

РГПУ имени А.И. Герцена

Институт Компьютерных Наук и Технологического Образования

Информатика и вычислительная техника

Работу выполнил Цирулик Иван

Лабораторная работа №2.

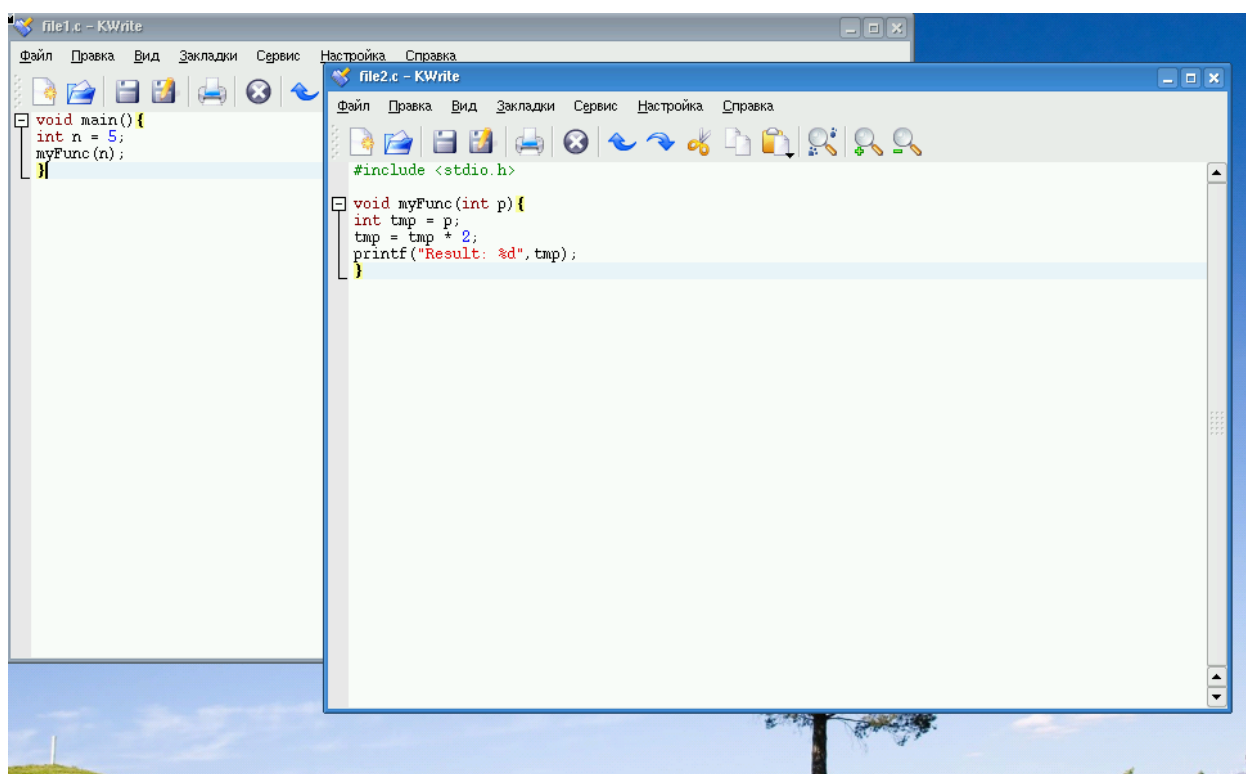
Практическое знакомство со стандартной утилитой GNU make для построения проектов в ОС UNIX

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 C (C++) в ОС Unix, а также получить практические навыки использования утилиты GNU make при создании и сборке проекта.

