

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

ОТЧЕТ по лабораторной
работе №1
по дисциплине «Программирование»
Тема: СОЗДАНИЕ MAKEFILE

Студент гр. 7304

Давыдов А.А.

Преподаватель

Берленко Т.А.

Санкт-Петербург

2017

Цель работы.

Создание Makefile по проекту, состоящему из файлов main.c, get_name.c, print_str.c, print_str.h, get_name.h и его сборка при помощи утилиты make.

Основные теоретические положения.

Заголовочные файлы стандартной библиотеки языка C, необходимые для выполнения данной лабораторной работы:

[stdio.h](#)

[stdlib.h](#)

[string.h](#)

Прототип функции вывода строки str:

```
int puts(const char *str);
```

Прототип функции конкатенации строк:

```
char * strncat( char * destptr, char * srcptr, size_t num );
```

Описание функции для ввода массива символов name (предполагается, что строка не содержит более 80 символов):

```
char* get_name(){
    char* name =(char*)malloc(80*sizeof(char));
    int i = 0;    char ch;
    while ((ch = getchar()) != '\n')
    {
        name[i] = ch;
        i++;
    }
    name[i] = '\0';
    return name;
}
```

Описание главной функции:

```
int main(){
    char hello[90] = "Hello, ";
    char* result;
    result = get_name();
```

```
    print_str(strncat(hello, result, 80));  
    free(result);  
    return 0;  
}
```

Сборка проекта - это процесс получения **исполняемого файла** из **исходного кода**.

Сборка проекта вручную может стать довольно утомительным занятием, особенно, если исходных файлов больше одного и требуется задавать некоторые параметры компиляции/линковки. Для этого используются **Makefile** - список инструкций для утилиты **make**, которая позволяет собирать проект сразу целиком.

Если запустить утилиту

`make`

то она попытается найти файл с именем **Makefile** в текущей директории и выполнить из него инструкции.

Если требуется задать какой-то конкретный Makefile, это можно сделать с помощью ключа `-f`

`make -f AnyMakefile`

Любой make-файл состоит из списка целей зависимостей этих целей команд, которые требуется выполнить, чтобы достичь эту цель
цель: зависимости

`[tab]` команда

Для сборки проекта обычно используется цель `all`, которая находится самой первой и является целью по умолчанию. (фактически, первая цель в файле)
Сборка проекта - это процесс получения исполняемого файла из исходного кода.

Сборка проекта вручную может стать довольно утомительным занятием, особенно, если исходных файлов больше одного и требуется задавать некоторые параметры компиляции/линковки. Для этого используются **Makefile** - список

инструкций для утилиты **make**, которая позволяет собирать проект сразу целиком.

Если запустить утилиту

make

то она попытается найти файл с именем **Makefile** в текущей директории и выполнить из него инструкции.

Если требуется задать какой-то конкретный Makefile, это можно сделать с помощью ключа `-f make -f AnyMakefile` и является целью по-умолчанию)

Также, рекомендуется создание цели `clean`, которая используется для очистки всех результатов сборки проекта

Использование нескольких целей и их зависимостей особенно полезно в больших проектах, так как при изменении одного файла не потребуется пересобирать весь проект целиком. Достаточно пересобрать измененную часть Пример:

