Sistemas Operacionais

Especificação do Trabalho Prático (Implementação)

Orientações Gerais

O trabalho de implementação da disciplina de SO, a ser desenvolvido por 4 grupos com 3 (três) componentes e dois grupos com dois alunos, compreenderá as seguintes fases:

- Estudo teórico relacionado ao assunto do trabalho;
- Apresentação da solução teórica dada ao problema;
- Implementação da solução proposta, em qualquer linguagem de programação;
- Relatório explicando o processo de construção e o uso da aplicação.

Orientações Específicas

1 Problema

A Universidade de Brasília (UnB) ampliou o acesso ao ensino superior nos últimos anos. Como conseqüência, houve um considerável aumento na quantidade de alunos que freqüentam as instalações do campus Darcy Ribeiro, o principal desta instituição. Contudo, infelizmente, ainda não houve tempo para que houvesse uma ampliação na infraestrutura da universidade. Isso ocasionou, entre outros problemas, a falta de instalações para laboratórios e salas de aula.

Como medida emergencial, o novo reitor da UnB decidiu que iria reformar os banheiros masculinos, transformando-os em salas de aula. Em contrapartida, os banheiros femininos iriam ser mantidos, mas convertendo-os em banheiros comuns. Isso é, a universidade agora só terá um tipo de banheiro, que deve ser compartilhado entre homens e mulheres. Contudo, para evitar constrangimentos e demais problemas, é necessário que homens e mulheres não compartilhem certo banheiro ao mesmo tempo. Ou seja, quando uma ou mais mulher estiver no banheiro, outra mulher pode entrar, desde que respeitada a capacidade máxima da instalação. Da mesma forma, quando um ou mais homem estiver no banheiro, é permitido que outro homem entre, mas não uma mulher. O desafio aqui **é gerenciar a entrada das pessoas**, evitando **que somente homens ou somente mulheres monopolizem o recurso ao mesmo tempo**, e que o banheiro seja utilizado **em toda sua capacidade**.

Para resolver esse problema, a administração da universidade **resolveu lançar um desafio para os alunos da computação** que estivessem cursando a matéria de **sistemas operacionais** (ou seja, vocês)**.** O objetivo do desafio é ter boas soluções que sejam capazes de resolver o problema de maneira eficiente e eficaz. Para isso, os alunos vão precisar implementar uma solução, que deve ser desenvolvida como um programa multitarefa.

Nesse programa, cada pessoa que chegar ao banheiro deve ser representada por **uma tarefa distinta**. Além disso, no banheiro há ***n* cabines** que devem ser utilizadas por uma única pessoa por vez. Logo, há um **pool de *n threads*** executando concomitantemente, onde somente *threads* do tipo homem ou do tipo mulher podem compartilhar o espaço. Uma *thread* do tipo homem ou do tipo mulher possui dois estados possíveis. Ou seja, uma pessoa pode estar usando uma única cabine ou esperando para usar o banheiro.

Deve-se notar que nesse problema podem existir diferentes estratégias para escalonar as cabines no banheiro. Uma estratégia pode consistir em ser mais justa, permitindo que tanto homens quanto mulheres utilizem o banheiro de forma mais igualitária, enquanto outra pode ser mais eficiente, permitindo que mais pessoas utilizem o banheiro, embora um gênero possa ser mais privilegiado que outro.

1.1 Atribuições do Problema

- Homens e mulheres não podem usar o banheiro ao mesmo tempo;
- Cada pessoa, seja homem ou mulher, gasta algum tempo esperando o banheiro;
- Cada pessoa, seja homem ou mulher, gasta algum tempo usando uma cabine do banheiro;
- As pessoas não gostam de ficar esperando para usar o banheiro;
- Algumas pessoas utilizam o banheiro em mais ou menos tempo que outras.

1.2 Considerações do Problema

Nesse problema, os principais fatores a serem considerados são o desempenho, a justiça (ou igualdade de gênero), a negação de uso, a garantia de não ter *deadlock* e nem *starvation.*

O **desempenho** é medido pelo tempo médio de espera por pessoa esperando para usar o banheiro. Quanto menor o tempo de espera médio por pessoa, independentemente do gênero, melhor o desempenho. O tempo médio de espera é calculado entre o momento em que uma pessoa requisita o banheiro e o momento em que ela consegue utilizá-lo.

A **justiça** (ou igualdade de gênero) consiste em permitir que tanto homens quanto mulheres possam utilizar o banheiro, evitando que haja espera eterna para certo tipo de gênero. Uma estratégia justa deve permitir que todas as pessoas entrem no banheiro, o que pode, eventualmente, implicar numa diminuição do desempenho. Deve-se lembrar que na UnB existem diversos movimentos sociais. Caso a solução prejudique o acesso de mulheres e favoreça o acesso de homens ao banheiro, por exemplo, ela poderá causar transtornos no ambiente acadêmico.

