#### CSC-27 – Computação Distribuída

# Laboratório 2: Algoritmo de Ricart-Agrawala para exclusão mútua

Nomes: Daniel Araujo Cavassani (COMP 25)

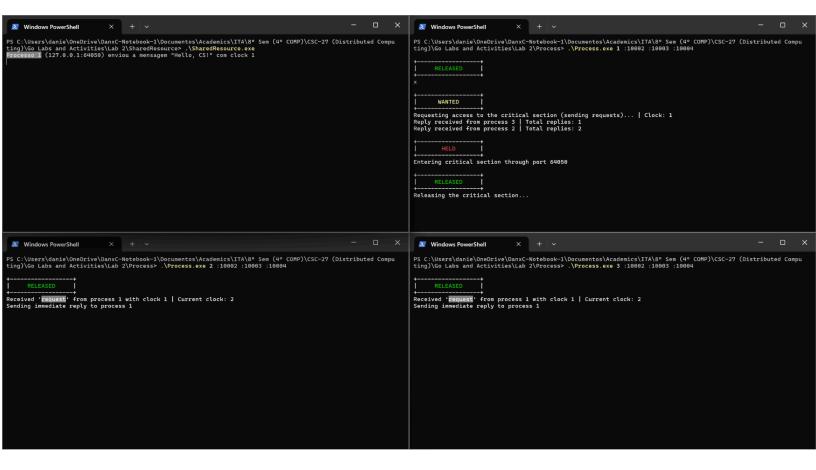
Data: 10/09/2024

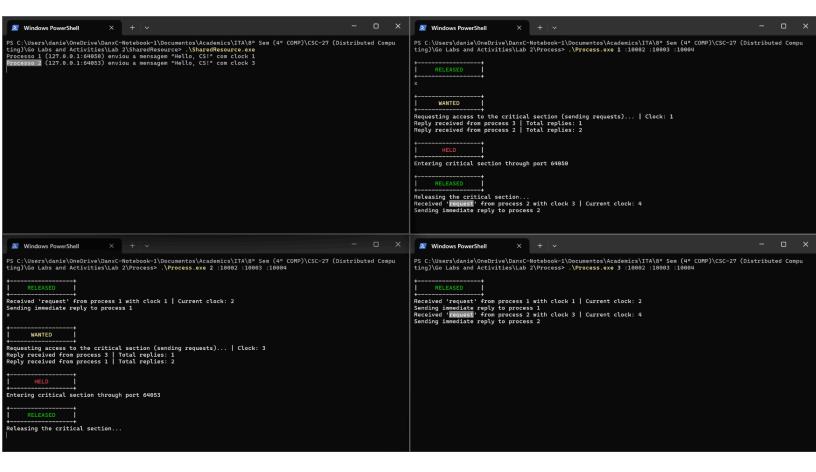
#### **Testes Realizados**

Para validar o funcionamento correto do algoritmo de Ricart-Agrawala, realizamos uma série de testes com três processos simultâneos. Os cenários abaixo demonstram como o algoritmo lida com diferentes situações de requisição e liberação da seção crítica (Critical Section - CS).

## Caso 1: Um processo solicita a CS. Após liberá-la, outro processo solicita a CS.

- P1 solicita a CS.
- P1 usa a CS e depois a libera.
- P2 usa a CS e depois a libera.

