Sincronização entre threads

O bar *Pega Leve* decidiu liberar um número específico de rodadas gratuitas para seus ***n*** clientes presentes no estabelecimento. O bar possui ***x*** garçons e cada garçom consegue atender a um número limitado de clientes (***C***) por vez. No caso das rodadas liberadas, cada garçom apenas vai buscar o pedido na copa quando todos os ***C*** clientes que ele pode atender tiverem feito o pedido ou não houver mais nenhum cliente a ser atendido. Após ter seu pedido atendido, um cliente pode fazer um novo pedido após consumir sua bebida (o consumo da bebida leva um tempo aleatório) e após ser definida uma nova rodada liberada. Uma nova rodada somente pode ocorrer quando foram recebidos todos os pedidos dos clientes. Por definição, nem todos os clientes precisam pedir uma bebida a cada rodada.

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

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

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

- O trabalho deve, necessariamente, usar semáforos para controlar a concorrência e a sincronização entre as *threads*. Para evitar conflito e condições de disputa, a implementação do programa deve ser feita utilizando semáforos (primitivas sem_init, sem_wait, sem_post,…) da biblioteca *pthreads*, linguagem C, ambiente Linux. Não utilizar pthread_lock e pthread_unlock.
- 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 somente pode ir para a copa quando tiver recebido seus ***C*** pedidos.
- O programa deve exibir a evolução (trace) 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.

Orientações e informações importantes:

- Trabalho individual.
- **Não** devem ser inseridos comentários no código.
- O programa deve estar modularizado (funções) e utilizar passagem de parâmetros (evitar variáveis globais quando possível).
- As dúvidas dos alunos devem se ater à especificação (o que fazer) e não a decisões de implementação (como fazer).
- Não serão analisados códigos .c antes do período de entrega. Favor não insistir.

- O arquivo **sincroniza.c**, que implementa a solução do trabalho, deve ser enviado pelo Moodle. Não serão aceitos trabalhos enviados fora da plataforma Moodle. Não serão aceitos trabalhos após o prazo.
- A apresentação do trabalho será realizada conforme agendamento. Trabalhos não apresentados serão desconsiderados.