Directives	**Clauses it use**
parallel	copyin, default, private, firstprivate, reduction, shared
sections	private, firstprivate, lastprivate, reduction
section	private, firstprivate, lastprivate, reduction

Directives	**Description**
parallel end parallel	Defines a parallel region.

do
end do
	Identifies an iterative worksharing construct in which the iterations of the associated loop should be executed in parallel.

sections end sections	Identifies a non-iterative worksharing construct that specifies a set of structured blocks that are to be divided among threads in a team.
section	Indicates that the associated structured block should be executed in parallel as part of the enclosing sections construct.
master end master	Identifies a construct that specifies a structured block that is executed by only the master thread of the team.
critical[*lock*] end critical[*lock*]	Identifies a construct that restricts execution of the associated structured block to a single thread at a time. Each thread waits at the beginning of the critical construct until no other thread is executing a critical construct with the same *lock* argument.
barrier	Synchronizes all the threads in a team. Each thread waits until all of the other threads in that team have reached this point.
atomic	Ensures that a specific memory location is updated atomically, rather than exposing it to the possibility of multiple, simultaneously writing threads.
flush [(*list*)]	Specifies a "cross-thread" sequence point at which the implementation is required to ensure that all the threads in a team have a consistent view of certain objects in memory. The optional *list* argument consists of a comma-separated list of variables to be flushed.
ordered end ordered	The structured block following an ordered directive is executed in the order in which iterations would be executed in a sequential loop.

Clauses	**Description**
private (*list*)	Declares variables in *list* to be private To each thread in a team.
firstprivate (*list*)	Same as private, but the copy of each variable in the *list* is initialized using the value of the original variable existing before the construct.
lastprivate (*list*)	Same as private, but the original variables in *list* are updated using the values assigned to the corresponding private variables in the last iteration in the do construct loop or the last section construct.
copyprivate (*list*)	Uses private variables in *list* to broadcast values, or pointers to shared objects, from one member of a team to the other members at the end of a single construct.
nowait	Specifies that threads need not wait at the end of worksharing constructs until they have completed execution. The threads may proceed past the end of the worksharing constructs as soon as there is no more work available for them to execute.
shared (*list*)	Shares variables in *list* among all the threads in a team.
default (*mode*)	Determines the default data-scope attributes of variables not explicitly specified by another clause. Possible values for *mode* are private, shared, or none.
reduction ({operator|intrinsic}:*list*)	Performs a reduction on variables that appear in *list* with the operator operator or the intrinsic procedure name intrinsic; operator is one of the following: +, *, .and., .or., .eqv., .neqv.; intrinsic refers to one of the following: max, min, iand, ior, or ieor.
ordered end ordered	Used in conjunction with a do or sections construct to impose a serial order on the execution of a section of code. If ordered constructs are contained in the dynamic extent of the do construct, the ordered clause must be present on the do directive.
if (*scalar_logical_expression*)	The enclosed parallel region is executed in parallel only if the *scalar_logical_expression* evaluates to .true.; otherwise the parallel region is serialized.
num_threads(*scalar_integer_expression*)	Requests the number of threads specified by *scalar_integer_expression* for the parallel region.
schedule (*type*[,*chunk*])	Specifies how iterations of the do construct are divided among the threads of the team. Possible values for the *type* argument are static, dynamic, guided, and runtime. The optional *chunk* argument must be a positive scalar integer expression.
copyin (*list*)	Specifies that the master thread's data values be copied to the threadprivate's copies of the common blocks or variables specified in *list* at the beginning of the parallel region.