1
Example: Hello world

```
#include<stdio.h>
#include"mpi.h"
int main(int argc, char *argv[])
{
        int rc;
        rc = MPI_Init(&argc, &argv);
        printf("Hello world.\n");
        rc = MPI_Finalize();
        return 0;
}
```

2
Example : Environment Management Routine

```
#include "mpi.h"
#include <stdio.h>
int main(argc,argv)
int argc;
char *argv[]; {
int  numtasks, rank, len, rc;
char hostname[MPI_MAX_PROCESSOR_NAME];
rc = MPI_Init(&argc,&argv);
if (rc != MPI_SUCCESS) {
  printf ("Error starting MPI program. Terminating.\n");
  MPI_Abort(MPI_COMM_WORLD, rc);
  }
MPI_Comm_size(MPI_COMM_WORLD,&numtasks);
MPI_Comm_rank(MPI_COMM_WORLD,&rank);
MPI_Get_processor_name(hostname, &len);
printf ("Number of tasks= %d My rank= %d Running on %s\n", numtasks,rank,hostname);
/*******  do some work *******/
rc = MPI_Finalize();
return 0;
}
```

3

Example : Blocking Message Passing Routine

```
#include "mpi.h"
#include <stdio.h>
int main(argc,argv)
int argc;
char *argv[];  {
int numtasks, rank, dest, source, rc, count, tag=1;
char inmsg, outmsg='x';
MPI_Status Stat;
MPI_Init(&argc,&argv);
MPI_Comm_size(MPI_COMM_WORLD, &numtasks);
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
if (rank == 0) {
 dest = 1;
 source = 1;
 rc = MPI_Send(&outmsg, 1, MPI_CHAR, dest, tag, MPI_COMM_WORLD);
 rc = MPI_Recv(&inmsg, 1, MPI_CHAR, source, tag, MPI_COMM_WORLD, &Stat);
 }
else if (rank == 1) {
 dest = 0;
 source = 0;
 rc = MPI_Recv(&inmsg, 1, MPI_CHAR, source, tag, MPI_COMM_WORLD, &Stat);
 rc = MPI_Send(&outmsg, 1, MPI_CHAR, dest, tag, MPI_COMM_WORLD);
 }

rc = MPI_Get_count(&Stat, MPI_CHAR, &count);
printf("Task %d: Received %d char(s) from task %d with tag %d \n",
     rank, count, Stat.MPI_SOURCE, Stat.MPI_TAG);
MPI_Finalize();
return 0;
}
```

4

Example : Dead Lock

```
#include "mpi.h"
#include <stdio.h>
int main(argc,argv)
int argc;
char *argv[];  {
int numtasks, rank, dest, source, rc, count, tag=1;
char inmsg, outmsg='x';
MPI_Status Stat;
MPI_Init(&argc,&argv);
MPI_Comm_size(MPI_COMM_WORLD, &numtasks);
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
if (rank == 0) {
 dest = 1;
 source = 1;
 rc = MPI_Send(&outmsg, 1, MPI_CHAR, dest, tag, MPI_COMM_WORLD);
 rc = MPI_Recv(&inmsg, 1, MPI_CHAR, source, tag, MPI_COMM_WORLD, &Stat);
 }
else if (rank == 1) {
 dest = 0;
 source = 0;
 rc = MPI_Send(&outmsg, 1, MPI_CHAR, dest, tag, MPI_COMM_WORLD);
 rc = MPI_Recv(&inmsg, 1, MPI_CHAR, source, tag, MPI_COMM_WORLD, &Stat);
}

rc = MPI_Get_count(&Stat, MPI_CHAR, &count);
printf("Task %d: Received %d char(s) from task %d with tag %d \n",
     rank, count, Stat.MPI_SOURCE, Stat.MPI_TAG);
MPI_Finalize();
return 0;
}
```

5

Example : Non-Blocking Message Passing Routine

```
#include "mpi.h"
#include <stdio.h>
int main(argc,argv)
int argc;
char *argv[];  {
int numtasks, rank, next, prev, buf[2], tag1=1, tag2=2;
MPI_Request reqs[4];
MPI_Status stats[2];
MPI_Init(&argc,&argv);
MPI_Comm_size(MPI_COMM_WORLD, &numtasks);
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
prev = rank-1;
next = rank+1;
if (rank == 0)  prev = numtasks - 1;
if (rank == (numtasks - 1))  next = 0;

MPI_Irecv(&buf[0], 1, MPI_INT, prev, tag1, MPI_COMM_WORLD, &reqs[0]);
MPI_Irecv(&buf[1], 1, MPI_INT, next, tag2, MPI_COMM_WORLD, &reqs[1]);
MPI_Isend(&rank, 1, MPI_INT, prev, tag2, MPI_COMM_WORLD, &reqs[2]);
MPI_Isend(&rank, 1, MPI_INT, next, tag1, MPI_COMM_WORLD, &reqs[3]);

    {  do some work  }
MPI_Waitall(4, reqs, stats);
MPI_Finalize();
return 0;
}
```

