

Experimento #5 – Sistemas Operacionais A



PUC
CAMPINAS
PONTIFÍCIA UNIVERSIDADE CATÓLICA

Adriano de Oliveira Munin	17066960
Fábio Seiji Irokawa	17057720
Lucas Rodrigues Coutinho	17776501
Marcos Lelis de F. Oliveira	16248387
Paulo M. Biocchi	16148363

Introdução

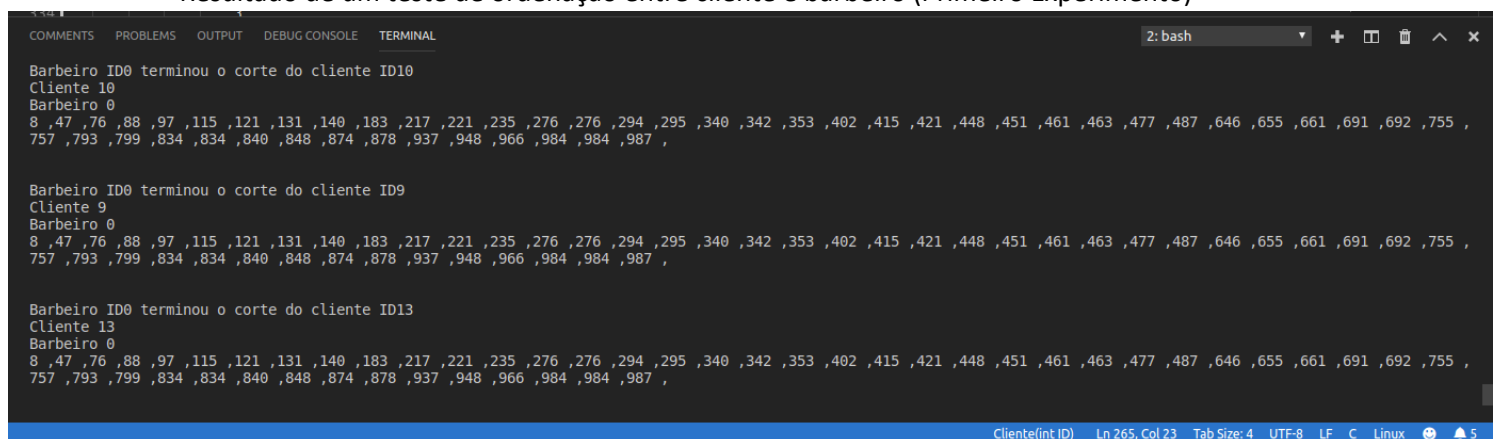
O objetivo do último experimento é a colocação em prática os conhecimentos dos conceitos aprendidos este semestre que são os processos, memória compartilhada, threads, semáforos, exclusão mútua e os mecanismos de troca de mensagens.

Como prática deste experimento foi pedido a resolução do problema do Barbeiro Dorminhoco em duas formas. A primeira usando processos e usando a fila de mensagens no lugar dos semáforos de barbeiros e clientes. No programa o cliente irá sentar em uma das sete cadeiras vazias disponíveis, que seria o buffer, e o barbeiro sempre verifica se há algum cliente (mensagem) no buffer para realizar o corte. Caso há alguma mensagem no buffer, o corte é realizado, caso contrário, quando não há clientes, o barbeiro fica esperando pela mensagem. Caso todas as cadeiras estejam cheias (buffer cheio), o próximo cliente que aparecer não encontrará lugar e irá embora sem ter o cabelo cortado.

A segunda forma utiliza-se threads no lugar dos processos, mutex e semáforos para os barbeiros e clientes no lugar da fila de mensagens. Um ponto importante a ser notado é que o acesso às cadeiras é uma seção crítica, ou seja, requer o uso da exclusão mútua caso todas as cadeiras estejam ocupadas evitando que novos clientes sentem em cadeiras “invisíveis”. A execução funciona quando um cliente surge para cortar o cabelo e senta em uma das cadeiras para ter seu cabelo cortado e, portanto, acorda o barbeiro que estava dormindo para que possa realizar o corte. Caso todas as cadeiras estejam ocupadas e novos clientes apareçam, em vez deles irem embora, eles vão esperar que uma nova cadeira seja esvaziada, ou como no documento do experimento relata, tomar um sorvete. Após o corte o cliente irá “apreciar o corte” do barbeiro, onde esta função irá devolver a duração desde que ele chegou à barbearia até o momento que foi terminado o corte.

