Trabalho 1 (T1)

INE 5410 - Programação Concorrente

Trabalho I - Simulador de Restaurante

Data de Entrega pelo Moodle: 1 de Outubro de 2018

Datas de Apresentação: 3 e 5 de Outubro de 2018

Enunciado

Dom Mario, dono da Trattoria di Mario, contratou vocês para implementarem um simulador de restaurante. Dom Mario quer saber quantos cozinheiros precisa contratar e quantos fogões precisa comprar para abrir filiais do seu restaurante.

O trabalho deve ser realizado em C usando *pthreads*. O simulador deve receber os seguintes argumentos de linha de comando:

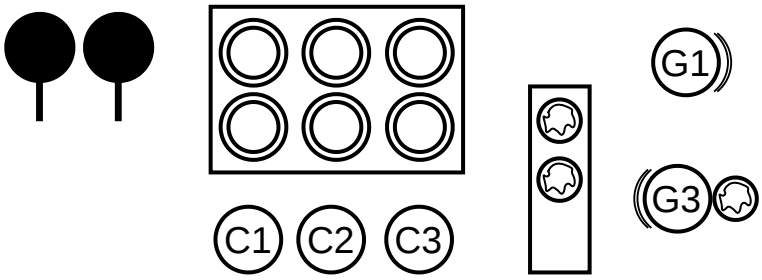
1. O número de **cozinheiros**;
2. O número de **fogões**;
3. O número de **bocas** de cada um desses fogões;
4. O número de **frigideiras**;
5. O número de **garçons**;
6. Quantos pratos cabem no **balcão** de pratos prontos da cozinha.

O programa deverá ler uma sequência de pedidos da entrada padrão. O sobrinho de Dom Mario **já implementou** um *parser* dos pedidos pra você, mas caiu de um penhasco quando estava de férias na Sicília e não pôde terminar o programa. Uma invocação do programa deve se parecer com o abaixo:

```
$ ./program --cozinheiros 3 --bocas 6 --fogoes 1 --frigideiras 2 --garcons 3 --balcao 3
Pedido 1 (SPAGHETTI) submetido!
Pedido 2 (CARNE) submetido!
Pedido 3 (SOPA) submetido!
Pedido 1 (SPAGHETTI) iniciando!
Pedido 2 (CARNE) iniciando!
Pedido 3 (SOPA) iniciando!
Pedido 1 (SPAGHETTI) entregue!
Pedido 2 (CARNE) entregue!
Pedido 3 (SOPA) entregue!
```

*(As mensagens na tela são meramente informativas, não precisam ser exatamente iguais. A corretude do programa será verificada a partir das chamadas à funções em **tarefas.h**, especialmente **notificar_prato_no_balcao()** e **entregar_pedido()**)*

Regras



- Cada cozinheiro só pode preparar um pedido por vez;
- Um pedido tem que ser preparado por um único cozinheiro;

- Um pedido tem apenas uma única receita (dentre as listadas abaixo);
- Só pode haver uma ou zero panelas em uma boca de fogão em um dado momento;
- Existem **infinitas panelas**, mas apenas um certo número de **frigideiras**
  - As tarefas que não mencionam frigideira são feitas em panelas
- Algumas tarefas (partes da receita) não necessitam da atenção do cozinheiro: ele pode iniciá-las e depois se dedicar a outra tarefa da mesma receita, mesmo que essa segunda tarefa exija atenção. As tarefas que exigem atenção são denominadas de Dedicação Exclusiva [DE];
- Um cozinheiro está **livre** se não está preparando um pedido;
- Um cozinheiro está **esperando** se está preparando um pedido, mas aguardando uma sub-tarefa do pedido que não exige sua atenção;
- Um cozinheiro está **ocupado** se está desempenhando uma tarefa que é parte de um pedido;
- Um pedido só pode ser preparado por um cozinheiro **livre**;
- Uma tarefa pode ser iniciada por um cozinheiro que esteja **esperando**;
  Quando um cozinheiro termina um pedido, ele o coloca no balcão de pedidos prontos:
  - Se não houver espaço no balcão, o cozinheiro fica segurando o pedido na mão, até que seja liberado um espaço;
- Qualquer garçom pode levar um pedido para qualquer cliente;
- Todo garçom pega pedidos prontos no balcão e os entrega para os clientes;
- Cada garçom pode entregar apenas um pedido de cada vez;
- Você deve garantir que após o retorno da função main:
  - Não haja nenhuma outra *thread* em execução;
  - Não exista nenhum pedido em um estágio intermediário de produção ou já produzido mas não entregue;
  - Não podem haver *memory leaks* (use o AddressSanitizer para confirmar).

- **Funções principais:**
  - **`notificar_prato_no_balcao`**`(prato_t* prato)`: deve ser invocada para cada pedido imediatamente antes de um prato com o pedido ser colocado no balcão.
  - **`entregar_pedido`**`(prato_t* prato)`: deve ser chamada quando um garçom entrega o prato correspondente a um pedido.

A cozinha deve operar com a melhor eficiência possível. Como as receitas possuem tarefas que não exigem a atenção do cozinheiro, é importante que ele inicie essas tarefas o mais cedo possível. **Dica**: para descobrir a estratégia ideal de início de tarefas, **construa um diagrama** para cada receita, representando as dependências entre as tarefas.

