1. Министерство образования и науки Российской Федерации
2. Санкт-Петербургский Политехнический Университет Петра Великого
3. —
4. Институт кибербезопасности и защиты информации

**ЛАБОРАТОРНАЯ РАБОТА № 1**

1. «Мониторинг событий в процессе»
2. по дисциплине «технологии разработки современного программного обеспечения»
3. Выполнил
4. студент гр. 5131001/20503 Черникова В. М.

<*подпись*>

1. Преподаватель
2. Завадский Е. В.

<*подпись*>

1. Санкт-Петербург
2. 2024

# ЦЕЛЬ РАБОТЫ

Целью работы – изучение этапов сборки программы с учетом особенностей статической и динамической компоновки и практическое освоение средств автоматизации данного процесса.

# ЗАДАЧИ

1. Изучить синтаксис файла конфигурации (Makefile) системы сборки make.

2. Разработать программу, которая формирует PNG-файла с нарисованным в нем текстовым сообщением с помощью шрифта, загруженного из файла \*.ttf с использованием библиотеки zlib [1], libpng [2], freetype [3].

3. Разработать Makefile для сборки трех версий программы:

⎯ со статической компоновкой трех библиотек;

⎯ с динамическая загрузка всех трех библиотек в начале работы;

⎯ в виде blob-файла, загружаемого с диска при помощи загрузчика elf-loader.

# Ход работы

Перед тем как начать программировать, надо собрать библиотеки командой make.

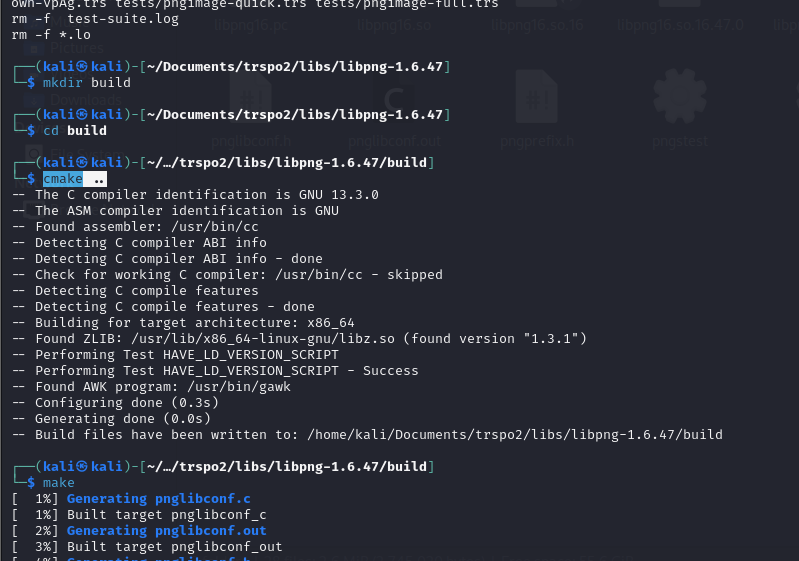
**

Рисунок 1 - Например, libpng

Далее структурировать репозиторий.

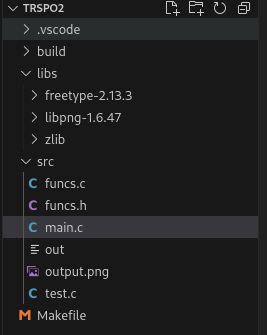
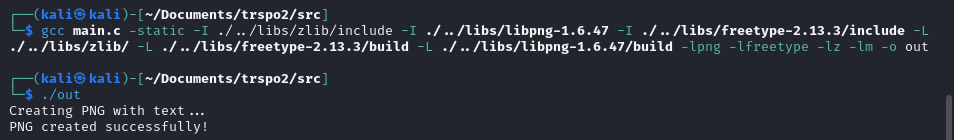


Рисунок 2 - структура директории

Потом была сделана тестовая программа, которая выводит текст в картинку. Программа собрана и работает :

*Рисунок 3 – запуск тестово*



*Рисунок 4 - картинка*

# Компилятор и флаги

CC = gcc

CFLAGS = -I ./src -I ./libs/zlib/include -I ./libs/libpng -I ./libs/libpng/include -I ./libs/freetype/include