6
Advanced Example : Monte-Carlo Simulation for PI

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

int main() {
const long num_step=100000000;
long i, cnt;
double pi, x, y, r;
printf("------------------------------------------------------------\n");
pi = 0.0;
cnt = 0;
r = 0.0;
for (i=0; i<num_step; i++) {
  x = rand() / (RAND_MAX+1.0);
  y = rand() / (RAND_MAX+1.0);
  r = sqrt(x*x + y*y);
  if (r<=1) cnt += 1;
}
pi = 4.0 * (double)(cnt) / (double)(num_step);
printf("PI = %17.15lf (Error = %e)\n", pi, fabs(acos(-1.0)-pi));
printf("------------------------------------------------------------\n");
return 0;
}
```

7

Advanced Example : Numerical integration for PI

```
#include <stdio.h>
#include <math.h>
int main() {
const long num_step=100000000;
long i;
double sum, step, pi, x;
step = (1.0/(double)num_step);
sum=0.0;
printf("------------------------------------------------------------\n");
for (i=0; i<num_step; i++) {
x = ((double)i - 0.5) * step;
sum += 4.0/(1.0+x*x);
}
pi = step * sum;

printf("PI = %5lf (Error = %e)\n", pi, fabs(acos(-1.0)-pi));
printf("------------------------------------------------------------\n");
return 0;
}
```

8

Example : Collective Communication

```
#include "mpi.h"
#include <stdio.h>
#define SIZE 4
int main(argc,argv)
int argc;
char *argv[];  {
int numtasks, rank, sendcount, recvcount, source;
float sendbuf[SIZE][SIZE] = {
  {1.0, 2.0, 3.0, 4.0},
  {5.0, 6.0, 7.0, 8.0},
  {9.0, 10.0, 11.0, 12.0},
  {13.0, 14.0, 15.0, 16.0}  };
float recvbuf[SIZE];
MPI_Init(&argc,&argv);
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
MPI_Comm_size(MPI_COMM_WORLD, &numtasks);

if (numtasks == SIZE) {
  source = 1;
  sendcount = SIZE;
  recvcount = SIZE;
  MPI_Scatter(sendbuf,sendcount,MPI_FLOAT,recvbuf,recvcount,
           MPI_FLOAT,source,MPI_COMM_WORLD);
  printf("rank= %d  Results: %f %f %f %f\n",rank,recvbuf[0],
        recvbuf[1],recvbuf[2],recvbuf[3]);
  }
else
  printf("Must specify %d processors. Terminating.\n",SIZE);
MPI_Finalize();
return 0;
}
```

9

Example : Contiguous Derived Data Type

```
#include "mpi.h"
#include <stdio.h>
#define SIZE 4
int main(argc,argv)
int argc;
char *argv[];  {
int numtasks, rank, source=0, dest, tag=1, i;
float a[SIZE][SIZE] =
  {1.0, 2.0, 3.0, 4.0,
   5.0, 6.0, 7.0, 8.0,
   9.0, 10.0, 11.0, 12.0,
   13.0, 14.0, 15.0, 16.0};
float b[SIZE];
MPI_Status stat;
MPI_Datatype rowtype;
MPI_Init(&argc,&argv);
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
MPI_Comm_size(MPI_COMM_WORLD, &numtasks);
MPI_Type_contiguous(SIZE, MPI_FLOAT, &rowtype);
MPI_Type_commit(&rowtype);

if (numtasks == SIZE) {
  if (rank == 0) {
    for (i=0; i<numtasks; i++)
      MPI_Send(&a[i][0], 1, rowtype, i, tag, MPI_COMM_WORLD);
    }
  MPI_Recv(b, SIZE, MPI_FLOAT, source, tag, MPI_COMM_WORLD, &stat);
  printf("rank= %d  b= %3.1f %3.1f %3.1f %3.1f\n",
        rank,b[0],b[1],b[2],b[3]);
  }
else
  printf("Must specify %d processors. Terminating.\n",SIZE);
MPI_Type_free(&rowtype);
MPI_Finalize();
return 0;
}
```

10

Example : Vector Derived Data Type