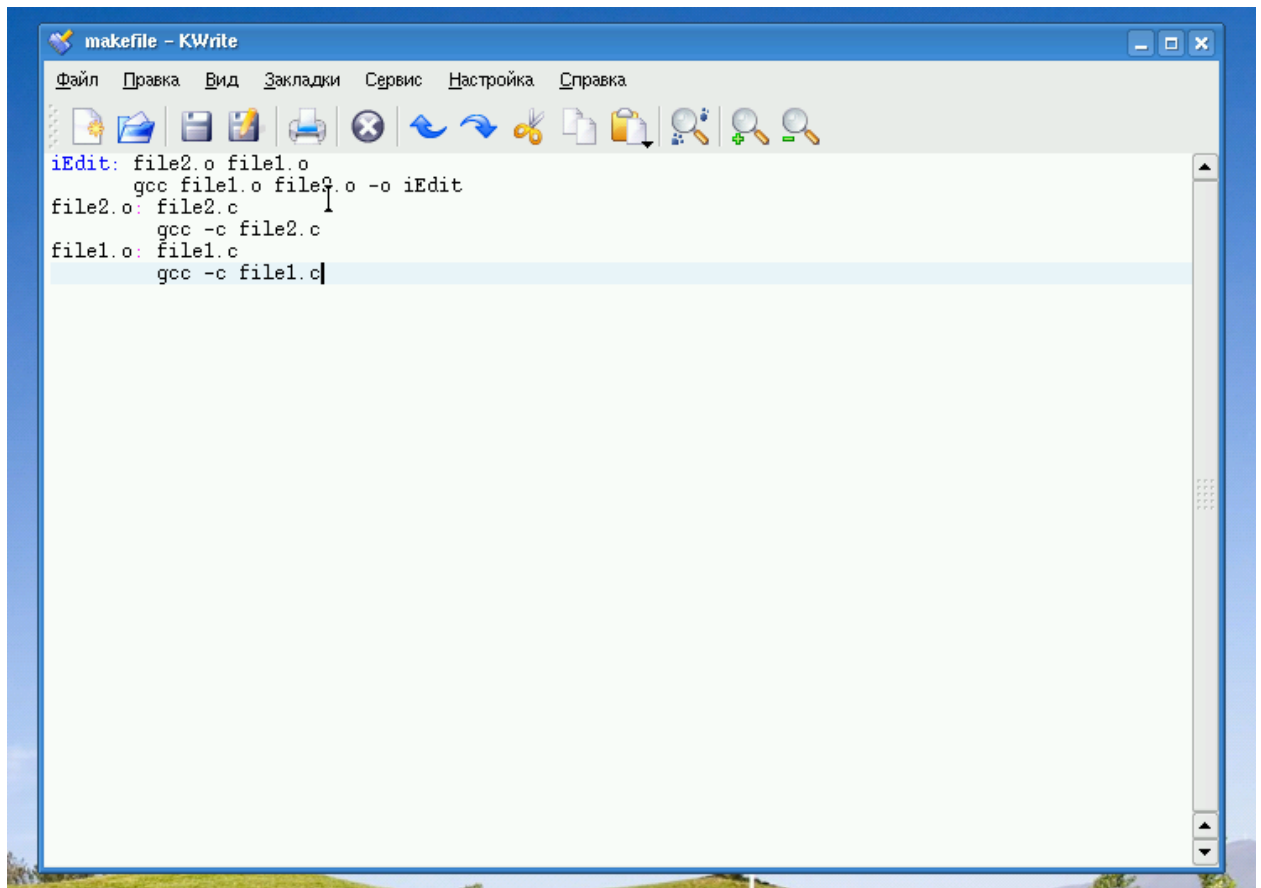
Задание 1.

