Aluno: Thierry Pierre Dutoit **DRE:** 119040432

LABORATÓRIO 4

Atividade 1:

1. A ideia do Programa é que inicialmente as duas Threads A imprimam “Hello” para que a Thread B em seguida imprima “Bye”.
2. Sim, sempre foram 2 “Hello” e um “Bye” nessa ordem. Portanto, a condição lógica é sempre atendida visto que se as duas Threads A não executarem antes da Thread B, ela se auto bloqueará. Dessa forma, quando as Threads A terminam, elas dão o sinal para a Thread B prosseguir e imprimir “Bye”.
3. Quando mudamos o número de Threads A para 1 a condição de desbloqueio da Thread B nunca será satisfeita (x == 2). Com isso, a Thread B não terminará sua execução e o join no fluxo principal nunca será satisfeito, fazendo com que o programa nunca termine.
4. Ao alterar o programa para o seu estado inicial ele volta a funcionar perfeitamente, satisfazendo a condição lógica.
5. Ao alterar o número de Threads B para 2 precisamos, além de possibilitar a criação da nova Thread B, trocar o signal na Thread A para broadcast. Assim, as Threads B não executarão antes da impressão dos 2 “Hello” e a Thread A possibilitará, após satisfazer x == 2, que todas as Threads B bloqueadas possam prosseguir.

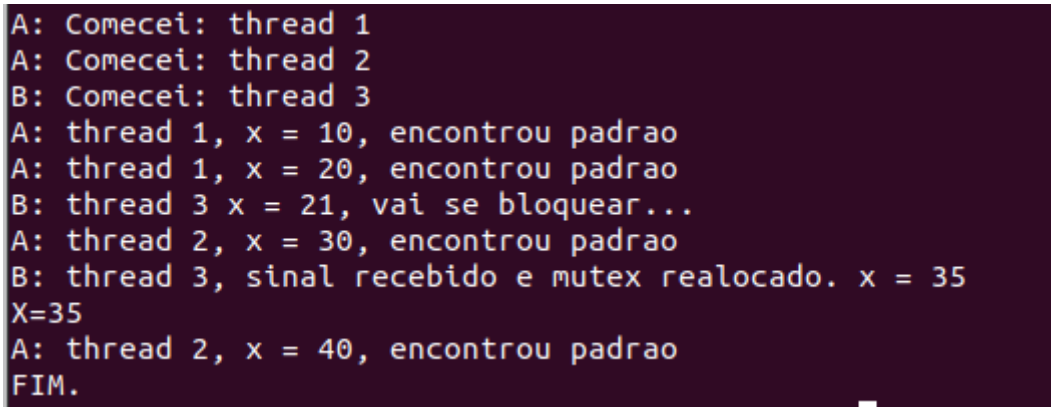
Atividade 2:

1. O programa tem a Thread B imprimindo “Hello” seguida das duas Threads A imprimindo “Bye” cada uma.
2. Sim, foi sempre o esperado pois as Threads A se bloqueiam ao executarem antes da Thread B (visto que x == 0), e a Thread B imprime “Hello” e dá um sinal de broadcast para as Threads A, ou seja, para

qualquer número de Threads A bloqueadas elas serão liberadas, imprimindo “BYE”.

Atividade 3:

1. A aplicação consiste no valor da variável global x ser somado de 1 em 1 até TCOUNT para cada Thread A, e enquanto essa soma ocorre podemos ter a execução da Thread B, que imprimirá o valor de x caso seja divisível por 10. Entretanto, caso a Thread B esteja bloqueada por não ter encontrado um valor de x divisível por 10, a Thread A manda um sinal de desbloqueio sempre que x for divisível por 10.
2. Sim, sempre que a Thread B tentava imprimir um valor de x que não fosse divisível por 10 ela era bloqueada. Quando a Thread B encontrava um valor de x divisível por 10, ela o imprimia e não executava mais. Um ponto a ser notado é que a Thread B sempre executa porque a Thread A termina deixando o valor de x como múltiplo de 10 e passando o sinal de desbloqueio.
3. Não podemos substituir as duas linhas pois quando a Thread B recebe o sinal de desbloqueio ela não necessariamente começa logo em seguida. Com isso, podemos ter o caso em que a Thread A esteja alterando o valor de x e a Thread B ao retomar sua execução, não satisfaça a condição que x é divisível por 10. Com o uso do while reforçamos a verificação para que seja impresso um múltiplo de 10.
4. Agora temos a possibilidade de imprimir um valor de x sem ser divisível por 10, não satisfazendo a condição lógica da aplicação. Um exemplo será mostrado a seguir:



No caso acima temos a Thread B sendo bloqueada por não encontrar um valor divisível por 10. Em seguida, uma Thread A encontra seu padrão e desbloqueia a Thread B, que volta de onde estava bloqueada (wait) e segue a execução, pois não precisa satisfazer a divisibilidade por 10, imprimindo X = 35.