Sistemas Operacionais

Trabalho de Programação #2

Período: 2022/2

Data de Entrega: 29/01/2023 ***Trabalho em Grupo!***
(Obs: as entrevistas para a apresentação do trabalho serão marcadas posteriormente)

Material a entregar

1. Todos os arquivos criados, incluindo *makefile*, com o código muito bem comentado. **Atenção: adicionar nos comentários iniciais do makefile a lista com os nomes completos dos componentes do grupo!**

   Valendo ponto: clareza, indentação e comentários no programa.

2. Vídeo explicando a solução do grupo, incluindo os testes de funcionalidades.

DESCRIÇÃO DO TRABALHO

Parte 1 - Controlando o acesso a um banheiro (Monitores e Threads)

Nesta primeira parte do trabalho você exercitará a programação *multithreading* explorando a biblioteca *pthreads* (POSIX Threads). É também objetivo desta parte exercitar a construção da estrutura de programação "Monitores". Para isso, você é solicitado a desenvolver um sistema para controlar o acesso a um banheiro de um bar, conforme descrição abaixo. Você deve desenvolver a sua solução em linguagem C, usando o padrão ***pthreads*** **(POSIX Threads)** para Linux 2.x. Você usará funções básicas de gerenciamento de *threads* (*pthread_create*,...), *mutexes* (*pthread_mutex_*...) e variáveis de condição (*pthread_cond_*...). Não é permitido o uso de outras bibliotecas de suporte à programação *multithread* (bibliotecas de semáforos, por exemplo).

O Problema...

Em um pequeno bar próximo a um estádio de futebol cabem até 20 clientes. No dia da decisão da Série A envolvendo Flamengo x Vasco, o dono estabeleceu que o bar comportaria no máximo 10 vascaínos e 10 flamenguistas, para não dar confusão. O bar possui apenas um banheiro, no qual cabem no máximo três pessoas. De acordo com as regras estabelecidas pelo dono do bar, o banheiro só pode ser ocupado por torcedores do mesmo time. Uma pessoa, ao tentar entrar no banheiro, ficará esperando se o banheiro estiver ocupado por pessoas do time oposto ou se estiver lotado. Sempre que houver um vascaíno na fila, ele(a) terá preferência sobre o(a)s flamenguistas para entrar no banheiro.

Foram instalados dois leitores de cartão magnético na porta do banheiro - um na entrada e outro na saída - para controlar o acesso das pessoas. Para abrir a porta, o(a) torcedor(a) deve passar o seu crachá pelo leitor de cartão.

Você foi incumbido de desenvolver o sistema de controle de acesso ao banheiro. Implemente o sistema na forma de um Monitor, usando *variáveis mutex* e *variáveis de condição*.

O Monitor possui os seguintes procedimentos de entrada:

- `void flamenguistaQuerEntrar()`
- `void flamenguistaSai()`
- `void vascainoQuerEntrar()`
- `void vascainoSai()`

Flamenguistas e vascaínos são *threads* que chamam esses procedimentos. Sempre que uma pessoa entrar no banheiro (ou seja, a sua respectiva *thread* finaliza a execução do procedimento `flamenguistaQuerEntrar()` ou `vascainoQuerEntrar()`) , o Monitor deve imprimir na tela o número de pessoas de cada time no banheiro (isso quer dizer que este "`printf`" deve ser o último comando antes do retorno desses dois procedimentos).

Implemente o Monitor na forma de um módulo (arquivo "`MonitorBanheiro.c`" separado), o qual exporta apenas os procedimentos de entrada. Este módulo deve ser implementado seguindo a filosofia dos monitores, ou seja, deve usar variáveis mutex para garantir exclusão mútua entre diferentes threads que chamam esses procedimentos simultaneamente. Além disso, quando uma thread fica bloqueada (seja porque o banheiro está ocupado por pessoas do time oposto ao seu, seja porque ele está lotado) usar para isto variáveis de condição.

Para *thread* o sistema, implemente um segundo módulo (`Torcedores.c`) onde são criadas 20 threads que representam o(a)s torcedore(a)s de cada time (10 flamenguistas e 10 vascaíno(a)s).

Código das threads de flamenguistas:

```
void thread_flamenguista(int *id)  // cada flamenguista terá um identificador de 1 a 10

{
 while (true){
       printf ("Eu sou flamenguista%d:...Estou apertado(a)! Vou no

       banheiro!\n",*id); flamenguistaQuerEntrar()

       printf ("Eu sou flamenguista-%d: ... UFA! Entrei no

       banheiro!\n",*id); sleep(5);

       flamenguistaSai();

       printf ("Eu sou flamenguista-%d: ... Estou aliviado! Vou

       torcer!\n",*id); sleep(5);

   }

}
```

Faça de modo similar para as threads de vascaíno(a)s:

```
void thread_vascaino (int *id) { ... }
```

Problema 2 - Dancei no Rock in Rio… (Semaphores POSIX)

Podemos encontrar dois tipos de semáforos em ambientes *Unix- like*: System V and POSIX. Em geral, sistemas mais antigos usam a versão System V e sistemas Li*nux-based* usam a versão POSIX, sendo a curva de aprendizado deste último bem menor.

