UNICAMP

DOCUMENTAÇÃO CÓDIGO-FONTE

```
*------------------ Sistemas Operacionais - Projeto 1 -------------------*/

//Grupo Miguezeiros
//Filipe Silveira Bernardo - 197054
//Lucas Watanuki - 202143
//Rodrigo Malosti Zanco - 20554

/*----------------------------- Bibliotecas ------------------------------*/
//Inclusão das bibliotecas essenciais
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <pthread.h> //Biblioteca POSIX Threads
#define TAM 20

/*------------------------- Variaveis globais-----------------------------*/
//Variáveis para manipulação de matrizes
float **matriz, elemento;
int       linha, coluna, n_threads;
int lin, col;

int erro; // -> Variável de retorno caso não seja encontrado valor na matriz

/*------------------------------- funcoes --------------------------------*/
float** Criar_Matriz(int linha, int coluna)
{
  int x, y; //Variáveis linha e coluna auxiliares

        float **matriz = (float**)malloc(linha * sizeof(float*)); //Alocação dinâmica

                for (x = 0; x < linha; x++){
                matriz[x] = (float*) malloc(coluna * sizeof(float));
                        for (y = 0; y < coluna; y++) {
                                        matriz[x][y] = 0.0; //Setando todos os valores da matriz
para nulo e prepara-la para receber os dados posteriormente
                                }
                }

return matriz;
}

void* Procurar_Elemento (void *arg)
```

```
	{
	while(col < linha){
		if(matriz[lin][col] == elemento){
			printf("Posicao --> Linha: %d - Coluna: %d\n", lin, col);
			erro = 0;
			}
		col++;
	}
	col = 0;
	lin++;
}

/*----------------------------- Programa -----------------------------*/

int main()
{
	int	x, y, auxiliar;
	char	arquivo[TAM];

	printf("Arquivo da matriz: ");
		gets(arquivo);
		fflush(stdin); //limpa buffer
	printf("Entre com as dimensoes da matriz: ");
		scanf("%d %d", &linha, &coluna);
	printf("Numero de Threads: ");
		scanf("%d", &n_threads);
	printf("Elemento a ser encontrado: ");
		scanf("%f", &elemento);
		fflush(stdin); //limpa buffer



//Verificar arquivo contendo dados p/ matriz
FILE* file = fopen(arquivo, "r"); //Ponteiro 'file' apontando para o arquivo em modo de leitura
	if(file == NULL){
		printf("Arquivo nao encontrado!\n");
		return 0;
	}

//Cria matriz com as dimensoes desejadas
	matriz = Criar_Matriz(linha, coluna);
	pthread_t thread_id[n_threads];

	for(lin = 0; lin < linha; lin++){
```

```
                        for(col = 0; col < coluna; col++){
                        fscanf(file, "%f", &matriz[lin][col]);


                }
        }

/*---------------------- Execucao das Threads ------------------------*/
lin = 0; col = 0;
erro = 1;
auxiliar = linha / n_threads;

for(y = 0 ;y < auxiliar; y++){
        for(x = 0; x < n_threads; x++){
                pthread_create(&thread_id[x], NULL, Procurar_Elemento, NULL);
                pthread_join(thread_id[x], NULL);
        }
}

//Alterna as threads se o numero delas for menor que o numero de linhas da matriz
if(n_threads < linha){

        auxiliar = linha - auxiliar * n_threads;

                for(x = 0; x < auxiliar; x++){
                        pthread_create(&thread_id[x], NULL, Procurar_Elemento, NULL);
                        pthread_join(thread_id[x], NULL);
                }
}

//Tratamento de erro, caso nao encontre elemento
        if(erro != 0){
                printf("Elemento nao consta na matriz!\n");
        }

return 0;
}
```