**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ**

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ   
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

**«РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ ПЕДАГОГИЧЕСКИЙ   
УНИВЕРСИТЕТ им. А. И. Герцена»**

ИНСТИТУТ КОМПЬЮТЕНЫХ НАУК И ТЕХНОЛОГИЧЕСКОГО ОБРАЗОВАНИЯ

**КАФЕДРА КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ И ЭЛЕКТРОННОГО ОБУЧЕНИЯ**

**ОТЧЕТ**

**По лабораторной работе №2**

**«ПРАКТИЧЕСКОЕ ЗНАКОМСТВО СО СТАНДАРТНОЙ УТИЛИТОЙ GNU MAKE ДЛЯ ПОСТРОЕНИЯ ПРОЕКТОВ В ОС UNIX»**

**по курсу «Операционные системы»**

**Студенки 2 курса**

**Волковой А. И.**

**Преподаватель:**

**профессор кафедры**

**компьютерных**

**технологий и**

**электронного**

**обучения**

**Абрамян Г. В.**

**Санкт-Петербург**

**2017 г.**

**Цель работы**

Ознакомиться с техникой компиляции программ на языке программирования C/C++в среде ОС семейства Unix, а также получить практические навыки использования утилиты GNU Make для сборки проекта.

**Задание**

Изучить особенности работы с утилитой make при создании проекта на языке С (С++) в ОС Unix, а также получить практические навыки использования утилиты GNU make при создании и сборке проекта

**Выполнение работы**

**работы**

**2. Создать простейшую программу на C/C++ из как минимум двух исходных файлов.**

**Файл source1.cpp:**

#include <iostream>

#include <stdio.h>

#include <conio.h>

#include <math.h>

#include "source2.h"

using namespace std;

int main(){

int n;

cout<<"Enter number \n";

cin>>n;

cout<<Multiply(n);

getch();

return 0;

}

**Файл source2.h:**

#pragma once

#include <iostream>

using namespace std;

int Multiply (int);

**Файл source2.cpp:**

int Multiply(int arg){

return arg \* arg;

}

3. **Для автоматизации сборки проекта утилитой Make создать простейший make - файл.**

iEdit: source1.o source2.o

g++ source1.o source2.o -o Programm

source1.o: source1.cpp

g++ -c source1.cpp

source2.o: source2.cpp

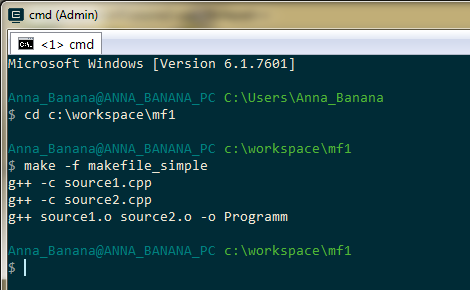
g++ -c source2.cpp

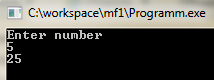
Флаги С++ - компилятора G ++:

• -c–компиляция без линковки, на выходе -для каждого исходного файла свой объектный файл.

• -o–file–поместить результат компиляции в файл file.

**4. Выполнить программу (скомпилировать, при необходимости отладить).**

****

****

**5. Показать, 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)**.

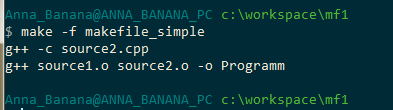
Внесем изменения в файл source2.cpp. Компилируется только измененный

файл.

int Multiply(int arg){

return arg \* arg\* arg;

}



**Задание 6. Создать make - файл с высоким уровнем автоматизированной обработки:**

• имя скомпилированной программы (выполняемый или бинарный

файл), флаги компиляции и имена каталогов с исходными файлами

и бинарными файлами (каталоги src, bin и т. п.) задаются с помощью

переменных в Makefile;

• зависимости исходных файлов на языке C/С++ и цели в make-файле должны формироваться динамически.

• наличие цели clean, удаляющей временные файлы;

Каталог проекта должен быть структурирован следующим образом:

• /src каталог с исходными файлами;

• /bin – каталог с бинарными файлами (скомпилированными);

• Makefile.

Makefile:

override compile\_flags := -pipe

source\_dir:= src

binar\_dir:= bin

search\_wildcards:=$(addsuffix /\*.cpp, $(source\_dir))

iEdit:$(notdir $(patsubst %.cpp, %.o,$(wildcard $(search\_wildcards))))

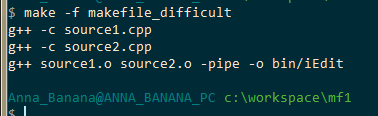
g++ $^ $(compile\_flags) -o $(addprefix $(addsuffix /,$(binar\_dir)),$@)

VPATH:=$(source\_dir)

%.o: %.cpp

g++ -c $<

**Результат работы**:

****

**Вывод**

В ходе данной лабораторной работы были изучены методы построения

программных проектов с использованием утилиты GNU Make.