|  |
| --- |
| 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 <unistd.h> |
|  |  |
|  | void\* print\_xs (void \*p) |
|  | { |
|  | int \*v = p; |
|  |  |
|  | for(int i = 0 ; i < 10 ; i++) |
|  | printf("%d ", v[i]); |
|  |  |
|  | return NULL; |
|  | } |
|  | int main () |
|  | { |
|  | int v[10]; |
|  |  |
|  | for(int i = 0 ; i < 10 ; i++) |
|  | { |
|  | v[i] = i + 1; |
|  | sleep(1); |
|  | } |
|  | pthread\_t thread\_id; |
|  | pthread\_create (&thread\_id, NULL, &print\_xs, v); |
|  | pthread\_join (thread\_id, NULL); |
|  |  |
|  | 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> |
|  | long int max1=0, max2=0, max3=0, max4=0; |
|  |  |
|  | void\* thread1 (void \*p) |
|  | { |
|  | long int \*v = p; |
|  | for(int i = 0 ; i < 12500 ; i++) |
|  | if(max1 < v[i]) |
|  | max1 = v[i]; |
|  | return NULL; |
|  | } |
|  | void\* thread2 (void \*p) |
|  | { |
|  | long int \*v = p; |
|  | for(int i = 12500 ; i < 2\*12500 ; i++) |
|  | if(max2 < v[i]) |
|  | max2 = v[i]; |
|  | return NULL; |
|  | } |
|  | void\* thread3 (void \*p) |
|  | { |
|  | long int \*v = p; |
|  | for(int i = 2\*12500 ; i < 3\*12500 ; i++) |
|  | if(max3 < v[i]) |
|  | max3 = v[i]; |
|  | return NULL; |
|  | } |
|  | void\* thread4 (void \*p) |
|  | { |
|  | long int \*v = p; |
|  | for(int i = 3\*12500 ; i < 50000 ; i++) |
|  | if(max4 < v[i]) |
|  | max4 = v[i]; |
|  | return NULL; |
|  | } |
|  | int main () |
|  | { |
|  | long int v[50000], max = 0; |
|  |  |
|  | for(int i = 0 ; i < 50000 ; i++) |
|  | v[i] = random(); |
|  |  |
|  | clock\_t begin = clock(); |
|  |  |
|  | for(int i = 0 ; i < 50000 ; i++) |
|  | if(v[i]>max) |
|  | max = v[i]; |
|  |  |
|  | clock\_t end = clock(); |
|  | double time\_spent = (double)(end - begin) / CLOCKS\_PER\_SEC; |
|  | printf("Valor maximo: %ld em %f segundos\n", max, time\_spent); |
|  |  |
|  | begin = clock(); |
|  |  |
|  | pthread\_t thread\_id1; |
|  | pthread\_t thread\_id2; |
|  | pthread\_t thread\_id3; |
|  | pthread\_t thread\_id4; |
|  | pthread\_create (&thread\_id1, NULL, &thread1, v); |
|  | pthread\_create (&thread\_id2, NULL, &thread2, v); |
|  | pthread\_create (&thread\_id3, NULL, &thread3, v); |
|  | pthread\_create (&thread\_id4, NULL, &thread4, v); |
|  | pthread\_join (thread\_id1, NULL); |
|  | pthread\_join (thread\_id2, NULL); |
|  | pthread\_join (thread\_id3, NULL); |
|  | pthread\_join (thread\_id4, NULL); |
|  |  |
|  | if(max2 > max1) |
|  | max1 = max2; |
|  | if(max4 > max3) |
|  | max3 = max4; |
|  | if(max3>max1) |
|  | max1 = max3; |
|  |  |
|  | end = clock(); |
|  | time\_spent = (double)(end - begin) / CLOCKS\_PER\_SEC; |
|  | printf("Valor maximo: %ld em %f segundos\n", max1, time\_spent); |
|  |  |
|  | return 0; |
|  | } |
|  |  |
|  | 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> |
|  | long double Soma1=0, Soma2=0, Soma3=0, Soma4=0; |
|  |  |
|  | void\* thread1 (void \*p) |
|  | { |
|  | long int \*v = p; |
|  | for(int i = 0 ; i < 12500 ; i++) |
|  | Soma1 += v[i]; |
|  | return NULL; |
|  | } |
|  | void\* thread2 (void \*p) |
|  | { |
|  | long int \*v = p; |
|  | for(int i = 12500 ; i < 2\*12500 ; i++) |
|  | Soma2 += v[i]; |
|  | return NULL; |
|  | } |
|  | void\* thread3 (void \*p) |
|  | { |
|  | long int \*v = p; |
|  | for(int i = 2\*12500 ; i < 3\*12500 ; i++) |
|  | Soma3 += v[i]; |
|  | return NULL; |
|  | } |
|  | void\* thread4 (void \*p) |
|  | { |
|  | long int \*v = p; |
|  | for(int i = 3\*12500 ; i < 50000 ; i++) |