Nesta parte do trabalho você deverá resolver o problema abaixo empregando semáforos POSIX.

O Problema...

Dizem por aí que durante uma certa edição do *Rock In Rio*, muitos estudantes da UFES tentaram comprar o ingresso no local do show com cambistas, mas eles relataram que esses ingressos acabaram logo. Então, os jovens estudantes acabaram descobrindo que havia um "esquema" para entrar no evento … em uma localidade vizinha ao local do show. Lá havia um barqueiro com um pequeno barco no qual cabiam 3 pessoas (além do barqueiro). Entre os fundos do local do show e esta localidade havia um lago (muito sujo...), e o barqueiro se propôs a fazer a travessia do lago pela módica quantia de R$100,00/pessoa. À medida que os jovens iam chegando, eles entravam no barco até que o mesmo ficasse cheio. O barqueiro então recolhia a "ajuda de custo" e saía com o barco para a travessia.

Chegando na outra margem, o barqueiro liberava a saída dos jovens roqueiros do barco. No entanto, visto que havia um matagal com uma grande cerca nos fundos do local do show, o barqueiro esperava que todos tivessem conseguido atravessar o mato e a cerca com sucesso, para então voltar com o barco e fazer uma nova travessia.

Claro que essa história (dizem que com h mesmo, pois pessoas conhecidas juram que aconteceu ...) não teve um final feliz já que, em um dado momento, um dos jovens, que não era tão jovem assim (seria um jovem professor?), ao atravessar o matagal, ficou "agarrado" na cerca... o que acionou a segurança do local e acabou com o esquema do barqueiro!

A Tarefa ...

Vocês terão que modelar essa situação através de “processos/*threads*”, escrevendo um programa concorrente na linguagem C, no Linux. Vocês devem criar dois tipos de processos/*threads*, um

representando os jovens e o outro representando o barqueiro. Considere a existência de 15 (quinze) jovens e 1 (um) barqueiro e que a sincronização das ações dos jovens e barqueiro será feita através de *semáforos POSIX.*

O jovem quando chega no local, tenta entrar no barco se houver vaga; se não houver, ele espera em uma fila até que o barco volte. Uma vez no barco, ele espera o barqueiro fazer a travessia e chegar ao outro lado da margem. Do outro lado, ele tenta então atravessar o matagal. Se ele conseguir, "termina" dentro do show. Se ele não conseguir atravessar, ele "termina" preso pelos seguranças e, neste momento, todos os outros processos deverão ser finalizados. Para tanto, o processo jovem terá um atributo para indicar se o mesmo é capaz ou não de atravessar o matagal. Atenção: durante a criação dos 15 jovens, 1 (um) deles, escolhido de forma aleatória, NÃO deve ser capaz de atravessar o matagal.

Algumas regras básicas:

1. A travessia do lago deve durar 3s de ida, e 2s na volta. A travessia do matagal deve durar 1s para cada jovem.

2. Cada atividade de um processo deverá ser reportada através de uma mensagem. Alguns exemplos de mensagens são:

   Jovem 01 vai tentar pegar o barco...
   Jovem 01 entrou no barco e espera que este saia.
   Jovem 01 sai com o barco.
   Jovem 03 vai tentar pegar o barco...
   Jovem 03 entrou na fila, tem 1 jovem na sua frente.
   Jovem 03 saiu da fila e entrou no barco...
   Barqueiro recolhe a ajuda de custo e sai com o barco...
   Barqueiro chega na outra margem e libera os 3 passageiros...
   Barqueiro espera todos atravessarem o matagal...
   Tudo certo! Barqueiro retorna para fazer outra viagem...
   Jovem 05 atravessando matagal...
   Jovem 05 entrou no show...
   Jovem 06 preso na cerca... Shii! Os seguranças me pegaram!
   Opa! Problemas! Os seguranças pegaram um! Acabou meu esquema...
   Opa! Os seguranças pegaram um ... não vou poder entrar no show... o jeito vai ser o telão mesmo!

   ... E para fechar: antes de "morrer" o barqueiro deverá criar um *Named Pipe* no qual o mesmo deixará uma mensagem para a prosperidade:

   "*O caminho mais curto nem sempre é o melhor ...*"