Pthreads parte 1

Nome:Hao Yue Zheng-10408948
Nome:Smuel Zheng-10395781
Nome:Vitor Pasquarelli Cinalli-10401806

Linker de Git
https://github.com/mauriciohao/Sistema-operacional.git

Considere o trecho de código em anexo para resolver o problema da multiplicação matriz-vetor utilizando pthreads. Entregue o link de seu repositório no git de preferência, ou submeta os arquivos-fonte.

Codigo

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

// Definições de número de threads e dimensões da matriz/vetor
#define NUM_THREADS 4
#define M 6
#define N 6

// Declaração dos arrays para a matriz A, vetor x e vetor resultado y
double A[M][N];
double x[N];
double y[M];

// Estrutura para passar argumentos para as threads
typedef struct {
    int rank;
} ThreadArg;

// Função que cada thread executará
void* Pth_mat_vect(void* arg) {
    int rank = ((ThreadArg*)arg)->rank;
    int local_m = M / NUM_THREADS;  // Divisão de trabalho por thread
    int my_first_row = rank * local_m;
    int my_last_row = (rank == NUM_THREADS - 1) ? M - 1 : (rank + 1) * local_m - 1;

    for (int i = my_first_row; i <= my_last_row; i++) {
        y[i] = 0.0;
        for (int j = 0; j < N; j++) {
            y[i] += A[i][j] * x[j];
        }
    }
    return NULL;
```

```
}

int main() {
    pthread_t threads[NUM_THREADS];
    ThreadArg args[NUM_THREADS];

    // Inicialização da matriz e do vetor
    for (int i = 0; i < M; i++) {
        for (int j = 0; j < N; j++) {
            A[i][j] = i + j;  // Exemplo de inicialização
        }
    }
    for (int j = 0; j < N; j++) {
        x[j] = 1;  // Exemplo de inicialização
    }

    // Criação das threads
    for (int i = 0; i < NUM_THREADS; i++) {
        args[i].rank = i;
        pthread_create(&threads[i], NULL, Pth_mat_vect, &args[i]);
    }

    // Espera todas as threads completarem
    for (int i = 0; i < NUM_THREADS; i++) {
        pthread_join(threads[i], NULL);
    }

    // Impressão do vetor resultante
    printf("Vetor resultante y:\n");
    for (int i = 0; i < M; i++) {
        printf("%f ", y[i]);
    }
    printf("\n");

    return 0;
}
```

Solução

```
 GNU nano 5.8
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>

// Definições de número de threads e dimensões da matriz/vetor
#define NUM_THREADS 4
#define M 6
#define N 6


// Declaração dos arrays para a matriz A, vetor x e vetor resultado y
double A[M][N];
double x[N];
double y[M];

// Estrutura para passar argumentos para as threads
typedef struct {
    int rank;
} ThreadArg;

// Função que cada thread executará
void* Pth_mat_vect(void* arg) {
    int rank = ((ThreadArg*)arg)->rank;
    int local_m = M / NUM_THREADS;  // Divisão de trabalho por thread
    int my_first_row = rank * local_m;
    int my_last_row = (rank == NUM_THREADS - 1) ? M - 1 : (rank + 1) * local_m - 1;

    for (int i = my_first_row; i <= my_last_row; i++) {
        y[i] = 0.0;
        for (int j = 0; j < N; j++) {
            y[i] += A[i][j] * x[j];
        }
    }
    return NULL;
}
```

```
[cloudshell-user@ip-10-132-38-160 ~]$ sudo nano
[cloudshell-user@ip-10-132-38-160 ~]$ gcc -o go p
pipes.c      proj1.c      proj1-o.c    pthreads1.c
[cloudshell-user@ip-10-132-38-160 ~]$ gcc -o go pthreads1.c
[cloudshell-user@ip-10-132-38-160 ~]$ ./go
Vetor resultante y:
15.000000 21.000000 27.000000 33.000000 39.000000 45.000000
[cloudshell-user@ip-10-132-38-160 ~]$
```