

Pontifícia Universidade Católica de Campinas Faculdade de Engenharia de Computação - FECOMP

Sistemas Operacionais A – Relatório Experimento 5

Beatriz Morelatto Lorente RA: 18071597

Cesar Marrote Manzano RA: 18051755

Fabricio Silva Cardoso RA: 18023481

Pedro Ignácio Trevisan RA: 18016568

**Sumário**

**1.Introdução.......................................................................................................3**

**2.Respostas das perguntas..............................................................................4**

**3.Resultados da execução do primeiro programa..........................................5**

**4.Resultados da execução do segundo programa.........................................9**

**5.Análise dos Resultados..................................................................................0**

**6.Conclusão........................................................................................................0**

**Introdução**

O experimento realizado permitiu o melhor entendimento de todos os conceitos adquiridos ao longo do semestre, como: criação de processos filhos, fila de mensagens, memória compartilhada, semáforos e threads. Foram feitos dois programas diferentes, visando a resolução do problema clássico do barbeiro dorminhoco.

No primeiro programa, era necessário o uso de processos filhos, fila de mensagens e memória compartilhada. O número de cadeiras disponíveis era igual a sete, e era necessário inicializar dois processos barbeiros e vinte processos clientes. Um processo cliente, envia uma mensagem ao barbeiro, com uma string de tamanho e números variáveis (entre 2 e 1023) e o barbeiro devolvia a mensagem com o resultado do serviço prestado. Caso o cliente não fosse atendido, era necessário mostrar o seu número. Caso contrário, era necessário imprimir o resultado da string que seria, de sua respectiva ordenação e o tempo de demora para ser atendido, ou seja, para iniciar o corte de cabelo.

No segundo programa, era necessário trocar os processos filhos por threads, mutex para exclusão mútua e semáforos ao invés de fila de mensagens. O número de cadeiras aumenta para sete, de barbeiros para sete e o de clientes para vinte e sete. A lógica e as impressões necessárias, funcionam de maneira similar ao primeiro programa.

**Respostas das perguntas**

***Pergunta 1: Qual é o recurso comum que necessita de exclusão mútua ?***

**Resposta:** O número de cadeiras disponíveis será um recurso comum do barbeiro e do cliente, portanto será necessário a exclusão mútua dessa variável.

***Pergunta 2: De que maneira (leitura, escrita, ambos) barbeiros e clientes vão acessar o recurso comum ?***

**Resposta:** No primeiro programa é necessário implementar um semáforo que trava o acesso ao recurso comum, que trava tanto o barbeiro quanto o cliente. No segundo programa isso é feito com o uso de mutex.

***Pergunta 3: Como os números foram colocados na String ?***

**Resposta:** Os números foram convertidos para string com o auxílio da função ‘sprintf()’. A cada número convertido, era adicionado um espaço em branco, para facilitar a conversão de string para número novamente.

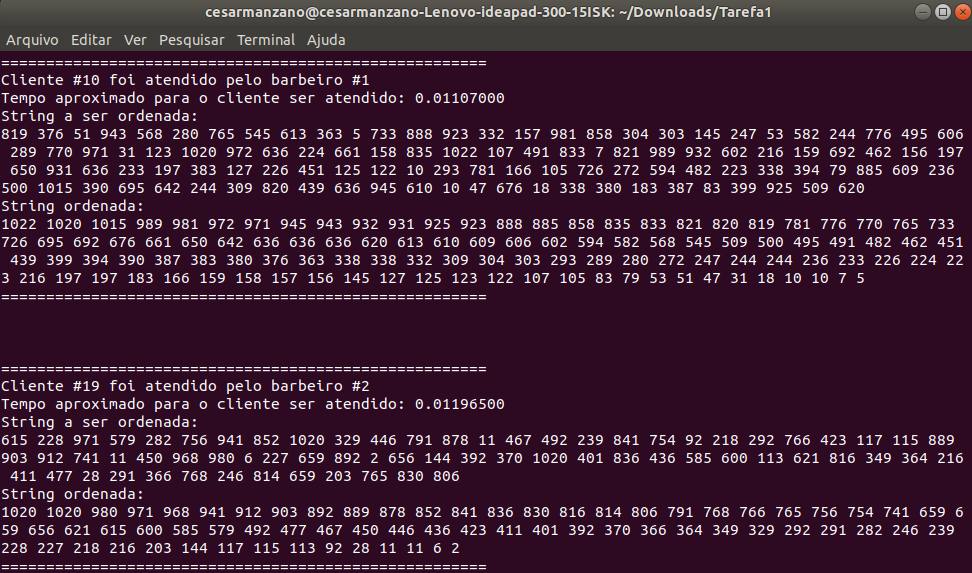
***Pergunta 4: Como o barbeiro vai ter acesso aos valores a serem ordenados ?***

