SCOPE OF VARIABLES

- In serial programming, the ***scope*** of a variable consists of those parts of a program in which the variable can be used.
- For example, a variable declared at the beginning of a C function has “function-wide” scope, that is, it can only be accessed in the body of the function.
- On the other hand, a variable declared at the beginning of ***a .c*** file but outside any function has “file-wide” scope, that is, any function in the filein which the variable is declared can access the variable
- In OpenMP, the ***scope*** of a variable refers to the set of threads that can access the variable in a ***parallel*** block. A variable that can be accessed by all the threads in the team has shared scope, whilea variable that can only be accessed by a single thread has private scope.
- In the “hello, world” program, the variables used by each thread (my_rank and thread_count) were declared in the ***Hello*** function, which is called inside the ***parallel*** block. Consequently, the variables used by each thread are allocated from the thread’s (private) stack, and hence all of the variables have private scope. This is almost the case in the trapezoidal rule program; since the parallel block is just a function call, all of the variables used by each thread in the ***Trap*** function are allocated from the thread’s stack.
- However, the variables that are declared in the *main* function (a, b, n,global_result, and thread_count) are all accessible to all the threads in the teamstarted by the ***parallel*** directive. Hence, the default scope for variables declaredbefore a ***parallel*** block is shared. In fact, we’ve made implicit use of this: eachthread in the team gets the values of a, b, and n from the call to ***Trap***. Since this calltakes place in the ***parallel*** block, it’s essential that each thread has access to a,b,and n when their values are copied into the corresponding formal arguments.