Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Факультет безопасности информационных технологий

Дисциплина:

«Программирование»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №4

«Работа с файлами и структуры данных на основе указателей»

Вариант 1-4-5-2

Выполнил:

Суханкулиев Мухаммет,

студент группы N3146

(подпись)

Проверила:

Сыдыкова Эмилия,

преподаватель программирования

(отметка о выполнении)

(подпись)

Санкт-Петербург

2024 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Разработать на языке C для ОС Linux программу, которая читает из файла заданного формата состояние заданной структуры данных, выполняет манипуляции со структурой в памяти, считывая команды из стандартного потока ввода, и сохраняет результат в файл.

Для достижения поставленной цели необходимо решить следующие задачи:

- Выполнить задание;
- Протестировать программу;
- Заархивировать папку проекта.

1 РАБОТА С ФАЙЛАМИ И СТРУКТУРЫ ДАННЫХ НА ОСНОВЕ УКАЗАТЕЛЕЙ

Выполнение манипуляции над списком в соответствии с вариантом 1-4-5-2.

1.1 Задание

Тип списка – Односвязный список.

Формат данных в строке – Доменное имя.

Команда: delete_even.

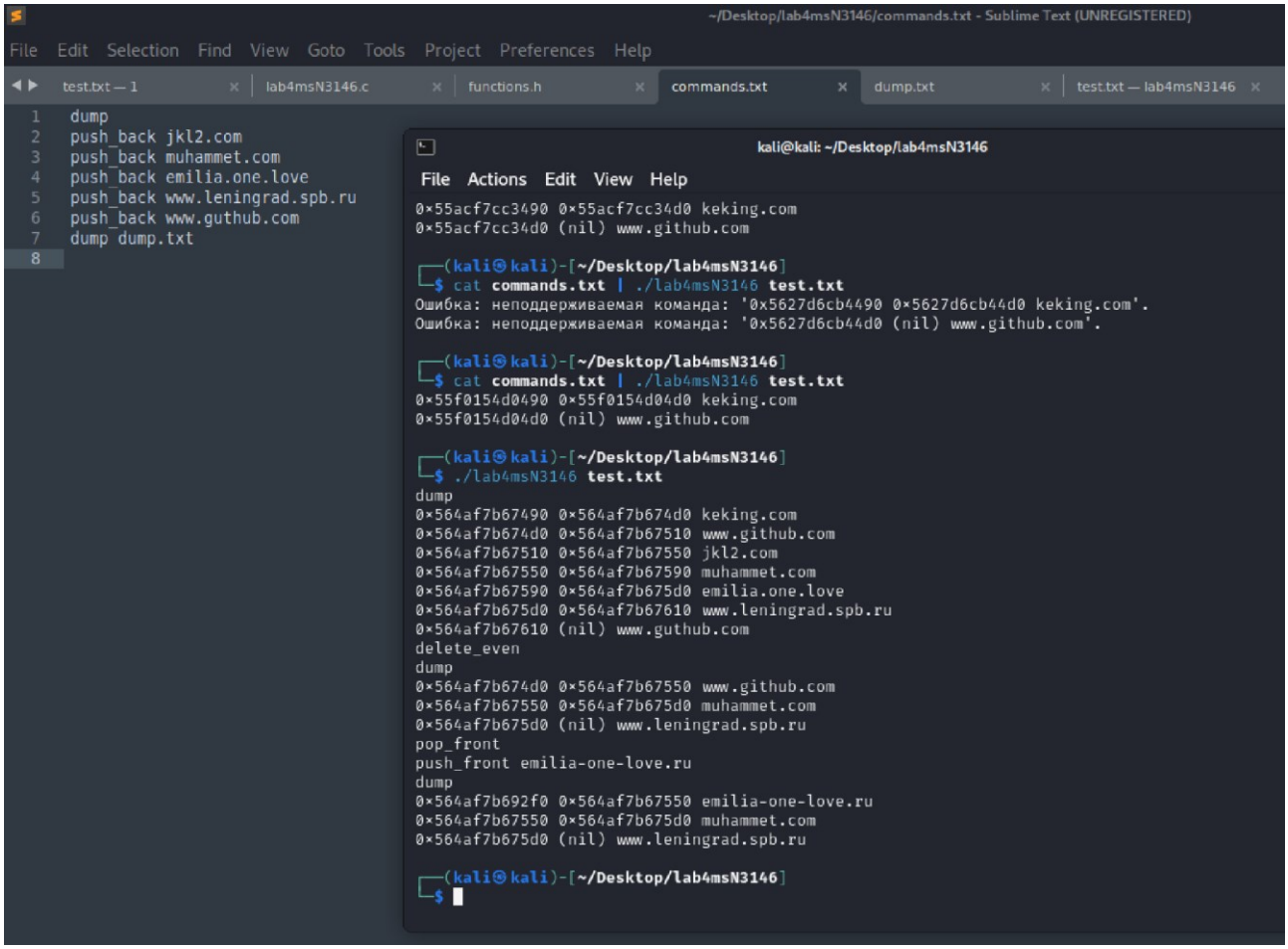
Описание: Удалить из списка элементы на четных позициях.

