H1 Sistemas Operacionais - Trabalho Prático 1

H4 Pedro Tavares de Carvalho - 2017014499 | Francisco Bonome Andrade - 2016006450

H2 Introdução

O problema apresentado pelo trabalho consiste basicamente em uma fila de pessoas que querem utilizar um microondas para esquentar seus alimentos. Essas pessoas possuem prioridades diferentes para executar essa tarefa e, após utilizarem o microondas, saem para comer, mas depois podem voltar novamente à fila para executar a tarefa novamente.
Com o intuito de representar essa estrutura, foi utilizada a biblioteca *pthread.h* , a qual nos permitiu definir cada uma das pessoas como uma *thread* diferente e o microondas como um *mutex*. Dessa forma, com as prioridades de cada *thread* devidamente estabelecidas, bloqueamos e liberamos o *mutex* conforme haja necessidade.

H2 Estruturas

Nossa solução conta com duas estruturas (classes) principais. A primeira é o Monitor, no qual estão implementadas todas as funções que são utilizadas para gerir o uso do microondas, assim como onde são criadas as *threads* para cada pessoa, a estrutura que contém informações de cada pessoa e os *mutexes* que serão utilizados no processo.
É importante ressaltar que foram utilizados dois *mutexes* distintos, sendo que um é associado ao uso do microondas (o que já era esperado), e o segundo é associado à entrada e saída de pessoas na fila para utilizar o microondas. Viu-se necessária a criação desse segundo *mutex* para permitir que o cálculo das prioridades dos membros da fila fosse feita de maneira correta, dado que a execução simultânea das *threads* estava impactando na atualização de informações do processo quando mais de uma *thread* entrava na fila/tentava utilizar o microondas ao mesmo tempo.
A segunda classe implementada é denominada Person, e ela armazena as informações individuais de cada uma das pessoas que desejam usar o microondas. Basicamente, quando o Monitor é iniciado ele cria dentro de si nove instâncias da classe Person, as quais armazenam o ID da *thread*, o nome da pessoa, quantas vezes ela utilizou o forno, se ela está acompanhada pelo(a) namorado(a), se ela está na fila e se ela entrou na fila antes do parceiro. Além disso, cada instância criada da classe Person possui sua própria variável de condição definida pela biblioteca pthread (*pthread_cond*). Essa variável é utilizada para determinar quando a *thread* deve ficar aguardando e quando ela deve continuar sua execução (utilizar o microondas).

H2 Lógica

A ideia base da execução do programa é criar um Monitor, o qual recebe como entrada o número de vezes que cada pessoa deve usar o microondas (inteiro fornecido como entrada pelo usuário). Após criado, o Monitor deve ser "iniciado", o que consiste em informá-lo qual função deverá ser associada às *threads* que vão ser criadas por ele. Após a criação das *threads* e suas respectivas execuções, o Monitor é deletado e o programa se encerra.
A função que é informada para "dar início" ao Monitor chama-se *go_to_work*, e ela define o comportamento que uma *thread* (uma pessoa) deve seguir. Esse comportamento é àquele o qual foi descrito na introdução: enquanto a pessoa não utilizou o microondas pelas n

vezes informadas pelo usuário, ela entra na fila para usar o microondas, usa o microondas e depois “vai para casa” - ou seja, libera o microondas para a próxima pessoa da fila e se ausenta durante um certo período de tempo antes de querer usar o aparelho novamente. Nesse ponto, a única pessoa/*thread* que possui comportamento diferenciado é o Raj, que ao invés de utilizar o microondas apenas fica “observando” os outros usarem.
O comportamento descrito acima foi definido por quatro funções distintas dentro do escopo de *go_to_work*, e a descrição lógica da execução de cada uma será dada a seguir. É importante dizer que a função referente ao Raj é executada apenas uma vez, enquanto as outras funções têm sua quantidade de vezes de execução definida com base no valor de entrada escolhido pelo usuário. As funções que podem ser executadas mais de uma vez estão descritas conforme sua ordem de execução.

`use_microwave`

Primeiramente seleciona-se qual pessoa -qual instância de Person- está executando essa atividade, e isso é definido por meio do ID da *thread* que realizou a chamada da função. Logo em seguida, o *mutex* responsável pela fila é bloqueado e os dados daquela instância de Person são atualizados (a pessoa é definida como estando na fila, e é feita a verificação se ela está acompanhada e se chegou antes de seu parceiro na fila), e é informado ao usuário que a pessoa X deseja usar o forno.Em seguida, é feita a verificação se aquela pessoa se encontra no primeiro lugar da fila. Caso não esteja, a pessoa recebe um sinal para aguardar (*pthread_cond_wait*). Caso esteja no primeiro lugar da fila, o *mutex* da fila é liberado, e o do microondas é bloqueado, de forma a permitir que apenas aquela pessoa utilize o microondas.Após esse bloqueio, informa-se ao usuário que a pessoa X começou a esquentar algo, e é definido que aquela pessoa não se encontra mais na fila (nesse momento é executado um comando de *sleep*, que define a quantidade de tempo que a pessoa gastará utilizando o microondas).

