Escola de Artes, Ciências e Humanidades
da Universidade de São Paulo

EDGAR HENRIQUE DE OLIVEIRA LIRA (12717266)
PEDRO AUGUSTO DE MELO DELAMURA SOARES (12542800)

SISTEMAS OPERACIONAIS

EP2

SÃO PAULO

2022

EDGAR HENRIQUE DE OLIVEIRA LIRA (12717266)
PEDRO AUGUSTO DE MELO DELAMURA SOARES (12542800)

SISTEMAS OPERACIONAIS
EP2

Trabalho apresentado à Escola de Artes, Ciências e Humanidades da Universidade de São Paulo como requisito da disciplina de Sistemas Operacionais.

Prof. Dr.ª Gisele da Silva Craveiro.

SÃO PAULO
2022

SUMÁRIO

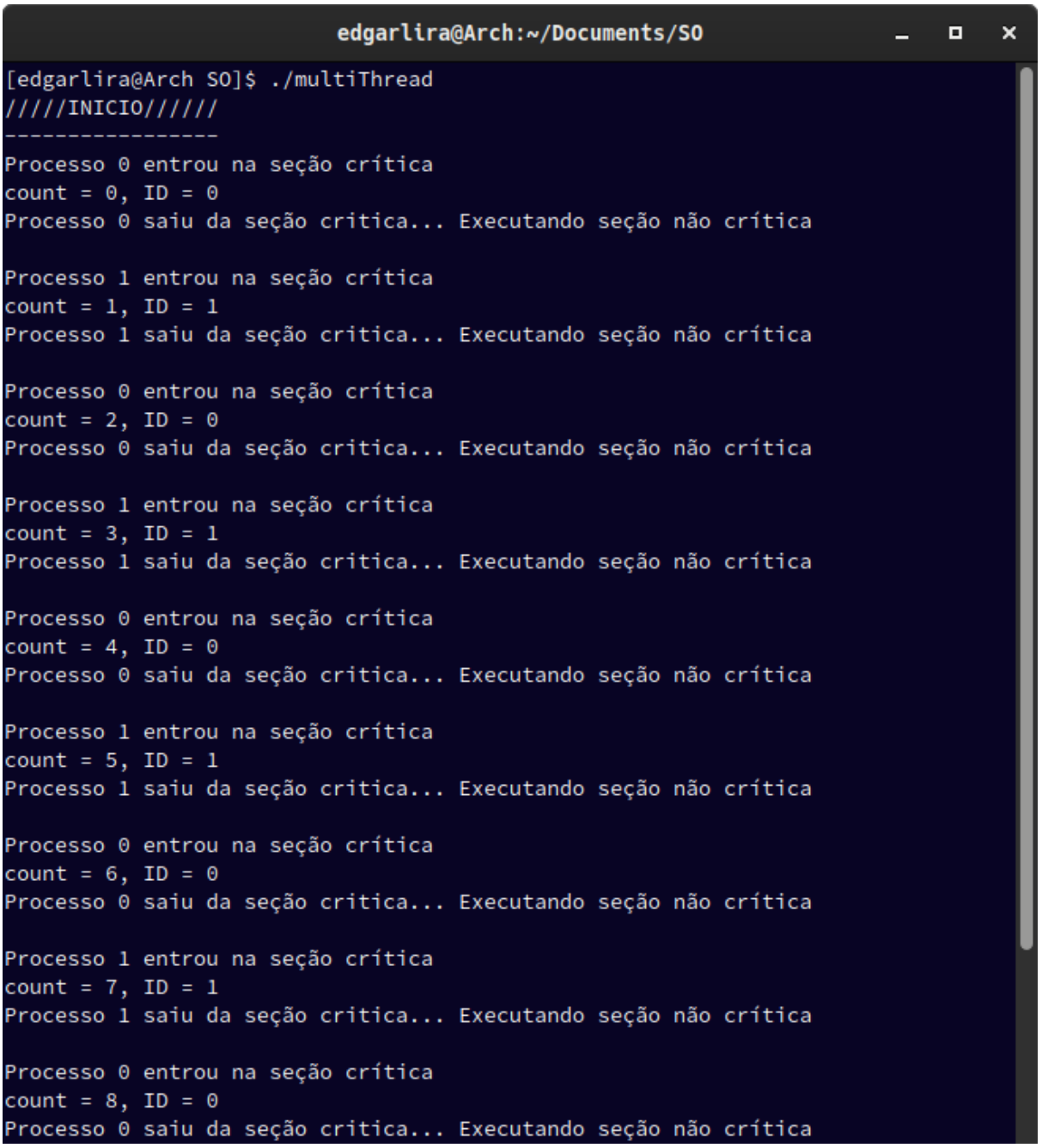
1 CÓDIGO

1.1 Código

```
#include <pthread.h>
#include <stdbool.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int vez = 0;
int count = 0;

void secao_nao_critica(int x) {
 printf("Processo %d saiu da seção crítica... Executando seção não
crítica\n", x);
}

void secao_critica(int y) {
 printf("Processo %d entrou na seção crítica\n", y);
 sleep(1.5);
 printf("count = %d, ID = %d\n", count, y);
 count++;

}

void *P0(void *id) {

 int meu_id = 0;
 int outro = 1;

 while (1) {

   while (vez != meu_id)
     ;
   secao_critica(meu_id);
   vez = outro;
   secao_nao_critica(meu_id);
 }

 return NULL;
}
```

```
void *P1(void *id) {

 int meu_id = 1;
 int outro = 0;

 while (1) {

   while (vez != meu_id)
     ;
   secao_critica(meu_id);
   vez = outro;
   secao_nao_critica(meu_id);
 }

 return NULL;
}

int main(void) {

 printf("/////INICIO//////\n-----------------\n");

 pthread_t id0, id1;

 pthread_create(&id0, NULL, P0, (void *)&id0);
 pthread_create(&id1, NULL, P1, (void *)&id1);

 pthread_exit(NULL);

 return 0;
 }
```

1.2 Print da saída



2 Seção Crítica

2.1 Condições para solução do problema da seção crítica

- Exclusão mútua:

Para provar que a exclusão mútua está sendo atendida, devemos observar que só um dos processos pode entrar na seção crítica por vez, ou seja, P0 só entra na seção crítica se for sua vez e P1 só entra em sua seção crítica se for sua vez . Observamos também que, se os dois processos podem executar suas seções críticas ao mesmo tempo, necessariamente a flag de P0 e P1 tem o mesmo valor, o que não é verdade. Uma outra possibilidade seria se a flag da “vez” pudesse ter os valores 0 e 1 ao mesmo tempo, mas isso também não é possível e, portanto, é impossível que ambos entrem na seção crítica ao mesmo tempo.

