**梯形积分法OpenMP编程**

|  |
| --- |
| C #include <stdio.h> #include <stdlib.h> #include <omp.h>  void Trap(double a, double b, int n, double\* global\_result\_p);  int main(int argc, char\* argv[]) {  double global\_result = 0.0;  double a, b;  int n;  int thread\_count;   // 从命令行获取线程数  thread\_count = strtol(argv[1], NULL, 10);   printf("输入 a、b 和 n\n");  scanf("%lf %lf %d", &a, &b, &n);   // 使用指定数量的线程进行OpenMP并行计算  #pragma omp parallel num\_threads(thread\_count)  Trap(a, b, n, &global\_result);   // 输出结果  printf("当 n = %d 个梯形时，我们的估算\n", n);  printf("从 %f 到 %f 的积分 = %.14e\n", a, b, global\_result);   return 0; } /\* main \*/  // 并行执行梯形法则计算的函数 void Trap(double a, double b, int n, double\* global\_result\_p) {  double h, x, my\_result;  double local\_a, local\_b;  int i, local\_n;  int my\_rank = omp\_get\_thread\_num();  int thread\_count = omp\_get\_num\_threads();   // 计算每个梯形的宽度  h = (b - a) / n;   // 计算每个线程分配到的梯形数  local\_n = n / thread\_count;   // 计算每个线程的局部区间  local\_a = a + my\_rank \* local\_n \* h;  local\_b = local\_a + local\_n \* h;   // 对分配给每个线程的梯形执行梯形法则计算  my\_result = (f(local\_a) + f(local\_b)) / 2.0;  for (i = 1; i <= local\_n - 1; i++) {  x = local\_a + i \* h;  my\_result += f(x);  }  my\_result = my\_result \* h;   // 用临界区保证全局求和正确性  #pragma omp critical  \*global\_result\_p += my\_result; } /\* Trap \*/ |