LDFLAGS = -L ./libs/zlib/lib -L ./libs/libpng/build -L ./libs/freetype/build

LDLIBS = -lpng -lfreetype -lz -lm

# Директории

SRC\_DIR = src

BUILD\_DIR = build

OBJ\_DIR = $(BUILD\_DIR)/obj

BIN\_DIR = $(BUILD\_DIR)/bin

# Цели для разных версий

TARGET=BIN\_DIR

# Исходные файлы и объекты

SRCS = $(wildcard $(SRC\_DIR)/\*.c)

TARGET\_STATIC = $(BIN\_DIR)/app-static

TARGET\_DYNAMIC = $(BIN\_DIR)/app-dynamic

TARGET\_BLOB = $(BIN\_DIR)/app-blob

# Основная цель

all: static dynamic #blob

# Статическая версия (полностью статически линкованная)

static: $(TARGET\_STATIC)

# Сборка исполняемого файла

$(TARGET\_STATIC): $(OBJS) | $(STATIC\_DIR)

$(CC) src/main.c -static $(CFLAGS) $(LDFLAGS) $^ $(LDLIBS) -o $@

# Создание директорий

$(BIN\_DIR):

mkdir -p $@

# Очистка

clean:

rm -rf $(BUILD\_DIR)

# Пересборка

rebuild: clean all

.PHONY: all clean rebuild

Проверка, что файл статической компоновки:

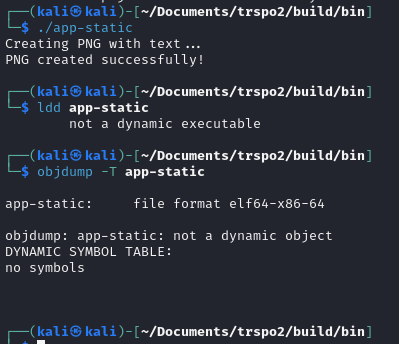


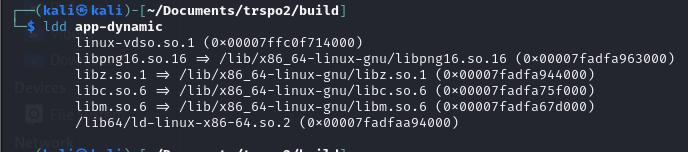
Рисунок - проверка на зависимости static

Часть кода - проверка на зависимости и запуск программы

objdump -T app-dynamic

ldd app-dynamic

./app-dynamic /usr/share/fonts/truetype/freefont/FreeSans.ttf output.png $'Hello, World!!!))' 36



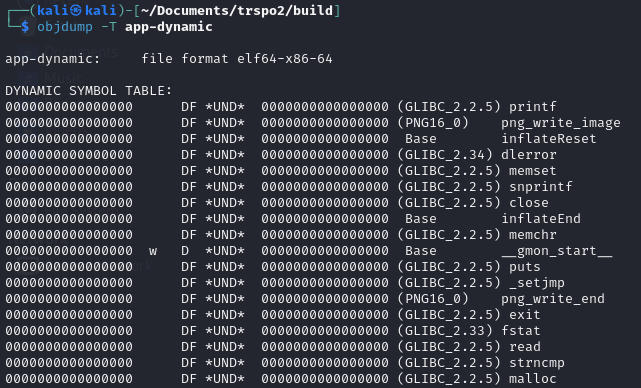
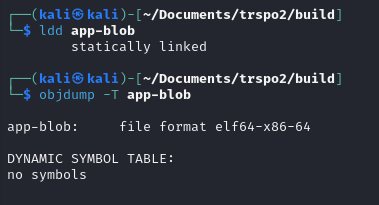


Рисунок - проверка на зависимости dynamic

Blob:



*Рисунок 7 - проверка на зависимости blob*

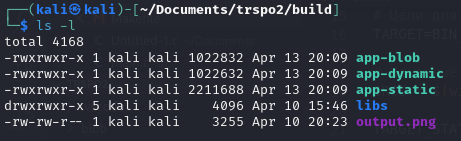


Рисунок - исполняемые файлы

# Вывод

В результате лабораторной работы было изучены этапы сборки программы с учетом особенностей статической и динамической компоновки и практическое освоение средств автоматизации данного процесса