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



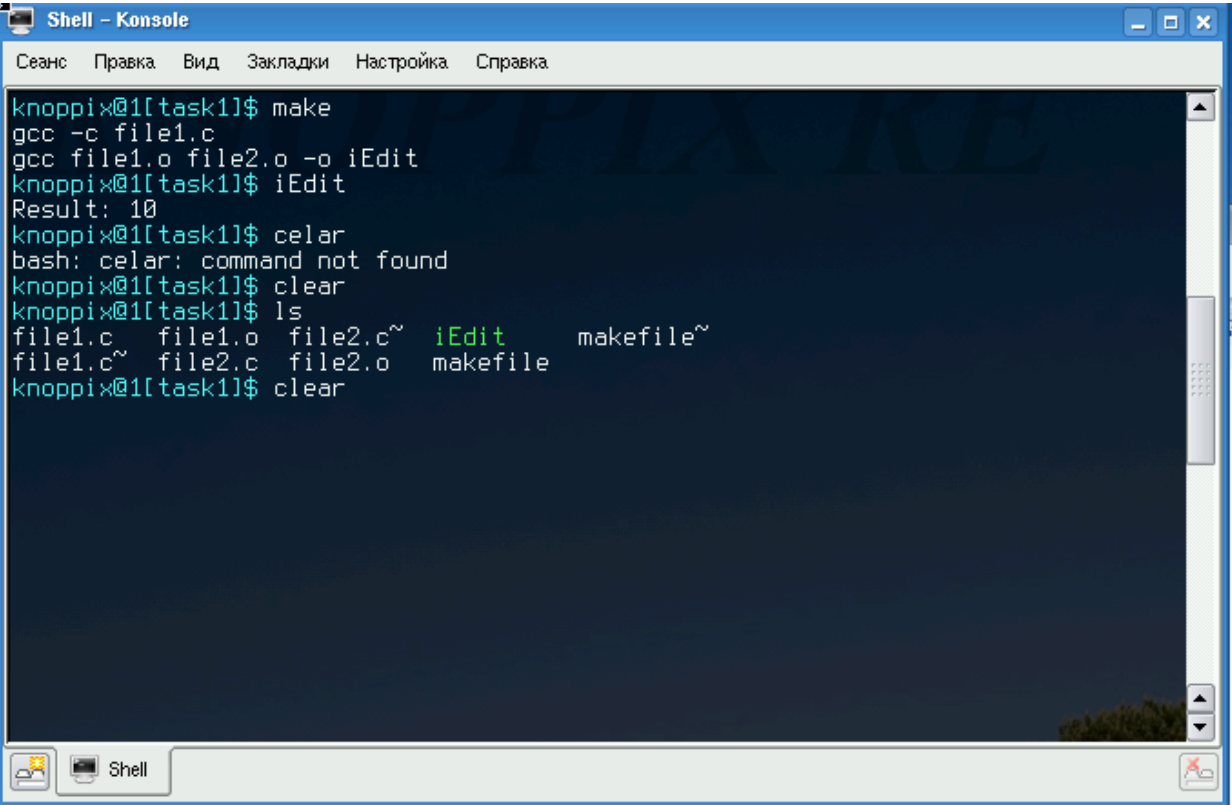
Задание 2.

Для автоматизации сборки проекта утилитой make создать make-файл.



Задание 3.

Выполнить программу (скомпилировать, при необходимости отладить).