Receitas

Legenda:

[DE] -- Dedicação exclusiva
[Xmin] -- tarefa dura **X** minutos -- simulação do tempo já **pronta** em tarefas.{h,c}

- **Carne**:
  1. Cortar a carne [5min] [DE]
  2. Temperar a carne [3min] [DE]
  3. Grelhar a carne em uma frigideira [3min] [DE]
  4. Empratar o pedido [1min] [DE]
- **Spaghetti**:
  - Esquentar o molho [5min]
  - Ferver água [3min]
  - Cozinhar o Spaghetti (na água fervente) [5min]
  - Dourar o bacon em uma frigideira [2min]
  - Empratar o pedido [3min] [DE]
- **Sopa**:
  1. Cortar legumes [10min] [DE]
  2. Ferver a água [3min]
  3. Fazer o caldo (com a água fervente, **precisa** de boca de fogão) [2min]
  4. Cozinhar os legumes no caldo [8min]
  5. Empratar o pedido [1min] [DE]

Código inicial

O *parsing* de argumentos de linha de comando, as tarefas que fazem parte das receitas e os objetos (**agua_t**, **carne_t**, **legumes_t**, ...) já estão presentes no código fornecido. Os arquivos fornecidos são:

- **main.c:** *Parsing* de argumentos. Chama algumas funções que **ainda**não estão definidas. Você pode alterar esse arquivo, mas mantenha o comportamento dele na leitura de argumentos inalterado!
- **cozinha.h**: Algumas funções usadas do main.c, que vocês devem implementar. Se necessário, os nomes e assinaturas dessas funções podem ser modificados.
- **pedido.{c,h}:** Funções que auxiliam no *parsing* dos pedidos.
- **tarefas.{c,h}**: **NÃO ALTERE ESSES ARQUIVOS**. Descreve todas as **tarefas que fazem parte das receitas.** Para realizar as tarefas, inclua o arquivo .h e chame as funções correspondentes. Os insumos das receitas podem ser considerados infinitos. Por exemplo, basta chamar **create_carne()** para ter um pedaço de carne crua.

Grupos e avaliação

O trabalho deverá ser realizado em grupos de **2 alunos**. Os grupos deverão ser formados com auxílio da ferramenta **Escolha de Grupos (T1 - Turma A)**, no caso de alunos matriculados na **Turma A**, ou **Escolha de Grupos (T1 - Turma B)**, no caso de alunos matriculados na **Turma B**. **Não será permitida a formação de grupos contendo alunos matriculados em turmas distintas.**

Os trabalhos serão apresentados nos dias definidos no cronograma disponível no Moodle. O professor irá avaliar a corretude, o desempenho e a clareza da solução proposta. A data/hora limite para o envio dos trabalhos é **01/10/2018 às 23h59min**. **Não será permitida a entrega de trabalhos fora desse prazo.**

Durante a apresentação, o professor irá avaliar o conhecimento **individual** dos alunos sobre os conteúdos teóricos e práticos vistos em aula e sobre a solução adotada no trabalho. A nota atribuída a cada aluno ***i*** no trabalho ($NotaTrabalho_i$) será calculada da seguinte forma, onde A_i é a nota referente à apresentação do aluno ***i*** e ***S*** é a nota atribuída à solução do trabalho:

$$NotaTrabalho_i = (A_i * S)/10$$

Como indicado pela fórmula mostrada acima, a nota atribuída à solução adotada será ponderada pelo desempenho do aluno durante a apresentação do trabalho. Por exemplo, se o professor atribuir nota 10 para a solução adotada pelo grupo mas o aluno receber nota 5 pela apresentação - devido ao desconhecimento dos conteúdos teóricos, práticos e/ou da solução do trabalho - a sua nota final do trabalho será 5. A ausência no dia da apresentação ou recusa de realização da apresentação do trabalho implicará em nota zero na apresentação, fazendo com que a nota atribuída ao aluno também seja zero.

Entrega do Exercício

Submeta um arquivo **.tar.gz** gerado usando o comando **make submission**. Esse comando irá realizar uma checagem preliminar do programa para identificar **alguns** desvios da especificação e **alguns**erros triviais.

Diferentemente das atividades de laboratório, a **nota não será automática**.

Atualizações

- Regra re-escrita para deixar mais claro que um cozinheiro pode executar uma tarefa DE enquanto aguarda que uma ou mais tarefas que não são DE terminem.
- Erro `unreconized option '--frigideiras'` corrigido no **main.c**. Basta adicionar `{"frigideiras", required_argument, 0, 'r'}` após a linha 12 em main.c
- Lembrete de que os sleeps referentes aos tempos de cada tarefa já estão prontos no tarefas.c.
- Deixado explícito que o preparo do caldo acontece com a panela de água fervente na boca do fogão.
- **make submission** deixava trabalhos com **segfaults** serem empacotados. Execute o **patch-make-submission.sh** (chmod +x no arquivo) para corrigir.
- Para alguns, uma mensagem "`-bash: [: 1 .: integer expression expected`" pode aparecer no **make submission**. Execute o **patch-make-submission-2.sh** (chmod+x ou simplesmente bash patch-make-submission-2.sh) para resolver. Inclui a correção do antigo **patch-make-submission.sh**

Esqueleto

(Atualizado em 19/09/2018 às 19:30)

main.c
patch-make-submission-2.sh
patch-make-submission.sh
t1.tar.gz