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 04 - Dormir e Acordar

1.1.1 - Controle de Assinaturas

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

int assinaturas = 1;

int *lista_chamada;

pthread_mutex_t possedalista;

void* aluno(void *id){
    int i = *(int *) id;
    pthread_mutex_init(&possedalista, NULL);
    pthread_mutex_lock(&possedalista);
    lista_chamada[assinaturas] = i;
    assinaturas++;
    sleep(1);
    printf("Aluno %d assinou a lista\n", i);
    free(id);
    pthread_mutex_unlock(&possedalista);
```

```
    pthread_exit(NULL);
}

int main(){
    int n;
    scanf("%d", &n);
    lista_chamada = (int *) calloc(n + 1, sizeof(int));
    pthread_t threads[n + 1];
    for(int j = 1; j <= n; j++) {
        int *i = calloc(1, sizeof(int));
        *i = j;
        pthread_create(&threads[(*i)], NULL, aluno, (void *) i);
    }

    for(int k = 1; k <= n; k++)
        pthread_join(threads[k], NULL);

    printf("A ordem de assinatura da lista foi:");
    for(int i = 1; i <= n; i++){
        printf(" %d", lista_chamada[i]);
    }
    printf("\n");
    free(lista_chamada);
    pthread_mutex_destroy(&possedalista);
    return 0;
}
```

2.1.1 - Tentativa de fazer a Visitação do museu.

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

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

#define NUMTURISTAS 90
#define CAPACIDADE_SALA1 10
#define CAPACIDADE_SALA2 6
#define CAPACIDADE_SALA3 18

sem_t sala1;
sem_t sala2;
sem_t sala3;
sem_t guia;

void *turista(void *id){
    int i = *(int *) id;
    int sala1_completa;
    int sala2_completa;
    int sala3_completa;

    sem_wait(&sala1);
    sem_getvalue(&sala1, &sala1_completa);
    sem_wait(&guia);
    if(sala1_completa == 0){
        printf("Sala 1 completa, começando explicação.\n");
        sleep(2);
    }
    sem_post(&guia);
```

```
    sem_post(&sala1);

    sem_wait(&sala2);
    sem_getvalue(&sala2, &sala2_completa);
    sem_wait(&guia);
    if(sala2_completa == 0){
        printf("Sala 2 completa, começando explicação.\n");
        sleep(2);
    }
    sem_post(&guia);
    sem_post(&sala2);

    sem_wait(&sala3);
    sem_getvalue(&sala3, &sala3_completa);
    sem_wait(&guia);
    if(sala3_completa == 0){
        printf("Sala 3 completa, começando explicação.\n");
        sleep(2);
    }
    sem_post(&guia);
    sem_post(&sala3);
}

int main(){
    pthread_t t[NUMTURISTAS];
    int *id;
    sem_init(&sala1, 0, CAPACIDADE_SALA1);
    sem_init(&sala2, 0, CAPACIDADE_SALA2);
    sem_init(&sala3, 0, CAPACIDADE_SALA3);
    sem_init(&guia, 0, 1);

    for(int i = 0; i < NUMTURISTAS; i++){
        id = (int *) calloc(1, sizeof(int));
        *id = i;
        pthread_create(&t[(*id)], NULL, turista, (void *) id);
    }
```

```
    for(int i = 0; i < NUMTURISTAS; i++){
        pthread_join(t[i], NULL);
    }

    sem_destroy(&sala1);
    sem_destroy(&sala2);
    sem_destroy(&sala3);
    sem_destroy(&guia);
    return 0;
}
```

3.1.1 - Entrega de Pacotes

```c
// autores: Henrique Mendes de Freitas Mariano e Leonardo Rodrigues de Souza
// arquivo: 3-1-1-agencia-de-correios.c
// atividade: 3.1.1

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

pthread_cond_t packets;
pthread_mutex_t mutex;
int countPackets;
pthread_t client, deliveryman;

typedef struct s_client {
    int num, i;
    struct s_client* next;
} Client;
Client* queue;

void *clientRoutine() {
    pthread_mutex_lock(&mutex);
    countPackets++;
    if (queue->num)
        printf("Cliente %d solicitou entrega de um pacote\n", queue->i),
queue->num--;
    Client* last = queue;
    while(last->next != NULL)
        last = last->next;

    last->next = queue, queue = queue->next, last->next->next = NULL;
    if (countPackets >= 10) pthread_cond_signal(&packets);
    pthread_mutex_unlock(&mutex);
```

```
}

void *deliverymanRoutine() {
    pthread_mutex_lock(&mutex);
    do {
        pthread_cond_wait(&packets, &mutex);
        printf("Saindo para entregar 10 pacotes\n");
        sleep(2), countPackets = 0;
    } while (queue->num > 0);
    pthread_mutex_unlock(&mutex);
}

int main(){
    int n;
    countPackets = 0;
    pthread_mutex_init(&mutex, NULL);
    pthread_cond_init(&packets, NULL);

    do {
        printf("Número de clientes (1 a 30): ");
        scanf("%d", &n);
    } while (n < 1 || n > 30);
    pthread_create(&deliveryman, NULL, deliverymanRoutine, NULL);

    queue = (Client*) malloc(sizeof(Client)), queue->num = 30, queue->i = 0;
    Client* last = queue;
    for (int i = 0; i < n - 1; i++)
        last->next = (Client*) malloc(sizeof(Client)), last->next->i =
last->i + 1, last->next->num = 30, last->next->next = NULL, last =
last->next;

    while (queue->num > 0)
        pthread_create(&client, NULL, clientRoutine, NULL),
pthread_join(client, NULL);

    pthread_join(deliveryman, NULL);
    last = queue;
```

```
    while (last != NULL) {
        Client* next = last->next;
        free(last), last = next;
    }

    pthread_mutex_destroy(&mutex);
    pthread_cond_destroy(&packets);
    return 0;
}
```