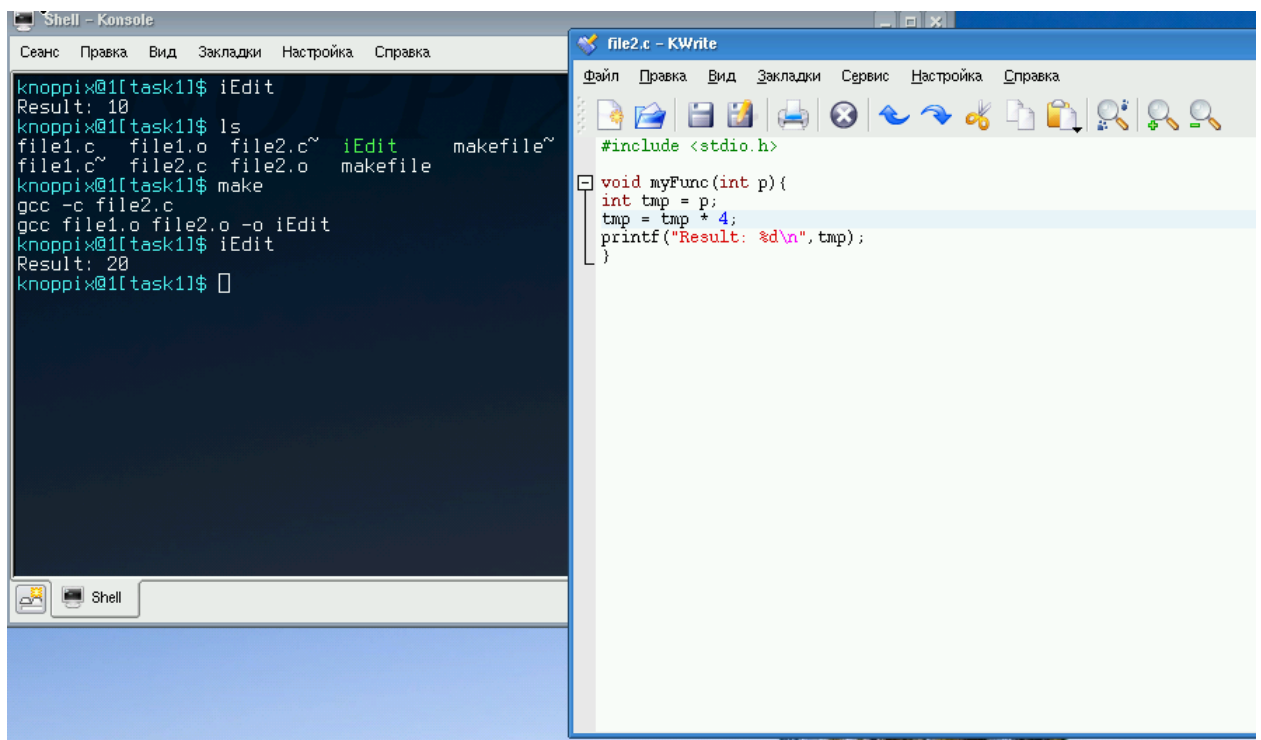
```
Shell - Konsole
Сеанс  Правка  Вид  Закладки  Настройка  Справка

knoppix@1[task1]$ make
gcc -c file1.c
gcc file1.o file2.o -o iEdit
knoppix@1[task1]$ iEdit
Result: 10
knoppix@1[task1]$ celar
bash: celar: command not found
knoppix@1[task1]$ clear
knoppix@1[task1]$ ls
file1.c  file1.o  file2.c~  iEdit    makefile~
file1.c~ file2.c  file2.o  makefile
knoppix@1[task1]$ clear
```

Задание 4.

Показать, что при изменении одного исходного файла и последующем вызове make будут исполнены только необходимые команды компиляции.

При изменении одного файла и последующей сборки всей программы, происходит перекомпиляция только одного файла, в нашем случае file2.c



The screenshot shows a terminal window titled "Shell - Konsole" and a text editor window titled "file2.c - KWrite".

The terminal window displays the following commands and output:

```
knoppi@1[task1]$ iEdit
Result: 10
knoppi@1[task1]$ ls
file1.c  file1.o  file2.c~  iEdit  makefile~
file1.c~ file2.c  file2.o  makefile
knoppi@1[task1]$ make
gcc -c file2.c
gcc file1.o file2.o -o iEdit
knoppi@1[task1]$ iEdit
Result: 20
knoppi@1[task1]$
```