Resultado das Execuções

Resultado de um teste de ordenação entre cliente e barbeiro (Primeiro Experimento)



```
Barbeiro ID0 terminou o corte do cliente ID10
Cliente 10
Barbeiro 0
8 ,47 ,76 ,88 ,97 ,115 ,121 ,131 ,140 ,183 ,217 ,221 ,235 ,276 ,276 ,294 ,295 ,340 ,342 ,353 ,402 ,415 ,421 ,448 ,451 ,461 ,463 ,477 ,487 ,646 ,655 ,661 ,691 ,692 ,755 ,
757 ,793 ,799 ,834 ,834 ,840 ,848 ,874 ,878 ,937 ,948 ,966 ,984 ,984 ,987 ,

Barbeiro ID0 terminou o corte do cliente ID9
Cliente 9
Barbeiro 0
8 ,47 ,76 ,88 ,97 ,115 ,121 ,131 ,140 ,183 ,217 ,221 ,235 ,276 ,276 ,294 ,295 ,340 ,342 ,353 ,402 ,415 ,421 ,448 ,451 ,461 ,463 ,477 ,487 ,646 ,655 ,661 ,691 ,692 ,755 ,
757 ,793 ,799 ,834 ,834 ,840 ,848 ,874 ,878 ,937 ,948 ,966 ,984 ,984 ,987 ,

Barbeiro ID0 terminou o corte do cliente ID13
Cliente 13
Barbeiro 0
8 ,47 ,76 ,88 ,97 ,115 ,121 ,131 ,140 ,183 ,217 ,221 ,235 ,276 ,276 ,294 ,295 ,340 ,342 ,353 ,402 ,415 ,421 ,448 ,451 ,461 ,463 ,477 ,487 ,646 ,655 ,661 ,691 ,692 ,755 ,
757 ,793 ,799 ,834 ,834 ,840 ,848 ,874 ,878 ,937 ,948 ,966 ,984 ,984 ,987 ,

Cliente(int ID) Ln 265, Col 23 Tab Size: 4 UTF-8 LF C Linux
```