all: hello

hello: main.o f1.o f2.o

gcc main.o f1.o f2.o -o hello

main.o: main.c

gcc -c main.c

f1.o: f1.c

gcc -c f1.c

f2.o: f2.c

gcc -c f2.c

clean: rm

-rf *.o hello

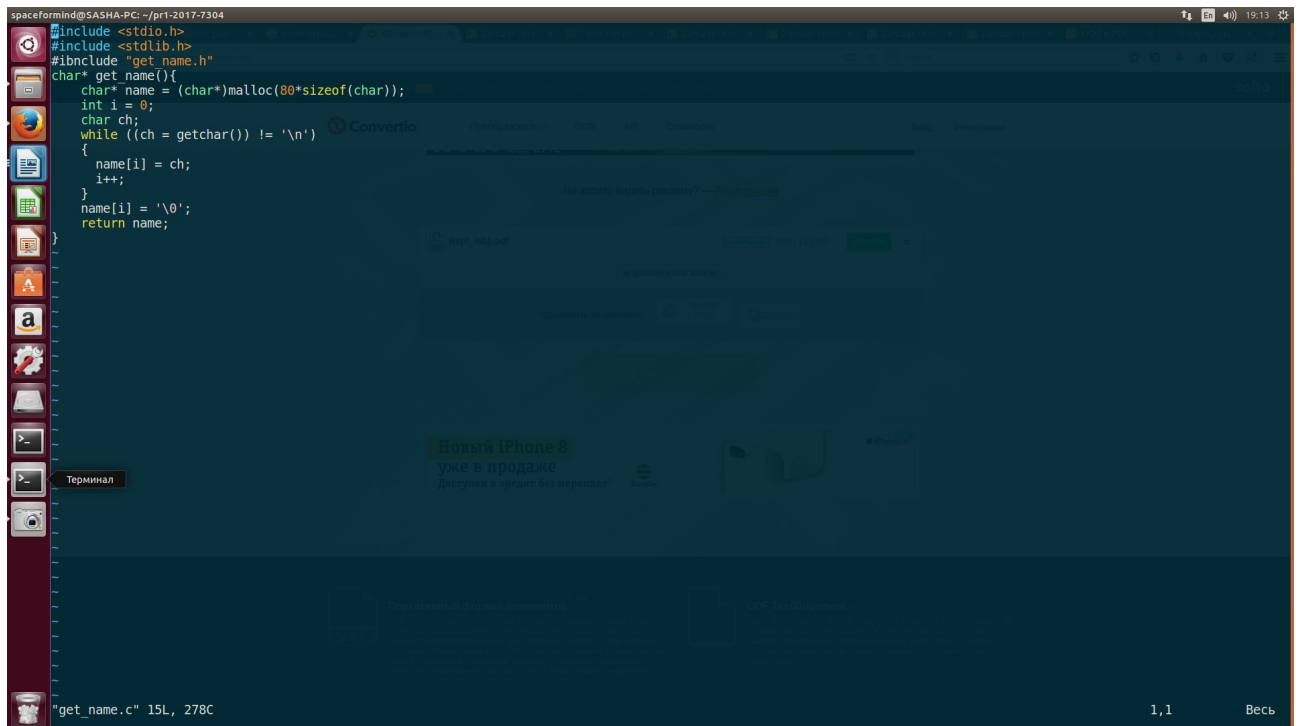
Таким образом, чтобы выполнить цель `all`, требуется выполнить цель `hello`

Для выполнения цели `hello`, а именно вызова `gcc` для объектных файлов, требуется что бы были выполнены цели соответствующие этим объектным файлам.

Для выполнения цели для каждого объектного файла требуется скомпилировать исходный код

Экспериментальные результаты.

