ASM 11 - File Programming Questions
Author: ThanhTH

1. Write a program to implement the user defined wc command.

```c
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>

void wc(const char *filename)
{
    FILE *file = fopen(filename, "r");
    if (file == NULL)
    {
        perror("Error opening file");
        return;
    }
```

```c
    int lines = 0, words = 0, characters = 0;
    int inWord = 0;
    char c;
```

```c
    while ((c = fgetc(file)) != EOF)
    {
        characters++;
        if (c == '\n')
        {
            lines++;
        }
```

```c
        if (isspace(c))
        {
            inWord = 0;
        }
        else if (inWord == 0)
        {
            inWord = 1;
            words++;
        }
    }
```

```c
    fclose(file);
```

```c
    printf(" %d %d %d %s\n", lines, words, characters, filename);
}
```

```c
int main(int argc, char *argv[])
{
    if (argc < 2)
    {
        fprintf(stderr, "Usage: %s <file1> <file2> ... <fileN>\n", argv[0]);
        return 1;
    }
```

```c
    for (int i = 1; i < argc; i++)
    {
        wc(argv[i]);
    }
```

```c
    return 0;
```

```
}
```

```
// ./program file1.txt
```

2. Write a program to implement the user defined grep command.

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

void grep(const char *filename, const char *searchTerm)
{
    FILE *file = fopen(filename, "r");
    if (file == NULL)
    {
        perror("Error opening file");
        return;
    }
```

```
    char line[1000];
    int lineNumber = 1;
    int check = 0;
```

```
    while (fgets(line, sizeof(line), file))
    {
        if (strstr(line, searchTerm) != NULL)
        {
            printf("File name: %s\n Line number: %d\n Line: %s\n", filename, lineNumber, line);
            check++;
        }
```

```
        lineNumber++;
    }
    if (!check)
    {
        printf("Not found!\n");
    }
```

```
    fclose(file);
}
```

```
int main(int argc, char *argv[])
{
    if (argc < 3)
    {
        fprintf(stderr, "Usage: %s <searchTerm> <file1> <file2> ... <fileN>\n", argv[0]);
        return 1;
    }
```

```
    const char *searchTerm = argv[1];
    for (int i = 2; i < argc; i++)
    {
        grep(argv[i], searchTerm);
    }
```

```
    return 0;
}
// ./program keyword file.txt
```

3. Write a program to replace all occurrences of substr with reverse of it.

```
#include <stdio.h>
#include <string.h>
```

```c
#include <stdlib.h>

#define BUFFER_SIZE 1000
```

```c
void reverseString(char *str)
{
    int left = 0, right = strlen(str) - 1;
    while (left < right)
    {
        char temp = str[left];
        str[left++] = str[right];
        str[right--] = temp;
    }
}
```

```c
void replaceOccurrences(FILE *inputFile, FILE *outputFile, const char *substr, const char
*reversedSubstr)
{
    char line[BUFFER_SIZE];
    while (fgets(line, sizeof(line), inputFile))
    {
        char *pos = line;
        while ((pos = strstr(pos, substr)) != NULL)
        {
            fwrite(line, 1, pos - line, outputFile);
            fputs(reversedSubstr, outputFile);
            pos += strlen(substr);
            memmove(line, pos, strlen(pos) + 1);
            pos = line;
        }
        fputs(line, outputFile);
    }
}
```

```c
int main(int argc, char *argv[])
{
    if (argc != 4)
    {
        fprintf(stderr, "Usage: %s <filename> <substr> <newfile>\n", argv[0]);
        return 1;
    }
```

```c
    const char *filename = argv[1];
    const char *substr = argv[2];
    const char *newfile = argv[3];
```

```c
    // Cấp phát động cho chuỗi đảo ngược
    int substrLen = strlen(substr);
    char *reversedSubstr = malloc(substrLen + 1);
    if (reversedSubstr == NULL)
    {
        perror("malloc");
        return 1;
    }
```

```c
    strcpy(reversedSubstr, substr);
    reverseString(reversedSubstr);
```

```c
    FILE *inputFile = fopen(filename, "r");
    if (!inputFile)
    {
```

```
        perror("Error opening input file");
        free(reversedSubstr);
        return 1;
    }
```

```
    FILE *outputFile = fopen(newfile, "w");
    if (!outputFile)
    {
        perror("Error opening output file");
        fclose(inputFile);
        free(reversedSubstr);
        return 1;
    }
```

```
    replaceOccurrences(inputFile, outputFile, substr, reversedSubstr);
```

```
    fclose(inputFile);
    fclose(outputFile);
    free(reversedSubstr);
```

```
    printf("Replaced all occurrences of '%s' with '%s' in %s and saved to %s.\n", substr,
reversedSubstr, filename, newfile);
    return 0;
}
```

