Universidade Federal de Roraima
Departamento de Ciência da Computação
Sistemas Operacionais

Atividade - Aula 19/08

Atenção: Vale ressaltar que esta atividade será usada como critério para a contabilização de sua frequência de aula.

Prazo de Entrega: 23/08/2021

Aluno: GUILHERME LUCAS PEREIRA BERNARDO

1. Com relação ao **Problema do Produtor/Consumidor** que consiste em dois processos compartilham um buffer de tamanho fixo. O processo produtor coloca dados no buffer e o processo consumidor retira dados do buffer, apresente:

A) A implementação de um programa, na linguagem C, que execute o problema do produtor/consumidor com um buffer de tamanho 10, usando primitivas sleep/wakeup.

R:

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

#define THREAD_NUM 2

pthread_mutex_t mutexBuffer;

int buffer[10];
int count = 0;

void* producer(void* args){
    while (1) {
        //produce
        int x = rand() % 100;

        //add to the buffer
        pthread_mutex_lock(&mutexBuffer);
        if (count < 10) {
            sleep(1);
            buffer[count] = x;
            count++;
        } else {
            printf("Skipped %d\n", x);
        }
            pthread_mutex_unlock(&mutexBuffer);
    }
}

void* consumer(void* args){
    while (1) {
```

```
        int y = -1;

        //remove from the buffer
        pthread_mutex_lock(&mutexBuffer);
        if (count > 0){
            y = buffer[count-1];
            count--;
        }
        pthread_mutex_unlock(&mutexBuffer);

        //consume
        printf("Got %d\n", y);
    }
}

int main(int argc, char** argv[]){
    srand(time(NULL));
    pthread_t th[THREAD_NUM];
    pthread_mutex_init(&mutexBuffer, NULL);
    int i;
    for (i = 0; i < THREAD_NUM; i++) {
        if (i % 2 == 0) {
            if (pthread_create(&th[i], NULL, &producer, NULL) !=
   0) {
                perror ("failed to create thread");
            }
        } else {
            if (pthread_create(&th[i], NULL, &consumer, NULL) !=
   0){
                perror ("failed to create thread");
            }
        }
    }
    for (i = 0; i < THREAD_NUM; i++){
        if (pthread_join(th[i], NULL)!= 0){
        perror ("failed to join thread");
        }
    }
    pthread_mutex_destroy(&mutexBuffer);
    return 0;
}
```

B) Descreva o problema relacionado ao uso de primitivas sleep/wakeup em relação ao produtor/consumidor.

R: primitivas como sleep/wakeup trazem consigo o problema da exclusão mútua e e o busy waiting.

C) Apresente um solução extra para a implementação do problema do produtor/consumidor. Dica: Pesquise por mutex.