#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <arpa/inet.h>

#include <sys/socket.h>

#include <unistd.h>

#define PORT 8080

#define BUF\_SIZE 1024

#define MAX\_ROWS 100

#define MAX\_COLS 100

int main() {

int sockfd;

struct sockaddr\_in server\_addr, client\_addr;

char buffer[BUF\_SIZE];

int matrix[MAX\_ROWS][MAX\_COLS];

socklen\_t addr\_len = sizeof(client\_addr);

int num\_rows = 0, num\_cols = 0;

// Create socket

if ((sockfd = socket(AF\_INET, SOCK\_DGRAM, 0)) < 0) {

perror("Socket creation failed");

exit(EXIT\_FAILURE);

}

memset(&server\_addr, 0, sizeof(server\_addr));

memset(&client\_addr, 0, sizeof(client\_addr));

// Server address setup

server\_addr.sin\_family = AF\_INET;

server\_addr.sin\_addr.s\_addr = INADDR\_ANY;

server\_addr.sin\_port = htons(PORT);

// Bind socket to the address and port

if (bind(sockfd, (const struct sockaddr \*)&server\_addr, sizeof(server\_addr)) < 0) {

perror("Bind failed");

close(sockfd);

exit(EXIT\_FAILURE);

}

printf("Server is running and waiting for matrix rows...\n");

// Receive the number of rows and columns

recvfrom(sockfd, &num\_rows, sizeof(int), 0, (struct sockaddr \*)&client\_addr, &addr\_len);

recvfrom(sockfd, &num\_cols, sizeof(int), 0, (struct sockaddr \*)&client\_addr, &addr\_len);

printf("Expecting a matrix of size %d x %d\n", num\_rows, num\_cols);

// Receive each row of the matrix

for (int i = 0; i < num\_rows; i++) {

recvfrom(sockfd, buffer, BUF\_SIZE, 0, (struct sockaddr \*)&client\_addr, &addr\_len);

sscanf(buffer, "%d %d %d %d", &matrix[i][0], &matrix[i][1], &matrix[i][2], &matrix[i][3]); // Assuming max 4 columns for simplicity

printf("Received row %d: %s\n", i + 1, buffer);

}

// Display the received matrix

printf("Received matrix:\n");

for (int i = 0; i < num\_rows; i++) {

for (int j = 0; j < num\_cols; j++) {

printf("%d ", matrix[i][j]);

}

printf("\n");

}

close(sockfd);

return 0;

}

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <arpa/inet.h>

#include <sys/socket.h>

#include <unistd.h>

#define PORT 8080

#define BUF\_SIZE 1024

int main() {

int sockfd;

struct sockaddr\_in server\_addr;

char buffer[BUF\_SIZE];

int matrix[4][4] = {

{1, 2, 3, 4},

{5, 6, 7, 8},

{9, 10, 11, 12},

{13, 14, 15, 16}

};

int num\_rows = 4, num\_cols = 4;

// Create socket

if ((sockfd = socket(AF\_INET, SOCK\_DGRAM, 0)) < 0) {

perror("Socket creation failed");

exit(EXIT\_FAILURE);

}

memset(&server\_addr, 0, sizeof(server\_addr));

// Server address setup

server\_addr.sin\_family = AF\_INET;

server\_addr.sin\_port = htons(PORT);

server\_addr.sin\_addr.s\_addr = INADDR\_ANY;

// Send the number of rows and columns

sendto(sockfd, &num\_rows, sizeof(int), 0, (const struct sockaddr \*)&server\_addr, sizeof(server\_addr));

sendto(sockfd, &num\_cols, sizeof(int), 0, (const struct sockaddr \*)&server\_addr, sizeof(server\_addr));

// Send each row of the matrix

for (int i = 0; i < num\_rows; i++) {

snprintf(buffer, BUF\_SIZE, "%d %d %d %d", matrix[i][0], matrix[i][1], matrix[i][2], matrix[i][3]);

sendto(sockfd, buffer, strlen(buffer), 0, (const struct sockaddr \*)&server\_addr, sizeof(server\_addr));

printf("Sent row %d: %s\n", i + 1, buffer);

}

close(sockfd);

return 0;

}