Exercises: Distributed Memory Programming with MPI

1

Greeting

Changing `strlen(greeting) + 1` to `strlen(greeting)` would exlude the terminating null character from the greeting string. But, as it turns out, when passing the non-null-terminated string to `MPI_Send`, MPI checks for the terminating null character and adds one if one is not already there. So, everything works as intended.

If `MAX_STRING` is passed instead, then the message might include bytes beyond the terminating null character. But, string functions like `printf` et cetera should work just fine because that includes the terminating null character.

`mpicc` also uses the same options as `gcc`, so conditional preprocessing can be done to produce three variants of the code by using appropriate `gcc` options.

```
1  #include <mpi.h>
2  #include <stdio.h>
3  #include <string.h>
4
5  const int MAX_STRING = 100;
6
7
8  int main(void) {
9          char greeting[MAX_STRING];
10         int comm_sz;
11         int my_rank;
12
13         MPI_Init(NULL, NULL);
14         MPI_Comm_size(MPI_COMM_WORLD, &comm_sz);
15         MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);
16
17         if (my_rank != 0) {
18                 sprintf(greeting, "Greetings from process %d of
                   ↪  %d!",\
19                                 my_rank, comm_sz);
20                 MPI_Send(
21                                 greeting,
22 #ifdef NORMAL
23                                 strlen(greeting)+1,
```

```
24  #endif
25  #ifdef NONULL
26                                  strlen(greeting),
27  #endif
28  #ifdef FULLBUFFER
29                                  MAX_STRING,
30  #endif
31                                  MPI_CHAR,
32                                  0, 0,
33                                  MPI_COMM_WORLD);
34
35          } else {
36                  printf("Greetings from process %d of %d!\n",
                    ↪  my_rank,\
37                                  comm_sz);
38                  for (int q = 1; q < comm_sz; q++) {
39                          MPI_Recv(greeting, MAX_STRING, MPI_CHAR,
                            ↪  q, 0,\
40                                  MPI_COMM_WORLD,
                                    ↪  MPI_STATUS_IGNORE);
41                          puts(greeting);
42                  }
43          }
44
45          MPI_Finalize();
46          return 0;
47  }
```

Listing 1: Greetings program with preprocessor directives
Use `mpicc -D NONULL -o nonull ex3_1.c` to get the code in listing ??.

```
8709  # 4 "ex3_1.c" 2
8710
8711  const int MAX_STRING = 100;
8712
8713
8714  int main(void) {
8715          char greeting[MAX_STRING];
8716          int comm_sz;
8717          int my_rank;
8718
8719          MPI_Init(NULL, NULL);
8720          MPI_Comm_size(MPI_COMM_WORLD, &comm_sz);
8721          MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);
8722
8723          if (my_rank != 0) {
8724                  sprintf(greeting, "Greetings from process %d of
                        ↪  %d!",\
8725                                  my_rank, comm_sz);
8726                  MPI_Send(
```

```
                                greeting,
                                strlen(greeting),
                                MPI_CHAR,
                                0, 0,
                                MPI_COMM_WORLD);
        } else {
                printf("Greetings from process %d of %d!\n",
                 ↪  my_rank,\
                                comm_sz);
                for (int q = 1; q < comm_sz; q++) {
                        MPI_Recv(greeting, MAX_STRING, MPI_CHAR,
                         ↪  q, 0,\
                                MPI_COMM_WORLD,
                                 ↪  MPI_STATUS_IGNORE);
                        puts(greeting);
                }
        }

        MPI_Finalize();
        return 0;
}
```

Listing 2: Without the terminating null character

Similarly, using `mpicc -D MAX_STRING -o maxstring ex3_1.c` gives.

```
# 4 "ex3_1.c" 2

const int MAX_STRING = 100;


int main(void) {
        char greeting[MAX_STRING];
        int comm_sz;
        int my_rank;

        MPI_Init(NULL, NULL);
        MPI_Comm_size(MPI_COMM_WORLD, &comm_sz);
        MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);

        if (my_rank !=0) {
```

```
8724                    sprintf(greeting, "Greetings from process %d of
                        ↪  %d!",\
8725                                    my_rank, comm_sz);
8726                    MPI_Send(
8727                                    greeting,
8728
8729
8730
8731
8732
8733
8734
8735                                    MAX_STRING,
8736
8737                                    MPI_CHAR,
8738                                    0, 0,
8739                                    MPI_COMM_WORLD);
8740
8741            } else {
8742                    printf("Greetings from process %d of %d!",
                        ↪  my_rank,\
8743                                    comm_sz);
8744                    for (int q = 1; q < comm_sz; q++) {
8745                            MPI_Recv(greeting, MAX_STRING, MPI_CHAR,
                            ↪  q, 0,\
8746                                    MPI_COMM_WORLD,
                                    ↪  MPI_STATUS_IGNORE);
8747
8748                    }
8749            }
8750
8751            MPI_Finalize();
8752            return 0;
8753    }
```

Trapezoid Rule