

Programme Name: BCS

Course Code:CSC2624

Course Name:Distributed and Parallel Computing

Individual Project Work

Date of Submission:26th August 2022

**Submitted By: Submitted To:**

Student Name:Udit Kumar Mahato Faculty Name:Sumanta Silwal

IUKL ID: 0042003900006 Department: PO office(BCS)

Semester: 4th Semester

Intake:September 2020

1. The following C program sums up all the values in array "data" and displays the sum total.

Code:-

#include <stdio.h>

#include <mpi.h>

#define NUMDATA 10000

int data[NUMDATA];

void LoadData(int data[])

{

for (int i = 0; i < NUMDATA; i++)

{

data[i] = 1;

}

}

int AddUp(int data[], int count)

{

int sum = 0;

for (int i = 0; i < count; i++)

{

sum += data[i];

}

return sum;

}

int main(void)

{

int sum;

int size;

int rank;

int tag = 0;

int chunksize;

int start;

int result;

chunksize = NUMDATA / size;

MPI\_Init(NULL, NULL);

MPI\_Comm\_size(MPI\_COMM\_WORLD, &size);

MPI\_Comm\_rank(MPI\_COMM\_WORLD, &rank);

chunksize = NUMDATA / size;

if (rank == 0)

{

LoadData(data);

for (int i = 1; i < size; i++)

{

start = i \* chunksize;

MPI\_Send(&(data[start]), chunksize, MPI\_INT, i, tag, MPI\_COMM\_WORLD);

}

sum = AddUp(data, chunksize);

for (int i = 1; i < size; i++)

{

MPI\_Recv(&result, 1, MPI\_INT, MPI\_ANY\_SOURCE, tag,

MPI\_COMM\_WORLD, MPI\_STATUS\_IGNORE);

sum += result;

}

printf("The total sum of data is %d\n", sum);

}

else

{

MPI\_Recv(data, chunksize, MPI\_INT, 0, tag, MPI\_COMM\_WORLD,

MPI\_STATUS\_IGNORE);

sum = AddUp(data, chunksize);

MPI\_Send(&sum, 1, MPI\_INT, 0, tag, MPI\_COMM\_WORLD);

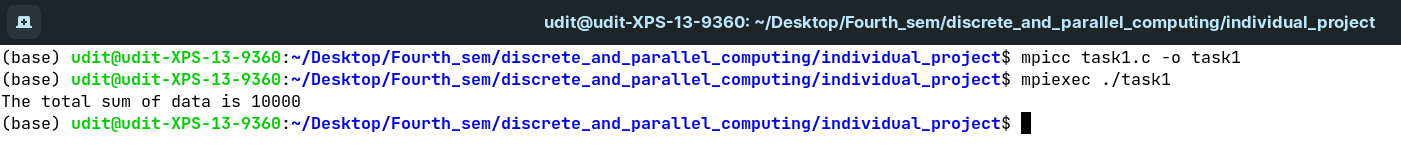
}

MPI\_Finalize();

return 0;

}

Output:-



2. The file "WarAndPeace.txt" attached herewith contains the entire text of the book "War and Peace" by Leo Tolstoy. Write an MPI program to count the number of times each letter of the alphabet occurs in the book. Count both the upper case and the lowercase as the same. Ignore any letter with accents such as " é " and so on.

Your MPI program should work with any number of processes from 1 to 100 process. Only Process rank 0 (zero) should read in the file and send the appropriate chunk of file to each other process. The other processes should not read in the file. You should submit this program as "individualtask1.c" as part of your individual/group Assignment.

Solution:-

Code:-

#include <stdio.h>

#include <stdlib.h>

#include <mpi.h>

int main(int argc, char \*\*argv[])

{

// counting letters

int templettercount[26] = {0};

int tag = 0;

int ch;

// settting up MPI program

int size;

int rank;

MPI\_Init(&argc, argv);

MPI\_Comm\_size(MPI\_COMM\_WORLD, &size);

MPI\_Comm\_rank(MPI\_COMM\_WORLD, &rank);

if (rank == 0)

{

char \*filename = "WarAndPeace.txt"; // this is the file name that we are using

char \*chararray;

FILE \*charfile = fopen(filename, "r"); //lets open the file

long filelength = 0;

int lettercount[26] = {0};

int i = 0;

int chunksize = 0;

int start = 0;

int rem = 0;

int charcount = 0;

// checking the file if not found then returning error

if (charfile == NULL)

{

printf("Error: File not found");

exit(0);

}

// finding the number of letters in the file

while ((ch = fgetc(charfile)) != EOF)

{

if ((ch >= 'a' && ch <= 'z') || (ch >= 'A' && ch <= 'Z'))

{

filelength++;

}

}

rewind(charfile);

printf("File has %ld letters\n", filelength); // displaying the length

chararray = malloc(sizeof(char) \* filelength); // allocating memory for the array

// reading the file and storing it in the array

while ((ch = fgetc(charfile)) != EOF)

{

if ((ch >= 'a' && ch <= 'z') || (ch >= 'A' && ch <= 'Z'))

{

chararray[i] = ch;

i++;

}

}

// sending the array to the other processes

chunksize = filelength / size;

rem = filelength % size;

charcount = filelength / size;

printf("keeping %d leters in rank %d\n", charcount, rank);

for (int receiver = 1; receiver < size; receiver++)

{

start = charcount \* receiver + 1;

if (receiver == size - 1)

{

charcount += rem;

}

printf("sending %d letters to rank %d\n", charcount, receiver);

MPI\_Send(&chararray[start], charcount, MPI\_CHAR, receiver, tag, MPI\_COMM\_WORLD);

}

// calculating results with rank 0

for (int letter = 0; letter <= chunksize; ++letter)

{

ch = chararray[letter];

if (ch >= 65 && ch <= 90)

{

lettercount[(ch - 65)]++;

}

else if (ch >= 97 && ch <= 122)

{

lettercount[(ch - 97)]++;

}

}

// receiving results from other processes

for (int sender = 1; sender < size; sender++)

{

MPI\_Recv(templettercount, 26, MPI\_INT, sender, tag, MPI\_COMM\_WORLD, MPI\_STATUS\_IGNORE);

int index = 0;

for (int count = 0; count < 26; count++)

{

lettercount[count] += templettercount[index];

index++;

}

}

printf("The letter count is:\n");

for (int count = 0; count < 26; count++)

{

printf("The Letter %c occurs %d times\n", count + 'a', lettercount[count]);

}

// freeing up memory

free(chararray);

}

else

{

MPI\_Status status;

int charcount;

char \*chararray;

MPI\_Probe(0, tag, MPI\_COMM\_WORLD, &status);

MPI\_Get\_count(&status, MPI\_CHAR, &charcount);

// allocating memory for the array

chararray = malloc(charcount \* sizeof(char));

// receiving the message

MPI\_Recv(chararray, charcount, MPI\_CHAR, 0, tag, MPI\_COMM\_WORLD, MPI\_STATUS\_IGNORE);

// calculating results

for (int letter = 0; letter <= charcount; ++letter)

{

ch = chararray[letter];

if (ch >= 65 && ch <= 90)

{

templettercount[ch - 65]++;

}

else if (ch >= 97 && ch <= 122)

{

templettercount[ch - 97]++;

}

}

// sending back the results

MPI\_Send((&templettercount[0]), 26, MPI\_INT, 0, tag, MPI\_COMM\_WORLD);

// freeingup memory

free(chararray);

}

MPI\_Finalize();

return 0;

}

Output:-