The text editor window shows the content of file2.c:

```
#include <stdio.h>

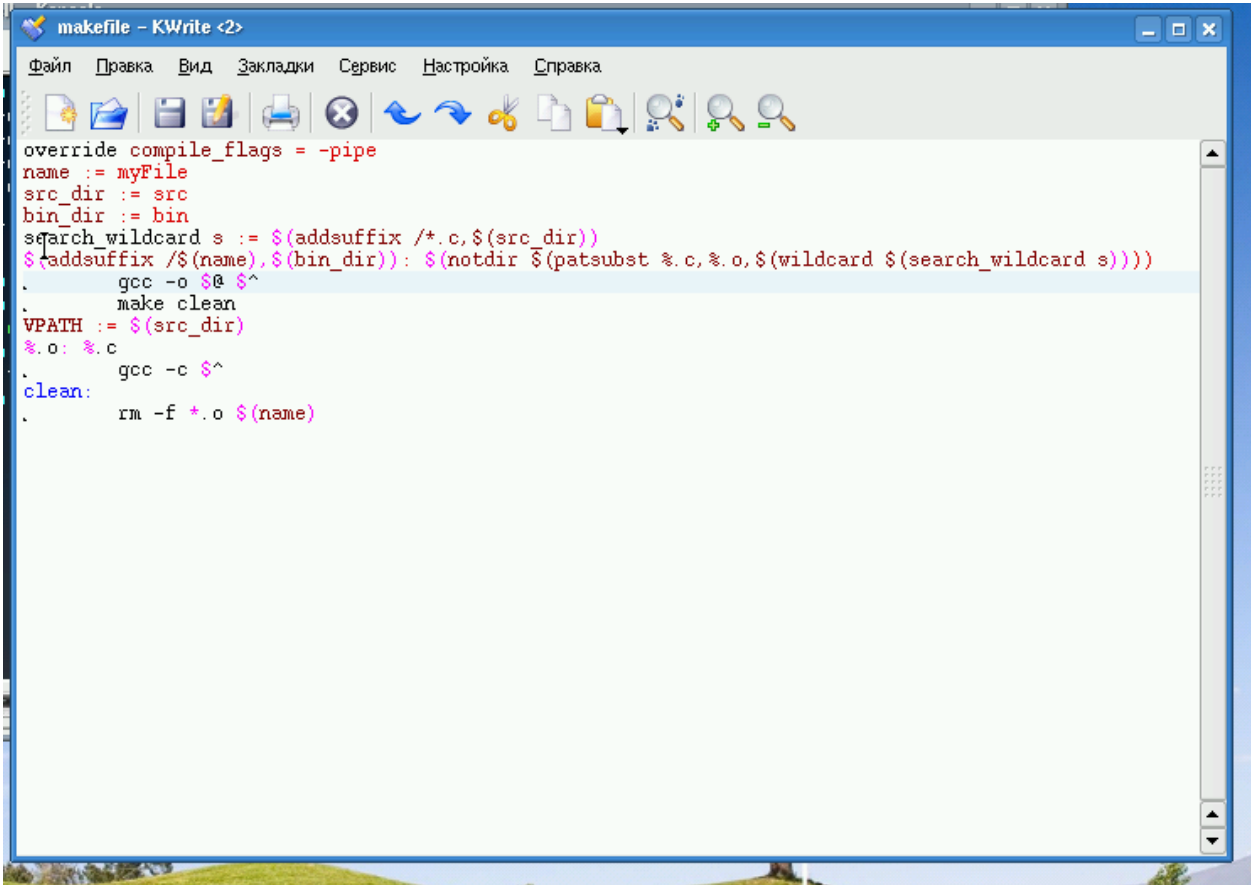
void myFunc(int p){
    int tmp = p;
    tmp = tmp + 4;
    printf("Result: %d\n", tmp);
}
```

Задание 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.

