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 C под Unix.
Инструментарий программиста в Unix»

Выполнил Н. В. Климкович

Проверил Н. Ю. Гриценко

Минск 2024

СОДЕРЖАНИЕ

1 Постановка задачи 3
2 Теоретические сведения 4
3 Результат выполнения 5
Выводы 6
Список использованных источников 7
Приложение А (обязательное) Листинг кода 8

1 ПОСТАНОВКА ЗАДАЧИ

Целью выполнения данной лабораторной работы является изучение среды программирования и основных инструментов: сборщик, компилятор, gcc, управление обработкой проекта make и языка makefile, библиотеки. Практическое использование основных библиотек и системных вызовов: ввод и вывод, работа с файлами, обработка текста, распределение памяти, управление выполнением.

2 ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Язык программирования C является одним из наиболее популярных и мощных языков программирования.[1] Он известен своей эффективностью, скоростью выполнения и широким применением в различных областях, от системного программирования до разработки приложений.

Основные концепции языка C включают в себя переменные, операторы, условные выражения, циклы, функции, массивы, указатели и структуры данных.[2]

Makefile представляет собой текстовый файл, который содержит инструкции для сборки и компиляции программного проекта.[3] Он описывает зависимости между исходными файлами и целями сборки, а также команды, которые необходимо выполнить для получения этих целей.

Основная цель makefile является автоматизация процесса сборки программного проекта, что делает его более эффективным и удобным для разработчиков.

Язык makefile позволяет определять переменные, цели, правила компиляции и линковки, а также использовать условия и циклы для более гибкого управления процессом сборки.

Одним из ключевых элементов makefile является описание правил компиляции, где указываются зависимости между исходными файлами и объектными файлами, а также команды, которые необходимо выполнить для компиляции их в объектные файлы или сборки исполняемого файла.[4] Команда make используется для выполнения makefile и автоматической сборки проекта в соответствии с указанными в нем правилами и зависимостями.

Unix представляет собой семейство многозадачных и многопользовательских операционных систем, изначально разработанных в Bell Labs в начале семидесятых годов.

Основные принципы Unix включают в себя работу с файловой системой, управление процессами и потоками, а также командную оболочку (shell), которая предоставляет интерфейс для взаимодействия с операционной системой.

В Unix часто используется компилятор GCC (GNU Compiler Collection), который поддерживает множество языков программирования, включая C. Сборка проектов обычно осуществляется с помощью утилиты make и языка описания сборки makefile.

Unix предоставляет широкий набор библиотек, которые можно использовать для разработки приложений. Это включает в себя стандартные библиотеки, такие как libc, а также различные библиотеки для работы с файлами, сетевыми соединениями, графикой и другими аспектами разработки программного обеспечения.

3 РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ

В результате выполнения лабораторной работы было создано приложение, которое выполняет преобразование символов потока в комбинации азбуки Морзе. Результат работы приложения представлен на рисунке 3.1.

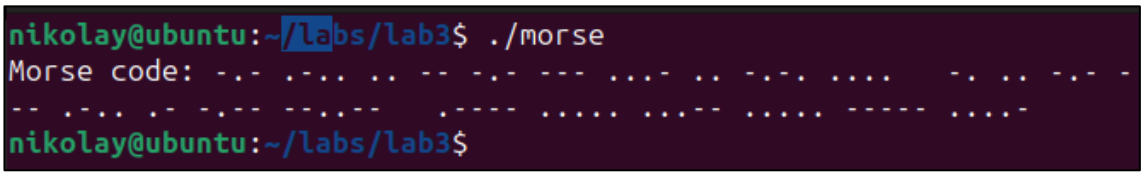

Рисунок 3.1 – Результат работы приложения

Входные данные представлены на рисунке 3.2.

Рисунок 3.2 – Входные данные

Код данной программы был многомодульным, был написан в соответствии с вариантом задания, а также при помощи создания makefile для управления обработкой проекта. Программа, которая выполняет преобразование символов потока в комбинации азбуки Морзе, реализованная на языке C под Unix, представляет собой эффективный инструмент для обработки данных в потоковом режиме, позволяющее передавать на вход файл с данными и на выходе получать новый файл с комбинацией азбуки Морзе.

