**Task5**

**Description**

The program recursively scans a chosen directory to find all the file sizes within the directory then the files are organized into bins based on a specified bin width and displays a histogram to visualize the file sizes within the directory.

**Code**

#include <stdio.h>  
#include <stdlib.h>  
#include <dirent.h>  
#include <sys/stat.h>  
#include <string.h>  
  
#define MAX\_BINS 1000  // Adjust as needed  
  
int histogram[MAX\_BINS] = {0};  
int bin\_width;  
  
void process\_file(const char \*filepath) {  
    struct stat file\_stat;  
    if (stat(filepath, &file\_stat) == 0 && S\_ISREG(file\_stat.st\_mode)) {  
        int bin = file\_stat.st\_size / bin\_width;  
        if (bin < MAX\_BINS) {  
            histogram[bin]++;  
        }  
    }  
}  
  
void traverse\_directory(const char \*dirpath) {  
    struct dirent \*entry;  
    DIR \*dir = opendir(dirpath);  
    if (!dir) {  
        perror("opendir failed");  
        return;  
    }  
  
    while ((entry = readdir(dir)) != NULL) {  
        if (strcmp(entry->d\_name, ".") == 0 || strcmp(entry->d\_name, "..") == 0)  
            continue;  
  
        char path[1024];  
        snprintf(path, sizeof(path), "%s/%s", dirpath, entry->d\_name);  
  
        struct stat entry\_stat;  
        if (stat(path, &entry\_stat) == 0) {  
            if (S\_ISDIR(entry\_stat.st\_mode)) {  
                traverse\_directory(path);  
            } else if (S\_ISREG(entry\_stat.st\_mode)) {  
                process\_file(path);  
            }  
        }  
    }  
    closedir(dir);  
}  
  
void print\_histogram() {  
    printf("File Size Histogram (bin width = %d bytes):\n", bin\_width);  
    for (int i = 0; i < MAX\_BINS; i++) {  
        if (histogram[i] > 0) {  
            printf("%d - %d: %d files\n", i \* bin\_width, (i + 1) \* bin\_width - 1, histogram[i]);  
        }  
    }  
}  
  
int main(int argc, char \*argv[]) {  
    if (argc != 3) {  
        fprintf(stderr, "Usage: %s <directory> <bin\_width>\n", argv[0]);  
        return 1;  
    }  
  
    bin\_width = atoi(argv[2]);  
    if (bin\_width <= 0) {  
        fprintf(stderr, "Invalid bin width\n");  
        return 1;  
    }  
  
    traverse\_directory(argv[1]);  
    print\_histogram();  
  
    return 0;  
}

**Gitlink**

<https://github.com/FirasAhmed2/Operating-systems-coursework.git>