C Programs by using UNIX File System Calls

7. a. Develop a C program to emulate the UNIX ls –li command, which lists all the attributes of the files in a specified directory.

Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <dirent.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <pwd.h>
#include <grp.h>
#include <time.h>

void listFiles(char *path) {
    DIR *dir;
    struct dirent *entry;
    struct stat fileStat;

    // Open the directory
    if ((dir = opendir(path)) == NULL) {
        perror("Error opening directory");
        exit(EXIT_FAILURE);
    }

    // Print header
    printf("%-10s %-8s %-8s %-8s %-12s %s\n", "Permissions", "Links", "Owner", "Group", "Size",
"Last Modified");

    // Read each entry in the directory
    while ((entry = readdir(dir)) != NULL) {
        // Construct the full path
        char filePath[1024];
        snprintf(filePath, sizeof(filePath), "%s/%s", path, entry->d_name);

        // Get file status
        if (stat(filePath, &fileStat) < 0) {
            perror("Error getting file status");
            exit(EXIT_FAILURE);
        }
```

```
        // Get owner and group names
        struct passwd *owner = getpwuid(fileStat.st_uid);
        struct group *group = getgrgid(fileStat.st_gid);

        // Print file attributes
        printf("%s %2lu %-8s %-8s %8ld %s %s\n",
               (S_ISDIR(fileStat.st_mode)) ? "d" : "-",
               fileStat.st_nlink,
               owner->pw_name,
               group->gr_name,
               fileStat.st_size,
               ctime(&fileStat.st_mtime),
               entry->d_name);
    }

    // Close the directory
    closedir(dir);
}

int main(int argc, char *argv[]) {
    // Check if the correct number of arguments is provided
    if (argc != 2) {
        fprintf(stderr, "Usage: %s <directory>\n", argv[0]);
        exit(EXIT_FAILURE);
    }

    // Call the function to list files
    listFiles(argv[1]);

    return 0;
}
```

Compile the above code using gcc command

Explanation:

- listFiles function: Opens the specified directory, reads each entry, gets the file status, and prints the file attributes in a formatted way.
- main function: Checks if the correct number of command-line arguments is provided. Calls listFiles with the specified directory path.
- Compile the program and run it by providing the target directory as a command-line argument:
- Assuming you compiled the program and the executable is named a.out, you should run it like this:

./a.out /path/to/directory

- Replace /path/to/directory with the actual path of the directory you want to list. Make sure you provide the directory as a command-line argument. If you are still facing issues, please double-check the command and ensure that the specified directory exists.

b. Write a C program to remove empty files from the given directory using system calls.

Code:

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

void removeEmptyFiles(char *path) {
    DIR *dir;
    struct dirent *entry;
    struct stat fileStat;

    // Open the directory
    if ((dir = opendir(path)) == NULL) {
        perror("Error opening directory");
        exit(EXIT_FAILURE);
    }

    // Read each entry in the directory
    while ((entry = readdir(dir)) != NULL) {
        // Construct the full path
        char filePath[1024];
        snprintf(filePath, sizeof(filePath), "%s/%s", path, entry->d_name);

        // Get file status
        if (stat(filePath, &fileStat) < 0) {
            perror("Error getting file status");
            exit(EXIT_FAILURE);
        }

        // Check if the file is empty and remove it
        if (S_ISREG(fileStat.st_mode) && fileStat.st_size == 0) {
            if (unlink(filePath) == 0) {
                printf("Removed empty file: %s\n", entry->d_name);
```

```
            } else {
                perror("Error removing file");
            }
        }
    }

    // Close the directory
    closedir(dir);
}

int main(int argc, char *argv[]) {
    // Check if the correct number of arguments is provided
    if (argc != 2) {
        fprintf(stderr, "Usage: %s <directory>\n", argv[0]);
        exit(EXIT_FAILURE);
    }

    // Call the function to remove empty files
    removeEmptyFiles(argv[1]);

    return 0;
}
```

Compilation Steps:
Assuming you save the program in a file named remove_empty_files.c, open a terminal and follow these steps:

1. gcc remove_empty_files.c -o remove_empty_files

This command uses the GCC compiler to compile the C program and produces an executable named remove_empty_files.

2. ./remove_empty_files /path/to/directory

Replace /path/to/directory with the actual path of the directory you want to process.

Explanation:

- The removeEmptyFiles function opens the specified directory, reads each entry, checks if it's a regular file with zero size, and removes it using the unlink system call.
- The main function checks if the correct number of command-line arguments is provided, then calls removeEmptyFiles with the specified directory path.
- Make sure to run the program with proper permissions to delete files in the specified directory.