```
//  ./program input.txt substr output.txt
```

5.Write a program to replace all occurrences of substr with another substr(should work with different and same lengths)

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

#define BUFFER_SIZE 1000
```

```
void replaceOccurrences(FILE *inputFile, FILE *outputFile, const char *substr, const char
*replaceWith)
{
    char line[BUFFER_SIZE];
    while (fgets(line, sizeof(line), inputFile))
    {
        char *pos = line;
        while ((pos = strstr(pos, substr)) != NULL)
        {
            fwrite(line, 1, pos - line, outputFile);
            fputs(replaceWith, outputFile);
            pos += strlen(substr);
            memmove(line, pos, strlen(pos) + 1);
            pos = line;
        }
        fputs(line, outputFile);
    }
}
```

```
int main(int argc, char *argv[])
{
    if (argc != 4)
    {
```

```
        fprintf(stderr, "Usage: %s <filename> <substr> <replaceWith>\n", argv[0]);
        return 1;
    }
```

```
    const char *filename = argv[1];
    const char *substr = argv[2];
    const char *replaceWith = argv[3];
```

```
    FILE *inputFile = fopen(filename, "r");
    if (!inputFile)
    {
        perror("Error opening input file");
        return 1;
    }
```

```
    char newFilename[256];
    snprintf(newFilename, sizeof(newFilename), "output.txt");
    FILE *outputFile = fopen(newFilename, "w");
    if (!outputFile)
    {
        perror("Error opening output file");
        fclose(inputFile);
        return 1;
    }
```

```
    replaceOccurrences(inputFile, outputFile, substr, replaceWith);
```

```
    fclose(inputFile);
    fclose(outputFile);
```

```
    printf("Replaced all occurrences of '%s' with '%s' in %s and saved to %s.\n", substr,
replaceWith, filename, newFilename);
    return 0;
}
```

```
// ./program input.txt substr replaceWith
```

8. Write a program to Sort the lines of a file according to their length and update back to same file.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAX_LINE_LENGTH 256
#define INITIAL_SIZE 10
```

```
int compareLines(const void *a, const void *b)
{
    const char *lineA = *(const char **)a;
    const char *lineB = *(const char **)b;
    return strlen(lineA) - strlen(lineB);
}
```

```
void sortLinesInFile(const char *filename)
{
    FILE *file = fopen(filename, "r");
    if (!file)
    {
        perror("Error opening file");
        return;
    }
```

```
    // Đọc các dòng vào mảng
    int capacity = INITIAL_SIZE;
    int count = 0;
    char **lines = malloc(capacity * sizeof(char *));
    if (!lines)
    {
        perror("Memory allocation failed");
        fclose(file);
        return;
    }
```

```
    char buffer[MAX_LINE_LENGTH];
    while (fgets(buffer, sizeof(buffer), file))
    {
        buffer[strcspn(buffer, "\n")] = '\0';
```

```
        if (count >= capacity)
        {
            capacity *= 2;
            lines = realloc(lines, capacity * sizeof(char *));
            if (!lines)
            {
                perror("Memory allocation failed");
                fclose(file);
                return;
            }
        }
```

```
        lines[count] = strdup(buffer);
        if (!lines[count])
        {
            perror("Memory allocation failed");
            fclose(file);
            return;
        }
        count++;
    }
    fclose(file);
```

```
    // Sort by leng
    qsort(lines, count, sizeof(char *), compareLines);
```

```
    file = fopen(filename, "w");
    if (!file)
    {
        perror("Error opening file for writing");
        for (size_t i = 0; i < count; i++)
        {
            free(lines[i]);
        }
        free(lines);
        return;
    }
```

```
    for (int i = 0; i < count; i++)
    {
        fprintf(file, "%s\n", lines[i]);
        free(lines[i]);
    }
    free(lines);
```

```
    fclose(file);
}
```

```
int main(int argc, char *argv[])
{
    if (argc != 2)
    {
        fprintf(stderr, "Usage: %s <filename>\n", argv[0]);
        return 1;
    }
```

```
    sortLinesInFile(argv[1]);
    printf("Sorted lines by length and updated the file: %s\n", argv[1]);
    return 0;
}
```