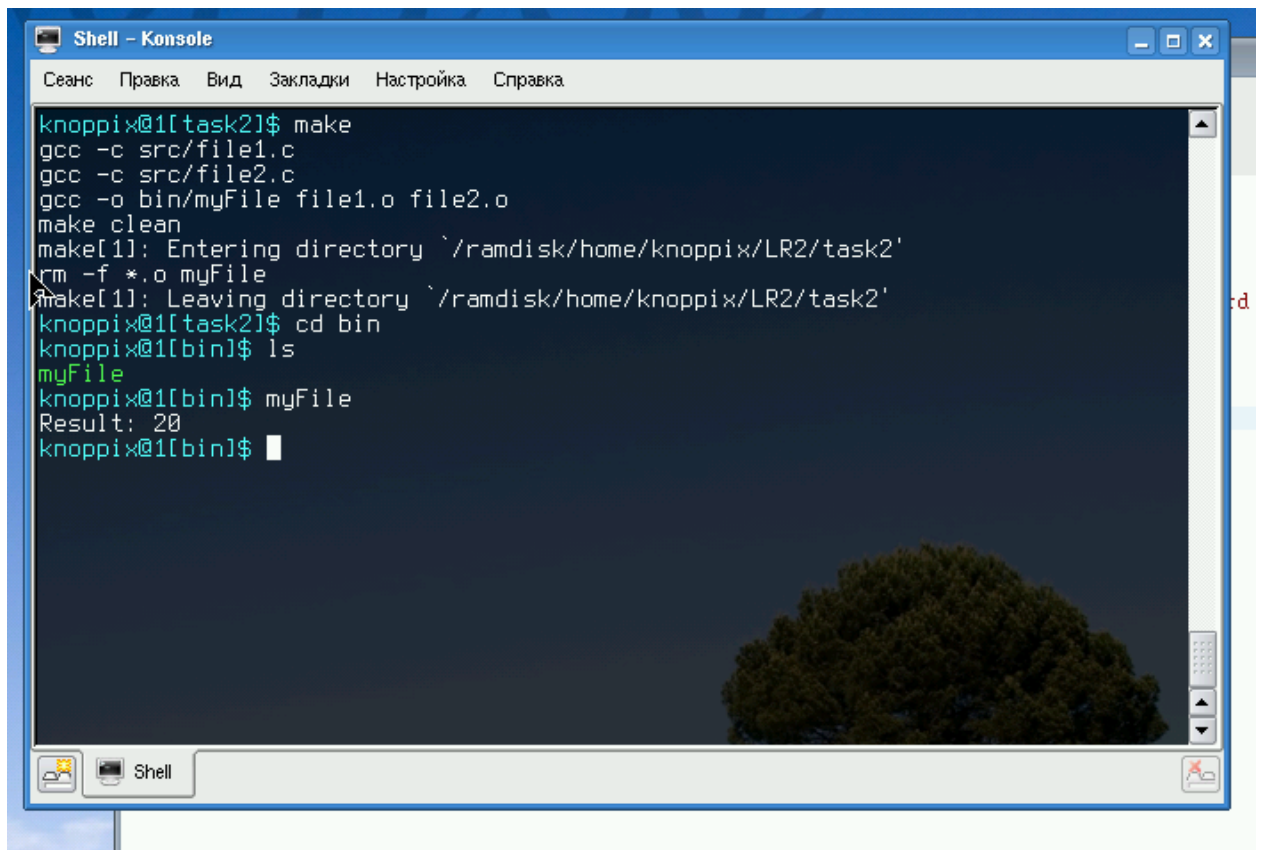
зависимости исходных файлов на языке C (C++) и цели в make-файле должны формироваться динамически;

наличие цели clean, удаляющей временные файлы



```
makefile - KWrite <2>
Файл  Правка  Вид  Закладки  Сервис  Настройка  Справка

override compile_flags = -pipe
name := myFile
src_dir := src
bin_dir := bin
search_wildcard s := $(addsuffix /*.c,$(src_dir))
$(addsuffix /$(name),$(bin_dir)): $(notdir $(patsubst %.c,%.o,$(wildcard $(search_wildcard s))))
    gcc -o $@ $^
    make clean
VPATH := $(src_dir)
%.o: %.c
    gcc -c $^
clean:
    rm -f *.o $(name)
```



```
knoppix@1[task2]$ make
gcc -c src/file1.c
gcc -c src/file2.c
gcc -o bin/myFile file1.o file2.o
make clean
make[1]: Entering directory `/ramdisk/home/knoppix/LR2/task2'
rm -f *.o myFile
make[1]: Leaving directory `/ramdisk/home/knoppix/LR2/task2'
knoppix@1[task2]$ cd bin
knoppix@1[bin]$ ls
myFile
knoppix@1[bin]$ myFile
Result: 20
knoppix@1[bin]$
```

Make-файл был значительно автоматизирован и стал более универсален. Здесь использованы функция wildcard, которая получает список файлов с заданным шаблоном в выбранном каталоге, и функция patsubst, которая заменяет заданную подстроку в заданной строке. Эти функции позволяют автоматически построить список объектных файлов программы. Еще в данном Make-файле автоматизирован поиск исходных файлов по нескольким директориям с использованием переменных search _wildcard s и VPATH.

Выводы

В ходе данной лабораторной работы была изучена утилита Make, которая позволяет собирать программу из множества разрозненных файлов. Данная утилита имеет большое количество возможностей по автоматизации сборки проекта и позволяет создать такой Make-файл, который подойдет ко многим программам и при сборке каждой из них потребует незначительное количество изменений. Была использована сборка knoppix и Oracle VirtualBox.