Формат файла:
(Начало файла)
Поле offset (целое без знака [4 байта]). Смещение области индексов в байтах от начала файла.
Область строк s_0, ..., s_{N-1}. Строки в кодировке UTF-8 располагаются одна за другой без промежутков. Каждая строка заканчивается нулевым байтом.
Область произвольных данных data. Область произвольного размера (возможно, нулевого).
Область индексов i_0, ..., i_{N-1}. Индексы соответствующих строк s_0, ..., s_{N-1}, определяющие порядок строк в списке (N целых без знака [2 байта]). Область индексов представляет собой массив, расположенный в конце файла. Количество индексов определяет количество строк в файле.
(Конец файла)

1.2 Make-файл

```
.PHONY: all clean

APP=lab4msN3146
CFLAGS=-Wall -Wextra -Werror -g

all: $(APP)

$(APP): $(APP).c
	gcc -o $(APP) $(CFLAGS) $(APP).c

clean:
	rm $(APP)
```

1.2.1 Примеры работы программы




```
┌──(kali㉿kali)-[~/Desktop/1]
└─$ cd ~/Desktop/lab4msN3146

┌──(kali㉿kali)-[~/Desktop/lab4msN3146]
└─$ make all
gcc -o lab4msN3146 -Wall -Wextra -Werror -g lab4msN3146.c

┌──(kali㉿kali)-[~/Desktop/lab4msN3146]
└─$ LAB4DEBUG=1 ./lab4msN3146 -v
Суханкулиев Мухаммет, гр. N3146
Вариант: 1-4-5-2

┌──(kali㉿kali)-[~/Desktop/lab4msN3146]
└─$ LAB4DEBUG=1 ./lab4msN3146
Использование: └─$./lab4msN3146 имя_файла
(-v для вывода информации о студенте)

┌──(kali㉿kali)-[~/Desktop/lab4msN3146]
└─$ LAB4DEBUG=1 ./lab4msN3146 test.txt
dump
push_front kek
Ошибка: строка 'kek' не является доменным именем.
push_front keking.com
push_back www.github.com
dump
0×55f3a56cb740 0×55f3a56cd480 keking.com
0×55f3a56cd480 (nil) www.github.com

┌──(kali㉿kali)-[~/Desktop/lab4msN3146]
└─$
```

