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:

студент гр. 153503

Татаринов В.В.

Проверил:

Гриценко Н.Ю.

Минск 2024

**СОДЕРЖАНИЕ**

[1 Цель работы 3](#_Toc146836467)

[2 Теоретические сведения 4](#_Toc146836468)

[3 Полученные результаты 5](#_Toc146836469)

[Выводы 6](#_Toc146836470)

[Список использованных источников 7](#_Toc146836471)

[Приложение А (обязательное) листинг кода 8](#_Toc146836472)

# **1 ЦЕЛЬ РАБОТЫ**

Изучить среды программирования и основные инструменты: компилятор/сборщик («коллекция компиляторов») *gcc*, управление обработкой проекта *make* (и язык *makefile*). Практически использовать основные библиотеки и системные вызовы: ввод-вывод и работа с файлами, обработка текста.

Написать программу, представляющую собой инвертирующий фильтр для символов. Создать *makefile* для управления обработкой проекта, собрать и протестировать исполняемый файл.

**2 ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ** 

*GСС* – это свободно доступный оптимизирующий компилятор для языков *C*, *C++*. 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. Опция *-c* означает «только компиляция». Опция *-o* задает имя исполняемого файла.

В процессе компоновки очень часто приходится использовать библиотеки. Библиотекой называют набор объектных файлов, сгруппированных в единый файл и проиндексированных. Когда команда компоновки обнаруживает некоторую библиотеку в списке объектных файлов для компоновки, она проверяет, содержат ли уже скомпонованные объектные файлы вызовы для функций, определенных в одном из файлов библиотек [1].

*Makefile* – это файл с инструкциями для утилиты make, которая нужна для автоматической сборки проекта*.* Его обычно помещают в корень проекта. Он выступает и как документация, и как исполняемый код. *Makefile* скрывает за собой детали реализации и структурированно раскладывает команды, а утилита *make* запускает их из того *makefile*, который находится в текущей директории.

Компоненты, из которых состоит *makefile*:

* Цель. Она представляет собой файл, который программа *make* должна сгенерировать после запуска. Это может быть модуль или исполняемый файл. В *makefile* указываются его будущее имя и расширение.
* Зависимости. Это начальные условия, файлы и действия, от которых зависит цель. Их может не быть – тогда *make* начнет сразу выполнять команды ниже. Но если зависимости есть, утилита сначала проверит их и выполнит все связанные с ними действия. Только после этого она перейдет к цели.
* Действия. Это команды для консоли, системные вызовы и так далее. Они описываются после цели и зависимостей. Когда начальные условия будут выполнены, утилита *make* перейдет к действиям, выполнит их, а результат запишет в цель [2].

# **3 ПОЛУЧЕННЫЕ РЕЗУЛЬТАТЫ**

В результате выполнения лабораторной работы был написана программа-фильтр для инвертирования символов.

Программа реализует инверсию символов в каждой строке потока, при этом порядок самих строк не изменяется. Длина строк ограничена 100 символами. Программа считывает данные из исходного файла, производит обработку и записывает полученный результат в другой файл. Исходный текст для обработки содержится в файле *input.txt* (рисунок 1).

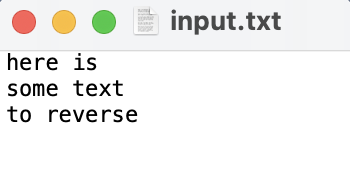


Рисунок 1 – Файл с исходным текстом для обработки

Обработанный текст с инвертированными символами сохраняется в файл *output.txt* (рисунок 2).

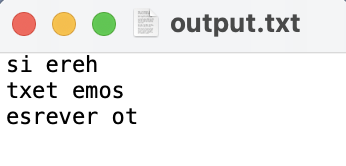


Рисунок 2 – Файл c обработанным текстом

# **ВЫВОДЫ**

В результате выполнения лабораторной работы были изучены среды программирования и основные инструменты: компилятор/сборщик («коллекция компиляторов») *gcc*, управление обработкой проекта *make* (и язык *makefile*). Практически использовались системные вызовы: ввод-вывод и работа с файлами, обработка текста.

Написана программа, представляющая собой инвертирующий фильтр для символов. Создан *makefile* для управления обработкой проекта, собран и протестирован исполняемый файл.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

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

# **ПРИЛОЖЕНИЕ А (обязательное) Листинг кода**

Листинг 1 **–** Файл *invert.c*:

#include <stdio.h>

#include <string.h>

#include "reverse.h"

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

char line[100];

FILE \*output\_file = fopen(argv[1], "w");

if (output\_file == NULL) {

perror("Error opening output file");

return 1;

}

while (fgets(line, sizeof(line), stdin) != NULL) {

char \*newline = strchr(line, '\n');

if (newline != NULL) {

\*newline = '\0';

}

reverseString(line);

fprintf(output\_file, "%s\n", line);

}

fclose(output\_file);

return 0;

}

Листинг 2 **–** Файл *reverse.c*:

#include "reverse.h"

#include <string.h>

void reverseString(char \*str) {

int length = strlen(str);

int start = 0;

int end = length - 1;

while (start < end) {

char temp = str[start];

str[start] = str[end];

str[end] = temp;

start++;

end--;

}

}

Листинг 3 **–** Файл *reverse.h*:

#ifndef REVERSE\_H

#define REVERSE\_H

void reverseString(char \*str);

#endif

Листинг 4 **–** Файл *makefile*:

CMP = gcc

TARGET = invert

all: $(TARGET)

$(TARGET): invert.o reverse.o

$(CMP) -o $@ $^

invert.o: invert.c reverse.h

$(CMP) -c $<

reverse.o: reverse.c reverse.h

$(CMP) -c $<

clean:

rm -f \*.o