

Disciplina: DIM0612 — Programação Concorrente
Docente: Everton Ranielly de Sousa Cavalcante
Discente: Felipe Cortez de Sá

Sincronização de um banheiro unissex virtual

1 Introdução

Este relatório descreve a implementação de algoritmos para sincronização de *threads* usando duas estratégias diferentes.

2 Modelo

As partes comuns às duas estratégias de sincronização estão no arquivo `common.py`.

O banheiro é implementado pela classe `Toilet`, que possui internamente um contador `counter` responsável por guardar o número de pessoas usando o toalete e uma lista de tamanho fixo l , sua capacidade máxima. O método `enter` recebe como parâmetro um objeto da classe `Person` e troca o primeiro espaço vago da lista pelo objeto. O método `leave` recebe como parâmetro uma pessoa a ser retirada do banheiro, atualizando a lista com um espaço vago onde a pessoa estava. Os espaços vagos são codificados como uma *string* `EMPTYSTR`.

Uma pessoa é representada como instância da classe `Person`, que possui um atributo `gender`, podendo assumir os valores de *string* 'M' ou 'F', determinados aleatoriamente na instanciação. Além disso, cada pessoa recebe um nome aleatório para visualização durante a execução do programa.

A função `main` recebe como parâmetro uma classe que implementa o método `personThread`, responsável pela lógica de sincronização. Os arquivos `locks.py` e `semaphores.py` implementam as classes e chamam a função `main`. Assim, os programas funcionam independentemente, cada um com seu próprio ponto de entrada `main`.

Foi utilizada a biblioteca `argparse` para leitura de parâmetros de linha de comando. O único parâmetro lido é `l`, a capacidade do banheiro. Os programas podem ser executados através dos comandos `python3 [semaphores|locks].py -l 10`.

3 Lock

O *lock* é adquirido por uma *thread* sempre ao ler o estado do banheiro. Se o banheiro estiver vazio ou tiver apenas pessoas do mesmo sexo e não estiver lotado, a pessoa entra no banheiro e libera o *lock*. Caso contrário, a pessoa não entra e libera o *lock*,

repetindo as tentativas. Em outras palavras, o *lock* apenas serve como uma maneira de limitar o acesso ao objeto `toilet`, transformando a leitura e escrita numa operação atômica.

4 Semáforo e variável de condição

Foi utilizada a classe `BoundedSemaphore` do Python, inicializada com limite determinado pelo usuário representando o número de pessoas que podem utilizar o banheiro simultaneamente. Para não permitir que pessoas do mesmo sexo entrem no banheiro, foi utilizada uma variável de condição `empty`, que faz a *thread* esperar se o banheiro estiver ocupado por pessoas do sexo oposto. Ao final da execução de uma *thread*, as *threads* em espera são notificadas através do método `notify_all` caso o banheiro fique vazio.