Restaurante Futurístico

Trabalho 1 – Sistemas Operacionais – Processos

Introdução

JuClino++ é um empresário do ramo de bares e restaurantes sofisticados. Seus clientes, acostumados com o requinte de seus ambientes, são pessoas exigentes que buscam boas comidas com facilidade de atendimento e rapidez. Sabendo disso, JuClino++ investiu em um restaurante considerado "futurístico". Nesse modelo de restaurante, os clientes não mais esperam pelos garçons para fazerem seus pedidos. Cada mesa é composta por uma tela sensível ao toque com um sistema computadorizado conectado a um servidor central. Os pedidos efetuados pelos clientes são recebidos pelo servidor que os enfileira e repassa-os aos chefes de cozinha. Também não há necessidade de caixas, uma vez que o pagamento é realizado ali mesmo pelo cartão de crédito.

Após o preparo, o chefe de cozinha entrega a comida para um dos robôs humanoides, capazes de servir corretamente na mesa correspondente ao pedido. Essa inovação chamou a atenção da clientela, e JuClino++, por sua vez, ficou bastante satisfeito, pois reduziu o custo com os caixas e garçons, e pôde investir em mais chefes de cozinha. Fazendo as contas, no modelo anterior, num restaurante com 60 mesas, eram necessários em média 15 garçons (1 para cada 4 mesas), 3 caixas (1 para cada 20) e no mínimo 6 chefes de cozinha (1 para cada 10). Agora, com esta tecnologia, JuClino++ espera contratar mais chefes de cozinha (1 para 4 mesas), melhorando assim a rapidez do atendimento!

JuClino++ está procurando alguém capaz de fornecer este sistema complexo e contratou você para esta tarefa!

O Sistema Computacional

O sistema que você implementará será o software responsável por processar os eventos do restaurante. Este, inicializará lendo dois números inteiros ***n*** e ***m*** do terminal, indicando, respectivamente, a quantidade de chefes de cozinha ($1 \leq n \leq 100$), e a quantidade de mesas do restaurante ($n \leq m \leq n * 4$). Exemplo:

```
5 20 // Há 5 chefes de cozinha e 20 mesas no restaurante
```

Quando o cliente realiza um pedido, seu software é notificado por uma mensagem composta por um número inteiro e uma string, conforme o exemplo do parágrafo seguinte. O número inteiro corresponde ao número da mesa que fez o pedido; e a string corresponde ao item que foi pedido, ou a palavra "fim", indicando o término do pedido. Cada mesa deve ser atendida por um único e exclusivo processo, que armazenará os itens dos pedidos em um log (arquivo de texto), e se encerará após o término dos pedidos. O limite de processos em execução está relacionado à quantidade de chefes de cozinha contratados pelo restaurante, ou seja, se o restaurante possui 10 chefes de cozinha, existirão no máximo 11 processos em execução (1 lendo os eventos da entrada do terminal, mais 10 recebendo os pedidos e escrevendo em um arquivo de log). Caso não

haja mais disponibilidade para a criação de processos (não há chefe de cozinha disponível para atender ao pedido, pois todos eles estão atendendo mesas com pedidos ainda em aberto), seu programa deverá escrever essa informação em um arquivo de log separado (`MesasNaoAtendidas.txt`), ou então, mais eficientemente, colocar estes pedidos em uma lista de espera, que será consumida de acordo com a liberação dos chefes de cozinha.

```
07 hamburguer com batatas fritas // pedido da mesa 7
03 bife acebolado // pedido da mesa 3
18 salada de tomate
03 suco natural de laranja (200ml)
15 picanha na chapa
03 fim // mesa 3 finalizou o pedido
```

No exemplo acima, o cliente da mesa 7 é o primeiro a chegar. Este, pede hambúrguer com batatas fritas. Por ser o primeiro cliente a realizar um pedido, certamente haverá um chefe de cozinha disponível para atende-lo. Em seguida, a mesa 3 pede um bife acebolado. Sendo esta uma nova mesa, é necessário que haja um segundo chefe de cozinha (disponível), ou então esse pedido não será atendido. Caso haja disponibilidade, esse segundo chefe de cozinha atende e espera por um próximo item da mesa 3. Observe que outros pedidos podem surgir na sequencia, e obviamente, serão necessários outros chefes de cozinha (processos) para atende-los, já que o primeiro chefe de cozinha está ocupado com a mesa 7 e o segundo com a mesa 3.

