Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Операционные среды и системное программирование

ОТЧЕТ

к лабораторной работе №3

на тему

**ОСНОВЫ ПРОГРАММИРОВАНИЯ НА С ПОД UNIX**

Студент М. Ю. Пухов

Преподаватель С. И. Сиротко

Минск 2024

СОДЕРЖАНИЕ

[1 Цель работы 3](#_Toc146752068)

[2 Теоретические сведения 4](#_Toc146752069)

[3 Результат выполнения 5](#_Toc146752070)

[Заключение 6](#_Toc146752071)

[Список использованных источников 7](#_Toc146752072)

[Приложение А (обязательное) Листинг кода 8](#_Toc146752073)

1. **ЦЕЛЬ РАБОТЫ**

Изучение среды программирования и основных инструментов: компилятор/сборщик («коллекция компиляторов») gcc, управление обработкой проекта make (и язык makefile), библиотеки и т.д.   
Практическое использование основных библиотек и системных вызовов: ввод-вывод и работа с файлами, обработка текста, распределение памяти, управление выполнением и т.п.

1. **ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**

Азбука Морзе – система кодирования букв, цифр и знаков препинания с помощью комбинаций точек и тире. Разработана в 1840-х годах американским художником и изобретателем Сэмюэлом Морзе для использования в телеграфной связи.

Основными элементами азбуки Морзе являются:

* Точка – короткий сигнал
* Тире – длинный сигнал
* Пауза между буквами в слове – пауза в 3 точки
* Пауза между словами – пауза в 7 точек

Каждой букве латинского алфавита, цифре и некоторым знакам пунктуации соответствует своя комбинация точек и тире.

Для реализации фильтра морзянки необходимо:

* Определить словарь соответствия символов и комбинаций Морзе
* Реализовать функции кодирования (преобразования символа в комбинацию Морзе) и декодирования (преобразования комбинации Морзе в символ)
* Организовать чтение данных из входного потока и запись в выходной поток с преобразованием согласно выбранному режиму работы (кодирование или декодирование)

1. **РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ**

Начальный файл input.txt с сообщением дл шифровки (рисунок 1).



Рисунок 1 – Файл с начальным сообщением

После выполнения программы сообщение шифруется в азубу Морзе (рисунок 2).



Рисунок 2 – Шифрованное сообщение

# ЗАКЛЮЧЕНИЕ

В ходе выполнения данной лабораторной работы был реализован фильтр для кодирования текста с использованием азбуки Морзе.

Был разработан словарь соответствия букв, цифр и знаков препинания и их кодов Морзе. На его основе реализованы функции прямого и обратного преобразования.

Фильтр осуществляет посимвольное чтение данных из stdin, преобразование с помощью разработанных функций в соответствии с выбранным режимом работы и вывод результата в stdout.

Режим работы фильтра (кодирование или декодирование) задается опцией командной строки при запуске.

Были подготовлены тестовые входные данные и проверена корректность работы фильтра как в режиме кодирования, так и декодирования. Полученные результаты соответствуют ожидаемым.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

# Азбука Морзе [Электронный ресурс]. – Режим доступа: https://morsedecoder.com/ru/.

# Introduction au Script Bash [Электронный ресурс]. – Режим доступа: https://www.hostinger.fr/tutoriels/introduction-au-script-bash-avec-exemples.

# ПРИЛОЖЕНИЕ А (обязательное) Листинг кода

Листинг 1 – Файл morse*.h*

void charToMorse(char ch, char\* result);

void stringToMorse(const char\* string, char\* result);

const char\* mapToMorse(char ch);