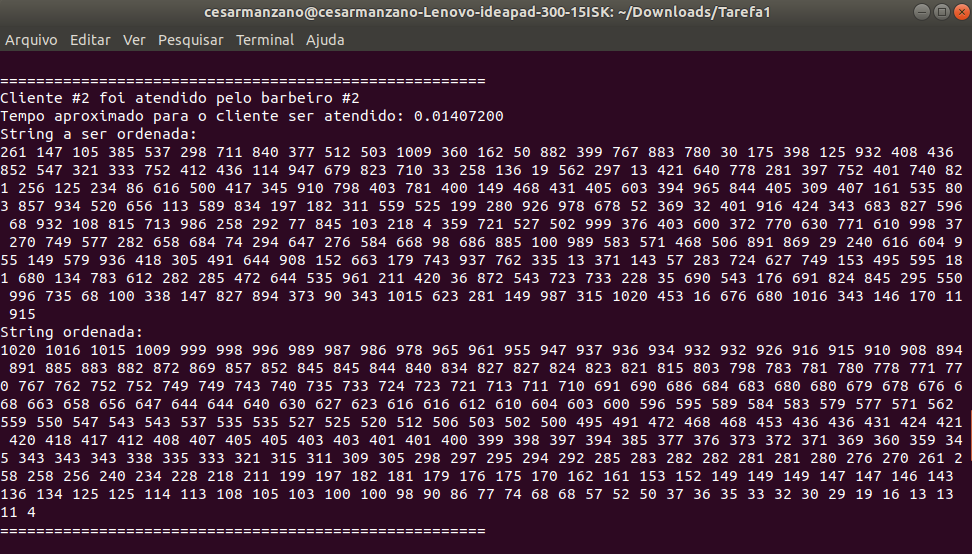
**Resposta:** No primeiro programa, o cliente passa uma mensagem com a string para ser ordenada, é necessário converter em um vetor de inteiros e então ordenar em forma decrescente. No segundo programa, é implementado um vetor de struct, no qual o barbeiro acessa e pega a string do cliente.

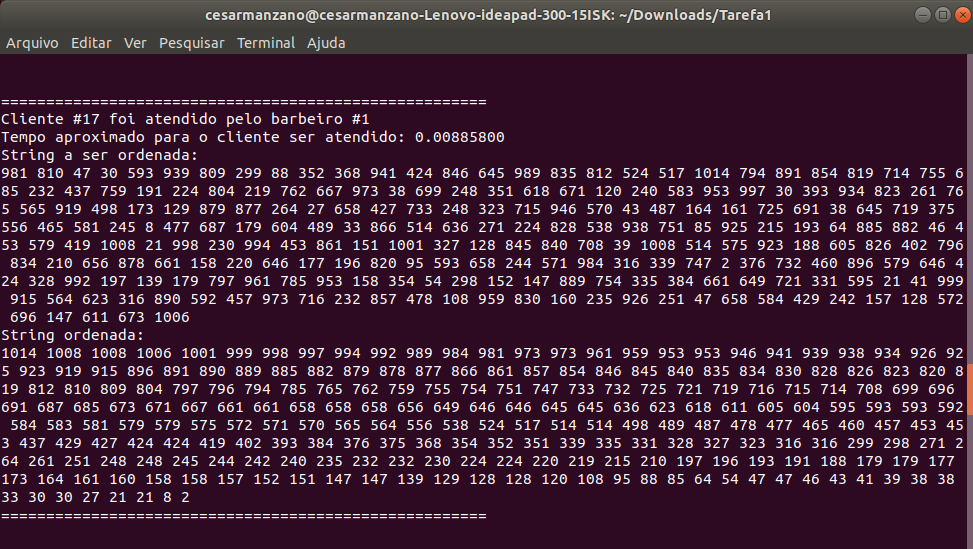
***Pergunta 5: Como o cliente vai ter acesso aos resultados?***

