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 № 4

на тему «Управление процессами и взаимодействие процессов»

Выполнил             А. К. Хрищанович

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, на практике необходимо написать программу, на языке программирования С, реализующую шифрование и дешифрование символов по азбуке 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 полностью написан на языке программирования С.

2 Стандартные библиотеки языка С: для работы с функциями ввода и вывода, строками и символами были использованы стандартные библиотеки языка С.

3 Компилятор GCC: для компиляции и сборки программы был использован компилятор GCC, который включает в себя компилятор языка С.

4 Система сборки make: для автоматизации процесса компиляции и сборки программы был создан makefile, в котором описаны все правила для компиляции и сборки различных модулей программы.

5 Модульное программирование: программа разделена на несколько модулей, каждый из которых отвечает за определенный функционал.

6 Тестирование исполняемого файла: при помощи makefile и цели test был реализован способ тестирования исполняемого файла.

# **3 РЕЗУЛЬТАТЫ ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ             РАБОТЫ**

В ходе выполнения лабораторной была разработана программа, которая позволяет шифровать и дешифровать текст по азбуке Морзе. Пользователю необходимо ввести данные в виде последовательного набора символов из азбуки Морзе, или в виде последовательного набора символ из латинского алфавита. В зависимости от введенных данных пользователь может использовать функцию char, для перевода символов из азбуки Морзе в латиницу, или функцию morse для обратного действия. Результат работы программы представлен на рисунке 3.1.

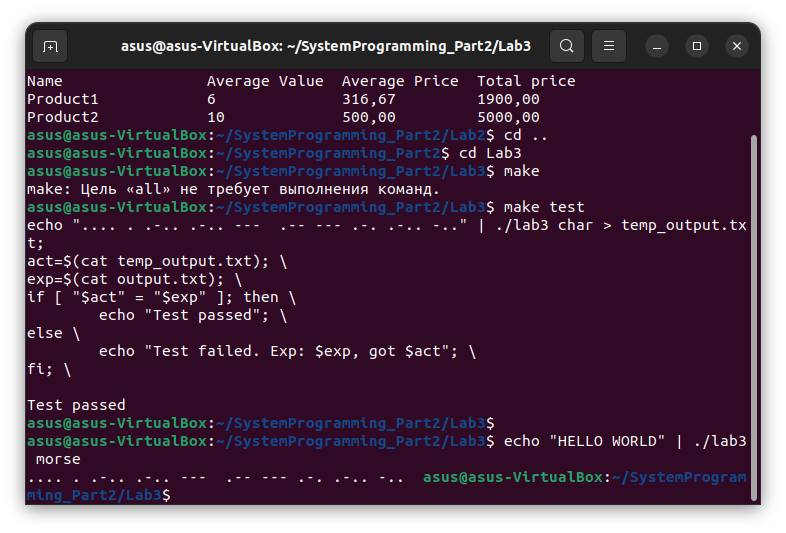


Рисунок 3.1 – Результат работы программы при использовании ввода данных

При помощи утилиты make была организована возможность тестирования. Ожидаемые результаты и фактически полученные результаты должны быть занесены в определенные файлы перед тестированием. Результат тестирования представлен на рисунке 3.2.

