

Iniciado em	sexta, 8 jul 2022, 10:03
Estado	Finalizada
Concluída em	sexta, 8 jul 2022, 10:06
Tempo empregado	2 minutos 28 segundos
Avaliar	Ainda não avaliado

[Atendimento](#)

Questão 1

Correto

Atingiu 1,00 de 1,00

Quantas threads serão criadas após as linhas de código a seguir? Quantas coexistirão?

```
void* funcao_thread_1(void *arg);
void* funcao_thread_2(void *arg);

int main (int argc, char** argv)
{
    pthread_t t1, t2;
    pthread_create(&t1, NULL, funcao_thread_1, NULL);
    pthread_create(&t2, NULL, funcao_thread_2, NULL);
    return 0;
}
```

2 threads criadas e 3 coexistindo



```
void* funcao_thread_1(void *arg);
void* funcao_thread_2(void *arg);

int main (int argc, char** argv)
{
    pthread_t t1, t2;
    pthread_create(&t1, NULL, funcao_thread_1, NULL);
    pthread_create(&t2, NULL, funcao_thread_2, NULL);
    pthread_join(t1, NULL);
    pthread_join(t2, NULL);
    return 0;
}
```

2 threads criadas e 3 coexistindo



Atendimento

```
void* funcao_thread_1(void *arg);
void* funcao_thread_2(void *arg);

int main (int argc, char** argv)
{
    pthread_t t1, t2;
    pthread_create(&t1, NULL, funcao_thread_1, NULL);
    pthread_join(t1, NULL);
    pthread_create(&t2, NULL, funcao_thread_2, NULL);
    pthread_join(t2, NULL);
    return 0;
}
```

2 threads criadas e 2 coexistindo



Sua resposta está correta.

A resposta correta é:

```
void* funcao_thread_1(void *arg);
void* funcao_thread_2(void *arg);

int main (int argc, char** argv)
{
    pthread_t t1, t2;
    pthread_create(&t1, NULL, funcao_thread_1, NULL);
    pthread_create(&t2, NULL, funcao_thread_2, NULL);
    return 0;
}
```

→ 2 threads criadas e 3 coexistindo,

```
void* funcao_thread_1(void *arg);
void* funcao_thread_2(void *arg);

int main (int argc, char** argv)
{
    pthread_t t1, t2;
    pthread_create(&t1, NULL, funcao_thread_1, NULL);
    pthread_create(&t2, NULL, funcao_thread_2, NULL);
    pthread_join(t1, NULL);
    pthread_join(t2, NULL);
    return 0;
}
```

Atendimento

→ 2 threads criadas e 3 coexistindo,

```
void* funcao_thread_1(void *arg);
void* funcao_thread_2(void *arg);

int main (int argc, char** argv)
{
    pthread_t t1, t2;
    pthread_create(&t1, NULL, funcao_thread_1, NULL);
    pthread_join(t1, NULL);
    pthread_create(&t2, NULL, funcao_thread_2, NULL);
    pthread_join(t2, NULL);
    return 0;
}
```

→ 2 threads criadas e 2 coexistindo.

Questão **2**

Completo

Vale 1,00 ponto(s).

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

```
void* funcao (void* arg){
    int *valor = (int *) arg;
    printf("%d\n", *valor);

    return NULL;
}
```

```
int main(void){
    pthread_t thread1_id;
    int i;
    for ( i = 1; i < 11; i++)
    {
        pthread_create(&thread1_id, NULL, funcao, &i);
        sleep(1);
    }
    sleep(1);

    return 0;
}
```

 [_main.c](#)

Atendimento

Questão 3

Completo

Vale 1,00 ponto(s).

