| 1) |
| --- |
|  | #include<mpi.h> |
|  | #include<string.h> |
|  | #define BUFFER\_SIZE 32 |
|  |  |
|  | int main(int argc, char\* argv[]) |
|  | { |
|  | int rank,numprocs,tag=0,root=3,temp=1; |
|  | char msg[BUFFER\_SIZE]; |
|  | MPI\_Init(&argc,&argv); |
|  | MPI\_Status status; |
|  | MPI\_Comm\_rank(MPI\_COMM\_WORLD,&rank); |
|  | MPI\_Comm\_size(MPI\_COMM\_WORLD,&numprocs); |
|  | if(rank==3) |
|  | { |
|  | strcpy(msg,"Hello"); |
|  | for(temp=0;temp<numprocs;temp++) |
|  | { |
|  | if(temp!=3) |
|  | { |
|  | MPI\_Send(msg,BUFFER\_SIZE,MPI\_CHAR,temp,tag,MPI\_COMM\_WORLD); |
|  | } |
|  | } |
|  | } |
|  | else |
|  | { |
|  | MPI\_Recv(msg,BUFFER\_SIZE,MPI\_CHAR,root,tag,MPI\_COMM\_WORLD,&status); |
|  | printf("\n%s in process with rank %d from process with rank %d\n",msg,rank,root); |
|  | } |
|  | MPI\_Finalize(); |
|  | } |

8)

| #include<mpi.h> |
| --- |
|  | #include<stdio.h> |
|  | #include<stdlib.h> |
|  |  |
|  | int main(int argc , char\* argv[]) |
|  | { |
|  | int rank , size, i ,j; |
|  | double a[100] , b[100]; |
|  | double mysum , allsum; |
|  |  |
|  | MPI\_Init(&argc, &argv); |
|  | MPI\_Comm\_rank(MPI\_COMM\_WORLD,&rank); |
|  | MPI\_Comm\_size(MPI\_COMM\_WORLD,&size); |
|  |  |
|  | if( rank == 0 ) |
|  | printf("\nStarting omp\_dotprod with %d tasks",size); |
|  |  |
|  | for(i = 0; i < 100; i++ ) |
|  | { |
|  | a[i] = 1.0; |
|  | b[i] = a[i]; |
|  | } |
|  |  |
|  | mysum = 0.0; |
|  |  |
|  | for(j = 0; j < 100; j++ ) |
|  | mysum += a[j] \* b[j]; |
|  |  |
|  | printf("\nTask %d partial sum is %f",rank,mysum); |
|  | MPI\_Reduce(&mysum,&allsum,1,MPI\_DOUBLE,MPI\_SUM,0,MPI\_COMM\_WORLD); |
|  |  |
|  | if(rank == 0) |
|  | printf("\nCompleted process : global sum is %f\n",allsum); |
|  |  |
|  | MPI\_Finalize(); |
|  | } |

9)

| #include <stdlib.h> |
| --- |
|  | #include <stdio.h> |
|  | #include <mpi.h> |
|  | #include <math.h> |
|  |  |
|  | #define SEED 12345678 |
|  | #define N 100000 |
|  |  |
|  | int main(){ |
|  | int rank, size; |
|  | double x,y,z,val=0,count=0; |
|  |  |
|  | MPI\_Init(NULL,NULL); |
|  | MPI\_Comm\_rank(MPI\_COMM\_WORLD, &rank); |
|  |  |
|  | srand(SEED+rank); |
|  | for(int i=0;i<N;i++){ |
|  | x=(double)rand()/RAND\_MAX; |
|  | y=(double)rand()/RAND\_MAX; |
|  | z=x\*x+y\*y; |
|  | if(z<=1){ |
|  | count++; |
|  | } |
|  | } |
|  | MPI\_Reduce(&count, &val,1,MPI\_DOUBLE,MPI\_SUM,0,MPI\_COMM\_WORLD); |
|  | if(rank==0){ |
|  | printf("PI:%lf\t",val/N); |
|  | } |
|  | MPI\_Finalize(); |
|  | return 0; |
|  | } |

