**OPERATING SYSTEM LAB MID SEMESTER EXAMINATION**

**T KARTHIKEYAN**

**CED18I064**

**01 NOV 2020**

**QUESTION**

**Develop a C program that generates the histogram (frequency count**

**of unique characters in the input text) using multiple processes (thread**

**numbers to match with unique character count) , each thread performing the**

**counting for that respective character. Compare the efficiency of the**

**multiprocessing version over its equivalent serial version.**

**LOGIC/PSEUDOCODE**

**Multiprocess version/Multithreaded version**

**Given the string str , string length slen**

**NUM\_THREAD/NUM\_PROCESS = number of unique characters in the string**

**Every new character will be given a unique id 0 … NUM\_THREAD/NUM\_PROCESS-1**

**Integer array brr = mapping from ascii number to unique id**

**Character array trr = mapping from unique id to ascii number**

**Integer array frr = stores the frequency of character based on unique id allotted**

**Number of threads/child processes created will be NUM\_THREAD/NUM\_PROCESS**

**Each thread/child process is given the responsibility to calculate frequency of respective character and store it in frr**

**Print the frr**

**CODE**

**Multiprocess version**

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <unistd.h>**

**#include <string.h>**

**#include <sys/types.h>**

**#include <sys/wait.h>**

**#include <math.h>**

**#include <time.h>**

**int main()**

**{**

**char \*str = malloc(sizeof(char)\*(1000));**

**int slen;**

**printf("Enter the string :");**

**fgets(str, 1000, stdin);**

**slen = strlen(str);**

**slen--;**

**printf("String is :%s", str);**

**printf("String length is %d\n", slen);**

**\_Bool arr[127] = {0}; // to indicate whether particular character exists or not**

**int NUM\_PROCESS = 0;**

**for(int i = 0; i < slen; i++)**

**{**

**if(arr[str[i]] == 0)**

**{**

**arr[str[i]] = 1;**

**NUM\_PROCESS++;**

**}**

**}**

**printf("NUM\_PROCESS : %d\n", NUM\_PROCESS);**

**int