Resultado de um teste de ordenação entre cliente e barbeiro (Segundo Experimento)

```
marcos@marcos-Inspiron-7559: ~/Área de Trabalho/Sistemas Operacionais/Teste
Arquivo Editar Ver Pesquisar Terminal Ajuda
-----
211         Up[0].sem_num = 1;
212         semop(&SemID,
213               {&SemID, O_CREAT, 0666, 1}, 1);
214         int i;
-----
Cliente atendido No. 25. Barbearia = barbearia dados[id].Id Cliente = id;
Barbeiro que atendeu foi o No. 0. ad mutex pthread_mutex_lock(&mutex_print);
0 cliente 25 demorou 1094.00 microsegundos na barbearia. ignored time(NULL));
qtdelementos = rand()%100;
pthread_mutex_unlock(&mutex_print);
-----
String a ser ordenada:
294 ,14 ,54 ,922 ,575 ,953 ,1021 ,139 ,760 ,351 ,81 ,94 ,923 ,978 ,496 ,98 ,585 ,498 ,644 ,409 ,95 ,573 ,671 ,545 ,974
,189 ,373 ,62 ,851 ,710 ,716 ,120 ,724 ,770 ,17 ,274 ,698 ,15 ,412 ,433 ,364 ,493 ,526 ,265 ,448 ,1022 ,363 ,10 ,497
,1005 ,418 ,590 ,554 ,66 ,110 ,505 ,255 ,483 ,567 ,83 ,168 ,258 ,203 ,891 ,5 ,218 ,142 ,702 ,233 ,554 ,110 ,596 ,22 ,6
36 ,861 ,468 ,634 ,199 ,479 ,108 ,181 ,897 ,697 ,733 ,961 ,807 ,215 ,193 ,266 ,783 ,276 ,432 ,18 ,477 ,300 ,22 ,695 ,4
41 ,722 ,926 ,993 ,832 ,499 ,1016 ,444 ,335 ,461 ,55 ,534 ,938 ,161 ,714 ,810 ,858 ,424 ,748 ,641 ,640 ,941 ,905 ,398
,192 ,312 ,414 ,667 ,613 ,436 ,337 ,29 ,133 ,241 ,1022 ,964 ,738 ,1013 ,385 ,51 ,450 ,440 ,583 ,365 ,599 ,274 ,151 ,43
3 ,697 ,899 ,51 ,312 ,816 ,954 ,710 ,1008 ,243 ,101 ,653 ,854 ,536 ,988 ,883 ,667 ,206 ,881 ,608 ,943 ,871 ,993 ,994 ,
296 ,408 ,554 ,660 ,1008 ,827 ,811 ,418 ,501 ,685 ,467 ,813 ,478 ,396 ,498 ,462 ,639 ,597 ,90 ,469 ,108 ,55 ,327 ,776
,262 ,185 ,361 ,180 ,32 ,330 ,151 ,328 ,738 ,703 ,988 ,721 ,505 ,776 ,116 ,1004 ,439 ,583 ,792 ,917 ,979 ,267 ,354 ,59
4 ,863 ,444 ,38 ,971 ,500 ,365 ,722 ,762 ,551 ,61 ,940 ,583 ,391 ,68 ,911 ,104 ,769 ,875 ,826 ,250 ,628 ,940 ,231 ,42
,499 ,1 ,958 ,453 ,266 ,289 ,24 ,106 ,734 ,62 ,53 ,211 ,428 ,775 ,971 ,977 ,834 ,888 ,537 ,200 ,954 ,423 ,303 ,698 ,27
5 ,106 ,948 ,902 ,21 ,157 ,944 ,520 ,156 ,877 ,974 ,422 ,144 ,998 ,527 ,878 ,36 ,578 ,64 ,462 ,328 ,12 ,416 ,140 ,898
,951 ,338 ,829 ,349 ,641 ,
-----
String a ja ordenada:
5 ,10 ,14 ,15 ,17 ,18 ,22 ,22 ,29 ,51 ,54 ,55 ,62 ,66 ,81 ,83 ,94 ,95 ,98 ,108 ,110 ,110 ,120 ,133 ,139 ,142 ,151 ,161
,168 ,181 ,189 ,192 ,193 ,199 ,203 ,215 ,218 ,233 ,241 ,255 ,258 ,265 ,266 ,274 ,274 ,276 ,294 ,300 ,312 ,335 ,337 ,3
51 ,363 ,364 ,365 ,373 ,385 ,398 ,409 ,412 ,414 ,418 ,424 ,432 ,433 ,433 ,436 ,440 ,441 ,444 ,448 ,450 ,461 ,468 ,477
,479 ,483 ,493 ,496 ,497 ,498 ,499 ,505 ,526 ,534 ,545 ,554 ,554 ,567 ,573 ,575 ,583 ,585 ,590 ,596 ,599 ,613 ,634 ,63
6 ,640 ,641 ,644 ,667 ,671 ,695 ,697 ,697 ,698 ,702 ,710 ,714 ,716 ,722 ,724 ,733 ,738 ,748 ,760 ,770 ,783 ,807 ,810 ,
832 ,851 ,858 ,861 ,891 ,897 ,899 ,905 ,922 ,923 ,926 ,938 ,941 ,953 ,961 ,964 ,974 ,978 ,993 ,1005 ,1013 ,1016 ,1021
,1022 ,1022 ,
-----
Cliente 19
-----
```

Análise das Execuções

Foi possível notar durante os testes, que a execução no programa que utilizou threads (segundo experimento) foi mais eficiente que no programa que utilizou processos filhos (primeiro experimento), isso se dá provavelmente pôr o primeiro programa utilizar fila de mensagens e o segundo utilizar memória compartilhada.

Respostas das Perguntas

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

R: O recurso comum é a seção crítica do programa que necessita a exclusão mútua

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

R: Tanto os barbeiros como os clientes vão acessar o recurso comum através de uma fila de mensagens, onde a fila funciona como um “semáforo”, permitindo o acesso ao recurso comum se houver alguma mensagem no buffer e travando o “semáforo” caso não exista mensagem no buffer no momento.

Pergunta 3: Como os números foram colocados no string?

R: Cada número inteiro tem seus bits divididos na metade, onde cada parte possui 8 bits, e cada metade de 8 bits é guardada em uma posição da string, garantindo assim que sempre terá espaço suficiente para o armazenamento.

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

R: Os valores a serem ordenados são guardados em uma estrutura global, onde cada cliente possui a sua separadamente.

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

R: Os resultados serão colocados em uma estrutura global, possibilitando tanto o acesso dos barbeiros como dos clientes,

Conclusão

Foi possível revisar com este experimento o conteúdo apresentado em aula durante este semestre, os processos, fila de mensagens, threads, mutex, semáforos e os mecanismos de troca de mensagens através deste experimento, que serviu como uma introdução à matéria e que possivelmente tais conhecimentos serão necessários futuramente para a realização de novos experimentos.