# ASSIGNMENT 8
# 22/10/24

NAME : SHRESTH SONKAR
REGNO : 20214272
GROUP : CS7D
TOPIC : DISTRIBUTED SYSTEM
CODE : CS-17201

```
// Q1 : list of users who owns a file having maximum
size in the current working directory using map reduce

#include <stdio.h>
#include <stdlib.h>
#include <dirent.h>
#include <sys/stat.h>
#include <unistd.h>
#include <pwd.h>
#include <string.h>

#define MAX_FILES 1024

typedef struct {
    char filename[256];
    char owner[256];
    off_t size;
} FileInfo;

FileInfo fileInfoList[MAX_FILES];
int fileCount = 0;

void map(const char *dirname) {
    struct dirent *entry;
    struct stat fileStat;
    DIR *dir = opendir(dirname);

    if (dir == NULL) {
        perror("Unable to open directory");
        exit(EXIT_FAILURE);
    }

    while ((entry = readdir(dir)) != NULL) {
        char path[512];
        snprintf(path, sizeof(path), "%s/%s", dirname,
entry->d_name);

        if (stat(path, &fileStat) == 0 &&
S_ISREG(fileStat.st_mode)) {
            struct passwd *pw =
getpwuid(fileStat.st_uid);
            strncpy(fileInfoList[fileCount].filename,
entry->d_name, 256);
            strncpy(fileInfoList[fileCount].owner, pw-
>pw_name, 256);
```

```
            fileInfoList[fileCount].size =
fileStat.st_size;

            fileCount++;
            if (fileCount >= MAX_FILES) {
                fprintf(stderr, "Too many files,
increase MAX_FILES limit.\n");
                exit(EXIT_FAILURE);
            }
        }
    }
    closedir(dir);
}

void reduce() {
    if (fileCount == 0) {
        printf("No files found in the current
directory.\n");
        return;
    }

    off_t maxSize = 0;

    for (int i = 0; i < fileCount; i++) {
        if (fileInfoList[i].size > maxSize) {
            maxSize = fileInfoList[i].size;
        }
    }

    printf("Users owning files with maximum size (%lld
bytes):\n", maxSize);
    for (int i = 0; i < fileCount; i++) {
        if (fileInfoList[i].size == maxSize) {
            printf("Owner: %s, File: %s\n",
fileInfoList[i].owner, fileInfoList[i].filename);
        }
    }
}

int main() {
    char cwd[512];
    if (getcwd(cwd, sizeof(cwd)) == NULL) {
        perror("getcwd() error");
        return EXIT_FAILURE;
    }
```

```c
    map(cwd);
    reduce();

    return 0;
}
```

```
.../sem7/dsys/2024-10-22

(base) ~/desktop/cse/ASSGN/sem7/dsys/2024-10-22
 → clang q1.c -o q1
(base) ~/desktop/cse/ASSGN/sem7/dsys/2024-10-22
 → ./q1
Users owning files with maximum size (34288 bytes):
Owner: ShresthS, File: q1
(base) ~/desktop/cse/ASSGN/sem7/dsys/2024-10-22
 → ls -l
total 96
-rwxr-xr-x@ 1 ShresthS  staff  34288 Oct 22 21:07 q1
-rw-r--r--@ 1 ShresthS  staff   2119 Oct 22 21:06 q1.c
-rw-r--r--@ 1 ShresthS  staff   1680 Oct 22 21:06 q2c.c
-rw-r--r--@ 1 ShresthS  staff   1875 Oct 22 21:06 q2s.c
(base) ~/desktop/cse/ASSGN/sem7/dsys/2024-10-22
 → |
```

```c
// Q2 : RPC Server

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>

#define PORT 8080
#define BUFFER_SIZE 1024

void send_file(FILE *file, int socket_fd) {
    char buffer[BUFFER_SIZE];
    while (fgets(buffer, BUFFER_SIZE, file) != NULL) {
        send(socket_fd, buffer, strlen(buffer), 0);
        memset(buffer, 0, BUFFER_SIZE);
    }
    printf("File sent successfully.\n");
}

int main() {
    int server_fd, new_socket;
    struct sockaddr_in address;
    int addrlen = sizeof(address);

