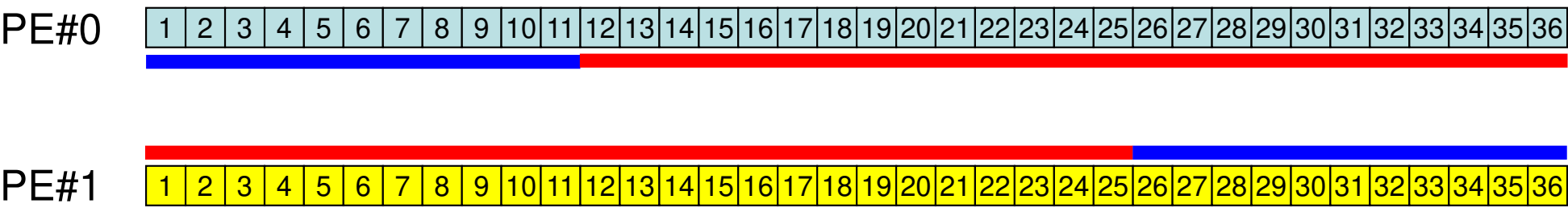


利用例(2): 配列の送受信(1/4)

- PE#0, PE#1間 で8バイト実数配列VECの値を交換する。
- PE#0⇒PE#1
 - PE#0: VEC(1)～VEC(11)の値を送る(長さ:11)
 - PE#1: VEV(26)～VEC(36)の値として受け取る
- PE#1⇒PE#0
 - PE#1: VEC(1)～VEC(25)の値を送る(長さ:25)
 - PE#0: VEV(12)～VEC(36)の値として受け取る



演習

- VEC(:)の初期状態を以下のようにする:
 - PE#0 VEC(1-36)= 101,102,103,~,135,136
 - PE#1 VEC(1-36)= 201,202,203,~,235,236
- 以下を使用したプログラムを作成せよ
 - MPI_Isend/Irecv/Waitall
- 正解は以下にある
- これらを </work/gt00/t00XXX/pFEM/mpi/S2> にコピーせよ

</work/gt00/z30088/pFEM/answer/t1>

配列の送受信:注意

#PE0

send:

VEC(start_send) ~
VEC(start_send+length_send-1)

#PE1

send:

VEC(start_send) ~
VEC(start_send+length_send-1)

#PE0

recv:

VEC(start_recv) ~
VEC(start_recv+length_recv-1)

#PE1

recv:

VEC(start_recv) ~
VEC(start_recv+length_recv-1)

- 送信側の「length_send」と受信側の「length_recv」は一致している必要がある。
 - PE#0⇒PE#1, PE#1⇒PE#0
- 「送信バッファ」と「受信バッファ」は別のアドレス

解答例: C (1/3)

C

Isend/Irecv/Waitall

```
$> cd /work/gt00/t00XXX/pFEM/mpi/S2
$> module load fj
$> mpifccpx -Nclang -Kfast ex2a.c
$> 実行(2プロセス) pjsub go2.sh
```

```
#include <stdio.h>
#include <stdlib.h>
#include "mpi.h"

int main(int argc, char **argv){
    int i, neib;
    int MyRank, PeTot;
    double VEC[36];
    int Start_Send, Length_Send;
    int Start_Recv, Length_Recv;

    MPI_Status *StatSend, *StatRecv;
    MPI_Request *RequestSend, *RequestRecv;

    MPI_Init(&argc, &argv);
    MPI_Comm_size(MPI_COMM_WORLD, &PeTot);
    MPI_Comm_rank(MPI_COMM_WORLD, &MyRank);
```

解答例: C (2/3)

C

Isend/Irecv/Waitall

```
Start_Send= 1;

if(MyRank == 0) {
    neib= 1;
    Length_Send= 11;
    Length_Recv= 25;
    for (i=0;i<36;i++){VEC[i]=100.0+i+1;}}

if(MyRank == 1) {
    neib= 0;
    Length_Send= 25;
    Length_Recv= 11;
    for (i=0;i<36;i++){VEC[i]=200.0+i+1;}}

Start_Recv= 1 + Length_Send;

StatSend = malloc(sizeof(MPI_Status) * 1);
StatRecv = malloc(sizeof(MPI_Status) * 1);
RequestSend = malloc(sizeof(MPI_Request) * 1);
RequestRecv = malloc(sizeof(MPI_Request) * 1);

for (i=0;i<36;i++) {
    printf("%s%2d%5d%8.0f¥n", "### before", MyRank, i, VEC[i]);}
```

解答例: C (3/3)

C

Isend/Irecv/Waitall

```
MPI_Isend(&VEC[Start_Send-1], Length_Send, MPI_DOUBLE, neib, 0,
          MPI_COMM_WORLD, &RequestSend[0]);

MPI_Irecv(&VEC[Start_Recv-1], Length_Recv, MPI_DOUBLE, neib, 0,
          MPI_COMM_WORLD, &RequestRecv[0]);

MPI_Waitall(1, RequestRecv, StatRecv);

MPI_Waitall(1, RequestSend, StatSend);

for (i=0;i<36;i++) {
    printf("%s%2d%5d%8.0f¥n", "### after ", MyRank, i, VEC[i]);}

MPI_Finalize();
return 0;
}
```

