

ANDREA MINDLIN TESSLER 10381702
Lucas Kenzo Kawamoto 10396359

Projeto 1 SO

Link github : <https://github.com/LUCASBR8/SO-PROJETO/tree/main>

Explicação:

Semana 1 - Uso de Threads:

Threads dos clientes são criadas para fazer pedidos, cada um em sua própria thread.

Os pedidos são processados por threads que representam chefs.

Semana 2 - Sincronização Básica com Mutex:

O acesso à fila de pedidos compartilhada é protegido por um mutex para evitar condições de corrida, garantindo que apenas uma thread manipule a fila de pedidos por vez.

Semana 3 - Comunicação entre Threads com Variáveis de Condição:

As threads dos chefs e dos clientes se comunicam por meio de variáveis de condição.


Clientes esperam por espaço na fila para fazer o pedido e notificam os chefs quando há um novo pedido disponível.

Os chefs esperam por novos pedidos e notificam quando o pedido foi processado, liberando espaço na fila.

Semana 4 - Sincronização Avançada e Controle de Concorrência com Semáforos:

Um semáforo controla o número de chefs disponíveis. O semáforo garante que o número de chefs simultaneamente processando pedidos não exceda o limite definido.

Cada chef deve esperar até que o semáforo permita processar um novo pedido, garantindo o controle eficiente do recurso "chef".

 **OnlineGDB**

online compiler and debugger for c/c++

code, compile, run, debug, share.

IDE

My Projects

Classroom new

Learn Programming

Programming Questions

Sign Up

Login

main.c

```
116 // Espera todas as threads dos clientes terminarem
117 for (int i = 0; i < NUM_CLIENTES; i++) {
118     pthread_join(threads_clientes[i], NULL);
119 }
120
121 // Sinaliza que todos os clientes já foram atendidos
122 pthread_mutex_lock(&mutex_fila);
123 finalizar = 1;
124 pthread_cond_broadcast(&cond_novo_pedido); // Notifica todas as threads dos chefs
125 pthread_mutex_unlock(&mutex_fila);
126
127 // Espera todas as threads dos chefs terminarem
128 for (int i = 0; i < NUM_CHEFS; i++) {
129     pthread_join(thread_chef[i], NULL);
130 }
131
132 // Destrói mutex, condições e semáforo
133 pthread_mutex_destroy(&mutex_fila);
134 pthread_cond_destroy(&cond_novo_pedido);
135 pthread_cond_destroy(&cond_pedido_pronto);
136 sem_destroy(&sem_chefs);
137
138 printf("Todos os clientes foram atendidos.\n");
139 return 0;
140
141
142
```

close ad

LEARN MORE

Input

```
Pedido do Cliente 8 está pronto!
Cliente 10 fez o pedido. (Pedidos na fila: 1)
Chef está preparando o pedido do Cliente 10. (Pedidos restantes: 0)
Pedido do Cliente 9 está pronto!
Pedido do Cliente 10 está pronto!
Todos os clientes foram atendidos.

...Program finished with exit code 0
Press ENTER to exit console.
```

About • FAQ • Blog • Terms of Use • Contact Us •

GDB Tutorial • Credits • Privacy

© 2016 - 2024 GDB Online