A **negação** de uso é um problema que pode aparecer ao escolher o critério para aplicação da justiça, definido acima. Deve-se ter cuidado para que não haja um individuo que nunca tem a

chance de utilizar o banheiro. Por exemplo, se o banheiro está em uso por pessoas de um determinado gênero e a estratégia de escalonamento permite que mais membros daquele gênero continuem usando o banheiro, então às pessoas do gênero oposto é negado o uso. Esse não é um caso de *deadlock*, uma vez que mesmo que se algumas pessoas fiquem sem utilizar o banheiro, ele ainda poderá ser utilizado pelas pessoas do outro gênero. A negação de uso é um critério que deve ser implementado, pois é da própria natureza do problema segregar o uso concomitante do banheiro entre gêneros distintos. Contudo, deve-se tomar cuidado como esse critério é escolhido.

O ***deadlock*** é um problema que pode aparecer, dependendo dos critérios implementados na negação de uso e na justiça. É o estado onde todas as pessoas que estão esperando para usar o banheiro nunca tenham as condições para usar. Isso significa que cada pessoa está esperando por uma cabine ficar disponível. Em teoria, como existem $n \geq 1$ cabines, ao menos uma pessoa pode ter a chance de usar o banheiro e o *deadlock* nunca ocorre. Contudo, quando são colocadas as restrições para evitar que duas pessoas utilizem a mesma cabine e que o banheiro só tenha pessoas do mesmo sexo, *deadlocks* podem ocorrer, por isso, devem ser evitados.

O ***starvation*** é um problema que pode ocorrer porque há a necessidade de garantir o uso exclusivo do banheiro por gênero, assim, se houver uma pessoa do gênero masculino, por exemplo, esperado na filha, e quando estiver chegando a vez dele, chegar uma pessoa do gênero feminino e entrar no banheiro, pode ser que essa situação aconteça sempre que o homem estiver prestes a entrar, e com isso, ele ter que ficar “eternamente” na fila. Para evitar situações constrangedoras nas filas dos banheiros da UnB, essa situação também deve ser evitada.

1.3 Estratégias para Implementação

Para este trabalho, pode-se utilizar diferentes estratégias para organizar as pessoas e a utilização das cabines. A solução ideal seria aquela que não tivesse *deadlocks*, evitasse negações, garantisse que nunca teria *starvation*, pudesse ser 100% justa e minimizasse ao máximo o tempo de espera das pessoas na fila dos banheiros. Obviamente, todas as estratégias para resolver o problema aqui proposto devem evitar *deadlocks* e *starvation*. Contudo, visando otimizar o desempenho da solução, não é possível obter justiça para 100% das pessoas, pois uma estratégia 100% justa, na qual tanto homens quanto mulheres possam ser atendidos logo que cheguem no banheiro, terá pouca performance. Da mesma maneira, uma estratégia com o máximo de desempenho pode não ser justa. A solução deve buscar balancear desempenho e justiça. Para isso, a solução implementada deve garantir uma boa relação entre justiça x desempenho. Assim sendo, as soluções podem seguir adaptações entre os seguintes critérios:

1. Pessoas que chegam ao banheiro irão utilizar as cabines de acordo com a ordem de chegada (FIFO, *First In, First Out*). Essa estratégia é 100% justa para todos os indivíduos. Se alguém do gênero oposto daquele que está utilizando o banheiro chegar, deve esperar até que o banheiro fique vazio. Nesse caso, há uma fila única de espera.
2. Se uma ou mais pessoa do gênero oposto está no banheiro, então deve esperar que o banheiro fique vazio. Enquanto isso, ***n-1*** pessoas do sexo oposto que estão esperando podem entrar no banheiro a medida que as pessoas no banheiro vão saindo. Essa estratégia não é 100% justa, mas reduz a espera ocupada das cabines. No caso, duas filas de espera são criadas, uma para cada gênero. Quando o banheiro estiver vazio, até ***n*** membros do gênero oposto podem entrar.
3. Se o banheiro estiver em uso por pessoas do gênero oposto, então a pessoa que chega deve esperar até o banheiro ficar vazio. Para evitar a **espera ocupada**, aplica-se a negação de acesso, evitando-se que novos membros do sexo oposto entrem no banheiro. Essa estratégia não é muito justa, pois algumas pessoas podem usar o banheiro antes de outras do gênero oposto que chegaram anteriormente. Contudo, essa estratégia é mais justa do que a anterior, pois um indivíduo do gênero oposto será a próxima pessoa a entrar. Nesse caso, existem duas filas de espera. Uma para cada gênero. Quando o banheiro estiver vazio, até ***n*** pessoas de um gênero podem entrar.

