```c
// pi: calculate pi in parallel

#include <stdio.h>
#include <omp.h>

#define NUM_THREADS 4

int main(){
    double pi = 0.0;
    size_t num_steps = 1000000000;
    int nthreads;

    double step = 1.0/num_steps;
    double start_time = omp_get_wtime();

    omp_set_num_threads(NUM_THREADS);

    #pragma omp parallel
    {
        double sum = 0.0; double x;
        int id = omp_get_thread_num();
        if(id == 0) nthreads = omp_get_num_threads();
        #pragma omp barrier

        for (size_t i = id; i < num_steps; i += nthreads)
        {
            x = (i + 0.5) * step;
            sum += 4.0/(1.0 + x*x);
        }

        #pragma omp critical
        pi += sum*step;
    }
    double time = omp_get_wtime() - start_time;
    printf("Pi calculated with %d threads: %lf\n", nthreads, pi);
    printf("Calculated in %lf s\n", time);


    return 0;
}
```