Disciplina: Sistema Operacionais Embarcados **Código:** 120961 **Turma:** A
Professor: Diogo Caetano Garcia
Aluno/Matrícula: Fábio Barbosa Pinto – 11/0116356

Questionário: 08_Threads_Mutexes_2

1. Crie um programa em C que cria uma thread, e faça com que o programa principal envie os valores 1, 2, 3, 4, 5, 6, 7, 8, 9 e 10 para a thread, com intervalos de 1 segundo entre cada envio. Depois de o programa principal enviar o número 10, ele aguarda 1 segundo e termina a execução. A thread escreve na tela cada valor recebido, e quando ela receber o valor 10, ela termina a execução.

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

#define PARADA 10

int count = 0;

void* print(void* inteiro)
{
	int * contador = (int *) inteiro;
	while(1 || *contador != 10)
	{
		printf("%d\n", *contador);
		sleep(1);
	}
	return NULL;
}

int main ()
{
	int i;
	pthread_t thread_id1;

	pthread_create (&thread_id1, NULL, &print, &count);

	for(i = 0; i < PARADA; i++)
	{
		count = i + 1;

		sleep(1);
	}
```

```
	return 0;
}
```

2. Crie um programa em C que preenche o vetor long int v[50000] completamente com valores aleatórios (use a função random()), e que procura o valor máximo do vetor por dois métodos:

(a) Pela busca completa no vetor v[];

(b) Separando o vetor em 4 partes, e usando 4 threads para cada uma encontrar o máximo de cada parte. Ao final das threads, o programa principal compara o resultado das quatro threads para definir o máximo do vetor.

Ao final do programa principal, compare os resultados obtidos pelos dois métodos.

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

#define PARADA 12500

struct vetor
{
	long int v[50000];
	long int maximo_total;
	long int maximo_thread[4];
};

void* encontra_maximo_pt1 (void *vetor)
{
	struct vetor* vect = (struct vetor *) vetor;
	long int maior, temp[PARADA];
	int i;

	for(i = 0; i < PARADA; i++)
	{
		temp[i] = vect->v[i];
	}

	maior = temp[0];

	for(i = 1; i < PARADA; i++)
	{
		if (maior < temp[i])
		{
			maior = temp[i];
		}
```

```
	}

	vect->maximo_thread[0] = maior;

	return NULL;
}

void* encontra_maximo_pt2 (void *vetor)
{
	struct vetor* vect = (struct vetor *) vetor;
	long int maior, temp[PARADA];
	int i;

	for(i = 0; i < PARADA; i++)
	{
		temp[i] = vect->v[PARADA + i];
	}

	maior = temp[0];

	for(i = 1; i < PARADA; i++)
	{
		if (maior < temp[i])
		{
			maior = temp[i];
		}
	}

	vect->maximo_thread[1] = maior;

	return NULL;
}

void* encontra_maximo_pt3 (void *vetor)
{
	struct vetor* vect = (struct vetor *) vetor;
	long int maior, temp[PARADA];
	int i;

	for(i = 0; i < PARADA; i++)
	{
		temp[i] = vect->v[2*PARADA + i];
	}

	maior = temp[0];

	for(i = 1; i < PARADA; i++)
```

```
	{
		if (maior < temp[i])
		{
			maior = temp[i];
		}
	}

	vect->maximo_thread[2] = maior;

	return NULL;
}

void* encontra_maximo_pt4 (void *vetor)
{
	struct vetor* vect = (struct vetor *) vetor;
	long int maior, temp[PARADA];
	int i;

	for(i = 0; i < PARADA; i++)
	{
		temp[i] = vect->v[3 * PARADA + i];
	}

	maior = temp[0];

	for(i = 1; i < PARADA; i++)
	{
		if (maior < temp[i])
		{
			maior = temp[i];
		}
	}

	vect->maximo_thread[3] = maior;

	return NULL;
}


int main ()
{
	int i;
	pthread_t thread_id1;
	pthread_t thread_id2;
	pthread_t thread_id3;
	pthread_t thread_id4;
	struct vetor random_vetor;
```

```
	long int maior;

	for(i = 0; i < 4*PARADA; i++)
		random_vetor.v[i] = rand();

	pthread_create (&thread_id1, NULL, &encontra_maximo_pt1, &random_vetor);
	pthread_create (&thread_id2, NULL, &encontra_maximo_pt2, &random_vetor);
	pthread_create (&thread_id3, NULL, &encontra_maximo_pt3, &random_vetor);
	pthread_create (&thread_id4, NULL, &encontra_maximo_pt4, &random_vetor);

	maior = random_vetor.v[0];
	for(i = 1; i < PARADA; i++)
	{
		if(maior < random_vetor.v[i])
		{
			maior = random_vetor.v[i];
		}
	}

	random_vetor.maximo_total = maior;

	pthread_join (thread_id1, NULL);
	pthread_join (thread_id2, NULL);
	pthread_join (thread_id3, NULL);
	pthread_join (thread_id4, NULL);

	maior = random_vetor.maximo_thread[0];

	for(i = 1; i < 4; i++)
	{
		if (maior < random_vetor.maximo_thread[i])
		{
			maior < random_vetor.maximo_thread[i];
		}

	}

	printf("%ld <- busca completa \n", random_vetor.maximo_total);
	printf("%ld <- thread \n", maior);


	return 0;
}
```