```
#include "mpi.h"
#include <stdio.h>
#define SIZE 4
int main(argc,argv)
int argc;
char *argv[];  {
int numtasks, rank, source=0, dest, tag=1, i;
float a[SIZE][SIZE] =
  {1.0, 2.0, 3.0, 4.0,
   5.0, 6.0, 7.0, 8.0,
   9.0, 10.0, 11.0, 12.0,
  13.0, 14.0, 15.0, 16.0};
float b[SIZE];
MPI_Status stat;
MPI_Datatype columntype;
MPI_Init(&argc,&argv);
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
MPI_Comm_size(MPI_COMM_WORLD, &numtasks);

MPI_Type_vector(SIZE, 1, SIZE, MPI_FLOAT, &columntype);
MPI_Type_commit(&columntype);

if (numtasks == SIZE) {
  if (rank == 0) {
     for (i=0; i<numtasks; i++)
       MPI_Send(&a[0][i], 1, columntype, i, tag, MPI_COMM_WORLD);
       }

  MPI_Recv(b, SIZE, MPI_FLOAT, source, tag, MPI_COMM_WORLD, &stat);
  printf("rank= %d  b= %3.1f %3.1f %3.1f %3.1f\n",
       rank,b[0],b[1],b[2],b[3]);
  }
else
  printf("Must specify %d processors. Terminating.\n",SIZE);

MPI_Type_free(&columntype);
MPI_Finalize();
return 0;
}
```

11

Example : Indexed Derived Data Type

```
#include "mpi.h"
#include <stdio.h>
#define NELEMENTS 6
int main(argc,argv)
int argc;
char *argv[];  {
int numtasks, rank, source=0, dest, tag=1, i;
int blocklengths[2], displacements[2];
float a[16] =
  {1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0,
   9.0, 10.0, 11.0, 12.0, 13.0, 14.0, 15.0, 16.0};
float b[NELEMENTS];
MPI_Status stat;
MPI_Datatype indextype;
MPI_Init(&argc,&argv);
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
MPI_Comm_size(MPI_COMM_WORLD, &numtasks);
blocklengths[0] = 4;
blocklengths[1] = 2;
displacements[0] = 5;
displacements[1] = 12;

MPI_Type_indexed(2, blocklengths, displacements, MPI_FLOAT, &indextype);
MPI_Type_commit(&indextype);
if (rank == 0) {
  for (i=0; i<numtasks; i++)
    MPI_Send(a, 1, indextype, i, tag, MPI_COMM_WORLD);
  }

MPI_Recv(b, NELEMENTS, MPI_FLOAT, source, tag, MPI_COMM_WORLD, &stat);
printf("rank= %d  b= %3.1f %3.1f %3.1f %3.1f %3.1f %3.1f\n",
    rank,b[0],b[1],b[2],b[3],b[4],b[5]);

MPI_Type_free(&indextype);
MPI_Finalize();
}
```

12

Example : Struct Derived Data Type

```
#include "mpi.h"
#include <stdio.h>
#define NELEM 25
int main(argc,argv)
int argc;
char *argv[];  {
int numtasks, rank, source=0, dest, tag=1, i;
typedef struct {
  float x, y, z;
  float velocity;
  int  n, type;
  }        Particle;
Particle     p[NELEM], particles[NELEM];
MPI_Datatype particletype, oldtypes[2];
int         blockcounts[2];
/* MPI_Aint type used to be consistent with syntax of */
/* MPI_Type_extent routine */
MPI_Aint    offsets[2], extent;

MPI_Status stat;
MPI_Init(&argc,&argv);
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
MPI_Comm_size(MPI_COMM_WORLD, &numtasks);

/* Setup description of the 4 MPI_FLOAT fields x, y, z, velocity */
offsets[0] = 0;
oldtypes[0] = MPI_FLOAT;
blockcounts[0] = 4;

/* Setup description of the 2 MPI_INT fields n, type */
/* Need to first figure offset by getting size of MPI_FLOAT */
MPI_Type_extent(MPI_FLOAT, &extent);
offsets[1] = 4 * extent;
oldtypes[1] = MPI_INT;
blockcounts[1] = 2;

/* Now define structured type and commit it */
MPI_Type_struct(2, blockcounts, offsets, oldtypes, &particletype);
MPI_Type_commit(&particletype);
```

```
/* Initialize the particle array and then send it to each task */
if (rank == 0) {
 for (i=0; i<NELEM; i++) {
   particles[i].x = i * 1.0;
   particles[i].y = i * -1.0;
   particles[i].z = i * 1.0;
   particles[i].velocity = 0.25;
   particles[i].n = i;
   particles[i].type = i % 2;
   }
 for (i=0; i<numtasks; i++)
   MPI_Send(particles, NELEM, particletype, i, tag, MPI_COMM_WORLD);
 }