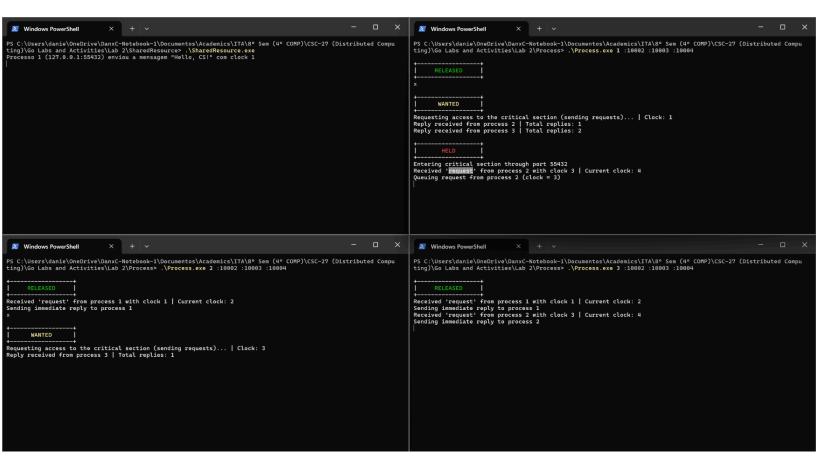




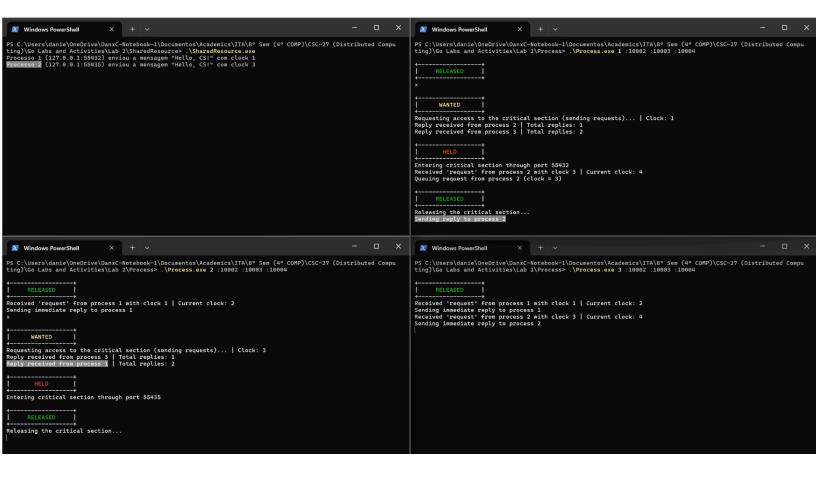
#### Caso 2: Um processo solicita a CS enquanto outro processo está usando a CS

P2 solicita a CS enquanto P1 está utilizando a CS

Aqui, podemos verificar que o request de P2 chega em P1 enquanto o P1 está HELD (usando a CS)



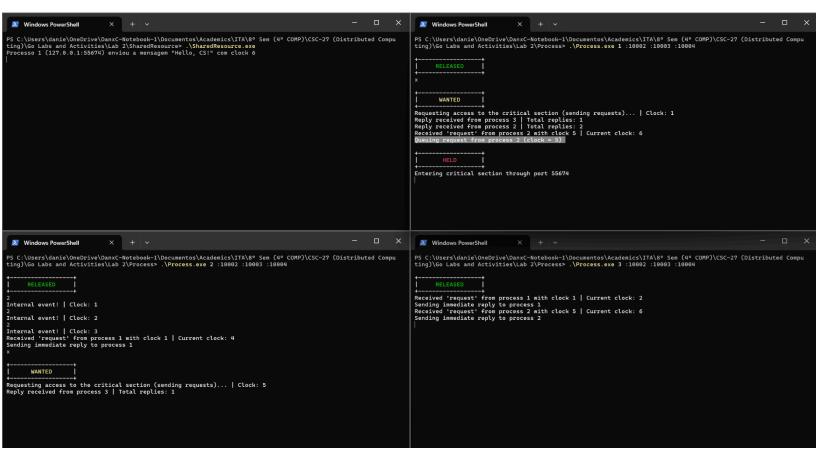
Após sair de HELD, ele consulta sua queue e da reply para o request de P2, permitindo P2 a entrar na CS



## Caso 3: Processo com requisição anterior tem preferência para acessar a Seção Crítica

- P1 solicita a CS (está WANTED)
- P2 também solicita a CS (está WANTED)
- P3 da REPLY para ambas as requisições
- Como P1 fez a solicitação com menor clock, ele entrará na CS primeiro, e logo após, o P2 entrará.

P1 e P2 entram em WANTED ao mesmo tempo. Note que P1 recebe de P2 um request com clock 5, sendo que o clock que ele tinha quando mandou request para a CS era 1... logo ele tem preferência e da enqueue no request recebido. Já P2, recebe de P1 o request de clock 1, sendo que enviou um request com clock 5... logo, ele sabe que não tem prioridade.



Após sair da CS, o P1 consulta sua queue de requests e da reply em todos eles. O P2, então, pode entrar na CS com este reply que estava faltando