1) get_name.c



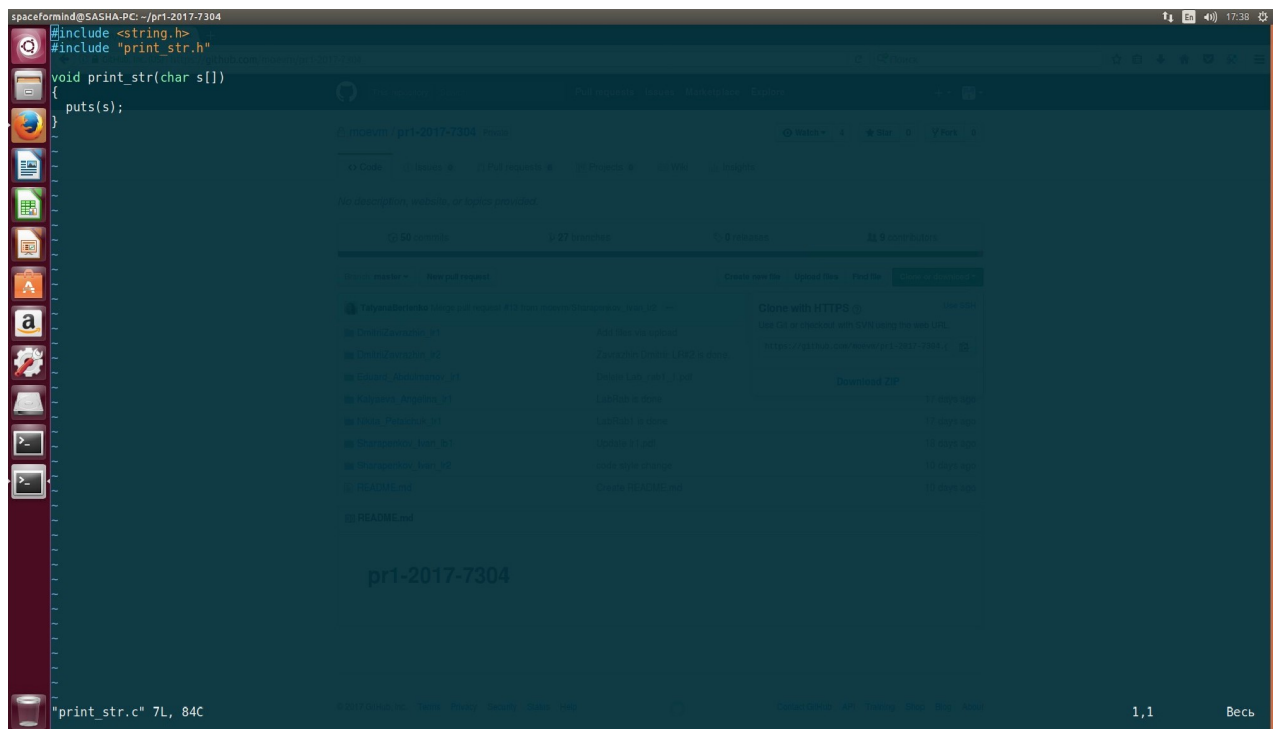
The screenshot shows a Linux desktop environment. A code editor window is open, displaying the source code for `get_name.c`. The code includes `<stdio.h>` and `<stdlib.h>`, and defines a `get_name()` function that allocates memory and reads input characters until a newline is encountered. A terminal window is also open, showing the command `gcc get_name.c -o get_name` and its output, which indicates that the file `get_name.c` was successfully compiled into the executable `get_name` with a size of 15L and 278C.

```
#include <stdio.h>
#include <stdlib.h>
#include "get_name.h"
char* get_name(){
    char* name = (char*)malloc(80*sizeof(char));
    int i = 0;
    char ch;
    while ((ch = getchar()) != '\n')
    {
        name[i] = ch;
        i++;
    }
    name[i] = '\0';
    return name;
}
```

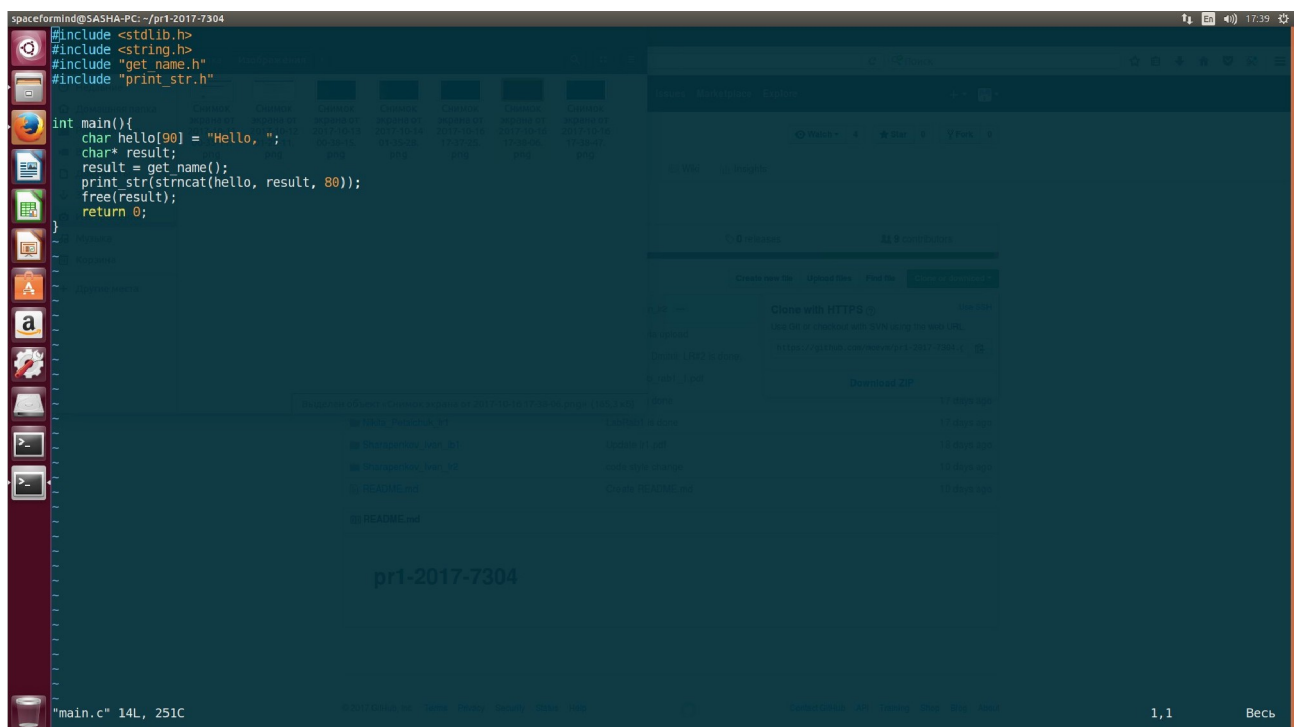
Terminal output:

```
gcc get_name.c -o get_name
"get_name.c" 15L, 278C
```

