Sistemas Operacionais "A"
Profa. Patrícia Pitthan
Trabalho Prático: Sincronização

informática

Observações:

- O trabalho pode ser realizado individualmente ou em duplas.
- O trabalho deve ser implementado em C, ambiente Linux, utilizando semáforos e *threads*.
- O programa deve estar devidamente comentado !!!
- O arquivo .c que implementa o trabalho deve ser enviado (pitthan@inf.ufsm.br), utilizando como Assunto da mensagem: SO-sincr-<nome_do_aluno1, nome_aluno2>.
- Data de entrega: até às 23:59 do dia **20/05/2015**.
- A apresentação do trabalho será agendada posteriormente.

Sincronização de threads

Um bar resolveu liberar um número específico de rodadas grátis para seus ***n*** clientes presentes no estabelecimento. Esse bar possui ***x*** garçons. Cada garçom consegue atender a um número limitado (***C***) de clientes por vez. Como essas rodadas são liberadas, cada garçom somente vai para a copa para buscar o pedido quando todos os ***C*** clientes que ele pode atender tiverem feito o pedido ou não houver mais clientes a serem atendidos. Após ter seu pedido atendido, um cliente pode fazer um novo pedido após consumir sua bebida (o que leva um tempo aleatório) e a definição de uma nova rodada liberada. Uma nova rodada somente pode ocorrer quando foram atendidos todos os clientes que fizeram pedidos. Por definição, nem todos os clientes precisam pedir uma bebida a cada rodada.

Implemente uma solução que permita a passagem por parâmetro *(i)* o número de clientes presentes no estabelecimento, *(ii)* o número de garçons que estão trabalhando, *(iii)* a capacidade de atendimento dos garçons e *(iv)* o número de rodadas que serão liberadas no bar. Cada garçom e cada cliente devem ser representados por *threads,* estruturalmente definidos como os pseudo-códigos que seguem:

```
thread cliente {
  while (!fechouBar){
    fazPedido();
    esperaPedido();
    recebePedido();
    consomePedido(); //tempo variavel
  }
}
```

```
thread garçom {
  while (existemClientesNoBar){
    recebeMaximoPedidos();
    registraPedidos();
    entregaPedidos();
    rodada++;  //serve como param. p/ fechar o bar
    }
}
```

- O trabalho deve, necessariamente, usar semáforos para controlar a concorrência e a sincronização entre as *threads*.
- A ordem de chegada dos pedidos dos clientes na fila de pedidos de cada garçom deve ser respeitada. Sua solução não deve permitir que clientes furem essa fila.
- O garçom só pode ir para a copa quando tiver recebido seus *C* pedidos.
- O programa deve exibir a evolução de modo a permitir o acompanhamento da execução. Sendo assim, deve ficar claro o que está acontecendo no bar a cada rodada. Os pedidos dos clientes, os atendimentos pelos garçons, os deslocamentos para o pedido, a garantia de ordem de atendimento, etc.