- 汎用性がある
- 「データが全て」という気がして来ないだろうか？

結果

0 #BEFORE# 1	101.
0 #BEFORE# 2	102.
0 #BEFORE# 3	103.
0 #BEFORE# 4	104.
0 #BEFORE# 5	105.
0 #BEFORE# 6	106.
0 #BEFORE# 7	107.
0 #BEFORE# 8	108.
0 #BEFORE# 9	109.
0 #BEFORE# 10	110.
0 #BEFORE# 11	111.
0 #BEFORE# 12	112.
0 #BEFORE# 13	113.
0 #BEFORE# 14	114.
0 #BEFORE# 15	115.
0 #BEFORE# 16	116.
0 #BEFORE# 17	117.
0 #BEFORE# 18	118.
0 #BEFORE# 19	119.
0 #BEFORE# 20	120.
0 #BEFORE# 21	121.
0 #BEFORE# 22	122.
0 #BEFORE# 23	123.
0 #BEFORE# 24	124.
0 #BEFORE# 25	125.
0 #BEFORE# 26	126.
0 #BEFORE# 27	127.
0 #BEFORE# 28	128.
0 #BEFORE# 29	129.
0 #BEFORE# 30	130.
0 #BEFORE# 31	131.
0 #BEFORE# 32	132.
0 #BEFORE# 33	133.
0 #BEFORE# 34	134.
0 #BEFORE# 35	135.
0 #BEFORE# 36	136.

0 #AFTER # 1	101.
0 #AFTER # 2	102.
0 #AFTER # 3	103.
0 #AFTER # 4	104.
0 #AFTER # 5	105.
0 #AFTER # 6	106.
0 #AFTER # 7	107.
0 #AFTER # 8	108.
0 #AFTER # 9	109.
0 #AFTER # 10	110.
0 #AFTER # 11	111.
0 #AFTER # 12	201.
0 #AFTER # 13	202.
0 #AFTER # 14	203.
0 #AFTER # 15	204.
0 #AFTER # 16	205.
0 #AFTER # 17	206.
0 #AFTER # 18	207.
0 #AFTER # 19	208.
0 #AFTER # 20	209.
0 #AFTER # 21	210.
0 #AFTER # 22	211.
0 #AFTER # 23	212.
0 #AFTER # 24	213.
0 #AFTER # 25	214.
0 #AFTER # 26	215.
0 #AFTER # 27	216.
0 #AFTER # 28	217.
0 #AFTER # 29	218.
0 #AFTER # 30	219.
0 #AFTER # 31	220.
0 #AFTER # 32	221.
0 #AFTER # 33	222.
0 #AFTER # 34	223.
0 #AFTER # 35	224.
0 #AFTER # 36	225.

1 #BEFORE# 1	201.
1 #BEFORE# 2	202.
1 #BEFORE# 3	203.
1 #BEFORE# 4	204.
1 #BEFORE# 5	205.
1 #BEFORE# 6	206.
1 #BEFORE# 7	207.
1 #BEFORE# 8	208.
1 #BEFORE# 9	209.
1 #BEFORE# 10	210.
1 #BEFORE# 11	211.
1 #BEFORE# 12	212.
1 #BEFORE# 13	213.
1 #BEFORE# 14	214.
1 #BEFORE# 15	215.
1 #BEFORE# 16	216.
1 #BEFORE# 17	217.
1 #BEFORE# 18	218.
1 #BEFORE# 19	219.
1 #BEFORE# 20	220.
1 #BEFORE# 21	221.
1 #BEFORE# 22	222.
1 #BEFORE# 23	223.
1 #BEFORE# 24	224.
1 #BEFORE# 25	225.
1 #BEFORE# 26	226.
1 #BEFORE# 27	227.
1 #BEFORE# 28	228.
1 #BEFORE# 29	229.
1 #BEFORE# 30	230.
1 #BEFORE# 31	231.
1 #BEFORE# 32	232.
1 #BEFORE# 33	233.
1 #BEFORE# 34	234.
1 #BEFORE# 35	235.
1 #BEFORE# 36	236.

1 #AFTER # 1	201.
1 #AFTER # 2	202.
1 #AFTER # 3	203.
1 #AFTER # 4	204.
1 #AFTER # 5	205.
1 #AFTER # 6	206.
1 #AFTER # 7	207.
1 #AFTER # 8	208.
1 #AFTER # 9	209.
1 #AFTER # 10	210.
1 #AFTER # 11	211.
1 #AFTER # 12	212.
1 #AFTER # 13	213.
1 #AFTER # 14	214.
1 #AFTER # 15	215.
1 #AFTER # 16	216.
1 #AFTER # 17	217.
1 #AFTER # 18	218.
1 #AFTER # 19	219.
1 #AFTER # 20	220.
1 #AFTER # 21	221.
1 #AFTER # 22	222.
1 #AFTER # 23	223.
1 #AFTER # 24	224.
1 #AFTER # 25	225.
1 #AFTER # 26	101.
1 #AFTER # 27	102.
1 #AFTER # 28	103.
1 #AFTER # 29	104.
1 #AFTER # 30	105.
1 #AFTER # 31	106.
1 #AFTER # 32	107.
1 #AFTER # 33	108.
1 #AFTER # 34	109.
1 #AFTER # 35	110.
1 #AFTER # 36	111.