    server_fd = socket(AF_INET, SOCK_STREAM, 0);
    if (server_fd == 0) {
        perror("Socket failed");
        exit(EXIT_FAILURE);
    }

    address.sin_family = AF_INET;
    address.sin_addr.s_addr = INADDR_ANY;
    address.sin_port = htons(PORT);

    if (bind(server_fd, (struct sockaddr *) &address,
sizeof(address)) < 0) {
        perror("Bind failed");
        close(server_fd);
        exit(EXIT_FAILURE);
    }

    if (listen(server_fd, 3) < 0) {
        perror("Listen failed");
        exit(EXIT_FAILURE);
```

```
    }

    printf("Waiting for connections...\n");

    new_socket = accept(server_fd, (struct sockaddr *)
&address, (socklen_t * ) & addrlen);
    if (new_socket < 0) {
        perror("Accept failed");
        exit(EXIT_FAILURE);
    }

    char filename[BUFFER_SIZE] = {0};
    recv(new_socket, filename, BUFFER_SIZE, 0);
    printf("Client requested file: %s\n", filename);

    FILE *file = fopen(filename, "r");
    if (file == NULL) {
        perror("File not found");
        close(new_socket);
        close(server_fd);
        exit(EXIT_FAILURE);
    }

    send_file(file, new_socket);
    fclose(file);
    close(new_socket);
    close(server_fd);

    return 0;
}

// Q2 : RPC Client

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>

#define PORT 8080
#define BUFFER_SIZE 1024

void receive_file(int socket_fd, const char
*output_filename) {
    char buffer[BUFFER_SIZE];
```

```
    FILE *file = fopen(output_filename, "w");
    if (file == NULL) {
        perror("Failed to open output file");
        exit(EXIT_FAILURE);
    }

    int bytes_received;
    while ((bytes_received = recv(socket_fd, buffer,
BUFFER_SIZE, 0)) > 0) {
        fwrite(buffer, sizeof(char), bytes_received,
file);
        memset(buffer, 0, BUFFER_SIZE);
    }
    printf("File received successfully.\n");
    fclose(file);
}

int main(int argc, char *argv[]) {
    if (argc != 3) {
        fprintf(stderr, "Usage: %s <server_ip>
<filename>\n", argv[0]);
        exit(EXIT_FAILURE);
    }

    const char *server_ip = argv[1];
    const char *filename = argv[2];

    int socket_fd;
    struct sockaddr_in server_address;

    socket_fd = socket(AF_INET, SOCK_STREAM, 0);
    if (socket_fd < 0) {
        perror("Socket creation failed");
        exit(EXIT_FAILURE);
    }

    server_address.sin_family = AF_INET;
    server_address.sin_port = htons(PORT);

    if (inet_pton(AF_INET, server_ip,
&server_address.sin_addr) <= 0) {
        perror("Invalid address");
        exit(EXIT_FAILURE);
    }
```

```c
    if (connect(socket_fd, (struct sockaddr *)
&server_address, sizeof(server_address)) < 0) {
        perror("Connection failed");
        close(socket_fd);
        exit(EXIT_FAILURE);
    }

    send(socket_fd, filename, strlen(filename), 0);
    receive_file(socket_fd, filename);
    close(socket_fd);
    return 0;
}
```

```
.../sem7/dsys/2024-10-22
(base) ~/desktop/cse/ASSGN/sem7/dsys/2024-10-22
 → clang q2s.c -o q2s
(base) ~/desktop/cse/ASSGN/sem7/dsys/2024-10-22
 → ./q2s
Waiting for connections...
Client requested file: test.txt
File sent successfully.
(base) ~/desktop/cse/ASSGN/sem7/dsys/2024-10-22
 →
```

```
(base) ~/desktop/cse/ASSGN/sem7/dsys/2024-10-22
 → clang q2c.c -o q2c
(base) ~/desktop/cse/ASSGN/sem7/dsys/2024-10-22
 → ./q2c 127.0.0.1 test.txt
File received successfully.
(base) ~/desktop/cse/ASSGN/sem7/dsys/2024-10-22
 →
```