```
/*  OpenMP program to demonstrate distributed iteration space.   */
/*  Critical section used to accumulate total.                   */

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

#define MAX_NUMS 2048
#define FNAME "nums"

int main()
{
/* Shared variables. */
   int * A ;
   int n   ;
   int total ;

/* Private variables. */
   int chunk_size     ;
   int start          ;
   int ptotal         ;
   int j, r, x, jend  ;
   int id, nprocs, n_mod_nprocs   ;
   FILE * fp          ;

#pragma omp parallel shared(n,A,total) \
        private(chunk_size,start,j,jend,id,nprocs,ptotal,fp,r,x,n_mod_nprocs)
   {
      nprocs = omp_get_num_threads() ;
      id = omp_get_thread_num()     ;

#pragma omp master
      {     /* The master thread does this. */
            A = (int *) malloc( MAX_NUMS * sizeof(int) ) ;
            if ( A == NULL ) {
                fprintf(stderr,"malloc failed !!\n" ) ;
                exit(1) ;
             }
             fp = fopen( FNAME, "r" ) ;
             if ( fp == NULL ) {
                fprintf(stderr,"unable to read file '%s'\n", FNAME ) ;
                exit(2) ;
             }
             n = 0 ;
             do {
                r = fscanf( fp, "%d", &x ) ;
                if ( ( r != EOF ) && ( n < MAX_NUMS ) ){
                   A[n] = x ;
                   n++ ;
                }
            } while ( r != EOF ) ;
            fclose( fp ) ;
            printf("%d numbers found.\n", n ) ;
            printf("%d procs.\n", nprocs ) ;

            total = 0 ;   /* Initialize the total. */
      }  /* end section: read-in data */

/* Every process must wait until thread 0 has updated A, n and total.    */
#pragma omp barrier
```

```
/*   Every process computes its chunk size and its start position in A.  */
   n_mod_nprocs = n % nprocs     ;
   if ( id < (n_mod_nprocs) ) {
       chunk_size = n / nprocs +  1 ;
       start = id * chunk_size      ;
   }
   else {
       chunk_size = n / nprocs ;
       start = (chunk_size + 1) * n_mod_nprocs  +
               ( id - n_mod_nprocs ) * chunk_size    ;
   }
   printf( "(%d):  chunk_size = %d, start = %d\n", id, chunk_size, start ) ;

   /* Every process computes a local sum. */
   ptotal = 0 ;
   jend = start + chunk_size ;
   for ( j = start ; j < jend ; j++ ) {
       ptotal += A[j] ;
   }
   /* printf( "(%d):  local sum = %d\n", id, ptotal ) ;   */

/*  Every process contributes its private total to the shared total. */
#pragma omp critical
{
   total += ptotal ;
}

/*  Wait for every process to finish updating 'total'. */
#pragma omp barrier

#pragma omp master
{
     printf("Total = %d\n", total ) ;
}

}  /* End parallel section. */

} /* End main() */
```

```
======================== Sample Session ========================
cosmos%
cosmos% gcc -fopenmp sum_demo.c
cosmos%
cosmos% setenv OMP_NUM_THREADS 7
cosmos%
cosmos% a.out
1024 numbers found.
7 procs.
(2):  chunk_size = 146, start = 294
(0):  chunk_size = 147, start = 0
(4):  chunk_size = 146, start = 586
(3):  chunk_size = 146, start = 440
(1):  chunk_size = 147, start = 147
(6):  chunk_size = 146, start = 878
(5):  chunk_size = 146, start = 732
Total = 519632
```