Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Programação Concorrente

Relatório de Entrega de Atividades

Aluno(s): Henrique Mendes de Freitas Mariano; Leonardo Rodrigues de Souza
Matrícula: 170012280; 170060543
Atividade: Aula Prática 05 - Locks

1.1.1 - Problema dos leitores e escritores

```c
// autores: Henrique Mendes de Freitas Mariano e Leonardo Rodrigues de Souza
// arquivo: 1-1-1-controle-de-assinatura.c
// atividade: 1.1.1

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <pthread.h>
#include <time.h>
#include <semaphore.h>

#define NUMLEITORES 10
#define NUMESCRITORES 3

char BD[10000];

int count = 1;

pthread_mutex_t mutex;

sem_t semaforo;

void *ler(void *args){
    while(1){
        sem_wait(&semaforo);
        printf("%s\n", BD);
```

```
        sem_post(&semaforo);
        sleep(rand() % 4);
    }
}

void *escrever(void *args){
    int *id = (int *) args;

    for(int i = 0; i < 1000; i++){
        pthread_mutex_lock(&mutex);
        BD[count] = (char) (*id + '0');
        count++;
        BD[count] = '\0';
        pthread_mutex_unlock(&mutex);
        sleep(rand() % 3);
    }
    free(id);
    pthread_exit(NULL);
}

int main(){

    pthread_t l[NUMLEITORES], e[NUMESCRITORES];

    int *id = NULL;

    BD[0] = 'A';
    BD[1] = '\0';

    sem_init(&semaforo, 0, 10);

    srand(time(0));

    pthread_mutex_init(&mutex, NULL);

    for(int i = 0; i < NUMESCRITORES; i++){
        id = (int *) calloc(1, sizeof(int));
```

```
        *id = i;
        pthread_create(&e[(*id)], NULL, escrever, (void *) id);
    }

    for(int j = 0; j < NUMLEITORES; j++){
        id = (int *) calloc(1, sizeof(int));
        *id = j;
        pthread_create(&l[(*id)], NULL, ler, (void *) id);
    }

    for(int k = 0; k < NUMESCRITORES; k++)
        pthread_join(e[k], NULL);

    for(int k = 0; k < NUMLEITORES; k++)
        pthread_join(l[k], NULL);

    pthread_mutex_destroy(&mutex);
    sem_destroy(&semaforo);

    return 0;
}
```

1.1.2 - Não pois o lock da escrita já impossibilita possíveis inconsistências.

2.1.1 - Lock Recursivo

```c
// autores: Henrique Mendes de Freitas Mariano e Leonardo Rodrigues de Souza
// arquivo: 2-1-1-lock-recursivo.c
// atividade: 2.1.1

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

pthread_mutex_t mutex;
pthread_mutexattr_t attr;

void bar() {
    printf("Tentando pegar o lock de novo.\n");
    pthread_mutex_lock(&mutex);
    printf("Estou com duplo acesso?\n");
    pthread_mutex_unlock(&mutex);
}
void *foo(void *empty){
    pthread_mutex_lock(&mutex);
    printf("Acesso a região crítica.\n");
    bar();
    pthread_mutex_unlock(&mutex);
}
int main() {
    pthread_t t;
    pthread_mutexattr_init(&attr);
    pthread_mutexattr_settype(&attr, PTHREAD_MUTEX_RECURSIVE_NP);
    pthread_mutex_init(&mutex, &attr);
    pthread_create(&t, NULL, foo, NULL);
    pthread_join(t, NULL);
    pthread_mutex_destroy(&mutex);
    pthread_attr_destroy(&attr);

    return 0;
```

```
}
```