brr[127] = {-1}; // store the newly assigned index**

**for(int i = 0; i < 127; i++)**

**brr[i] = -1;**

**char trr[NUM\_PROCESS]; // index to char mapping**

**int index = 0;**

**for(int i = 0; i < slen; i++)**

**{**

**if(brr[str[i]] == -1)**

**{**

**brr[str[i]] = index;**

**trr[index] = str[i];**

**index++;**

**}**

**}**

**int \*frr = malloc(sizeof(int)\*NUM\_PROCESS);**

**for(int i = 0; i < NUM\_PROCESS; i++)**

**frr[i] = 0;**

**// process**

**for(int i = 0; i < NUM\_PROCESS; i++)**

**{**

**if(vfork() == 0)**

**{**

**for(int j = 0; j < slen; j++)**

**{**

**if(str[j] == trr[i])**

**frr[brr[trr[i]]]++;**

**}**

**exit(0);**

**}**

**}**

**for(int i = 0; i < NUM\_PROCESS; i++)**

**wait(NULL);**

**for(int i = 0; i < NUM\_PROCESS; i++)**

**printf("%c : %d\n", trr[i], frr[i]);**

**return 0;**

**}**

**Multithreaded version**

**#include <pthread.h>**

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <unistd.h>**

**#include <string.h>**

**#include <sys/types.h>**

**#include <sys/wait.h>**

**#include <math.h>**

**#include <time.h>**

**struct block**

**{**

**char \*str;**

**int \*frr;**

**int slen;**

**char c;**

**int ci;**

**};**

**void \*runner(void \*param)**

**{**

**struct block \*b = param;**

**for(int i = 0; i < (b->slen); i++)**

**{**

**if(b->str[i] == (b->c))**

**b->frr[b->ci]++;**

**}**

**pthread\_exit(0);**

**}**

**int main()**

**{**

**char \*str = malloc(sizeof(char)\*(1000));**

**int slen;**

**printf("Enter the string :");**

**fgets(str, 1000, stdin);**

**slen = strlen(str);**

**slen--;**

**printf("String is :%s", str);**

**printf("String length is %d\n", slen);**

**\_Bool arr[127] = {0}; // to indicate whether particular character exists or not**

**int NUM\_THREAD = 0;**

**for(int i = 0; i < slen; i++)**

**{**

**if(arr[str[i]] == 0)**

**{**

**arr[str[i]] = 1;**

**NUM\_THREAD++;**

**}**

**}**

**printf("NUM\_THREAD : %d\n", NUM\_THREAD);**

**int brr[127] = {-1}; // store the newly assigned index**

**for(int