```
// ./program file.txt
```

9. Write a program to delete a line in a file

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAX_LINE_LENGTH 256
#define INITIAL_SIZE 10
```

```
// Hàm để xóa một dòng trong tệp tin
void deleteLineFromFile(const char *filename, int lineToDelete)
{
    FILE *file = fopen(filename, "r");
    if (!file)
    {
        perror("Error opening file");
        return;
    }
```

```
    // Đọc các dòng vào mảng
    int capacity = INITIAL_SIZE;
    int count = 0;
    char **lines = malloc(capacity * sizeof(char *));
    if (!lines)
    {
        perror("Memory allocation failed");
        fclose(file);
        return;
    }
```

```
    char buffer[MAX_LINE_LENGTH];
    while (fgets(buffer, sizeof(buffer), file))
    {
        // Loại bỏ ký tự newline nếu có
        buffer[strcspn(buffer, "\n")] = '\0';
```

```
        if (count >= capacity)
        {
            capacity *= 2;
            lines = realloc(lines, capacity * sizeof(char *));
            if (!lines)
            {
```

```
                perror("Memory allocation failed");
                fclose(file);
                return;
            }
        }
```

```
        lines[count] = strdup(buffer); // copy a strings and return a pointer to this string
copy
        if (!lines[count])
        {
            perror("Memory allocation failed");
            fclose(file);
            return;
        }
        count++;
    }
    fclose(file);
```

```
    // Kiểm tra số dòng để xóa có hợp lệ không
    if (lineToDelete >= count)
    {
        fprintf(stderr, "Error: Line %zu does not exist.\n", lineToDelete);
        for (int i = 0; i < count; i++)
        {
            free(lines[i]);
        }
        free(lines);
        return;
    }
```

```
    // Ghi lại nội dung còn lại vào tệp tin, bỏ qua dòng cần xóa
    file = fopen(filename, "w");
    if (!file)
    {
        perror("Error opening file for writing");
        for (int i = 0; i < count; i++)
        {
            free(lines[i]);
        }
        free(lines);
        return;
    }
```

```
    for (int i = 0; i < count; i++)
    {
        if (i != lineToDelete)
        {
            fprintf(file, "%s\n", lines[i]);
        }
        free(lines[i]);
    }
    free(lines);
    fclose(file);
```

```
    printf("Deleted line %zu from %s.\n", lineToDelete, filename);
}
```

```
int main(int argc, char *argv[])
{
    if (argc != 3)
    {
```

```
        fprintf(stderr, "Usage: %s <filename> <line_number>\n", argv[0]);
        return 1;
    }
```

```
    const char *filename = argv[1];
    int lineToDelete = (int)atoi(argv[2]);
```

```
    if (lineToDelete == 0)
    {
        fprintf(stderr, "Error: Line number must be greater than 0.\n");
        return 1;
    }
```

```
    deleteLineFromFile(filename, lineToDelete - 1); // Chuyển đổi sang chỉ số bắt đầu từ 0
    return 0;
}
```

```
// ./program file.txt numofline
```

11. Write a program to implement the save to file and sync from file function to Student data base.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAX_NAME_LENGTH 100
#define FILENAME "students.txt"
```

```
typedef struct
{
    int id;
    char name[MAX_NAME_LENGTH];
    float grade;
} Student;
```

```
void saveToFile(Student *students, int count)
{
    FILE *file = fopen(FILENAME, "w");
    if (!file)
    {
        perror("Error opening file for writing");
        return;
    }
```

```
    for (int i = 0; i < count; i++)
    {
        fprintf(file, "%d,%s,%f\n", students[i].id, students[i].name, students[i].grade);
    }
```

```
    fclose(file);
    printf("Data saved to %s.\n", FILENAME);
}
```

```
int syncFromFile(Student **students)
{
    FILE *file = fopen(FILENAME, "r");
    if (!file)
    {
        perror("Error opening file for reading");
```

```
        return 0;
    }
```

```
    int capacity = 10;
    int count = 0;
    *students = malloc(capacity * sizeof(Student));
    if (!*students)
    {
        perror("Memory allocation failed");
        fclose(file);
        return 0;
    }
```

```
    while (fscanf(file, "%d,%99[^,],%f\n", &(*students)[count].id, (*students)[count].name,
&(*students)[count].grade) == 3)
    {
        count++;
        if (count >= capacity)
        {
            capacity *= 2;
            *students = realloc(*students, capacity * sizeof(Student));
            if (!*students)
            {
                perror("Memory allocation failed");
                fclose(file);
                return 0;
            }
        }
    }
```

```
    fclose(file);
    printf("Data synced from %s.\n", FILENAME);
    return count;
}
```

```
int main()
{
    int studentCount;
```

```
    printf("Enter the number of students: ");
    scanf("%zu", &studentCount);
```

```
    Student *students = malloc(studentCount * sizeof(Student));
    if (!students)
    {
        perror("Memory allocation failed");
        return 1;
    }
    for (int i = 0; i < studentCount; i++)
    {
        printf("Enter details for student %zu\n", i + 1);
        printf("ID: ");
        scanf("%d", &students[i].id);
        getchar(); // read \n
```

```
        printf("Name: ");
        fgets(students[i].name, MAX_NAME_LENGTH, stdin);
        students[i].name[strcspn(students[i].name, "\n")] = 0;
```

```
        printf("Grade: ");
        scanf("%f", &students[i].grade);
```

```
        getchar();
    }

    saveToFile(students, studentCount);

    Student *loadedStudents = NULL;
    int loadedCount = syncFromFile(&loadedStudents);

    for (int i = 0; i < loadedCount; i++)
    {
        printf("ID: %d, Name: %s, Grade: %.2f\n", loadedStudents[i].id, loadedStudents[i].name,
loadedStudents[i].grade);
    }

    free(students);
    free(loadedStudents);

    return 0;
}
```