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

Relatório de Entrega de Atividades

Aluno(s): Amanda Oliveira Alves e Fillype Alves do Nascimento

Matrícula: 15/0116276 e 16/0070431

Atividade: Aula Prática 05 - Locks (Complementar)

1.1.1

```
// autor: Amanda Oliveira Alves e Fillype Alves do Nascimento
// arquivo: 1.1.1.c
// atividade: 1.1.1

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

#define ESCRITORES 3
#define LEITORES 10

int i, j, rc, total;
char vetor[10000];
char increment[1000];
sem_t mutex;
pthread_mutex_t bd;

void read_data_base();
void use_data_read();
void think_up_data();
void write_data_base();

void *reader(void *j){
```

```
  int i= *(int*) j;
  while(1){
    sem_wait(&mutex);
    rc=rc+1;
    if(rc==1) pthread_mutex_lock(&bd);
    sem_post(&mutex);
    read_data_base(i);
    sem_wait(&mutex);
    rc=rc-1;
    if(rc==0) pthread_mutex_unlock(&bd);
    sem_post(&mutex);
    use_data_read(i);
    sleep(rand() % 3);
  }
}

void *writer(void *j){
  int i= *(int*) j;
  while(1){
    think_up_data(i);
    pthread_mutex_lock(&bd);
    write_data_base(i);
    pthread_mutex_unlock(&bd);
    sleep(rand() % 2);
  }
}

void read_data_base(int i){
  printf("Leitor %d lê: %s\n",i,vetor);
}

void use_data_read(int i){
  printf("Leitor %d usa o que leu.\n",i);
}

void think_up_data(int i){
  printf("Escritor %d inventa um dado.\n",i);
```

```
  increment[0]='A';
  increment[1]= i;
}

void write_data_base(int i){
  strcat(vetor,increment);
  printf("Escritor %d: %s\n",i,vetor);
}

int main(){

  rc=0;
  pthread_t writerthreads[ESCRITORES], readerthreads[LEITORES];
  sem_init(&mutex,0,1);
  pthread_mutex_init(&bd,NULL);

  for(i=0;i<ESCRITORES;i++){
    pthread_create( &writerthreads[i], NULL, writer, &i);
  }

  for(i=0;i<LEITORES;i++){
    pthread_create( &readerthreads[i], NULL, reader, &i);
  }

  for(i=0;i<ESCRITORES;i++){
    pthread_join(writerthreads[i], NULL);
  }

  for(i=0;i<LEITORES;i++){
    pthread_join(readerthreads[i], NULL);
  }

  return(0);
}
```

1.2 - A partir da implementação apresentada no tópico anterior, se fossem removidos os sleeps das threads leituras, existe a chance de um problema acontecer. Que problema é esse e como ele ocorre?
R: Pode acontecer de os leitores nunca permitirem os escritores escreverem ao deixar as threads de leitura sempre ativas.

2.1.1

```c
// autor: Amanda Oliveira Alves e Fillype Alves do Nascimento
// arquivo: 2.1.c
// atividade: 2.1

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

pthread_mutexattr_t     attr;
pthread_mutex_t mutex;

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);
}
```

```c
int main(){
    pthread_t t;

    pthread_mutexattr_settype(&attr,PTHREAD_MUTEX_RECURSIVE);
    pthread_mutex_init(&mutex,&attr);
    pthread_create(&t, NULL, foo, NULL);
    pthread_join(t, NULL);
    pthread_mutex_destroy(&mutex);
    return 0;
}
```