`release_microwave`

De maneira análoga à função anterior, seleciona-se qual pessoa está executando essa atividade e, logo em seguida, bloqueia-se o *mutex* referente à fila e informa ao usuário que a pessoa X foi comer.Após isso, as informações da pessoa são atualizadas (ela é definida como não estando mais acompanhada e que não entrou na fila antes de seu parceiro) e é feita a busca por qual deve ser a próxima pessoa a utilizar o microondas (essa lógica será melhor detalhada em seção à parte dessa documentação).A partir do resultado dessa busca, podem ocorrer dois casos. O primeiro é quando não existe um deadlock e, portanto, a próxima pessoa da fila recebe um sinal para começar a utilizar o microondas. No segundo caso, há um deadlock, e o Raj recebe um sinal para liberar randomicamente alguma das pessoas envolvidas no deadlock (a resolução do deadlock será melhor detalhada mais à frente).Por fim, os dois *mutexes* são liberados, permitindo o prosseguimento do processo.

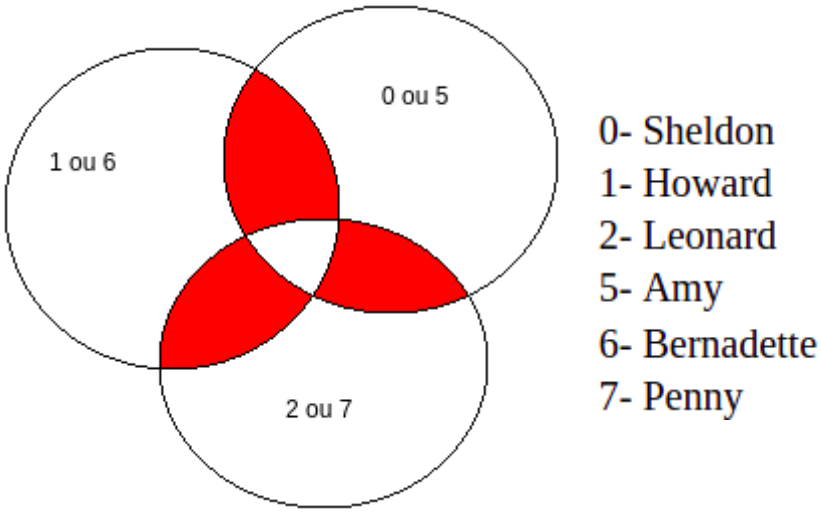
`go_home`

Essa função faz a *thread* “dormir” por um determinado intervalo de tempo, a fim de simular quando a pessoa termina de usar o microondas e vai para casa antes de voltar ao trabalho e querer usar o microondas novamente.
*activate_raj***:**Essa função faz o Raj ficar “observando” a fila enquanto todas as outras pessoas executam o número de interações com o microondas definido pelo usuário.A busca pela próxima pessoa a utilizar o microondas, mencionada na descrição da função *release_microwave*, retorna um valor específico quando ocorre um deadlock e, quando isso

acontece, o Raj resolve esse deadlock por meio da chamada da função *resolve_deadlock* (dentro do escopo de *activate_raj*).

H3 Escolha da *thread* a ser executada

A fim de escolher qual próxima *thread* da fila que será executada, primeiramente analisamos o caso no qual a pessoa que está utilizando o microondas estava acompanhada e havia entrado na fila antes de seu parceiro(a). Quando é esse o caso, a próxima pessoa a utilizar o microondas é o parceiro(a) da pessoa que está utilizando o microondas.
Caso contrário, implementamos uma lógica baseada em um diagrama de Venn, que funciona da seguinte forma:



Nós comparamos os casais dois a dois, e retornamos o membro de um dos dois casais que possui maior prioridade na fila (ou sinalizamos caso nenhum dos dois casais esteja na fila). Após realizar as comparações, temos três resultados (um para cada comparação), que são representados pela área vermelha no diagrama acima. Se os três resultados forem distintos, ou seja, cada resultado representa um membro de cada um dos casais (por exemplo 1, 5 e 2) isso implica que ocorreu um deadlock, e a ação do Raj é necessária. Por outro lado, se dois dos três resultados forem iguais, isso implica que aquela pessoa "repetida" é a próxima a usar o microondas. Por fim, se todos os resultados acusarem que nenhum membro dos casais comparados está na fila, isso indica que nenhum dos casais está na fila.No caso de nenhum dos membros dos casais estarem na fila, avaliamos se Stuart e/ou Kripke querem utilizar o microondas, e, caso queiram, liberamos suas respectivas *threads* conforme a regra descrita na documentação do Trabalho Prático (Kripke é sempre o último a executar).

H3 Resolução do Deadlock

A fim de resolvermos um eventual deadlock, primeiramente geramos um número aleatório entre 0 e 2 (esses números representam os personagens masculinos que possuem namoradas) e verificamos se a pessoa referente ao número gerado ou sua parceira estão na fila. Caso não estejam, são gerados números aleatórios (entre 0 e 2) até que essa condição seja válida.Após o número gerado ser válido, conferimos qual membro do casal está na fila (se é o homem ou a mulher), informamos ao usuário que um deadlock foi encontrado e que o Raj liberará o membro do casal que foi escolhido.Por fim, atualizamos o sistema de forma a definir o deadlock como resolvido, e enviamos um sinal à pessoa escolhida para que ela comece a utilizar o microondas.