

**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

\_\_\_\_\_

Вологдин М.Д.

Преподаватель

\_\_\_\_\_

Берленко Т.И.

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.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.

После создания проекта, написать для него Makefile, с помощью которого он будет собираться.

## Основные теоретические положения.

**Заголовочные файлы** стандартной библиотеки языка C, необходимые для создания проекта:

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 )" 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.

A. `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.

**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.

B. `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 выполнения лабораторной работы познакомился с операционной системой Linux, освоил основные функции терминала. Познакомился с системой контроля версий git. Освоил компиляцию кода через консоль вручную и с помощью утилиты make.

### **Исходный код проекта:**

- Файл "get\_name.h":

```
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":

```
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
```