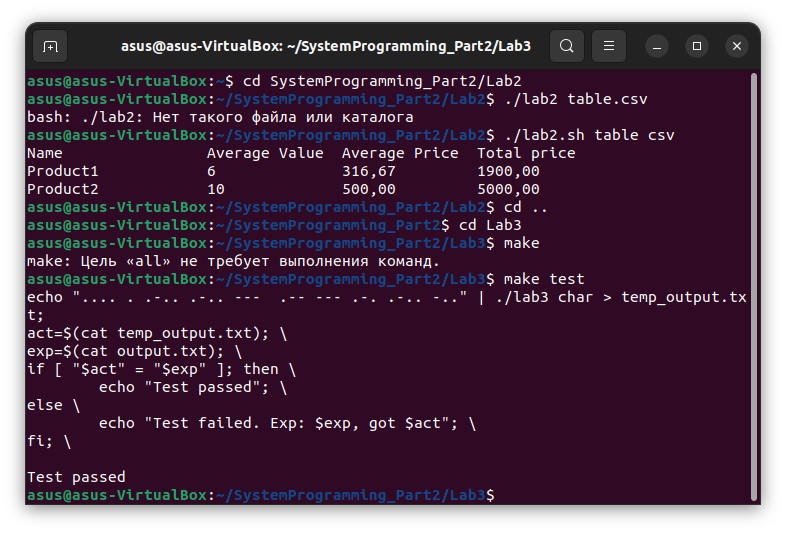


Рисунок 3.2 – Результат тестирования исполняемого файла

Таким образом, в ходе данной лабораторной работы была реализована программа шифрующая и дешифрующая символы по азбуке 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 шифрование и дешифрование символов по азбуке 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: 08.02.2024.

[2] Компилятор GCC [Электронный ресурс]. – Режим доступа: https://metanit.com/c/tutorial/1.3.php. – Дата доступа: 08.02.2024.

[3] Makefile [Электронный ресурс]. – Режим доступа:   
https://habr.com/ru/articles/155201/. – Дата доступа: 08.02.2024.

## **ПРИЛОЖЕНИЕ А**

## **(обязательное)**

## **Листинг исходного кода**

Листинг 1 – Программный код главной функции

#include <stdio.h>

#include <string.h>

#include <ctype.h>

#include "to\_char.h"

#include "to\_morse.h"

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

if (argc != 2) {

fprintf(stderr, "Usage: %s [morse/char]\n", argv[0]);

return 1;

}

if (strcmp(argv[1], "morse") == 0) {

int ch;

while ((ch = getchar()) != EOF) {

to\_morse(ch);

}

} else if (strcmp(argv[1], "char") == 0) {

char morse[50];

int index = 0;

int ch;

while ((ch = getchar()) != EOF) {

if (ch == ' ' || ch == '\n') {

morse[index] = '\0';

to\_char(morse);

index = 0;

} else {

morse[index++] = ch;

}

}

} else {

fprintf(stderr, "Unknown option: %s\n", argv[1]);

return 1;

}

return 0;

}

Листинг 2 – Программный код to\_char.c

#include <stdio.h>

#include <string.h>

#include <ctype.h>

void to\_char(const char \*morse) {

if (strcmp(morse, ".-") == 0) printf("A");

else if (strcmp(morse, "-...") == 0) printf("B");

else if (strcmp(morse, "-.-.") == 0) printf("C");

else if (strcmp(morse, "-..") == 0) printf("D");

else if (strcmp(morse, ".") == 0) printf("E");

else if (strcmp(morse, "..-.") == 0) printf("F");

else if (strcmp(morse, "--.") == 0) printf("G");

else if (strcmp(morse, "....") == 0) printf("H");

else if (strcmp(morse, "..") == 0) printf("I");

else if (strcmp(morse, ".---") == 0) printf("J");

else if (strcmp(morse, "-.-") == 0) printf("K");

else if (strcmp(morse, ".-..") == 0) printf("L");

else if (strcmp(morse, "--") == 0) printf("M");

else if (strcmp(morse, "-.") == 0) printf("N");

else if (strcmp(morse, "---") == 0) printf("O");

else if (strcmp(morse, ".--.") == 0) printf("P");

else if (strcmp(morse, "--.-") == 0) printf("Q");

else if (strcmp(morse, ".-.") == 0) printf("R");

else if (strcmp(morse, "...") == 0) printf("S");

else if (strcmp(morse, "-") == 0) printf("T");

else if (strcmp(morse, "..-") == 0) printf("U");

else if (strcmp(morse, "...-") == 0) printf("V");

else if (strcmp(morse, ".--") == 0) printf("W");

else if (strcmp(morse, "-..-") == 0) printf("X");

else if (strcmp(morse, "-.--") == 0) printf("Y");

else if (strcmp(morse, "--..") == 0) printf("Z");

else if (strcmp(morse, "-----") == 0) printf("0");

