Curitiba, 2022/2
Universidade Positivo
Programação Concorrente
Prof. Jean (jean.diogo@up.edu.br)

Especificação do **TRABALHO 2** (v1)

No presente bimestre tivemos como objetivo estudar a solução de alguns problemas clássicos de sincronização de *threads* utilizando técnicas como mutexes e semáforos através da linguagem C++. Neste trabalho sua tarefa aplicar o conhecimento adquirido resolvendo o problema a seguir, que é uma adaptação do "problema do banheiro único".

Um pet-shop passou a receber diariamente uma quantidade de cães e gatos para cuidar e alimentar. O pet-shop possui duas áreas, uma na qual os pets brincam e dormem e outra na qual são alimentados. Durante todo o dia cada pet se alterna entre brincar/dormir e comer. A área de brincar é grande e comporta um grande número de pets. A área de comer é pequena e comporta um número limitado de pets. Além disso, cães e gatos interagem cooperativamente na área de brincar, mas na área de comer pode ocorrer competição entre as duas espécies, sendo necessário evitar que cães e gatos a ocupem ao mesmo tempo. Assim, o pet-shop estabeleceu uma regra: enquanto houver cães na área de comida, os gatos que chegarem devem esperar, mas os cães que chegarem podem continuar entrando até o limite do espaço. Quando o último cão sair, os gatos que estavam esperando podem começar a entrar e os cães que chegarem devem esperar. (O processo é recíproco para o caso em que gatos ocupem a área primeiro.) Entretanto, após aplicar essa regra o pet-shop observou que caso um dos grupos ocupassem a área primeiro e não parassem de retornar a ela, os pets do outro grupo nunca tinham a chance de entrar. Por essa razão, uma regra adicional foi estabelecida: se houver cães comendo e chegar um gato, os cães que chegarem depois deste gato devem parar de entrar para que o gato possa entrar assim que o último cão que está lá dentro sair. (O processo é recíproco para caso haja gatos comendo e chegue um cão.) Portanto, o problema do pet-shop é o problema de sincronizar o acesso de dois grupos de processos à região crítica evitando que haja concorrência entre os grupos (mas não entre os indivíduos de cada grupo) e evitando que haja *starvation* para algum deles.

O código base para o trabalho está disponível no Blackboard com o nome "trabalho2.cpp". No topo do código estão algumas regras e instruções para a implementação. A defesa do trabalho ocorrerá durante a aula especificada no calendário do Blackboard da turma. A equipe deverá subir o código em um repositório GIT qualquer e terá cerca de 10 minutos para executá-lo e explicar seu funcionamento. Os trabalhos deverão ser preferencialmente realizados pelas mesmas equipes do primeiro trabalho. Caso haja alterações nas equipes, favor comunicar o professor. Trabalhos que não cumpram todos os requisitos do trabalho serão pontuadas apenas parcialmente. Dúvidas devem ser encaminhadas ao email do professor ou postadas no Blackboard.