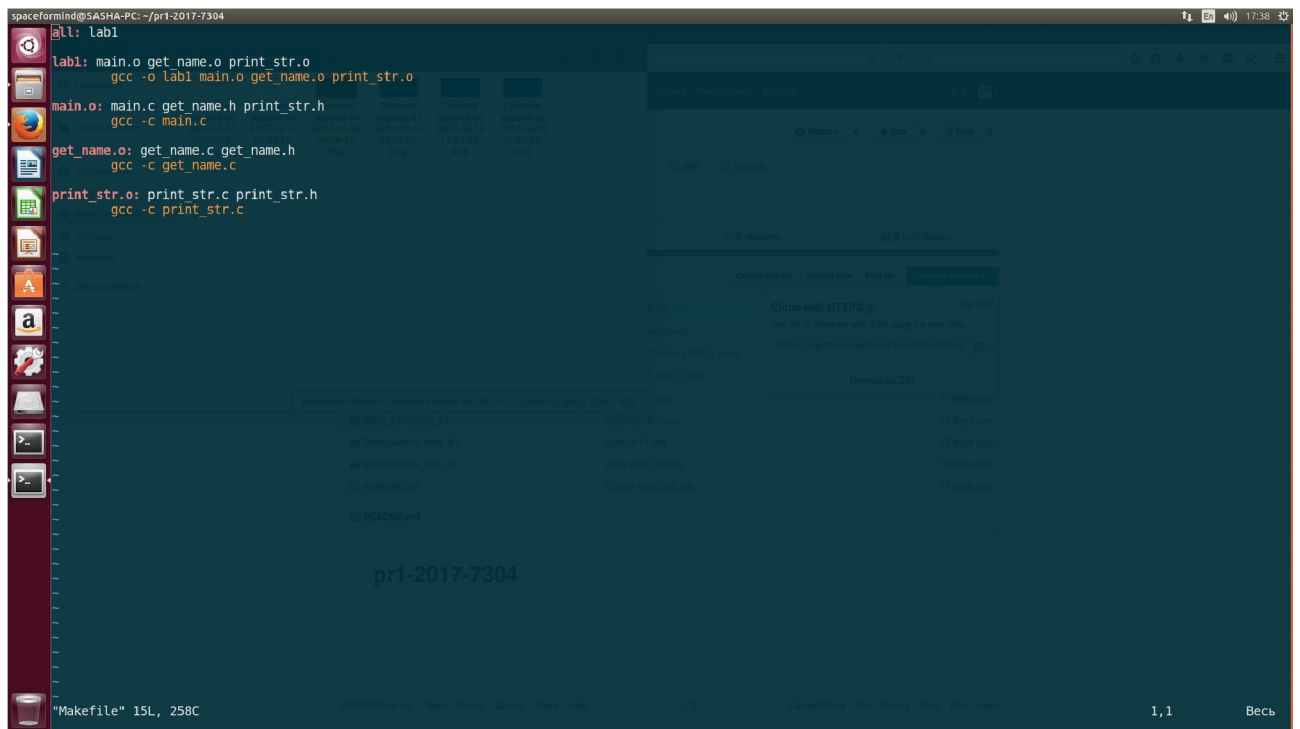
2) print_str.c



3)main.c



4) Makefile



Выводы.

Проделав лабораторную работу, ознакомился с основными командами сборки программ при помощи Makefile и собрал свой проект в соответствии с указаниями предыдущих уроков. Полученные знания ускорят мою работу с многофайловыми программами.