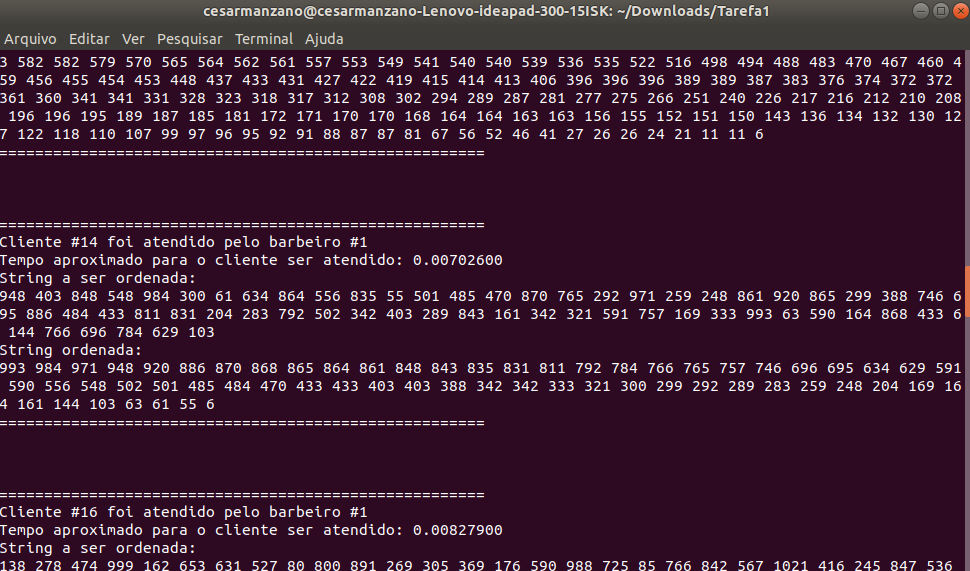
**Resposta:** No primeiro programa, por meio de uma mensagem enviada pelo barbeiro. No segundo programa, o cliente acessa a struct após o barbeiro inserir as informações necessárias, para assim imprimir de maneira correta.

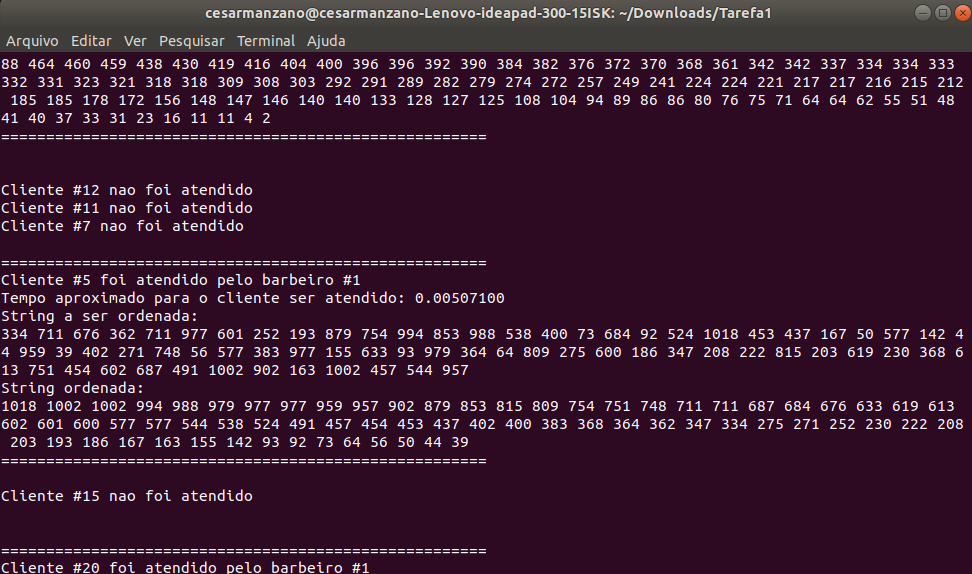
**Resultados da execução do primeiro programa**