- Progresso:

A propriedade do progresso é perdida se um dos processos é impedido de entrar na seção crítica, mas não é o que ocorre neste código. Por exemplo, P0 só ficará preso no loop e impedido de entrar na seção crítica se P1 está na seção crítica e ainda não passou a vez para P0, caso contrário seria a vez de P0 e ele conseguiria com tranquilidade entrar na seção crítica, já que quando um dos processos termina de executar a seção crítica, ele automaticamente passa a vez para o outro. Dessa maneira temos garantida a propriedade do progresso.

- Espera Limitada:

Já a propriedade da espera limitada é garantida da seguinte forma: P0 só entrará na seção crítica após no máximo uma entrada de P1, e vice-versa, lembrando que após uma execução da seção crítica de uma das threads, a outra já tem permissão para entrar na seção crítica, garantindo que a espera limitada seja atendida.

Dessa maneira a **exclusão mútua** é mantida, já que P0 só entra novamente na seção crítica quando P1 terminar e, após isso, P1 só voltará à seção crítica quando P0 terminar de executar. A **espera limitada** está garantida, observando que temos no máximo uma entrada de P0 ou de P1 na seção crítica em suas respectivas vezes. Assim como o **progresso** também é garantido, pois um dos processos só estará impedido de entrar na seção crítica se o outro já estiver executando-a. Portanto, atendendo-se às três condições previamente citadas, é possível concluir que o algoritmo é uma solução válida para o problema da seção crítica.

2.2 Análise do Código:

No código temos que a variável global “vez” inicializa com o valor 0, e P0 só inicia se a variável “vez” for igual a 0, dessa maneira P0 passa com sucesso do primeiro while, e entra na seção crítica, enquanto isso P1 está “travado” no while, já que a variável “vez” ainda vale 0 e a vez de P1 executar a seção crítica só ocorre se “vez” vale 1 **(exclusão mútua)**. Em um segundo momento, quando P0 termina de executar sua seção crítica ele transforma o valor da variável “vez” para 1, ou seja, P1 agora pode ultrapassar o while que estava travando-o e entrar na seção crítica **(progresso)**, e agora P0 só pode voltar a executar a seção crítica novamente quando P1 sair da seção crítica e transformar o valor da “vez” para 0 **(espera limitada)**.