else if (strcmp(morse, ".----") == 0) printf("1");

else if (strcmp(morse, "..---") == 0) printf("2");

else if (strcmp(morse, "...--") == 0) printf("3");

else if (strcmp(morse, "....-") == 0) printf("4");

else if (strcmp(morse, ".....") == 0) printf("5");

else if (strcmp(morse, "-....") == 0) printf("6");

else if (strcmp(morse, "--...") == 0) printf("7");

else if (strcmp(morse, "---..") == 0) printf("8");

else if (strcmp(morse, "----.") == 0) printf("9");

else printf(" ");

}

Листинг 3 – Программный код to\_char.h

#ifndef TO\_CHAR\_H

#define TO\_CHAR\_H

void to\_char(const char \*morse);

#endif

Листинг 4 – Программный код to\_morse.c

#include <stdio.h>

#include <string.h>

#include <ctype.h>

void to\_morse(char s) {

char symbol = toupper(s);

if (symbol == 'A') printf(".- ");

else if (symbol == 'B') printf("-... ");

else if (symbol == 'C') printf("-.-. ");

else if (symbol == 'D') printf("-.. ");

else if (symbol == 'E') printf(". ");

else if (symbol == 'F') printf("..-. ");

else if (symbol == 'G') printf("--. ");

else if (symbol == 'H') printf(".... ");

else if (symbol == 'I') printf(".. ");

else if (symbol == 'J') printf(".--- ");

else if (symbol == 'K') printf("-.- ");

else if (symbol == 'L') printf(".-.. ");

else if (symbol == 'M') printf("-- ");

else if (symbol == 'N') printf("-. ");

else if (symbol == 'O') printf("--- ");

else if (symbol == 'P') printf(".--. ");

else if (symbol == 'Q') printf("--.- ");

else if (symbol == 'R') printf(".-. ");

else if (symbol == 'S') printf("... ");

else if (symbol == 'T') printf("- ");

else if (symbol == 'U') printf("..- ");

else if (symbol == 'V') printf("...- ");

else if (symbol == 'W') printf(".-- ");

else if (symbol == 'X') printf("-..- ");

else if (symbol == 'Y') printf("-.-- ");

else if (symbol == 'Z') printf("--.. ");

else if (symbol == '0') printf("----- ");

else if (symbol == '1') printf(".---- ");

else if (symbol == '2') printf("..--- ");

else if (symbol == '3') printf("...-- ");

else if (symbol == '4') printf("....- ");

else if (symbol == '5') printf("..... ");

else if (symbol == '6') printf("-.... ");

else if (symbol == '7') printf("--... ");

else if (symbol == '8') printf("---.. ");

else if (symbol == '9') printf("----. ");

else {

printf(" ");

}

}

Листинг 5 – Программный код to\_morse.h

#ifndef TO\_MORSE\_H

#define TO\_MORSE\_H

void to\_morse(char s);

#endif

Листинг 6 – Программный код makefile

CC=gcc

CFLAGS=-Wall

.PHONY: all test clean

all: lab3

lab3: lab3.o to\_char.o to\_morse.o

$(CC) $(CFLAGS) lab3.o to\_char.o to\_morse.o -o lab3

lab3.o: lab3.c

$(CC) $(CFLAGS) -c lab3.c -o lab3.o

to\_char.o: to\_char.c

$(CC) $(CFLAGS) -c to\_char.c -o to\_char.o

to\_morse.o: to\_morse.c

$(CC) $(CFLAGS) -c to\_morse.c -o to\_morse.o

test: lab3 input.txt output.txt

echo ".... . .-.. .-.. --- .-- --- .-. .-.. -.." | ./lab3 char > temp\_output.txt;

act=$$(cat temp\_output.txt); \

exp=$$(cat output.txt); \

if [ "$$act" = "$$exp" ]; then \

echo "Test passed"; \

else \

echo "Test failed. Exp: $$exp, got $$act"; \

fi; \

clean:

rm -f lab3 lab3.o to\_char.o to\_morse.o