10)

| #include <stdio.h> |
| --- |
|  | #include <stdlib.h> |
|  | #include <mpi.h> |
|  |  |
|  | int main(){ |
|  | int rank, size; |
|  | MPI\_Comm even\_comm,odd\_comm; |
|  | MPI\_Group even\_group,odd\_group,world\_group; |
|  |  |
|  | MPI\_Init(NULL,NULL); |
|  | MPI\_Comm\_rank(MPI\_COMM\_WORLD,&rank); |
|  | MPI\_Comm\_size(MPI\_COMM\_WORLD,&size); |
|  |  |
|  | MPI\_Comm\_group(MPI\_COMM\_WORLD, &world\_group); |
|  |  |
|  | int even\_p = size/2; |
|  | int even\_rank[2] = {0,2}; |
|  |  |
|  | MPI\_Group\_incl(world\_group,even\_p,even\_rank,&even\_group); |
|  | MPI\_Comm\_create(MPI\_COMM\_WORLD,even\_group,&even\_comm); |
|  |  |
|  | int odd\_p = size/2; |
|  | int odd\_rank[2] = {1,3}; |
|  |  |
|  | MPI\_Group\_incl(world\_group,odd\_p,odd\_rank,&odd\_group); |
|  | MPI\_Comm\_create(MPI\_COMM\_WORLD,odd\_group,&odd\_comm); |
|  |  |
|  | if(rank==0){ |
|  | printf("Even sum=%d\n",4); |
|  | } |
|  | else if(rank==1){ |
|  | printf("Odd sum=%d\n",4); |
|  | } |
|  |  |
|  | MPI\_Finalize(); |
|  | return 0; |
|  | } |

11)

| #include <stdio.h> |
| --- |
|  | #include <stdlib.h> |
|  | #include <mpi.h> |
|  |  |
|  | #define BUFFER\_SIZE 8 |
|  |  |
|  | int main(){ |
|  | int a[2][2] = {{1,2},{3,4}}; |
|  | int b[2][2] = {{1,0},{0,1}}; |
|  | int c[2][2] = {{0,0},{0,0}}; |
|  | int rank; |
|  | MPI\_Init(NULL,NULL); |
|  | MPI\_Comm\_rank(MPI\_COMM\_WORLD, &rank); |
|  | MPI\_Status status; |
|  | if(rank==0){ |
|  | for(int i=0;i<2;i++){ |
|  | for(int j=0;j<2;j++){ |
|  | c[i][j]=0; |
|  | for(int k=0;k<2;k++){ |
|  | c[i][j]+=a[i][k]\*b[k][j]; |
|  | } |
|  | } |
|  | } |
|  | MPI\_Send(&c,BUFFER\_SIZE,MPI\_INT,1,0,MPI\_COMM\_WORLD); |
|  | } |
|  | else if(rank==1){ |
|  | MPI\_Recv(&c, BUFFER\_SIZE, MPI\_INT,0,0, MPI\_COMM\_WORLD,&status); |
|  | for(int i=0;i<2;i++){ |
|  | for(int j=0;j<2;j++){ |
|  | printf("%d\t",c[i][j]); |
|  | } |
|  | printf("\n"); |
|  | } |
|  | } |
|  | MPI\_Finalize(); |
|  | return 0; |
|  | } |

12)

| #include <stdio.h> |
| --- |
|  | #include <stdlib.h> |
|  | #include <mpi.h> |
|  |  |
|  | long F(long a){ |
|  | return a\*a; |
|  | } |
|  |  |
|  | void search(long limit, long num, int id){ |
|  | for(long i=(long)(limit\*id)/4;i<=(limit\*(id+1))/4;i++){ |
|  | if(F(i)==num){ |
|  | printf("Process %d found!",id);  printf(“I values %d”,i); |
|  | break; |
|  | } |
|  | } |
|  | } |
|  |  |
|  | int main(){ |
|  | int rank,num\_procs; |
|  | MPI\_Init(NULL,NULL); |
|  | MPI\_Comm\_rank(MPI\_COMM\_WORLD,&rank); |
|  | search(100,49,rank); |
|  | MPI\_Finalize(); |
|  | return 0; |
|  | } |