Os resultados foram iguais.

3. Repita o exercício anterior, mas calcule a média do vetor ao invés do valor máximo.

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

#define PARADA 12500

struct vetor
{
	long int v[4*PARADA];
	double media_total;
	double media_thread[4];
};

void* encontra_maximo_pt1 (void *vetor)
{
	struct vetor* vect = (struct vetor *) vetor;
	long int temp[PARADA];
	double soma;
	int i;

	for(i = 0; i < PARADA; i++)
	{
		temp[i] = vect->v[i];
	}

	soma = 0;

	for(i = 0; i < PARADA; i++)
	{
		soma += temp[PARADA];
	}

	vect->media_thread[0] = soma/(PARADA*4);
	printf("%lf <- tread 1\n", soma/(PARADA*4));

	return NULL;
}

void* encontra_maximo_pt2 (void *vetor)
{
	struct vetor* vect = (struct vetor *) vetor;
	long int temp[PARADA];
	double soma;
	int i;
```

```
	for(i = 0; i < PARADA; i++)
	{
		temp[i] = vect->v[PARADA + i];
	}

	soma = 0;

	for(i = 0; i < PARADA; i++)
	{
		soma += temp[i];
	}

	vect->media_thread[1] = soma/(PARADA*4);

	printf("%lf <- tread 2\n", soma/(PARADA*4));

	return NULL;
}

void* encontra_maximo_pt3 (void *vetor)
{
	struct vetor* vect = (struct vetor *) vetor;
	long int temp[PARADA];
	double soma;
	int i;

	for(i = 0; i < PARADA; i++)
	{
		temp[i] = vect->v[2*PARADA + i];
	}

	soma = 0;

	for(i = 0; i < PARADA; i++)
	{
		soma += temp[i];
	}

	vect->media_thread[2] = soma/(PARADA*4);

		printf("%lf <- tread 3\n", soma/(PARADA*4));

	return NULL;
}

void* encontra_maximo_pt4 (void *vetor)
```

```
{
	struct vetor* vect = (struct vetor *) vetor;
	long int temp[PARADA];
	double soma;
	int i;

	for(i = 0; i < PARADA; i++)
	{
		temp[i] = vect->v[3 * PARADA + i];
	}

	soma = 0;

	for(i = 0; i < PARADA; i++)
	{
		soma += temp[i];
	}

	vect->media_thread[3] = soma/(PARADA*4);

	printf("%lf <- tread 4\n", soma/(PARADA*4));

	return NULL;
}


int main ()
{
	int i;
	pthread_t thread_id1;
	pthread_t thread_id2;
	pthread_t thread_id3;
	pthread_t thread_id4;
	struct vetor random_vetor;
	double soma;

	for(i = 0; i < 4*PARADA; i++)
		random_vetor.v[i] = rand();



	pthread_create (&thread_id1, NULL, &encontra_maximo_pt1, &random_vetor);
	pthread_create (&thread_id2, NULL, &encontra_maximo_pt2, &random_vetor);
	pthread_create (&thread_id3, NULL, &encontra_maximo_pt3, &random_vetor);
	pthread_create (&thread_id4, NULL, &encontra_maximo_pt4, &random_vetor);

	soma = 0;
```

```
	for(i = 1; i < PARADA; i++)
	{
		soma += random_vetor.v[i];
	}

	random_vetor.media_total = soma/(4*PARADA);

	pthread_join (thread_id1, NULL);
	pthread_join (thread_id2, NULL);
	pthread_join (thread_id3, NULL);
	pthread_join (thread_id4, NULL);

	soma = 0;
	for(i = 0; i < 4; i++)
	{
		soma += random_vetor.media_thread[i];
	}

	printf("%lf <- busca completa \n", random_vetor.media_total);
	printf("%lf <- thread \n", soma);


	return 0;
}
```