MPI Functions

```
int MPI_Send(void *buf, int count, MPI_Datatype datatype,
     int dest, int tag, MPI_Comm comm)
int MPI_Ssend(void *buf, int count, MPI_Datatype datatype,
     int dest, int tag, MPI_Comm comm)
int MPI_Bsend(void *buf, int count, MPI_Datatype datatype,
     int dest, int tag, MPI_Comm comm)

int MPI_Recv(void *buf, int count, MPI_Datatype datatype,
     int source, int tag, MPI_Comm comm, MPI_Status *status)

int MPI_Irecv(void *buf, int count, MPI_Datatype datatype,
     int source, int tag, MPI_Comm comm, MPI_Request *request)
int MPI_Isend(void *buf, int count, MPI_Datatype datatype,
     int dest, int tag, MPI_Comm comm, MPI_Request *request)

int MPI_Test(MPI_Request *request, int *flag, MPI_Status
     *status)
int MPI_Wait(MPI_Request *request, MPI_Status *status)

int MPI_Probe(int source, int tag, MPI_Comm comm,
     MPI_Status *status)
int MPI_Get_count(MPI_Status *status, MPI_Datatype datatype,
     int *count)

int MPI_Bcast(void *buffer, int count, MPI_Datatype datatype,
     int root, MPI_Comm comm)
int MPI_Scatter(void *sendbuf, int sendcount,
     MPI_Datatype sendtype,void *recvbuf, int recvcount,
     MPI_Datatype recvtype, int root, MPI_Comm comm)
int MPI_Gather(void *sendbuf, int sendcount,
     MPI_Datatype sendtype, void *recvbuf, int recvcount,
     MPI_Datatype recvtype, int root, MPI_Comm comm)
int MPI_Reduce(void *sendbuf, void *recvbuf, int count,
     MPI_Datatype datatype, MPI_Op op, int root, MPI_Comm comm)

int MPI_Type_contiguous(int count, MPI_Datatype oldtype,
     MPI_Datatype *newtype)
int MPI_Type_vector(int count, int blocklength, int stride,
     MPI_Datatype oldtype, MPI_Datatype *newtype)
int MPI_Type_commit(MPI_Datatype *datatype)
int MPI_Type_free(MPI_Ddatatype *datatype)
int MPI_Group_incl(MPI_Group group, int n, int *ranks,
     MPI_Group *newgroup)
int MPI_Group_excl(MPI_Group group, int n, int *ranks,
     MPI_Group *newgroup)
```

```
int MPI_Comm_group(MPI_Comm comm, MPI_Group *group)
int MPI_Comm_create(MPI_Comm comm, MPI_Group group,
     MPI_Comm *newcomm)
int MPI_Comm_split(MPI_Comm comm, int color, int key,
     MPI_Comm *newcomm)
int MPI_Comm_free(MPI_Comm *comm)

Operators for Reduce:    MPI_MAX, MPI_MIN, MPI_SUM, MPI_PROD,
                         MPI_LAND, MPI_LOR, …

MPI_Datatypes:           MPI_INT, MPI_DOUBLE, MPI_FLOAT,
                         MPI_CHAR, MPI_SHORT, MPI_LONG,
                         MPI_UNSIGNED, MPI_UNSIGNED_SHORT,
                         MPI_UNSIGNED_LONG, …
```

OpenMP Directives

The **parallel** construct forms a team of threads and starts parallel execution.

#pragma omp parallel *[clause[[*, *]clause] ...] new-line*
{
        //structured-block
}

clauses:
 if(*scalar-expression***)**
num_threads(*integer-expression***)**
default(shared | **none)**
private(*list***)**
firstprivate(*list***)**
shared(*list***)**
copyin(*list***)**
reduction(*operator***:** *list***)**

The loop construct specifies that the iterations of loops will be distributed among and executed by the encountering team of threads. The most common form of the for loop is shown below:

```
for(var = lb; var relational-op b; var += incr)
{
        //do something
}
```

#pragma omp for *[clause[[,] clause] ...] new-line*
        for-loops

clauses:
private(*list***)**
firstprivate(*list***)**
lastprivate(*list***)**
reduction(*operator***:** *list***)**
schedule(*kind[, chunk_size]***)**
collapse(*n***)**
ordered
nowait

reduction operators: +, *, -, max, min, &, &&, |, ||, ^