1.3 Исходный текст программы

1.3.1 .c-файл

```c
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include "functions.h"

int main(int argc, char *argv[]) {
    // Проверка запуска с переменной среды, включающей отладочный вывод.
    // Пример запуска с установкой переменной LAB4DEBUG в 1:
    // $ LAB4DEBUG=1 ./lab4msN3146 test.txt
    char *DEBUG = getenv("LAB3DEBUG");
    if (DEBUG) {
        fprintf(stderr, "Включен вывод отладочных сообщений\n");
    }

    if(argc != 2) {
        fprintf(stderr, "Использование: └─$%s имя_файла\n(-v для вывода информации о
студенте)\n", argv[0]);
        return EXIT_FAILURE;
    }
    if(strcmp(argv[1], "-v") == 0) {
        if (DEBUG) {
            fprintf(stderr, "Опция -v включена\n");
            }
        fprintf(stderr, "Суханкулиев Мухаммет, гр. N3146\nВариант: 1-4-5-2\n");
        exit(EXIT_SUCCESS);
    }

    // Чтение данных из файла (добавил перехват ошибки переполнения стека, ибо таким
образом злоумышленик может получить доступ к своей функции)
    signal(SIGSEGV, handle_sigsegv);
    Node* head = readDataFromFile(argv[1]);

    // Обработка ввода пользователя
    char command[256];
    while(fgets(command, sizeof(command), stdin)) {
        command[strcspn(command, "\n")] = 0;
        if(strncmp(command, "push_front ", 11) == 0) {
        push_front(&head, command + 11);
        } else if(strncmp(command, "push_back ", 10) == 0) {
            push_back(&head, command + 10);
        } else if(strcmp(command, "pop_front") == 0) {
            pop_front(&head);
        } else if(strcmp(command, "pop_back") == 0) {
            pop_back(&head);
        } else if(strncmp(command, "dump", 4) == 0) {
            char* filename = NULL;
            if(strlen(command) > 5) {
                filename = command + 5;
            }
            dump(head, filename);
        } else if(strcmp(command, "delete_even") == 0) {
            delete_even(&head);
        } else {
            fprintf(stderr, "Ошибка: неподдерживаемая команда: '%s'.\n", command);
        }
    }
```

```
    if (head == NULL){
        createEmptyFile(argv[1]);
    } else {
        writeDataToFile(head, argv[1]);
        Node* current = head;
        while (current != NULL) {
            Node* next = current->next;
            free(current->data);
            free(current);
            current = next;
        }
    }
    return EXIT_SUCCESS;
}
```

