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

ОТЧЕТ

по лабораторной работе №1

по дисциплине «Программирование»

Тема: «Создание make-файла»

Студент гр. 7381 _____ Минуллин М. А.

Преподаватель _____ Михайлов В. В.

Санкт-Петербург

2017

Цель работы.

Познакомиться с операционной системой Linux.

Познакомиться с системой контроля версий git.

Создать проект, состоящий из пяти файлов: "get_name.h", "get_name.c", "print_str.h", "print_str.c", "main.c":

- Файл "get_name.h" должен содержать **прототип** функции "char* get_name()", которая считывает из входного потока имя пользователя и возвращает его.
- Файл "get_name.c" должен содержать **определение** функции "char* get_name()", которая считывает из входного потока имя пользователя и возвращает его.
- Файл "print_str.h" должен содержать **прототип** функции "print_str(char*)", которая принимает в качестве аргумента строку и выводит её.
- Файл "print_str.c" должен содержать **определение** функции "print_str(char*)", которая принимает в качестве аргумента строку и выводит её.
- Файл "main.c" содержит главную функцию "int main()", которая вызывает "get_name()" из файла "get_name.h", добавляет к результату выполнения функции строку "Hello, " и передаёт полученную строку в качестве аргумента в функцию вывода строки "print_str(char*)" из файла "print_str.h"

После создания проекта, написать Makefile, с помощью которого данный проект будет собираться.

Основные теоретические положения.

Заголовочные файлы, необходимые для создания проекта:

1. <stdio.h> - содержит прототип функции "void puts(const char* string)", выводящей в поток вывода строку string. Используется в определении функции "print_str(char*)".

Описание:

Функция "puts" выводит строку типа "char*", на которую указывает параметр "string" в стандартный поток вывод и добавляет символ новой строки '\n'.

Функция начинает копировать строку с адреса, указанного в "string", пока не достигнет нулевого символа ''. Этот заключительный, нулевой символ не копируется в стандартный поток вывод.

Параметры:

"const char* string" - С-строка для вывода на стандартный поток вывода.

Возвращаемое значение:

В случае успеха, возвращается неотрицательное значение.

В случае ошибки, функция возвращает значение EOF.

2. <string.h> - содержит прототип функции "char* strncat(char* destptr, char* srcptr, size_t num)", необходимая для склейки приветствия и имени.

Описание:

Функция добавляет первые num символов строки srcptr к концу строки destptr, плюс символ конца строки. Если строка srcptr больше чем количество копируемых символов num, то после скопированных символов неявно добавляется символ конца строки.

Параметры:

destptr – указатель на строку назначения, которая будет содержать результат конкатенации строк, включая символ завершения строки.

srcptr – строка, из которой будут копироваться первые num символов для конкатенации.

num – максимальное количество символов для конкатенации.

Возвращаемое значение:

Указатель на строку с результатом конкатенации.

3. <stdlib.h> - содержит функции для выделения и освобождения памяти.

```
void free(void* ptrmem);
```

Описание:

Функция free освобождает место в памяти. Блок памяти, ранее выделенный с помощью вызова malloc, calloc или realloc освобождается. То есть освобожденная память может дальше использоваться программами или ОС.

Обратите внимание, что эта функция оставляет значение ptr неизменным, следовательно, он по-прежнему указывает на тот же блок памяти, а не на нулевой указатель.

Параметры:

ptrmem – указатель на блок памяти, ранее выделенный функциями malloc, calloc или realloc, которую необходимо высвободить. Если в качестве аргумента передается нулевой указатель, никаких действий не происходит.

Возвращаемое значение:

Функция не имеет возвращаемое значение.

```
void* malloc(size_t sizemem);
```

Описание:

Функция `malloc` выделяет блок памяти, размером `sizemem` байт, и возвращает указатель на начало блока.

Содержание выделенного блока памяти не инициализируется, оно остается с неопределенными значениями.

Параметры:

`sizemem` – размер выделяемого блока памяти в байтах.

Возвращаемое значение

Указатель на выделенный блок памяти. Тип данных на который ссылается указатель всегда `void*`, поэтому это тип данных может быть приведен к желаемому типу данных.

Если функции не удалось выделить требуемый блок памяти, возвращается нулевой указатель.

Вывод:

В процессе работы над проектом, освоил основные функции терминала в Linux (переход между директориями, просмотр их содержимого, создание и удаление файлов, просмотр файлов, открытие файла в текстовом редакторе, запуск исполняемых файлов), познакомился с операционной системой Linux, системой контроля версий git (checkout, add, commit, status, log, diff, clone, merge, push, pull), освоил компиляцию кода через консоль вручную и с помощью утилиты make (понятия цели, реквизитов, переменных и пр.), флаги компиляции `-c` (компиляция файла, получение объектного файла), `-o` (указание имени исполняемого файла, полученного при сборке проекта), `-E` (запуск препроцессора без последующей компиляции), `-l` и `-L` (добавление путей для поиска библиотек), `-I` (добавление путей для поиска заголовочных файлов), `-O0`, `-O1`, `-O2`, `-O3` (оптимизация кода, `0-3` – уровень оптимизации, 0 – без неё).

Исходный код проекта:

- Файл "get_name.h":

```
#pragma once

char* get_name();
```

- Файл "get_name.c":

```
#include "get_name.h"
#include <stdlib.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;
}
```

- Файл "print_str.h":

```
#pragma once

void print_str(char* string);
```

- Файл "print_str.c":

```
#include "print_str.h"
#include <stdio.h>

void print_str(char* string) {
    puts(string);
}
```

- Файл "main.c":

```
#include "get_name.h"
```

```c
#include "print_str.h"
#include <string.h>

int main() {
    char hello[90] = "Hello, ";
    char* result;
    result = get_name();
    print_str(strncat(hello, result, 80));
    free(result);
    return 0;
}
```

- Файл Makefile:

```
all: main.o get_name.o print_str.o
   gcc -o lr1 main.o get_name.o print_str.o

main.o: main.c get_name.h print_name.h
   gcc -c main.c

get_name.o: get_name.c get_name.h
   gcc -c get_name.c

print_str.o: print_str.c print_str.h
   gcc -c print_str.c
```