Relatório EP2

ACH2044 - Sistemas Operacionais
Profª. Gisele S. Craveiro - Turma 94/2020

Alexandre Kenji Okamoto - 11208371
Mirela Mei - 11208392

Para compilar e executar os programas, utilizamos um notebook com Windows 10 (versão 10.0.19041 - compilação 19041) com Subsistema do Windows para Linux 2 (WSL 2) com a distribuição Ubuntu 20.04.1 LTS.

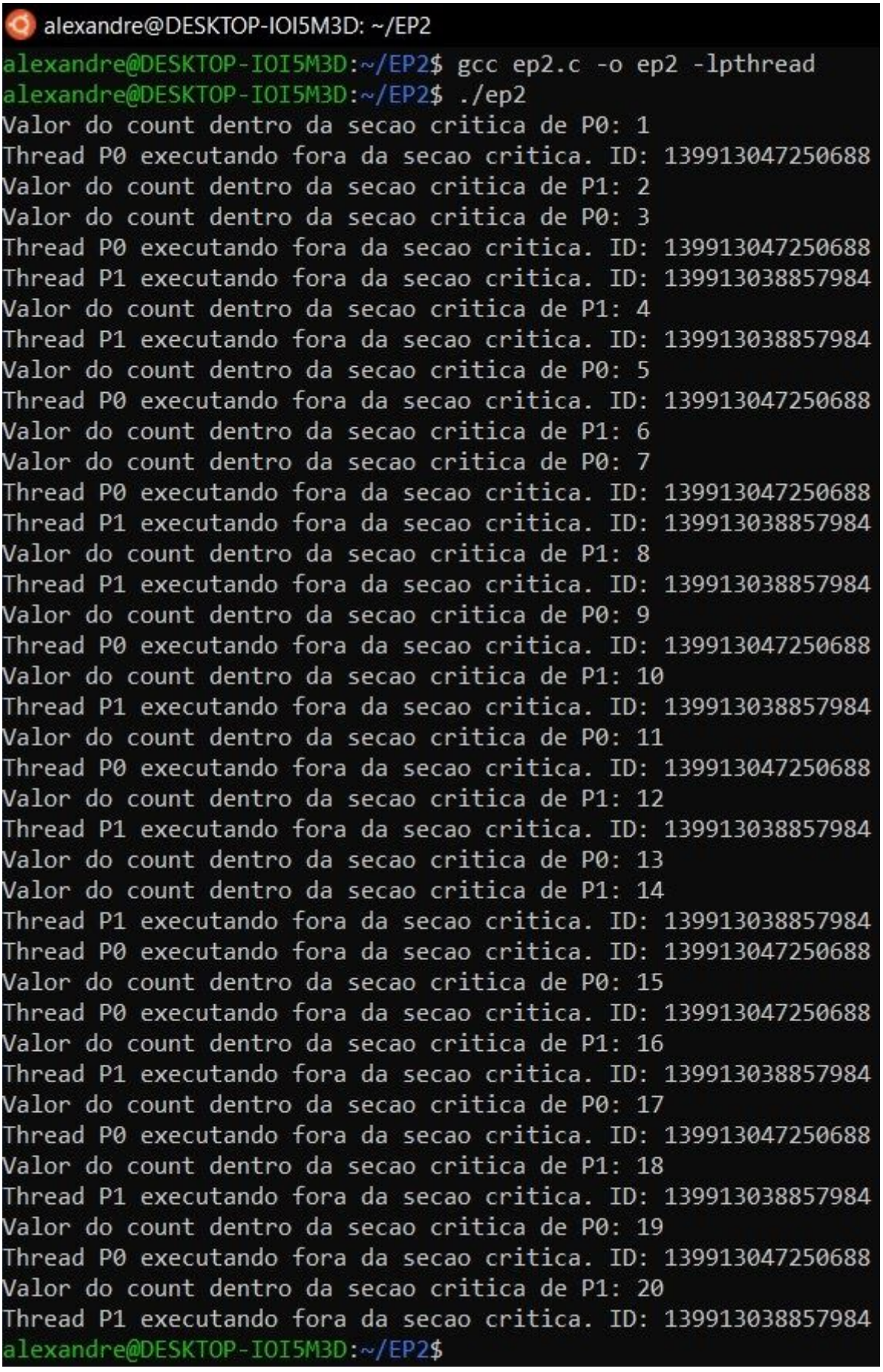
Na implementação das threads, foi utilizado o método de alternância explícita da espera ocupada, com a variável *vez* para checar o momento certo de executar cada seção crítica. O método utilizado não é muito eficiente, pois um processo não pode executar mais de uma vez seguida a sua região crítica, é preciso passar a vez ao outro, então não é recomendável quando um dos processos é muito mais lento. Além disso, o processo que está esperando sua vez fica preso em um loop infinito, gastando processamento sem necessidade.

