Sistemas Operacionais
Trabalho #2 de Programação
Período: 2023/1

Data de Entrega: 15/07/2023
Trabalho em Grupo!
Obs: as entrevistas 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!

2. Valendo ponto: clareza, indentação e comentários no programa.

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

DESCRIÇÃO DO TRABALHO

Parte I. O Problema dos Macacos

Suponha que haja macacos em ambas as margens de um rio e, de tempos em tempos, os macacos decidem passar para o outro lado à procura de comida. A passagem para a outra margem do rio é feita através de uma ponte de corda. Mais de um macaco pode atravessar a ponte ao mesmo tempo, mas isso só é possível se eles estiverem indo na mesma direção.

Implemente um programa que faça o controle da passagem de macacos pela ponte usando Semáforos. Para testar o sistema, crie 10 threads que representem os macacos, colocando inicialmente metade deles em cada margem do rio. Sempre que um macaco iniciar ou concluir a travessia, imprima uma mensagem na tela identificando o macaco.

Após testar o programa acima, crie agora uma nova versão do programa adicionando dois gorilas, um de cada lado do rio. Como os gorilas são muito pesados, eles só poderão atravessar a ponte sozinhos. Como os outros macacos têm medo dos gorilas, eles terão prioridade para fazer a travessia.

Parte II. Implementando Monitores

Em uma pequena empresa de marketing esportivo trabalham 20 pessoas - 10 corintianos e 10 palmeirenses. A sede desta empresa possui apenas um banheiro, no qual cabem no máximo três pessoas. De acordo com as regras estabelecidas pela empresa, para não dar confusão, o banheiro só pode ser ocupado por pessoas que torcem para o mesmo time. Uma pessoa, ao tentar entrar no banheiro, ficará esperando se o banheiro estiver ocupado por pessoas do time oposto ou se ele

estiver lotado. Sempre que houver um corintiano na fila, ela terá preferência sobre os palmeirenses para entrar no banheiro (hehehe).

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 funcionário deve passar o seu crachá de identificação pelo leitor de cartão.

Você foi incumbido de desenvolver um programa para controle de acesso ao banheiro. Implemente o programa na forma de um Monitor, usando mutexes e variáveis de condição.

O Monitor possui os seguintes procedimentos de entrada:

- *void corintianoQuerEntrar()*
- *void conrintianoSai()*
- *void palmeirenseQuerEntrar()*
- *void palmeirenseSai()*

Corintianos e palmeirenses são threads que chamam esses procedimentos. Sempre que uma pessoa entrar no banheiro (ou seja, a sua respectiva thread finaliza a execução do procedimento *corintianoQuerEntrar()* ou *palmeirenseQuerEntrar()*) , 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. Esse 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 testar o sistema, implemente um segundo módulo (*Funcionarios.c*) em que são criadas 20 *threads* que representam os funcionários da empresa (10 corintianos e 10 palmeirenses).

Código das *threads* dos corintianos:

```
void thread_corintiano(int *id)  // cada corintiano e palmeirense
terá um identificador de 1 a 10

{
 while (true){

       corintianoQuerEntrar();

       printf ("Eu sou corintiano-%d: ... UFA! Entrei no
banheiro!\n",*id);

       sleep(3);
```

```
        corintianoSai();

        printf ("Eu sou corintiano-%d: ... Estou aliviado! Vou
trabalhar!\n",*id);

        sleep(5);

    }

}
```

Faça de modo similar para as threads dos palmeirenses:

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

Bibliografia: (ver referências adicionais no site da disciplina)

[1] Kay A. Robbins, Steven Robbins, ***UNIX Systems Programming: Communication****, Concurrency and Threads*, 2nd Edition (Cap 1-3).

[2] W. Richard Stevens, Stephen A. Rago, ***Advanced Programming in the UNIX Environment***, Addison-Wesley.