Правительство Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»
(НИУ ВШЭ)

Московский институт электроники и математики им. А.Н. Тихонова

ОТЧЕТ
О ПРАКТИЧЕСКОЙ РАБОТЕ № 2.2
по дисциплине «Архитектура компьютера и системное программирование»

ПРАКТИЧЕСКОЕ ЗНАКОМСТВО С УТИЛИТОЙ GNU MAKE ДЛЯ СБОРКИ ПРОЕКТОВ В UNIX

Студент гр. БИБ232
Николаев М.А

Руководитель
Абдуллаев Ариф Биннатович
«14» декабря 2025 г.

Москва 2025

СОДЕРЖАНИЕ

1 Задание на практическую работу

Цели настоящей практической работы заключаются в следующем

В рамках практической работы необходимо выполнить следующее:

1. Изучить процесс компиляции программ на языке C (C++) в операционных системах Unix и приобрести практические навыки работы с утилитой GNU make для создания и сборки проектов.

2. Разобраться в особенностях применения утилиты make при разработке проектов на языке C (C++) в окружении Unix и получить опыт работы с GNU make в процессе создания и компиляции проектов.

Задание настоящей практической работы заключаются в следующем:

1 Ознакомиться с теоретическим материалом
2 Используйте в качестве основы лабораторную работу «Комбинирование разноязыковых модулей» для данной работы.
3 Для автоматизации сборки проекта утилитой make создайте make-файл.
4 Выполните программу (скомпилировать, при необходимости выполнить отладку).
5 Покажите, что при изменении одного исходного файла и последующем вызове make будут исполнены только необходимые команды компиляции (неизмененные файлы перекомпилированы не будут) и изменены атрибуты и/или размер объектных файлов (файлы с расширением .o).

2 Ход работы

2.1 Создание и написание make файла

Для выполнения задания я написал следующий make файл (прикреплен в Приложении 1). Обозначу что использовалось из методического материала.

OBJ := main.o pr1_2.o

В данной строке показано создание переменных для упрощенного вызова и изменения значений.

all: $(TARGET)

В данной строке показано использование переменных, зависимостей, а так же использование первой команды - вместо написания «make all» появляется возможность написать «make» и иметь такой же результат.

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

В данной строке показано использование переменных, так же как и использование автоматических переменных. «$<» получает значение первое переменной из зависимостей, «$@» получает значение имени цели.

.PHONY: all clean

В данной строке показано использование команды «.PHONY», которая определяет список целей Makefile, не являющихся именами файлов. Это необходимо, чтобы make всегда выполнял эти цели при вызове, даже если в директории появятся файлы с такими именами (например, файл «clean»).

2.2 Демонстрация работоспособности

Для начала покажу эквивалентность команд «make» и «make all» - на рисунках 1 и 2 соответственно.

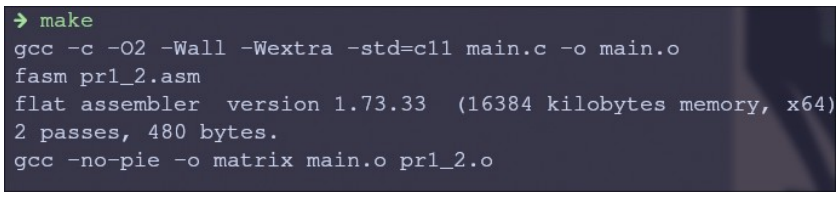

Рисунок 1 - Запуск «make» без дополнительных команд

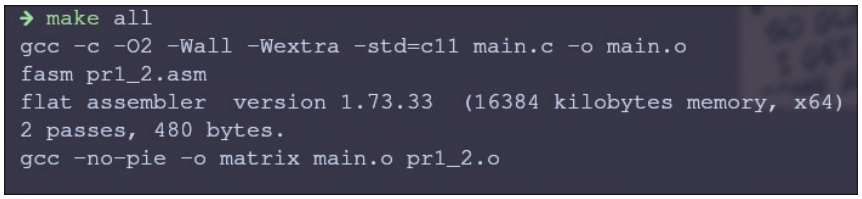

Рисунок 2 - Запуск «make all»

Демонстрация работы команды «make clean»