MPI_Recv(p, NELEM, particletype, source, tag, MPI_COMM_WORLD, &stat);
/* Print a sample of what was received */
printf("rank= %d   %3.2f %3.2f %3.2f %3.2f %d %d\n", rank,p[3].x,
   p[3].y,p[3].z,p[3].velocity,p[3].n,p[3].type);

MPI_Type_free(&particletype);
MPI_Finalize();
return 0;
}
```

13

Example : Group and Communicator Routine

```
#include "mpi.h"
#include <stdio.h>
#define NPROCS 8
int main(argc,argv)
int argc;
char *argv[];  {
int       rank, new_rank, sendbuf, recvbuf, numtasks,
          ranks1[4]={0,1,2,3}, ranks2[4]={4,5,6,7};
MPI_Group  orig_group, new_group;
MPI_Comm   new_comm;
MPI_Init(&argc,&argv);
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
MPI_Comm_size(MPI_COMM_WORLD, &numtasks);
if (numtasks != NPROCS) {
  printf("Must specify MP_PROCS= %d. Terminating.\n",NPROCS);
  MPI_Finalize();
  exit(0);
  }
sendbuf = rank;

/* Extract the original group handle */
MPI_Comm_group(MPI_COMM_WORLD, &orig_group);
/* Divide tasks into two distinct groups based upon rank */
if (rank < NPROCS/2) {
  MPI_Group_incl(orig_group, NPROCS/2, ranks1, &new_group);
  }
else {
  MPI_Group_incl(orig_group, NPROCS/2, ranks2, &new_group);
  }
/* Create new new communicator and then perform collective communications */
MPI_Comm_create(MPI_COMM_WORLD, new_group, &new_comm);
MPI_Allreduce(&sendbuf, &recvbuf, 1, MPI_INT, MPI_SUM, new_comm);
MPI_Group_rank (new_group, &new_rank);
printf("rank= %d newrank= %d recvbuf= %d\n",rank,new_rank,recvbuf);
MPI_Finalize();
return 0;
}
```

Example : Cartesian Virtual Topology (2/3)

```
#include "mpi.h"
#include <stdio.h>
#define SIZE 16
#define UP    0
#define DOWN  1
#define LEFT  2
#define RIGHT 3
int main(argc,argv)
int argc;
char *argv[];  {
int numtasks, rank, source, dest, outbuf, i, tag=1,
  inbuf[4]={MPI_PROC_NULL,MPI_PROC_NULL,MPI_PROC_NULL,MPI_PROC_NULL,},
  nbrs[4], dims[2]={4,4},
  periods[2]={0,0}, reorder=0, coords[2];
MPI_Request reqs[8];
MPI_Status stats[8];
MPI_Comm cartcomm;
MPI_Init(&argc,&argv);
MPI_Comm_size(MPI_COMM_WORLD, &numtasks);
if (numtasks == SIZE) {
 MPI_Cart_create(MPI_COMM_WORLD, 2, dims, periods, reorder, &cartcomm);
 MPI_Comm_rank(cartcomm, &rank);
 MPI_Cart_coords(cartcomm, rank, 2, coords);
 MPI_Cart_shift(cartcomm, 0, 1, &nbrs[UP], &nbrs[DOWN]);
 MPI_Cart_shift(cartcomm, 1, 1, &nbrs[LEFT], &nbrs[RIGHT]);

 printf("rank= %d coords= %d %d  neighbors(u,d,l,r)= %d %d %d %d\n",
      rank,coords[0],coords[1],nbrs[UP],nbrs[DOWN],nbrs[LEFT],
      nbrs[RIGHT]);
 outbuf = rank;
 for (i=0; i<4; i++) {
   dest = nbrs[i];
   source = nbrs[i];
   MPI_Isend(&outbuf, 1, MPI_INT, dest, tag,
           MPI_COMM_WORLD, &reqs[i]);
   MPI_Irecv(&inbuf[i], 1, MPI_INT, source, tag,
           MPI_COMM_WORLD, &reqs[i+4]);
   }
 MPI_Waitall(8, reqs, stats);

 printf("rank= %d               inbuf(u,d,l,r)= %d %d %d %d\n",
      rank,inbuf[UP],inbuf[DOWN],inbuf[LEFT],inbuf[RIGHT]);  }
else
 printf("Must specify %d processors. Terminating.\n",SIZE);

MPI_Finalize();
return 0;
}
```