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. Инструментарий программиста в Unix»

Выполнил             Е. А. Киселёва

Проверил                          Н. Ю. Гриценко

Минск 2024

**СОДЕРЖАНИЕ**

[1 Постановка задачи 3](#_Toc157722973)

[2 Краткие теоретические сведения 4](#_Toc157722974)

[3 Результаты выполнения лабораторной работы 5](#_Toc157722975)

[Выводы 6](#_Toc157722976)

[Список использованных источников 7](#_Toc157722977)

[Приложение А (обязательное) Листинг исходного кода 8](#_Toc157722978)

# **1 ПОСТАНОВКА ЗАДАЧИ**

Целью выполнения данной лабораторной работы является изучение среды программирования и основных инструментов, таких как компилятор/сборщик gcc, управление обработкой проекта make и языка makefile. Кроме того, лнеобходимо написать программу на языке программирования С, реализующую преобразование символов потока в комбинации азбуки Морзе. Также необходимо создать makefile для управления обработкой проекта, собрать и протестировать исполняемый файл.

# **2 КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**

Язык программирования С является языком общего назначения, который широко используется для разработки системного программного обеспечения, прикладных программ и встраиваемых систем.[1]

GCC – это коллекция компиляторов для различных языков программирования, включая С. Также в данную коллекцию входят компиляторы для языков С++, Go. В зависимости от расширения имени файла, передаваемых в качестве параметра, и дополнительных опций, GCC запускает необходимые препроцессоры, компиляторы, сборщики.[2]

Компиляция – это процесс преобразования исходного кода программы из языка высокого уровня в машинный код. Компиляция происходит с помощью компилятора, который анализирует исходный код. Процесс компиляции преобразует исходный код программы в объектные файлы. Сборщик же создает исполняемый файл из одного или нескольких объектных файлов, полученных в результате компиляции.

Make – это утилита для автоматизации процесса сборки программы из исходных файлов. Make использует файл makefile, который содержит правила для компиляции и сборки проекта.[3]

Для больших проектов использование makefile и утилиты make позволяет достаточно сократить время компиляции и сборки.

Для выполнения данной лабораторной работы были использованы следующие сведения и концепции:

1 Язык программирования С: код программы был написан на языке программирования С.

2 Стандартные библиотеки: используются стандартные библиотеки stdio.h, ctype.h для работы с вводом-выводом и проверкой типов символов.

3 Функции и внешние файлы: программа разделена на несколько файлов: main.c, morse\_filter.c, morse\_filter.h. Файлы .c содержат определения функций, а файл .h содержит объявления функций и внешние переменные.

4 Makefile: используется для автоматизации процесса компиляции программы, определяет зависимости между исходными файлами и исполняемым файлом, а также задает правила для компиляции и чистки проекта.

5 Обработка строки: функция morse\_filter() принимает строку в качестве аргумента и фильтрует каждый символ. Если символ является буквой, цифрой, знаком препинания или пробелом, он заменяется соответствующим кодом Морзе. Если символ не распознан, выводится сообщение «Unknown character».

6 Тестирование: в makefile определена цель test, которая выполняет программу с входными данными из файла input.txt и сравнивает вывод с ожидаемым результатом из файла right\_result.txt с помощью утилиты diff.

# **3 РЕЗУЛЬТАТЫ ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ             РАБОТЫ**

В ходе выполнения лабораторной была разработана программа, которая преобразует символы потока в комбинации азбуки Морзе. Пользователю необходимо ввести строку из последовательного набора символов латинского алфавита. Программа преобразует введенные символы в комбинацию азбуки Морзе. Результат работы программы предоставлен на рисунке 3.1.



Рисунок 3.1 – Результат работы программы при использовании ввода данных

При помощи утилиты make была организована возможность тестирования. Ожидаемые результаты должны быть занесены в определенные файлы перед тестированием, полученные результаты программа занесет в файл самостоятельно. Результат тестирования предоставлен на рисунке 3.2.

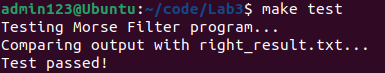


Рисунок 3.2 – Результат тестирования исполняемого файла

Таким образом, в ходе данной лабораторной работы была реализована программа, преобразующая символы потока в комбинации азбуки Морзе.

