```c
/* trap.c -- Parallel Trapezoidal Rule, first version
 *
 * Input: None.
 * Output:  Estimate of the integral from a to b of f(x)
 *    using the trapezoidal rule and n trapezoids.
 *
 * Algorithm:
 *    1.  Each process calculates "its" interval of
 *        integration.
 *    2.  Each process estimates the integral of f(x)
 *        over its interval using the trapezoidal rule.
 *    3a. Each process != 0 sends its integral to 0.
 *    3b. Process 0 sums the calculations received from
 *        the individual processes and prints the result.
 *
 * Notes:
 *    1.  f(x), a, b, and n are all hardwired.
 *    2.  The number of processes (p) should evenly divide
 *        the number of trapezoids (n = 1024)
 *
 */
#include <stdio.h>

/* We'll be using MPI routines, definitions, etc. */
#include "mpi.h"


main(int argc, char** argv) {
    int         my_rank;   /* My process rank           */
    int         p;         /* The number of processes   */
    float       a = 0.0;   /* Left endpoint             */
    float       b = 1.0;   /* Right endpoint            */
    int         n = 1024;  /* Number of trapezoids      */
    float       h;         /* Trapezoid base length     */
    float       local_a;   /* Left endpoint my process  */
    float       local_b;   /* Right endpoint my process */
    int         local_n;   /* Number of trapezoids for  */
                           /* my calculation            */
    float       integral;  /* Integral over my interval */
    float       total;     /* Total integral            */
    int         source;    /* Process sending integral  */
    int         dest = 0;  /* All messages go to 0      */
    int         tag = 0;
    MPI_Status  status;

    float Trap(float local_a, float local_b, int local_n,
              float h);    /* Calculate local integral  */

    /* Let the system do what it needs to start up MPI */
    MPI_Init(&argc, &argv);

    /* Get my process rank */
    MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);

    /* Find out how many processes are being used */
    MPI_Comm_size(MPI_COMM_WORLD, &p);

    h = (b-a)/n;    /* h is the same for all processes */
    local_n = n/p;  /* So is the number of trapezoids */

    /* Length of each process' interval of
     * integration = local_n*h.  So my interval
     * starts at: */
    local_a = a + my_rank*local_n*h;
    local_b = local_a + local_n*h;
    integral = Trap(local_a, local_b, local_n, h);

    /* Add up the integrals calculated by each process */
    if (my_rank == 0) {
        total = integral;
        for (source = 1; source < p; source++) {
            MPI_Recv(&integral, 1, MPI_FLOAT, source, tag,
                MPI_COMM_WORLD, &status);
            total = total + integral;
        }
    } else {
        MPI_Send(&integral, 1, MPI_FLOAT, dest,
            tag, MPI_COMM_WORLD);
    }

    /* Print the result */
    if (my_rank == 0) {
        printf("With n = %d trapezoids, our estimate\n",
            n);
        printf("of the integral from %f to %f = %f\n",
            a, b, total);
    }

    /* Shut down MPI */
    MPI_Finalize();
} /*  main  */


float Trap(
          float  local_a   /* in */,
          float  local_b   /* in */,
          int    local_n   /* in */,
          float  h         /* in */) {

    float integral;   /* Store result in integral  */
    float x;
    int i;

    float f(float x); /* function we're integrating */

    integral = (f(local_a) + f(local_b))/2.0;
    x = local_a;
    for (i = 1; i <= local_n-1; i++) {
        x = x + h;
        integral = integral + f(x);
    }
    integral = integral*h;
    return integral;
} /*  Trap  */


float f(float x) {
    float return_val;
    /* Calculate f(x). */
    /* Store calculation in return_val. */
    return_val = x*x;
    return return_val;
} /* f */
```