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 make-файла»

Студент гр. 7381 _____ Павлов А.П.

Преподаватель _____ Берленко Т.А.

Санкт-Петербург
2017

Цель работы.

Познакомиться с операционной системой Linux и научиться работать в терминале.

Познакомиться с системой контроля версий git, научиться делать pull request.

Создать проект, состоящий из пяти файлов: main.c, print_str.c, get_name.c, print_str.h, get_name.h и произвести его сборку при помощи make-файла.

- Файл get_name.c должен содержать **описание** функции, которая **считывает** из входного потока имя пользователя и возвращает его.
- Файл get_name.h должен содержать **прототип** функции, которая **считывает** из входного потока имя пользователя и возвращает его.
- Файл print_str.c должен содержать **описание** функции, которая **принимает** в качестве аргумента строку и выводит её (функция ничего не возвращает).
- Файл print_str.h должен содержать **прототип** функции, которая **принимает** в качестве аргумента строку и выводит её (функция ничего не возвращает).
- Файл main.c содержит главную функцию, которая вызывает функцию из файла get_name.h, добавляет к результату выполнения функции строку “Hello, ” и передает полученную строку в функцию вывода строки из print_str.h.

Основные теоретические положения.

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

- stdio.h
- stdlib.h
- string.h

1. <stdio.h> - содержит прототип функции "void puts(const char *str)", 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:

string Си-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)”, которая необходима 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 3.1 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.

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.

3.2 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 ходе лабораторной работы были освоены основные команды для работы в терминале операционной системы Linux(создание и удаление директорий, просмотр содержимого директорий, создание и удаление файлов, просмотр пути до файла), система контроля версий git(git clone, add, commit, push, pull), компиляция кода через консоль вручную и с помощью утилиты make(цель, реквизиты, переменные).

Исходный код проекта:

Файл "get_name.h":

```
#pragma once
char* get_name();
```

Файл "get_name.c":

```
#include <stdio.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;
}
```

Файл "print_str.h":

```
#pragma once
void print_str(const char *str);
```

Файл "print_str.c":

```
#include <stdio.h>
#include "print_str.h"
void print_str(const char *str) {
puts(str);
}
```

Файл "main.c":

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "get_name.h"
#include "print_str.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: hello
hello: main.o print_str.o get_name.o
	gcc -o a.out main.o print_str.o get_name.o
main.o: main.c get_name.h print_str.h
	gcc -c main.c
print_str.o: print_str.c print_str.h
	gcc -c print_str.c
get_name.o: get_name.c get_name.h
	gcc –c get_name.c
```