**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 и управление кодом с помощью git

|  |  |  |
| --- | --- | --- |
| Студенка гр. 6304 |  | Блинникова Ю. И. |
| Преподаватель |  | Кринкин К.В. |

Санкт-Петербург

2016

**Цель:** Научиться создавать makefile и управлять кодом с помощью Git.

**Задание:**

1. Создайте проект, состоящий из пяти файлов: main.c, print\_str.c, get\_name.c, print\_str.h, get\_name.h в каталоге, имя которого содержит Ваше имя, Фамилию и номер лабораторной.

* 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.

2. После того, как Ваш проект будет готов, создайте для него Makefile.

**Содержание:**

1. **Создание get\_name.c**

Cодержит описание функции, которая считывает из входного потока имя пользователя и возвращает его.

|  |
| --- |
| #include<stdio.h> |
| #include<string.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; |
| } |

1. **Создание get\_name.h**

Содержит прототип функции, которая считывает из входного потока имя пользователя и возвращает его.

char\* get\_name();

1. **Создание print\_str.c**

Содержит описание функции, которая принимает в качестве аргумента строку и выводит её (функция ничего не возвращает).

|  |
| --- |
| #include<stdio.h> |
| #include<stdlib.h> |
| #include<string.h> |
| char\* print\_str(char\*name){ |
| printf("%s\n",name); |
| } |

1. **Создание print\_str.h**

Содержит прототип функции, которая принимает в качестве аргумента строку и выводит её (функция ничего не возвращает).

void print\_str(char\*name);

1. **Создание 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.

|  |
| --- |
| #include<stdio.h> |
| #include<string.h> |
| #include<stdlib.h> |
| int main(){ |
| char hello[90] = "Hello, "; |
| char\* result; |
| result = get\_name(); |
| print\_str(strncat(hello, result, 80)); |
| free(result); |
| return 0; |
| } |

1. **Создание Makefile:**

main: get\_name.o print\_str.o main.o

#Линковка

gcc main.o -o main get\_name.o print\_str.o

#Исполняемый файл – main

rm \*.o

#Очистка

main.o: main.c

gcc -c main.c

#Создание объектного файла main.o

get\_name.o: get\_name.c get\_name.h

gcc -c get\_name.c

#Создание объектного файла get\_name.o

print\_str.o: print\_str.c print\_str.h

gcc -c print\_str.c

#Создание объектного файла print\_str.o

clean:

rm \*.o

#Очистка

1. **Отправка работы на репозиторий**
2. Git clone <Ссылка на репозиторий 6304>( скопировали себе репозиторий группы 6304 и стали работать в нем локально)
3. Git checkout –b “Blinnikova\_Yulia\_L\_1” (локально создали новую ветку от ветки master для нашего проекта)
4. Git branch (проверили, в какой ветке мы сейчас находимся)
5. Git push origin “ Blinnikova\_Yulia\_L\_1” отправили изменения на сервер.

В данном случае за “origin” считается репозиторий группы 6304 т.к. мы работаем в нем.

1. Git add \*.c \*.h makefile проиндексировали изменения
2. Git commit –m “L\_1” создали коммит наших изменений
3. Git push origin отправили изменения на репозиторий
4. Создали pull request в ветку master

**Вывод:** Выполнив данную лабораторную работу, мы освоили на практике написание собственного make файла для сборки проекта, а также работу с репозиторием GitHub.com, основные команды, связанные с индексированием изменений, созданием веток и синхронизации их с удалённым сервером.