**ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ**

Государственное образовательное учреждение

высшего профессионального образования

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**МОСКОВСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**СВЯЗИ И ИНФОРМАТИКИ**

Кафедра Математической кибернетики и информационных технологий

Лабораторная работа №2

на тему «Знакомство со стандартной утилитой GNU MAKE для построения проектов в ОС UNIX»

по дисциплине «Современные операционные системы»

Выполнила: студ. гр. ЗМПП1901

Иванова Е.В.

Проверила: И.о. зам. зав. кафедрой Мосева М.С.

Москва 2019

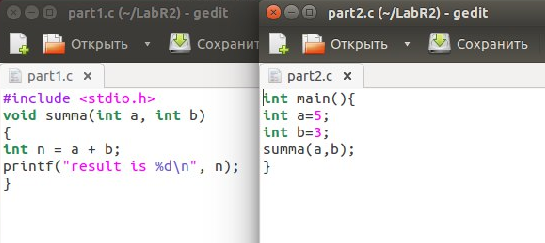
**Цель работы:** Ознакомиться с техникой компиляции программ на языке программирования C (C++) в среде ОС семейства Unix, а также получить практические навыки использования утилиты GNU make для сборки проекта.

**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.

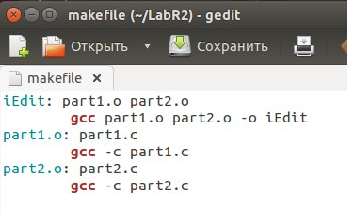
**Выполнение:**

**1. Используя любой текстовый редактор, создать простейшую программу на языке C (C++) с использованием как минимум двух исходных файлов (с программным кодом).**

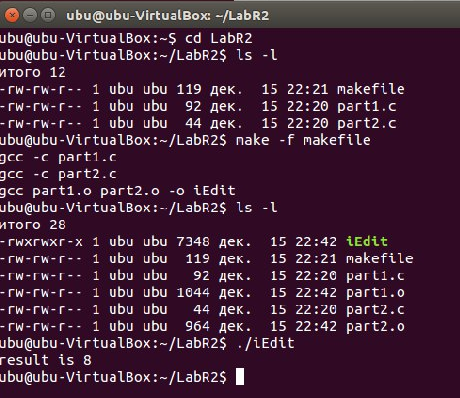
Создана программа, состоящая из двух исходных файлов, которая складывает две переменные.



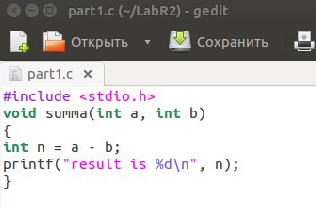
**2. Для автоматизации сборки проекта утилитой make создать makeфайл.**

****

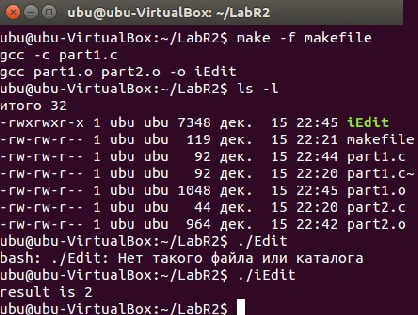
**3. Выполнить программу (скомпилировать, при необходимости отладить).**

****

**4. 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).**

****

При изменении одного исходного файла и последующем вызове make, происходит перекомпиляция только одного файла part1.o.

****

**5. Создать make-файл с высоким уровнем автоматизированной обработки исходных файлов программы согласно следующим условиям:**

**• 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 makeфайле должны формироваться динамически;**

**• наличие цели clean, удаляющей временные файлы;**

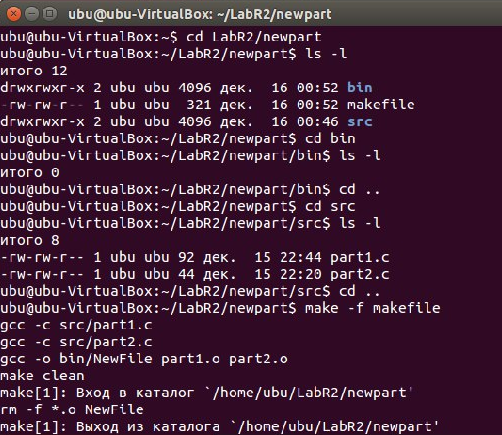
**• каталог проекта должен быть структурирован следующим образом: - src – каталог с исходными файлами; - bin – каталог с бинарными файлами (скомпилированными); - makefile.**

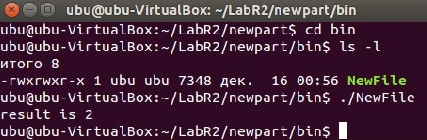
****

Автоматизированная обработка исходных файлов программы с помощью Make-файла стала более универсальна.

Функция wildcard получает список файлов с заданным шаблоном в выбранном каталоге, а функция patsubst заменяет заданную подстроку в заданной строке. Эти функции позволяют автоматически построить список объектных файлов программы.

Также с использованием переменных search \_wildcard s и VPATH стал автоматизирован поиск исходных файлов по нескольким директорием.

****

****

**Вывод:** в данной лабораторной работе были изучены особенности работы с утилитой make при создании проекта на языке С (С++) в ОС Unix, а также получены практические навыки использования утилиты GNU make при создании и сборке проекта. Утилита Make имеет большое количество возможностей по автоматизации сборки проекта и позволяет создать такой Make-файл, который универсален и при сборке разных программ потребует незначительное количество изменений в этом файле.