Observação: É importante lembrar que neste modelo de restaurante, cada chefe de cozinha atende unicamente a uma única mesa, podendo somente trocar de mesa após o cliente atual fechar os pedidos da mesa pela mensagem de fim.

Uso de buffers: Observe que no exemplo dado, caso uma nova mesa faça um pedido e não haja chefe de cozinha disponível, o dono do restaurante certamente terá prejuízo, pois terá que rejeitar o pedido por falta de recursos (chefe de cozinha). Contratar novos chefes de cozinha não é necessariamente uma boa ideia, pois vários destes novos empregados poderão ficar ociosos na maior parte do tempo. Uma boa estratégia é criar uma lista de espera para os pedidos que não podem ser atendidos prontamente. Assim, cada chefe de cozinha, ao ficar disponível consulta esta "lista de espera" e realiza o atendimento ao cliente.

Problemas relacionados a falta de recursos é comum nos Sistemas Operacionais, uma vez que eles são limitados e simultaneamente utilizados por diversos processos. Por isso, uma estratégia bastante utilizada é o uso de "buffers", que consistem em regiões de memória sendo utilizadas como área de armazenamento temporário de dados durante a transferência entre dispositivos. Essa estratégia não só ameniza o problema da limitação de recursos, como também permite que o processador não fique em espera ocupada ao fazer operações de entrada/saída.

Seu programa termina ao receber uma linha contendo uma única palavra fim, assim como descrito abaixo:

```
FIM
```

Entrada e Saída do seu Programa

Exemplo de entrada do seu programa:

```
3 10
4 hamburguer com batatas fritas
3 bife acebolado
8 salada de tomate
3 suco natural de laranja (250ml)
5 picanha na chapa
4 cachorro quente
8 espaguete ao molho branco
5 porcao de batatas fritas
5 fim
3 bolo de chocolate
4 fim
3 fim
8 suco natural de acerola (500ml)
4 bife a parmegiana
4 salada de alface
1 pizza de calabresa
4 fim
8 pudim de leite condensado
1 pizza de lombo ao creme
1 fim
8 fim
FIM
```

Exemplo dos arquivos de saída do seu programa:

Arquivo: `ChefeCozinha_1.txt`

```
ChefeCozinha_1

Mesa 4:
     - hamburguer com batatas fritas
     - cachorro quente

Mesa 5:
     - picanha na chapa
     - porcao de batatas fritas

Mesa 4:
     - bife a parmegiana
     - salada de alface
```

Arquivo: `ChefeCozinha_2.txt`

```
ChefeCozinha_2

Mesa 3:
     - bife acebolado
     - suco natural de laranja (250ml)
     - bolo de chocolate

Mesa 1:
     - pizza de calabresa
     - pizza de lombo ao creme
```

Arquivo: `ChefeCozinha_3.txt`

```
ChefeCozinha_3

Mesa 8:
     - salada de tomate
     - espaguete ao molho branco
     - suco natural de acerola (500ml)
     - pudim de leite condensado
```

Observações de notas:

- A implementação correta do atendimento dos pedidos vale 6.0 pontos;
- Ao ler a mesa que fez um pedido, é necessário saber ela já está sendo atendida por algum chefe de cozinha. O algoritmo responsável por essa verificação deve ter complexidade O(1). A implementação conforme se pede valerá 1.0 ponto;
- Ao saber que um cliente fez o primeiro pedido, é necessário verificar se há um chefe de cozinha disponível, e se sim, reserva-lo ao atendimento da mesa. Esta função deve ter complexidade O(1). A implementação conforme se pede valerá 1.0 ponto;
- A implementação da lista de espera vale 2.0 pontos (obviamente, com a lista de espera não é necessário manter o log das mesas não atendidas);

Para a realização do trabalho, é importante estudar as funções: fork e pipe. Além disso, é necessário estudar IPC (*interprocess communication* – comunicação entre processos). O livro **Sistemas Operacionais Modernos 3ºed – Tanenbaum** os ajudará. Além disso, me procurem ou mandem-me um email: kleberkruger@gmail.com

Data de entrega: 31/01/2017
Enviar por email para o endereço: **kleberkruger@gmail.com**

Bom trabalho a todos, um feliz ano novo e um feliz natal! Ho ho ho!