Essa implementação garante que apenas uma thread manipule por vez a variável compartilhada *count*, através da outra variável global *vez*. A variável *vez* define de quem é a vez de manipular a *count*. Essa transição é feita logo após uma thread sair da sua seção crítica, assim ela passa a vez para a outra.

Com esse algoritmo, um processo executando fora da sua região crítica pode bloquear um processo que poderia entrar na sua seção crítica. No caso de ter uma thread que executa muito mais rápido que a outra, ela tem que esperar a outra entrar na seção crítica para passar a vez. Só que essa última, ainda está na seção não crítica com a vez para ela. Logo, um processo que não está em sua região crítica está bloqueando uma que poderia entrar.

Nessa implementação, nenhum processo fica aguardando infinitamente para entrar em sua seção crítica. Eles executam de forma alternada e, obrigatoriamente, entram a mesma quantidade de vezes na sua região crítica.

Para compilar, utilizamos o comando “gcc ep2.c -o ep2 -lpthread” e para executar “./ep2”


alexandre@DESKTOP-IOI5M3D: ~/EP2
alexandre@DESKTOP-IOI5M3D:~/EP2$ gcc ep2.c -o ep2 -lpthread
alexandre@DESKTOP-IOI5M3D:~/EP2$ ./ep2
Valor do count dentro da secao critica de P0: 1
Thread P0 executando fora da secao critica. ID: 139913047250688
Valor do count dentro da secao critica de P1: 2
Valor do count dentro da secao critica de P0: 3
Thread P0 executando fora da secao critica. ID: 139913047250688
Thread P1 executando fora da secao critica. ID: 139913038857984
Valor do count dentro da secao critica de P1: 4
Thread P1 executando fora da secao critica. ID: 139913038857984
Valor do count dentro da secao critica de P0: 5
Thread P0 executando fora da secao critica. ID: 139913047250688
Valor do count dentro da secao critica de P1: 6
Valor do count dentro da secao critica de P0: 7
Thread P0 executando fora da secao critica. ID: 139913047250688
Thread P1 executando fora da secao critica. ID: 139913038857984
Valor do count dentro da secao critica de P1: 8
Thread P1 executando fora da secao critica. ID: 139913038857984
Valor do count dentro da secao critica de P0: 9
Thread P0 executando fora da secao critica. ID: 139913047250688
Valor do count dentro da secao critica de P1: 10
Thread P1 executando fora da secao critica. ID: 139913038857984
Valor do count dentro da secao critica de P0: 11
Thread P0 executando fora da secao critica. ID: 139913047250688
Valor do count dentro da secao critica de P1: 12
Thread P1 executando fora da secao critica. ID: 139913038857984
Valor do count dentro da secao critica de P0: 13
Valor do count dentro da secao critica de P1: 14
Thread P1 executando fora da secao critica. ID: 139913038857984
Thread P0 executando fora da secao critica. ID: 139913047250688
Valor do count dentro da secao critica de P0: 15
Thread P0 executando fora da secao critica. ID: 139913047250688
Valor do count dentro da secao critica de P1: 16
Thread P1 executando fora da secao critica. ID: 139913038857984
Valor do count dentro da secao critica de P0: 17
Thread P0 executando fora da secao critica. ID: 139913047250688
Valor do count dentro da secao critica de P1: 18
Thread P1 executando fora da secao critica. ID: 139913038857984
Valor do count dentro da secao critica de P0: 19
Thread P0 executando fora da secao critica. ID: 139913047250688
Valor do count dentro da secao critica de P1: 20
Thread P1 executando fora da secao critica. ID: 139913038857984
alexandre@DESKTOP-IOI5M3D:~/EP2$

Notamos que em alguns casos, os prints da seção crítica saem um seguido do outro, como o com *count* (2 e 3, 6 e 7, 13 e 14). Não entendemos o motivo, mas ainda sim é garantida a alternância entre as threads. E de vez em quando o programa termina com *count* valendo 20 ou 21, talvez seja porque no último ciclo ambos os processos já estavam dentro do while, por isso 1 a mais.

Nossas referências foram:

Slides “Sistemas Operacionais” de Prof. Jó Ueyama
(http://wiki.icmc.usp.br/images/7/76/Aula06.pdf)

Slides “Exclusão Mútua por Espera-Ocupada” de Walter Fetter Lages
(http://www.ece.ufrgs.br/~fetter/eng04008/busywait.pdf)

Capítulo 4: “O Problema da Exclusão Mútua” do livro “Sistemas Operacionais e Programação Concorrente” de S.S.Toscani, R.S. de Oliveira e A.S.Carissimi
(http://www.inf.ufrgs.br/~asc/livro/cap4-14.pdf)

Slides “Sistemas Operacionais” de Edson Moreno
(https://www.inf.pucrs.br/~emoreno/undergraduate/CC/sisop/class_files/Aula07/Aula07.pdf)

Livro Fundamentos de Sistemas Operacionais de Abraham Silberschatz, Peter B. Galvin e Greg Gagne, 9ª edição