#define MORSE\_CHAR\_MAX\_LEN 6

Листинг 2 – Файл morse*.c*

*#include "morse.h"*

*#include <string.h>*

*const char\* mapToMorse(char ch) {*

*switch (ch) {*

*case ' ':*

*return " ";*

*case 'A':*

*case 'a':*

*return ".-";*

*case 'B':*

*case 'b':*

*return "-...";*

*case 'C':*

*case 'c':*

*return "-.-.";*

*case 'D':*

*case 'd':*

*return "-..";*

*case 'E':*

*case 'e':*

*return ".";*

*case 'F':*

*case 'f':*

*return "..-.";*

*case 'G':*

*case 'g':*

*return "--.";*

*case 'H':*

*case 'h':*

*return "....";*

*case 'I':*

*case 'i':*

*return "..";*

*case 'J':*

*case 'j':*

*return ".---";*

*case 'K':*

*case 'k':*

*return "-.-";*

*case 'L':*

*case 'l':*

*return ".-..";*

*case 'M':*

*case 'm':*

*return "--";*

*case 'N':*

*case 'n':*

*return "-.";*

*case 'O':*

*case 'o':*

*return "---";*

*case 'P':*

*case 'p':*

*return ".--.";*

*case 'Q':*

*case 'q':*

*return "--.-";*

*case 'R':*

*case 'r':*

*return ".-.";*

*case 'S':*

*case 's':*

*return "...";*

*case 'T':*

*case 't':*

*return "-";*

*case 'U':*

*case 'u':*

*return "..-";*

*case 'V':*

*case 'v':*

*return "...-";*

*case 'W':*

*case 'w':*

*return ".--";*

*case 'X':*

*case 'x':*

*return "-..-";*

*case 'Y':*

*case 'y':*

*return "-.--";*

*case 'Z':*

*case 'z':*

*return "--..";*

*case '0':*

*return "-----";*

*case '1':*

*return ".----";*

*case '2':*

*return "..---";*

*case '3':*

*return "...--";*

*case '4':*

*return "....-";*

*case '5':*

*return ".....";*

*case '6':*

*return "-....";*

*case '7':*

*return "--...";*

*case '8':*

*return "---..";*

*case '9':*

*return "----.";*

*case '?':*

*return "..--..";*

*case '.':*

*return ".-.-.-";*

*case '!':*

*return "-.-.--";*

*case ',':*

*return "--..--";*

*case '(':*

*return "-.--.";*

*case ')':*

*return "-.--.-";*

*case ';':*

*return "-.-.-.";*

*case '+':*

*return ".-.-.";*

*case '-':*

*return "-....-";*

*case ':':*

*return "---...";*

*case '=':*

*return "-...-";*

*default:*

*return "";*

*}*

*}*

*void charToMorse(char ch, char\* result) {*

*strcpy(result, mapToMorse(ch));*

*}*

*void stringToMorse(const char\* str, char\* result) {*

*int len = strlen(str);*

*int pos = 0;*

*char morse[MORSE\_CHAR\_MAX\_LEN]; // +1 for separator*

*for (int i = 0; i < len; i++)*

*{*

*charToMorse(str[i], morse);*

*int morseLen = strlen(morse);*

*memcpy(result + pos, morse, morseLen);*

*pos+=morseLen;*

*memcpy(result + pos, " ", 1);*

*pos++;*

*}*

*result[--pos] = '\0';*

*}*

Листинг 3 – Файл *main.c*

*#include <stdlib.h>*

*#include <stdio.h>*

*#include <string.h>*

*#include "morse.h"*

*int main(int argc, char \*argv[]) {*

*// Проверка наличия правильного количества аргументов командной строки*

*if (argc != 3) {*

*printf("Using: %s input\_file output\_file\n", argv[0]);*

*return 1;*

*}*

*// Извлечение имен входного и выходного файлов из аргументов командной строки*

*char \*inputFileName = argv[1];*

*char \*outputFileName = argv[2];*

*// Открытие входного файла*

*FILE \*inputFile = fopen(inputFileName, "r");*

*if (inputFile == NULL) {*

*printf("Error open.\n");*

*return 1;*

*}*

*// Открытие выходного файла*

*FILE \*outputFile = fopen(outputFileName, "w");*

*if (outputFile == NULL) {*

*printf("Ошибка открытия выходного файла.\n");*

*fclose(inputFile);*

*return 1;*

*}*

*// Определение размера входного файла*

*fseek(inputFile, 0, SEEK\_END);*

*long fileSize = ftell(inputFile);*

*fseek(inputFile, 0, SEEK\_SET);*

*// Выделение памяти под текст из входного файла*

*char \*text = (char\*) malloc(fileSize + 1);*

*if (text == NULL) {*

*printf("Memory error\n");*

*fclose(inputFile);*

*return 1;*

*}*

*// Чтение содержимого входного файла в выделенную память*

*size\_t bytesRead = fread(text, 1, fileSize, inputFile);*

*text[bytesRead] = '\0';*

*// Расчет длины входного текста*

*size\_t len = strlen(text);*

*// Выделение памяти под результат в виде кода Морзе*

*char\* result = (char\*)malloc((len \* (MORSE\_CHAR\_MAX\_LEN + 1) + 1) \* sizeof(char));*

*// Преобразование входного текста в код Морзе*

*stringToMorse(text, result);*

*// Запись результата в коде Морзе в выходной файл*

*fprintf(outputFile, "%s", result);*

*fprintf(outputFile, "%s", "\n");*

*// Закрытие входного и выходного файлов*

*fclose(inputFile);*

*fclose(outputFile);*

*// Освобождение выделенной памяти*

*free(result);*

*free(text);*

*printf("Готово.\n");*

*return 0;*

*}*