ВЫВОДЫ

В результате выполнения лабораторной работы мы изучили основы программирования на языке C под операционной системой Unix. Освоили работу с компилятором GCC, утилитой make и языком описания сборки makefile. Создали приложение, которое преобразует символы в код азбуки Морзе.

Таким образом, выполнение данной лабораторной работы позволило ознакомиться с основными аспектами программирования на языке C под Unix, а также научиться эффективно использовать инструменты и библиотеки этой среды для разработки программного обеспечения.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Введение в Unix [Электронный ресурс]. – Режим доступа: https://unix.com. – Дата доступа: 01.02.2024.

[2] Makefile tutorial how to write [Электронный ресурс]. – Режим доступа: https://www.tutorialspoint.com/makefile. – Дата доступа: 02.02.2024.

[3] Функции C [Электронный ресурс]. – Режим доступа: https://metanit.com/cpp/tutorial/3.1.php. – Дата доступа: 04.02.2024.

[4] Классы C [Электронный ресурс]. – Режим доступа: https://www.w3schools.com/cpp/cpp_classes.asp. – Дата доступа: 04.02.2024.

ПРИЛОЖЕНИЕ А
(обязательное)
Листинг кода

Листинг 1 – Основной файл программы main.c

```
#include "morse.h"
#include <stdio.h>

int main() {
    const char *input_filename = "input.txt";
    const char *output_filename = "output.txt";

    text_to_morse_file(input_filename, output_filename);

    return 0;
}
```

Листинг 2 – Программная реализация азбуки Морзе morse.c

```
#include "morse.h"
#include <stdio.h>
#include <ctype.h>
#include <string.h>

const char* char_to_morse(char c) {
    switch (toupper(c)) {
        case 'A': return ".-";
        case 'B': return "-...";
        case 'C': return "-.-.";
        case 'D': return "-..";
        case 'E': return ".";
        case 'F': return "..-.";
        case 'G': return "--.";
        case 'H': return "....";
        case 'I': return "..";
        case 'J': return ".---";
        case 'K': return "-.-";
        case 'L': return ".-..";
        case 'M': return "--";
        case 'N': return "-.";
        case 'O': return "---";
        case 'P': return ".--.";
        case 'Q': return "--.-";
        case 'R': return ".-.";
        case 'S': return "...";
        case 'T': return "-";
        case 'U': return "..-";
        case 'V': return "...-";
        case 'W': return ".--";
        case 'X': return "-..-";
        case 'Y': return "-.--";
        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 "-.-.-.";
        case '=': return "-...-";
        case '+': return ".-.-.";
        case '-': return "-....-";
        case '_': return "..--.-";
        case '"': return ".-..-.";
        case '$': return "...-..-";
        case '@': return ".--.-.";
        case ' ': return " ";

        default: return "";
    }
}

void text_to_morse_file(const char *input_filename, const char
*output_filename) {
    FILE *input_file = fopen(input_filename, "r");
    FILE *output_file = fopen(output_filename, "w");

    if (input_file == NULL || output_file == NULL) {
        perror("Error opening files");
        return;
    }

    char morse_code[255];
    morse_code[0] = '\0';

    int ch;
    while ((ch = fgetc(input_file)) != EOF) {
        const char *morse = char_to_morse(ch);

        if (strlen(morse) > 0) {
            fprintf(output_file, "%s ", morse);
            strcat(morse_code, morse);
            strcat(morse_code, " ");
        }
    }

    printf("Morse code: %s\n", morse_code);

    fclose(input_file);
    fclose(output_file);
}
```

Листинг 3 – Файл morse.h заголовка для основной программы

```
#ifndef MORSE_H
#define MORSE_H

const char* char_to_morse(char c);
void text_to_morse_file(const char *input_filename, const char
*output_filename);

#endif
```

Листинг 4 – Файл makefile

```
CC = gcc
CFLAGS = -Wall -Wextra

TARGET = my_program
SOURCES = main.c morse.c
OBJECTS = $(SOURCES:.c=.o)

.PHONY: all clean

all: $(TARGET)

$(TARGET): $(OBJECTS)
	$(CC) $(CFLAGS) -o $@ $^

%.o: %.c
	$(CC) $(CFLAGS) -c $< -o $@

clean:
	rm -f $(TARGET) $(OBJECTS)
```