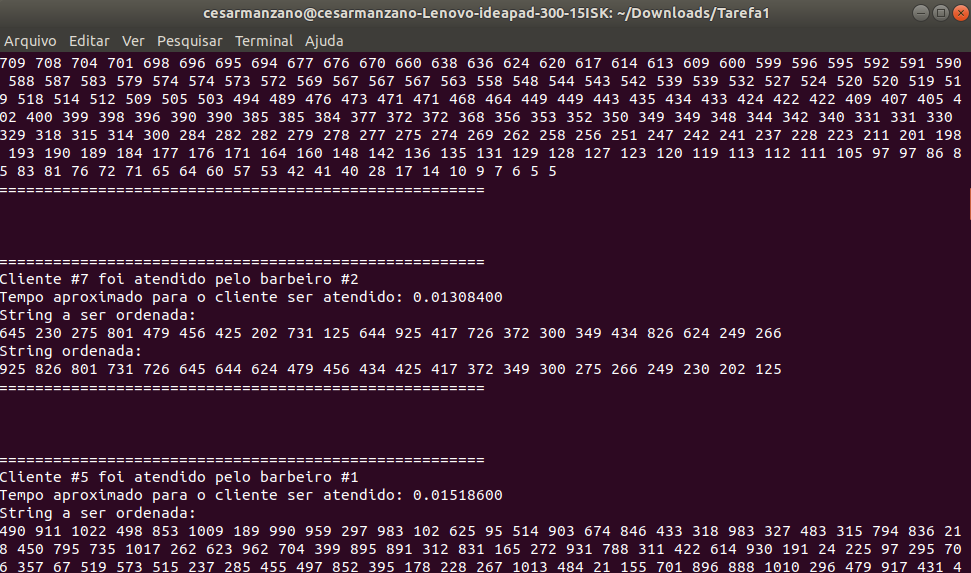
****

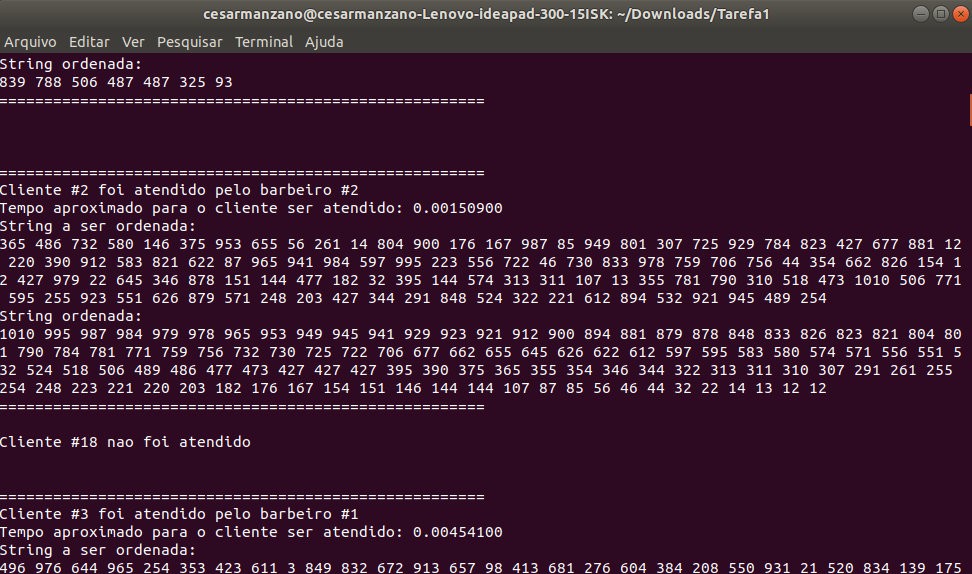
****

****

****

****

****

****

**Resultados da execução do segundo programa**

**Análise dos resultados**

**Programa 1**

Como podemos observar pelos resultados do primeiro programa, esta parte do experimento foi realizada com sucesso. Verifica-se que a string de cada processo filho foi ordenada de forma decrescente, como esperado e que alguns clientes também não foram atendidos.

Para podemos observar melhor alguns aspectos do programa e para uma melhor análise da tarefa, foram feitos alguns gráficos mostrados abaixo:

**Programa 2**

**Conclusão**

Como citado anteriormente, o experimento possibilitou o uso de todos os mecanismos vistos no semestre. Foi importante para que os conceitos fossem melhor compreendidos pelos alunos.

No primeiro programa, foi possível a revisão de funções relacionadas a fila de mensagens e memória compartilhada como: shmget(), shmat(), msgget(), entre outras. No segundo programa, todas as funções relacionadas a threads, mutex e semáforos também puderam ser revisadas, contribuindo para a conclusão do experimento