|  | Soma4 += v[i]; |
|  | return NULL; |
|  | } |
|  | int main () |
|  | { |
|  | long int v[50000]; |
|  | long double Soma = 0; |
|  |  |
|  | for(int i = 0 ; i < 50000 ; i++) |
|  | v[i] = random(); |
|  |  |
|  | clock\_t begin = clock(); |
|  |  |
|  | for(int i = 0 ; i < 50000 ; i++) |
|  | Soma += v[i]; |
|  |  |
|  | clock\_t end = clock(); |
|  | double time\_spent = (double)(end - begin) / CLOCKS\_PER\_SEC; |
|  | printf("Media: %Le em %f segundos\n", Soma/50000, time\_spent); |
|  |  |
|  | begin = clock(); |
|  |  |
|  | pthread\_t thread\_id1; |
|  | pthread\_t thread\_id2; |
|  | pthread\_t thread\_id3; |
|  | pthread\_t thread\_id4; |
|  | pthread\_create (&thread\_id1, NULL, &thread1, v); |
|  | pthread\_create (&thread\_id2, NULL, &thread2, v); |
|  | pthread\_create (&thread\_id3, NULL, &thread3, v); |
|  | pthread\_create (&thread\_id4, NULL, &thread4, v); |
|  | pthread\_join (thread\_id1, NULL); |
|  | pthread\_join (thread\_id2, NULL); |
|  | pthread\_join (thread\_id3, NULL); |
|  | pthread\_join (thread\_id4, NULL); |
|  |  |
|  | end = clock(); |
|  | time\_spent = (double)(end - begin) / CLOCKS\_PER\_SEC; |
|  | printf("Media: %Le em %f segundos\n", (Soma1+Soma2+Soma3+Soma4)/50000, time\_spent); |
|  |  |
|  | return 0; |
|  | } |
|  |  |
|  | 4. Repita o exercício anterior, mas calcule a variância do vetor ao invés da média. |
|  |  |
|  | #include <pthread.h> |
|  | #include <stdio.h> |
|  | #include <stdlib.h> |
|  | #include <unistd.h> |
|  | #include <time.h> |
|  | long double Soma1=0, Soma2=0, Soma3=0, Soma4=0; |
|  |  |
|  | void\* thread1 (void \*p) |
|  | { |
|  | long int \*v = p; |
|  | for(int i = 0 ; i < 12500 ; i++) |
|  | Soma1 += v[i]; |
|  | return NULL; |
|  | } |
|  | void\* thread2 (void \*p) |
|  | { |
|  | long int \*v = p; |
|  | for(int i = 12500 ; i < 2\*12500 ; i++) |
|  | Soma2 += v[i]; |
|  | return NULL; |
|  | } |
|  | void\* thread3 (void \*p) |
|  | { |
|  | long int \*v = p; |
|  | for(int i = 2\*12500 ; i < 3\*12500 ; i++) |
|  | Soma3 += v[i]; |
|  | return NULL; |
|  | } |
|  | void\* thread4 (void \*p) |
|  | { |
|  | long int \*v = p; |
|  | for(int i = 3\*12500 ; i < 50000 ; i++) |
|  | Soma4 += v[i]; |
|  | return NULL; |
|  | } |
|  | int main () |
|  | { |
|  | long int v[50000]; |
|  | long double Soma = 0; |
|  |  |
|  | for(int i = 0 ; i < 50000 ; i++) |
|  | v[i] = random(); |
|  |  |
|  | clock\_t begin = clock(); |
|  |  |
|  | for(int i = 0 ; i < 50000 ; i++) |
|  | Soma += v[i]; |
|  |  |
|  | clock\_t end = clock(); |
|  | double time\_spent = (double)(end - begin) / CLOCKS\_PER\_SEC; |
|  | printf("Media: %Le em %f segundos\n", Soma/50000, time\_spent); |
|  |  |
|  | begin = clock(); |
|  |  |
|  | pthread\_t thread\_id1; |
|  | pthread\_t thread\_id2; |
|  | pthread\_t thread\_id3; |
|  | pthread\_t thread\_id4; |
|  | pthread\_create (&thread\_id1, NULL, &thread1, v); |
|  | pthread\_create (&thread\_id2, NULL, &thread2, v); |
|  | pthread\_create (&thread\_id3, NULL, &thread3, v); |
|  | pthread\_create (&thread\_id4, NULL, &thread4, v); |
|  | pthread\_join (thread\_id1, NULL); |
|  | pthread\_join (thread\_id2, NULL); |
|  | pthread\_join (thread\_id3, NULL); |
|  | pthread\_join (thread\_id4, NULL); |
|  |  |
|  | end = clock(); |
|  | time\_spent = (double)(end - begin) / CLOCKS\_PER\_SEC; |
|  | printf("Media: %Le em %f segundos\n", (Soma1+Soma2+Soma3+Soma4)/50000, time\_spent); |
|  |  |
|  | return 0; |