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 <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <pthread.h>
#include <semaphore.h>
#include <sys/wait.h>
#include <sys/time.h>
#define n 50000
```

```
long int busca_vetor(long int *v, int tamanho){
    int i;
    long int v_max = v[0];
    for ( i = 1; i < tamanho; i++)
    {
        if (v[i] > v_max){
            v_max = v[i];
        }
    }
}
```

Atendimento

```
    }  
}  
return v_max;  
}  
  
struct max_thread_parms  
{  
    long int *v;  
    int tam;  
    long int v_max_thread;  
};  
  
void *max_thread(void *parms){  
    struct max_thread_parms* p =(struct max_thread_parms*) parms;  
    p->v_max_thread = busca_vetor(p->v,p->tam);  
    return NULL;  
}  
  
int main(){  
    pthread_t thread_id1,thread_id2,thread_id3,thread_id4;  
    long int v[n];  
    long int v_max, v_max_thread;  
    int i;  
    struct max_thread_parms max_1,max_2,max_3,max_4;  
  
    srandom(time(NULL));  
    for ( i = 0; i < n; i++)
```

Atendimento

```
{
    v[i] = random();
}

v_max = busca_vetor(v,n);
printf("Realizando a busca completa no vetor, temos que o valor máximo é de %ld\n",v_max);

// *****

max_1.v = &(v[0*n/4]);
max_1.tam = n/4;
max_2.v = &(v[0*n/4]);
max_2.tam = n/4;
max_3.v = &(v[0*n/4]);
max_3.tam = n/4;
max_4.v = &(v[0*n/4]);
max_4.tam = n/4;

pthread_create(&thread_id1,NULL,&max_thread,&max_1);
pthread_create(&thread_id2,NULL,&max_thread,&max_2);
pthread_create(&thread_id3,NULL,&max_thread,&max_3);
pthread_create(&thread_id4,NULL,&max_thread,&max_4);

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


v_max_thread = max_1.v_max_thread;
```

Atendimento


```
if (max_2.v_max_thread > v_max_thread)
{
    v_max_thread = max_2.v_max_thread;
}
if (max_3.v_max_thread > v_max_thread)
{
    v_max_thread = max_3.v_max_thread;
}
if (max_4.v_max_thread > v_max_thread)
{
    v_max_thread = max_4.v_max_thread;
}

printf("Utilizando o cálculo paralelizado, temos que o valor máximo é de %ld\n",v_max_thread);

return 0;
}
```

 [main.c](#)

Questão 4

Completo

Vale 1,00 ponto(s).

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 médio 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 valor médio de cada parte. Ao final das threads, o programa principal utiliza os resultados das quatro threads para definir a média do vetor.

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

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <pthread.h>
#include <semaphore.h>
#include <sys/types.h>
#define n 50000
```

```
double calcula_media(long int *v, int tamanho){
    int i;
    long int sum=0;
    double media=0.0;
    for ( i = 0; i < tamanho; i++)
    {
        sum +=v[i];
    }
    media += (double) sum/tamanho;
```

Atendimento

```
    return media;
}

struct media_thread_parms
{
    long int *v;
    int tam;
    long int v_media_thread;
};

void *media_thread(void *parms){
    struct media_thread_parms* p =(struct media_thread_parms*) parms;
    p->v_media_thread = calcula_media(p->v,p->tam);
    return NULL;
}

int main(){
    long int v[n];
    double media;
    pthread_t thread1,thread2,thread3,thread4;
    struct media_thread_parms media_1,media_2,media_3,media_4;

    srandom(time(NULL));
    for (size_t i = 0; i < n; i++)
    {
        v[i] = random();
    }
}
```

Atendimento

```
media = calcula_media(v,n);
printf("O valor médio encontrado foi de %lf\n",media);

//*****

media_1.v = &(v[0*n/4]);
media_1.tam = n/4;
media_2.v = &(v[0*n/4]);
media_2.tam = n/4;
media_3.v = &(v[0*n/4]);
media_3.tam = n/4;
media_4.v = &(v[0*n/4]);
media_4.tam = n/4;

pthread_create(&thread1,NULL,&media_thread, &media_1);
pthread_create(&thread2,NULL,&media_thread, &media_2);
pthread_create(&thread3,NULL,&media_thread, &media_3);
pthread_create(&thread4,NULL,&media_thread, &media_4);

pthread_join(thread1,NULL);
pthread_join(thread2,NULL);
pthread_join(thread3,NULL);
pthread_join(thread4,NULL);

media = (media_1.v_media_thread + media_2.v_media_thread + media_3.v_media_thread + media_4.v_media_thread)/4;
printf("O valor média encontrado paralelamente foi de %lf\n",media);

return 0;
}
```

Atendimento

 [main.c](#)[Atendimento](#)