso contrário
6. Um cliente terminará quando conseguir todos os recursos que precisa.
7. A execução termina quando todos os clientes terminarem.
8. O exemplo está utilizando a bilioteca de semáforos do sistema (*semaphore.h*), contudo, ainda é necessário que você defina a utilização do semáforo no código.
9. O semáforo deverá garantir que a thread cliente, definida na função **cliente**, realize o acesso mutuamente exclusivo aos recursos.
   – É importante observar que, por se tratar de uma simulação, o cliente não utiliza os recursos propriemante ditos.
   – Mas, ele aloca para ele os recursos e libera-os quando finaliza, como poderá ser visto nas matrizes do algoritmo.
   – Assim, para a execução correta deste simulador, é preciso garantir que a verificação da sua requisição, bem como a liberação dos recursos dos clientes, seja realizada de forma exclusiva.
10. Para definir o semáforo, você deve:
    (a) Criar um semáforo *mutex*, necessário para o controle do acesso aos recursos pelos clientes.
    (b) Inicialiar o semáforo para acesso exclusivo.
    (c) Garantir o acesso exclusivo aos recursos compartilhados.
    (d) Liberar o acesso aos recursos compartilhados.
    (e) Excluir o semáforo criado ao final.

- Por fim, as informações sobre sua implementação das funções do algoritmo do banqueiro, bem como a explicação de como o semáforo foi definido, deverão ser organizadas sob a forma de relatório, onde deverão ser discutidas as implementações dos algoritmos.

- O modelo de relatório seguirá o *Modelo para publicação de artigos* da Sociedade Brasileira de Computação (SBC):

  `http://bit.ly/SBCArtigos`

- O código-fonte da implemtnação do algoritmo deverá ser enviado juntamente com o relatório da atividade. No entanto, para melhor explicar a sua implementação, no relatório poderão ser inseridos trechos do código, ou algoritmo, conforme sua necessidade.

- O envio da atividade deverá ser feita pelo SIGAA, até a data estabelecida na tarefa cadastrada no sistema.