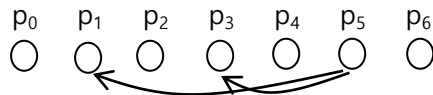


## Lab8 – Communicator and Grid

### ex1. Create new communicator group from old communicator.

Complete the following MPI program(odd.c) to create a new communicator with odd numbered processes and to broadcast the data from the biggest numbered process. The next figure shows an example of running 7 processes.



홀수 값만 뽑아내는 코드.

|  |  |
|--|--|
| <pre>#include &lt;stdio.h&gt; #include &lt;stdlib.h&gt; #include "mpi.h"  int main(int argc, char* argv[]) {     int pid, np, tag = 0, i;     MPI_Status status;      MPI_Group group_world, group_odd;     MPI_Comm comm_odd;     int *newrank;     int data;      MPI_Init(&amp;argc, &amp;argv);     MPI_Comm_rank(MPI_COMM_WORLD, &amp;pid);     MPI_Comm_size(MPI_COMM_WORLD, &amp;np);      data = 10 + pid;      newrank = (int*)(malloc(np/2*sizeof(int)));     for (i=0; i&lt;np/2; i++)         newrank[i] = ...;      MPI_Comm_group(..., &amp;group_world);     MPI_Group_incl(..., &amp;group_odd);     MPI_Comm_create(..., &amp;comm_odd);      if (...) {         MPI_Bcast(..., comm_odd);         printf("%d %d\n", pid, data);     }      free(newrank);     MPI_Finalize();     exit(0); }</pre> |  |
|--|--|

### ex2. Split a communicator

Complete the following MPI program(odd\_split.c) to split a communicator into even numbered processes and odd numbered processes and to broadcast from the biggest numbered process. The execution result is the same as the above exercise (ex1).

위와 같은 개념을 split 방식으로...

|  |  |
|--|--|
| <pre>#include &lt;stdio.h&gt; #include &lt;stdlib.h&gt; #include "mpi.h"  int main(int argc, char* argv[]) {     int pid, np, flag, tag = 0, i;     MPI_Status status;      MPI_Comm comm_odd;     int data;      MPI_Init(&amp;argc, &amp;argv);     MPI_Comm_rank(MPI_COMM_WORLD, &amp;pid);     MPI_Comm_size(MPI_COMM_WORLD, &amp;np);      data = pid;      flag = ...;     MPI_Comm_split(..., &amp;comm_odd);      if (...) {         MPI_Bcast(..., comm_odd);         printf("%d %d\n", pid, data);     }      MPI_Finalize();     exit(0); }</pre> |  |
|--|--|

### ex3. Grid topology(1)

Complete the following MPI program(grid.c) to shift the row or the column direction provided by command-line arguments.

Run the program "mpirun -n #proc prog direction shift" and find the out results. Use #proc = a number of square of 2, direction = 0 or 1, shift = 1 or -1.

값을 가지고 shift 해보는 코드.

|   |   |
|---|---|
| <pre>#include &lt;stdio.h&gt; #include &lt;stdlib.h&gt; #include &lt;math.h&gt; #include "mpi.h"  int main(int argc, char* argv[]) {     int np, pid, inp, jnp, i, j, k, pid_from, *buf;     ;     int dim_sizes[2], wrap_around[2];     int direct, shift, source, dest, reorder, tag=0;     MPI_Comm grid_comm;     MPI_Status status;      if (argc != 3) {         printf("usage %s direct shift\n", argv[0]);         exit(1);     }     direct = atoi(argv[1]); shift = atoi(argv[2]);      MPI_Init(&amp;argc, &amp;argv);      MPI_Comm_rank(MPI_COMM_WORLD, &amp;pid);     MPI_Comm_size(MPI_COMM_WORLD, &amp;np);      inp = sqrt(np);     // check np is a square number     if (np != inp*inp) {         if (pid == 0) fprintf(stderr, "#proc is not a square number of 2\n");         MPI_Finalize();         exit(0);     }</pre> | <pre>jnp = np/inp; dim_sizes[0] = inp; dim_sizes[1] = jnp; wrap_around[0] = wrap_around[1] = 1;  MPI_Cart_create(..., &amp;grid_comm); MPI_Cart_shift(grid_comm, direct, shift, &amp;source, &amp;dest);  MPI_Sendrecv(&amp;pid, ..., &amp;pid_from, ...);  // print the shifted matrix if (pid == 0 )     buf = (int *)malloc(sizeof(int)*np);  MPI_Gather(&amp;pid_from, 1, MPI_INT, buf, 1, MPI_INT, 0, MPI_COMM_WORLD);  if (pid == 0 ) {     k = 0;     for (i = 0; i &lt; inp; i++) {         for (j = 0; j &lt; jnp; j++)             printf("%2d ", buf[k++]);         printf("\n");     }     free(buf); }  MPI_Finalize(); exit(0);</pre> |
|---|---|

#### ex4. Grid topology(2)

Complete the following MPI program(diag.c) for the leftmost bottom process to broadcast data. Follow figure illustrates a case of using 16 processes.



|   |   |
|---|---|
| <pre> #include &lt;stdio.h&gt; #include &lt;stdlib.h&gt; #include &lt;string.h&gt; #include &lt;math.h&gt; #include "mpi.h"  int main(int argc, char* argv[]) {     int np, pid, inp, jnp, i, j, k, *buf;     int dim_sizes[2], wrap_around[2];     int direct, shift, source, dest, reorder, tag=0;     MPI_Comm grid_comm;     MPI_Status status;      int coordinates[2]; /* i and j location in the grid */      MPI_Group diag_world;     MPI_Group diag_group;     MPI_Comm diag_comm;     int *diag_rank;     int data;      MPI_Init(&amp;argc, &amp;argv);      MPI_Comm_rank(MPI_COMM_WORLD, &amp;pid);     MPI_Comm_size(MPI_COMM_WORLD, &amp;np);      inp = sqrt(np);     // check np is a square number     if (np != inp*inp) {         if (pid == 0) fprintf(stderr, "#proc is not a square number of 2\n");         MPI_Finalize();         exit(0);     }     jnp = np/inp;     dim_sizes[0] = inp;     dim_sizes[1] = jnp;     wrap_around[0] = wrap_around[1] = 1; </pre> | <pre> MPI_Cart_create(..., &amp;grid_comm); MPI_Cart_coords(..., coordinates);  // new communicator and broadcast diag_rank = (int*)malloc(inp*sizeof(int)); for (i = 0; i &lt; jnp; i++)     diag_rank[i] = ...;  MPI_Comm_group(..., &amp;diag_world); MPI_Group_incl(..., &amp;diag_group); MPI_Comm_create(..., &amp;diag_comm);  data = pid; if (...) coordinate [0], [1]을 더한 값이 3이 되면 된다.     MPI_Bcast(&amp;data, 1, MPI_INT, jnp-1, diag_comm);  // print the shifted matrix if (pid == 0 )     buf = (int *)malloc(sizeof(int)*np);  MPI_Gather(&amp;pid_from, 1, MPI_INT, buf, 1, MPI_INT, 0, MPI_COMM_WORLD);  if (pid == 0 ) {     k = 0;     for (i = 0; i &lt; inp; i++) {         for (j = 0; j &lt; jnp; j++)             printf("%2d ", buf[k++]);         printf("\n");     }     free(buf); }  MPI_Finalize(); exit(0); } </pre> |
|---|---|

Run the program using #processes as a number of square of 2 (4, 9, 16,...).

Submit diag.c when you are done.