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             В. А. Звягинцева

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**

Компилятор gcc распространяется по лицензии GNU, Фондом свободного программного обеспечения, для nix-подобных ОС и является C\C++ компилятором, который управляется с помощью командной строки.

Программа gcc, запускаемая из командной строки, представляет собой надстройку над группой компиляторов. В зависимости от расширений имен файлов, передаваемых в качестве параметров, и дополнительных опций, gcc запускает необходимые препроцессоры, компиляторы, линкеры.

Файлы с расширением .cc или .C рассматриваются, как файлы на языке C++, файлы с расширением .c как программы на языке C, а файлы c расширением .o считаются объектными. [1]

Makefile – это файл с инструкциями для утилиты make, которая нужна для автоматической сборки проекта.

Компилятор превращает набор кода в объектные файлы, или модули. С помощью линковщика они соединяет все воедино с учетом зависимостей и связей между исходниками. Результат – готовый исполняемый файл.

Утилита make автоматизирует процесс – разработчику достаточно набрать одну команду в консоли. Она преобразует исходный код в модули, а их – в исполняемые файлы, которые можно запустить. Так создаются библиотеки и разнообразные программные продукты. Makefile показывает программе правила, по которым нужно выполнять преобразования.

Make работает на Linux и системах на базе Unix – для этих ОС утилита считается основным средством сборки программ.

Чаще всего Makefile применяется для:

– выдачи инструкций программе make, которая автоматизирует работу по сборке;

– возможности собрать программный проект и превратить его в исполняемый файл в одно касание;

– правильной обработки зависимостей и сущностей в коде;

– корректной компиляции, преобразования кода;

– ускорения и упрощения программирования, в первую очередь на языках C и C++. [2]

# **3 РЕЗУЛЬТАТЫ ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ             РАБОТЫ**

В ходе выполнения лабораторной работы была разработана программа перевода текста в азбуку Морзе или из нее. Также был создан Makefile для управления компиляцией и сборкой этого проекта. После этого была запущена утилита make (рисунок 3.1).

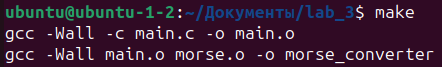


Рисунок 3.1 – Результат выполнения команды make

При запуске файла morse\_converter, пользователю предлагается ввести режим для использования. При выборе первого режима, текст, введенный пользователем будет конвертирован в азбуку Морзе (рисунок 3.2).

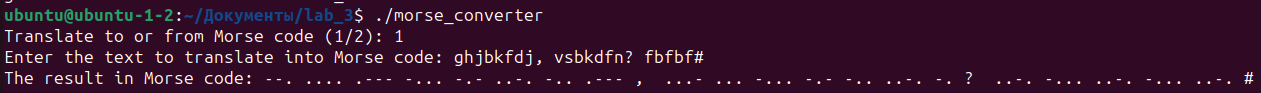


Рисунок 3.2 – Перевод текста в азбуку Морзе

При выборе второго режима, пользователю предлагается ввести код на азбуке Морзе, который будет переведен в текст (рисунок 3.3).

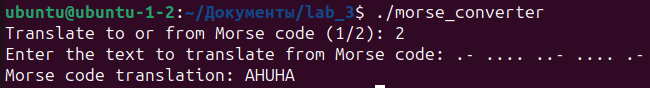


Рисунок 3.3 – Перевод текста из азбуки Морзе

# **ВЫВОДЫ**

В ходе лабораторной работы были изучены среда программирования и основные инструменты: компилятор/сборщик («коллекция компиляторов») gcc, управление обработкой проекта make (и язык makefile), библиотеки и т.д. А также была разработана программа перевода текста в азбуку Морзе и из нее.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Компилятор GCC [Электронный ресурс]. – Режим доступа: https://parallel.uran.ru/book/export/html/25. – Дата доступа: 19.02.2024.
2. Makefile [Электронный ресурс]. – Режим доступа: https://blog.skillfactory.ru/glossary/makefile/. – Дата доступа: 19.02.2024.

## **ПРИЛОЖЕНИЕ А**

## **(обязательное)**

## **Листинг исходного кода**

Листинг А.1 – main.c

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <locale.h>

#include "morse.h"

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

setlocale(0, "");

char input[100];

int choice;

printf("Translate to or from Morse code (1/2): ");

scanf("%d", &choice);

getchar();

switch (choice) {

case 1:

printf("Enter the text to translate into Morse code: ");

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

convert\_to\_morse(input);

break;

case 2:

printf("Enter the text to translate from Morse code: ");

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

convert\_from\_morse(input);

break;

default:

printf("Incorrect input. Choose 1 or 2.");

break;

}

return 0;

}

Листинг А.2 – morse.h

#ifndef MORSE\_H

#define MORSE\_H

void convert\_to\_morse(const char\* input);

void convert\_from\_morse(const char\* input);

#endif

Листинг А.3 – morse.c

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <string.h>

#include <ctype.h>

#include "morse.h"

typedef struct {

const char\* letter;

const char\* morse;

} MorseMap;

const MorseMap morse\_dict[] = {

{"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", "--.."}

};

void convert\_to\_morse(const char\* input) {

printf("The result in Morse code: ");

for (int i = 0; i < strlen(input); i++) {

char c = toupper(input[i]);

if (c >= 'A' && c <= 'Z') {

int found = 0;

for (int j = 0; j < sizeof(morse\_dict) / sizeof(MorseMap); j++)

if (morse\_dict[j].letter[0] == c) {

printf("%s ", morse\_dict[j].morse);

found = 1;

break;

}

if (!found) printf("%c ", c);

}

else if (c == ' ') printf(" ");

else printf("%c ", c);

}

printf("\n");

}

void convert\_from\_morse(const char\* input) {

printf("Morse code translation: ");

char\* token = strtok((char\*)input, " \t\n\r");

while (token != NULL) {

int found = 0;

for (int i = 0; i < sizeof(morse\_dict) / sizeof(MorseMap); i++)

if (strcmp(token, morse\_dict[i].morse) == 0) {

printf("%s", morse\_dict[i].letter);

found = 1;

break;

}

if (!found) printf("This isn\’t Morse code.");

token = strtok(NULL, " \t\n\r");

}

printf('\n');

}