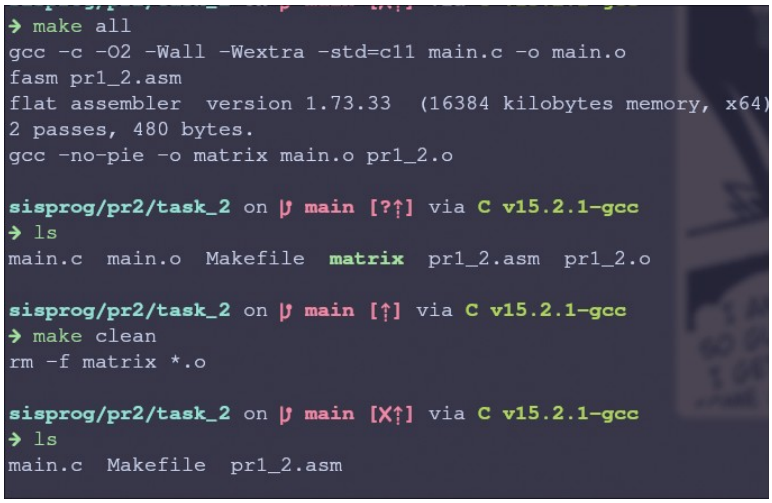

Рисунок 3 — Демонстрация работы «make clean»

Демонстрация отсутствия компиляции неизмененных файлов:

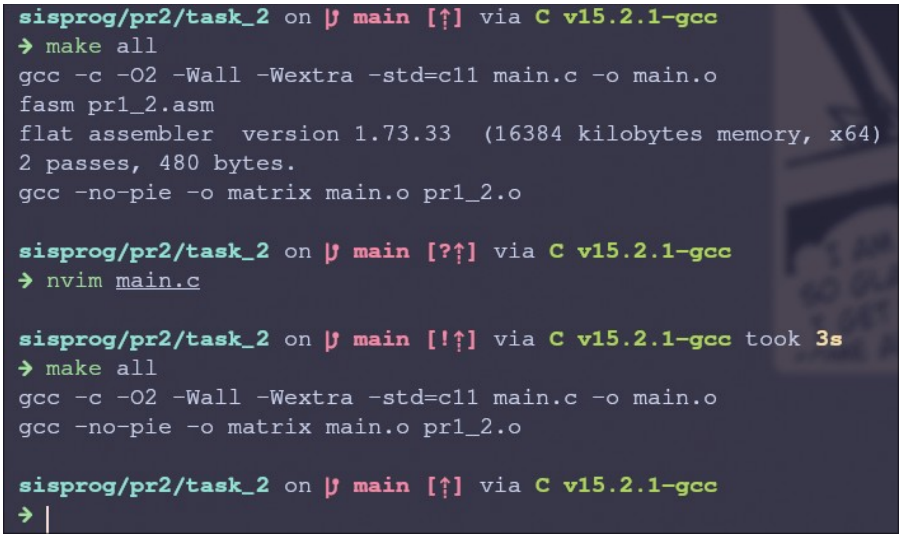

Рисунок 4 - Демонстрация отсутствия компиляции неизмененных файлов (pr1_2.asm) не был скомпилирован повторно.

3. Выводы о проделанной работе

В ходе практической работы были изучены принципы сборки программ с помощью утилиты make в Unix-среде. Был разработан make файл для автоматизации сборки проекта, включающего C- и ассемблерный модули. Получены практические навыки по использованию переменных, зависимостей и автоматических переменных в make. Также была подтверждена корректная работа системы зависимостей: при изменении одного исходного файла пересобирается только он, что повышает эффективность разработки.

Приложение 1 — код make файла

```
TARGET := matrix
SRC := main.c
ASM := pr1_2.asm
OBJ := main.o pr1_2.o

CC := gcc
ASM_CC := fasm

CFLAGS := -c -O2 -Wall -Wextra -std=c11
LDFLAGS := -no-pie

all: $(TARGET)

$(TARGET): $(OBJ)
	$(CC) $(LDFLAGS) -o $@ $^

main.o: $(SRC)
	$(CC) $(CFLAGS) $< -o $@

pr1_2.o: $(ASM)
	$(ASM_CC) $<

clean:
	rm -f $(TARGET) *.o

.PHONY: all clean
```