i = 0; i < 127; i++)**

**brr[i] = -1;**

**char trr[NUM\_THREAD]; // index to char mapping**

**int index = 0;**

**for(int i = 0; i < slen; i++)**

**{**

**if(brr[str[i]] == -1)**

**{**

**brr[str[i]] = index;**

**trr[index] = str[i];**

**index++;**

**}**

**}**

**int \*frr = malloc(sizeof(int)\*NUM\_THREAD);**

**for(int i = 0; i < NUM\_THREAD; i++)**

**frr[i] = 0;**

**struct block b[NUM\_THREAD];**

**for(int i = 0; i < NUM\_THREAD; i++)**

**{**

**b[i].str = str;**

**b[i].slen = slen;**

**b[i].frr = frr;**

**b[i].c = trr[i];**

**b[i].ci = brr[trr[i]];**

**}**

**pthread\_t tid[NUM\_THREAD];**

**pthread\_attr\_t attr[NUM\_THREAD];**

**for(int i = 0; i < NUM\_THREAD; i++)**

**pthread\_attr\_init(&attr[i]);**

**for(int i = 0; i < NUM\_THREAD; i++)**

**pthread\_create(&tid[i], &attr[i], runner, &b[i]);**

**for(int i = 0; i < NUM\_THREAD; i++)**

**pthread\_join(tid[i], NULL);**

**printf("HISTOGRAM\n");**

**for(int i = 0; i < NUM\_THREAD; i++)**

**printf("%c : %d\n", trr[i], frr[i]);**

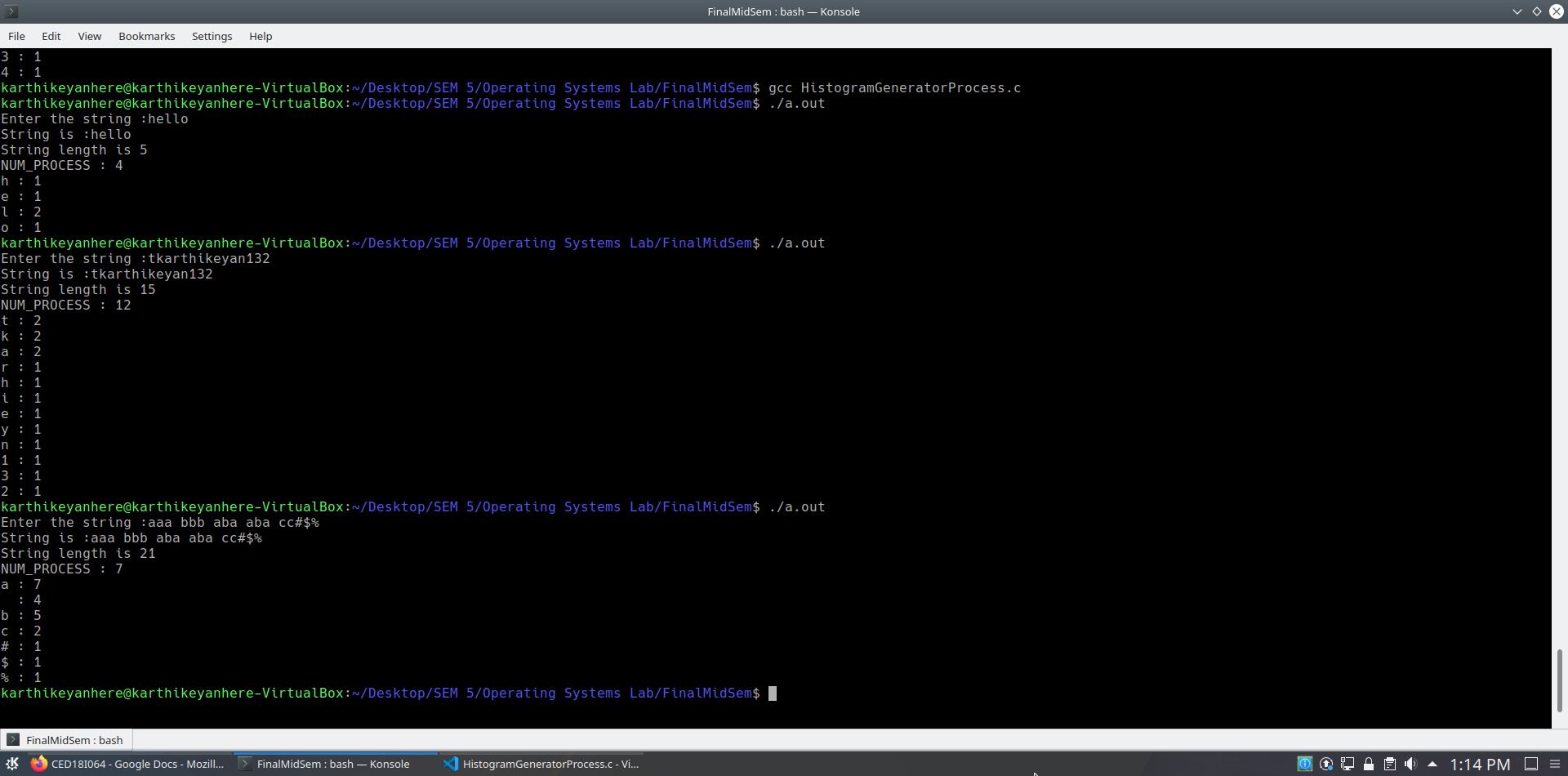
**free(str);**

**return 0;**

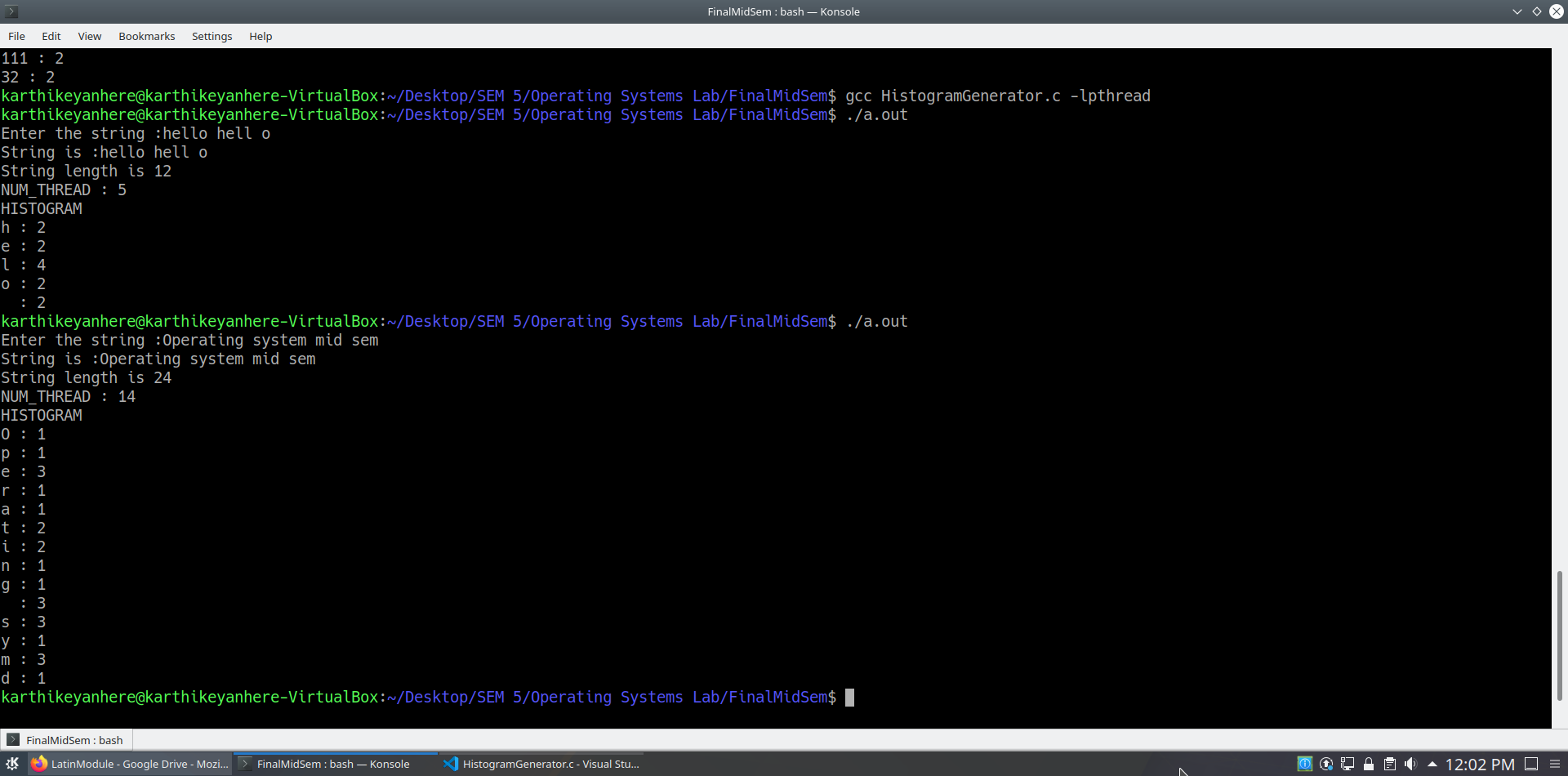
**}**

**OUTPUT**

**Multiprocess version**

****

**Multithreaded version**

****

**ANALYSIS/HOW PARALLEL VERSION IS BETTER**

**NOTE : n is length of the string, m is the number of processors**

**NOTE : Total time complexity = TC of finding unique characters + TC of frequency calculation**

|  |  |  |
| --- | --- | --- |
|  | **Serial version** | **Parallel version** |
| **Finding unique characters** | **O(n)** | **O(n)** |
| **Counting the frequency of one character** | **O(n)** | **O(n)** |
| **Counting the frequency of UNIQ characters**  **Where UNIQ characters possible is 126-31 = 95,**  **So UNIQ = O(1)** | **O(n\*UNIQ) = O(n)** | **O(n\*UNIQ) = O(n)** |
| **Number of processors** | **1** | **m** |
| **Time complexity of frequency calculation** | **O(n)** | **O(n/m)** |
| **Total time complexity** | **O(n) + O(n) = O(n)** | **O(n) + O(n/m) = O(n)** |