1.4 *Threads* (Pessoas)

Cada pessoa, seja homem ou mulher, deve ser representada por uma *thread*. Essas *threads* possuem dois estados: ou está esperando, ou está usando o banheiro. A lógica simples é representada abaixo

```
While(true) {
    Foreach(thread in Pessoa) {
        Espera(); // Estado 1: espera para usar o banheiro
        Adquire_cabine(); // mudança de estado
        Usa_banheiro();  // Estado 2
        Libera_cabine();  // Fim do segundo estado, a thread pode ser finalizada
    }
}
```

Nas simulações, tanto o estado de espera quanto o de uso do banheiro deve ser um tempo aleatório. Fixar o tempo garante que cada pessoa irá gastar uma quantidade de tempo previsível, o

que facilita a solução da implementação, mas não é realístico. Sendo assim, pode-se assumir que o tempo de uso seja determinado por um intervalo aleatório razoável.

1.5 Cabines

Além de gerenciar a entrada ao banheiro a fim de permitir que somente homens ou somente mulheres o compartilhem, é necessário gerenciar a distribuição das pessoas nas cabines, a fim de que duas pessoas não concorram a mesma cabine. Em termos de sistemas operacionais, deve-se evitar **condições de corrida**. Logo, nas simulações, as cabines são recursos disputados que devem ser representadas por posições de memória compartilhada. Mecanismos como semáforos *mutexes*, monitores ou passagem de mensagens devem ser utilizados para sincronizar esses recursos.

2 Estudo Teórico para a Solução

Cada grupo deverá buscar a solução para o compartilhamento e sincronização de recursos, de acordo com o problema proposto. É responsabilidade de cada grupo estudar a maneira mais eficiente para implementar a solução proposta. Contudo, é importante ressaltar que para o problema de compartilhamento de recursos não há soluções mágicas, as soluções possíveis são exatamente as mesmas vistas em sala de aula: semáforos (baixo nível), monitores (alto nível, mas devem ser implementados pela linguagem de programação, pois o compilador deve reconhecer o tipo monitor) e troca de mensagens (usadas preferencialmente para garantir a sincronização entre processos em máquinas diferentes, também podem ser usadas para processos na mesma máquina. Nesse caso, as mensagens trocadas passam por toda a pilha de protocolos como se estivessem sendo enviadas para outra máquina).

3 Implementação

O trabalho valerá 10 (dez) pontos, sendo 9 (nove) pontos dados à corretura dos algoritmos e 1 (um) ponto dado à qualidade do código. Portanto, o código-fonte deve seguir as bias práticas de programação: nomes de variáveis auto-explicativos, código comentado e identado. As métricas de medida a serem usada para avaliar a qualidade do código serão baseadas naquelas descritas em [2]. Dessa forma, funções grandes, classes com baixa coesão, código mal-identado e demais práticas erradas acarretarão no prejuízo da nota.

Contudo, é fundamental que apenas os padrões das linguagens, sem bibliotecas de terceiros, sejam utilizados. Ou seja, se você for executar um programa escrito em Java no Linux, que eu precise apenas do SDK padrão para recompilá-lo, se o programa for escrito em C, que eu

possa compilá-lo com o padrão –ansi ou c99, etc. As especificações de padrão devem ser explicitadas. Isso é só para evitar problemas de rodar na máquina de vocês e não rodar na minha. Padronização é fundamental!

A implementação poderá ser feita em qualquer linguagem e utilizando qualquer biblioteca, desde que o código seja compatível com ambientes UNIX. Programas que utilizarem bibliotecas ou recursos adicionais deverão conter no relatório informações detalhadas das condições para a reprodução correta do programa. Além disso, bibliotecas proprietárias não são permitidas.

4 Responsabilidades

Cada grupo é responsável por realizar, de maneira exclusiva, toda a implementação do seu trabalho e entregá-lo funcionando corretamente. Além disso, o grupo deverá explicar detalhamente todo o código desenvolvido. É fundamental que **todo o grupo** esteja presente no dia/horário definido para apresentar e explicar o código-fonte.

5 Material a ser entregue

O grupo deverá, além de apresentar o código executando, entregar um relatório impresso (mínimo de 3 e máximo de 4 páginas) contendo, obrigatoriamente, **no mínimo,** os seguintes itens:

- Descrição das ferramentas/linguagens usadas;
- Descrição teórica e prática da solução dada;
- Descrição das principais dificuldades encontradas durante a implementação;
- Para todas as dificuldades encontradas, explicar as soluções usadas;
- Descrever o papel/função de cada aluno na realização do trabalho;
- Bibliográfia.

6 Cronograma

O trabalho deverá ser enviado pelo Moodle até o dia 21/07/2022 às 23:59 h e apresentado no dia **0**

22/07/2022, das 8:00 às 9:40, no Laboratório de Projetos Especiais, no prédio do CiC/EST. Nenhum trabalho será recebido fora do prazo, por qualquer que seja o motivo. É obrigatória a presença de todo o grupo para a explicação do código desenvolvido.

7 Referências

[1] Stalling, W. *Operating Systems Internals and Design Principles*, Pearson Education, 2009.

[2] Martin, R. C. *Clean Code. A Handbook of Agile Software Craftsmanship*, Prentice Hall, 2008.