# **ВЫВОДЫ**

В ходе лабораторной работы были изучена среда программирования и основные инструменты, такие как компилятор/сборщик gcc, управление обработкой проекта make и языка makefile. Кроме того, была разработана программа на языке программирования С, реализующая преобразование символов потока в комбинации азбуки Морзе. Также был создан makefile для управления обработкой проекта, а также сборки и тестирования исполняемого файла.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

[1] Язык программирования С [Электронный ресурс]. – Режим доступа: https://metanit.com/c/tutorial/1.2.php. – Дата доступа: 17.02.2024.

[2] Компилятор GCC [Электронный ресурс]. – Режим доступа: https://metanit.com/c/tutorial/1.3.php. – Дата доступа: 17.02.2024.

[3] Makefile [Электронный ресурс]. – Режим доступа:   
https://habr.com/ru/articles/155201/. – Дата доступа: 18.02.2024.

## **ПРИЛОЖЕНИЕ А**

## **(обязательное)**

## **Листинг исходного кода**

Листинг 1 – Программный код главной функции

#include <stdio.h>

#include "morse\_filter.h"

int main() {

char input[100];

printf("Enter a string: ");

fgets(input, sizeof(input), stdin);

printf("Morse code: ");

morse\_filter(input);

return 0;

}

Листинг 2 – Программный код morse\_filter.c

#include "morse\_filter.h"

#include <ctype.h>

#include <stdio.h>

const char \*morse\_code[] = {

".-", // A

"-...", // B

"-.-.", // C

"-..", // D

".", // E

"..-.", // F

"--.", // G

"....", // H

"..", // I

".---", // J

"-.-", // K

".-..", // L

"--", // M

"-.", // N

"---", // O

".--.", // P

"--.-", // Q

".-.", // R

"...", // S

"-", // T

"..-", // U

"...-", // V

".--", // W

"-..-", // X

"-.--", // Y

"--..", // Z

"-----", // 0

".----", // 1

"..---", // 2

"...--", // 3

"....-", // 4

".....", // 5

"-....", // 6

"--...", // 7

"---..", // 8

"----.", // 9

"--..--",// ,

".-.-.-",// .

"..--..",// ?

"-.-.--" // !

};

void morse\_filter(char \*str) {

int i = 0;

while (str[i] != '\0') {

if (isalpha(str[i])) {

printf("%s ", morse\_code[toupper(str[i]) - 'A']);

} else if (isdigit(str[i])) {

printf("%s ", morse\_code[str[i] - '0' + 26]);

} else if (isspace(str[i])) {

printf(" ");

} else if (str[i] == ',') {

printf("%s ", morse\_code[36]);

} else if (str[i] == '.') {

printf("%s ", morse\_code[37]);

} else if (str[i] == '?') {

printf("%s ", morse\_code[38]);

} else if (str[i] == '!') {

printf("%s ", morse\_code[39]);

} else {

printf("Unknown character ");

}

i++;

}

printf("\n");

}

Листинг 3 – Программный код morse\_filter.h

#ifndef MORSE\_FILTER\_H

#define MORSE\_FILTER\_H

extern const char \*morse\_code[];

void morse\_filter(char \*str);

#endif /\* MORSE\_FILTER\_H \*/

Листинг 4 – Программный код makefile

CC = gcc

CFLAGS = -Wall -Wextra -Wpedantic

SRC = main.c morse\_filter.c

OBJ = $(SRC:.c=.o)

EXECUTABLE = morse\_filter

all: $(EXECUTABLE)

$(EXECUTABLE): $(OBJ)

$(CC) $(CFLAGS) $(OBJ) -o $@

clean:

rm -f $(OBJ) $(EXECUTABLE)

test: $(EXECUTABLE)

@echo "Testing Morse Filter program..."

@./$(EXECUTABLE) < input.txt > output.txt

@echo "Comparing output with right\_result.txt..."

@if diff output.txt right\_result.txt > /dev/null; then \

echo "Test passed!"; \

else \

echo "Test failed!"; \

fi

.PHONY: all clean test