1.3.2 .h-файл

```
#ifndef FUNCTIONS_H
#define FUNCTIONS_H

#include <stdint.h>
#include <unistd.h>
#include <regex.h>
#include <signal.h>

typedef struct Node {
    char* data;
    struct Node* next;
} Node;

Node* createNode(char* data) {
    Node* new_node = (Node*)malloc(sizeof(Node));
    if(new_node == NULL) {
        fprintf(stderr, "Ошибка выделения памяти\n");
        return NULL;
    }
    new_node->data = strdup(data);
    new_node->next = NULL;
    return new_node;
}

// Проверка Доменного имени
int isValidDomain(char* domain) {
    regex_t regex;
    int reti;
    reti = regcomp(&regex, "^([a-z0-9]+(-[a-z0-9]+)*\\.)+[a-z]{2,}$", REG_EXTENDED);
    if (reti) {
        fprintf(stderr, "Could not compile regex\n");
        return 0;
    }
    reti = regexec(&regex, domain, 0, NULL, 0);
    regfree(&regex);
    if (!reti) {
        return 1;
    } else if (reti == REG_NOMATCH) {
        return 0;
    } else {
        fprintf(stderr, "Regex match failed\n");
        return 0;
    }
}
```

```
void free_list(Node** head) {
    Node* tmp;
    while (*head != NULL) {
        tmp = *head;
        *head = (*head)->next;
        free(tmp->data);
        free(tmp);
    }
}

void push_front(Node** head, char* data) {
    if (isValidDomain(data)) {
        Node* new_node = createNode(data);
        if(new_node == NULL) {
            return;
        }
        new_node->next = *head;
        *head = new_node;
    } else {
        fprintf(stderr, "Ошибка: строка '%s' не является доменным именем.\n", data);
    }
}

void push_back(Node** head, char* data) {
    if (isValidDomain(data)) {
        Node* new_node = createNode(data);
        if(new_node == NULL) {
            return;
        }

        if (*head == NULL) {
            *head = new_node;
        } else {
            Node* current = *head;
            while (current->next != NULL) {
                current = current->next;
            }
            current->next = new_node;
        }
    } else {
        fprintf(stderr, "Ошибка: строка '%s' не является доменным именем.\n", data);
    }
}

void pop_front(Node** head) {
    if (*head == NULL) {
        return;
    }
    Node* next_node = (*head)->next;
    free((*head)->data);
    free(*head);
    *head = next_node;
}

void pop_back(Node** head) {
    if (*head == NULL) {
        return;
    }
    Node* current = *head;
    Node* prev = NULL;
    while (current->next != NULL) {
        prev = current;
```

```
        current = current->next;
    }
    if (prev == NULL) {
        *head = NULL;
    } else {
        prev->next = NULL;
    }
    free(current->data);
    free(current);
}

void dump(Node* head, char* filename) {
    FILE* stream;
    if (filename != NULL) {
        stream = fopen(filename, "w");
        if (stream == NULL) {
            fprintf(stderr, "Ошибка: не удалось открыть файл '%s' для записи.\n",
filename);
            return;
        }
    } else {
        stream = stdout;
    }
    Node* current = head;
    while (current != NULL) {
        fprintf(stream, "%p %p %s\n", (void*)current, (void*)current->next, current-
>data);
        current = current->next;
    }
    if (filename != NULL) {
        fclose(stream);
    }
}

// Удаление элементов на четных позициях из списка
void delete_even(Node** head) {
    Node* current = *head;
    Node* prev = NULL;
    int index = 0;
    while (current != NULL) {
        if (index % 2 == 0) {
            Node* next = current->next;
            if (prev == NULL) {
                *head = next;
            } else {
                prev->next = next;
            }
            free(current->data);
            free(current);
            current = next;
        } else {
            prev = current;
            current = current->next;
        }
        index++;
    }
}

// Запись данных из файла, заданного формата, в Односвязный список
Node* readDataFromFile(char* filename) {
    FILE *file = fopen(filename, "rb");
    if (file != NULL) {
        uint32_t offset;
```

```
        if (fread(&offset, sizeof(uint32_t), 1, file) != 1) {
            printf("Некорректный формат файла\n");
            exit(EXIT_FAILURE);
        }
        Node *head = NULL, *current = NULL;
        char buffer[256];
        uint16_t index;
        uint32_t i = 0;
        while (i < offset) {
            int j = 0;
            char c;
            while ((c = fgetc(file)) != '\0') {
                buffer[j++] = c;
            }
            buffer[j] = '\0';
            Node *node = (Node*)malloc(sizeof(Node));
            node->data = strdup(buffer);
            node->next = NULL;
            if (head == NULL) {
                head = node;
                current = node;
            } else {
                current->next = node;
                current = node;
            }
            i += j + 1;
        }
        if (i != offset) {
            printf("Некорректный формат файла\n");
            exit(EXIT_FAILURE);
        }
        fseek(file, offset, SEEK_SET);
        while (fread(&index, sizeof(uint16_t), 1, file) == 1) {
            Node *node = head;
            for (i = 0; i < index && node != NULL; i++) {
                node = node->next;
            }
        }
        fclose(file);
        return head;
    } else {
        return NULL;
    }
}

// Запись данных в файл, в определенном формате
void writeDataToFile(Node* head, char* filename) {
    FILE *file = fopen(filename, "wb");
    if (file == NULL) {
        printf("Не удалось открыть файл\n");
        return;
    }
    Node *current = head;
    uint32_t offset = 0;
    uint16_t index = 0;
    while (current != NULL) {
        offset += strlen(current->data) + 1;
        current = current->next;
    }
    offset += sizeof(uint16_t) * index;
    fwrite(&offset, sizeof(uint32_t), 1, file);
    current = head;
    while (current != NULL) {
```

```
        fwrite(current->data, sizeof(char), strlen(current->data) + 1, file);
        current = current->next;
    }
    current = head;
    while (current != NULL) {
        fwrite(&index, sizeof(uint16_t), 1, file);
        index++;
        current = current->next;
    }
    fclose(file);
}

void createEmptyFile(char* filename) {
    FILE *file = fopen(filename, "w");
    if (file != NULL) {
        fclose(file);
    } else {
        printf("Не удалось создать файл\n");
    }
}

void handle_sigsegv(int sig) {
    (void)sig;
    printf("Некорректный формат файла\n");
    exit(EXIT_FAILURE);
}

#endif
```

ЗАКЛЮЧЕНИЕ

Были выполнены манипуляции над списком [из файла].

Это позволило закрепить навыки работы с файлами и структурами данных на основе указателей.

Я все сделал, я молодец. =)

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Лекции Гирика Алексея Валерьевича по программированию – 2024. – URL : https://drive.google.com/drive/folders/1eAiMW4hD9TLhZH2vtpKPWzZkzKp10BnL (дата обращения: 25.04.2024).