GIT, GCC

практика

Задачка на подумать:

Попробуйте в уме разделить 30 на 1/2 и прибавить 10. Каким будет результат?

Работа с компилятором

- Работаем в терминале.
  - Создадим в домашней директории папку: «**helloWorldProj**».
  - Внутри папки создаем файл: «helloWorld.c».
  - Открываем созданный файл:

$ **nano helloWorld.c**

  - Набираем в файле текст:

```
#include <stdio.h>
int main ()
{
  printf ("Hello, World!\n")
  return 0;
}
```

  - Чтобы сохранить файл, используем сочетания клавиш: **ctrl+X**, затем **Enter**.
  - Запускаем компилятор из директории где находится созданный файл - «helloWorld.c» :

$ **gcc –o hello helloWorld.c**

  - Пробуем запустить программу:

$ ./hello

Работа с компилятором

Компиляция программы из нескольких файлов исходного кода может осуществляться также, как и программы из одного файла исходного кода путем перечисления всех файлов исходного кода вместо одного, но лучшим решением является компиляция файлов исходного кода по очереди с последующим связыванием.

Создайте 3 файла (см. ->).

Для компиляции программы “**test**” может использоваться следующая последовательность команд:

$ g++ -c test.cpp -o test.o

$ g++ -c main.cpp -o main.o

$ g++ test.o main.o -o test

В результате также будет создан исполняемый файл “**test”**:

$ ./test

Hello world

test.cpp:

```
#include <iostream>
void print_hello()
{
  std::cout << “Hello world”<< std::endl;
}
```

main.cpp:

```
#include <iostream>
#include "test.h"
int main()
{
        print_hello();
         return 0;
}
```

test.h:

```
#pragma once
void print_hello();
```

Установка, начальная настройка GIT

- Для установки:

$ sudo apt update

$ sudo apt install git

- Настройка:

В состав Git входит утилита **git config**, которая позволяет просматривать и настраивать параметры, контролирующие все аспекты работы **Git**

1. Чтобы посмотреть все установленные настройки и узнать где именно они заданы, используйте команду:

$ git config --list --show-origin

2. Укажем ваше имя и адрес электронной почты:

$ git config --global user.name «John Doe»
$ git config --global user.email johndoe@example.com

3. Самое время выбрать текстовый редактор

$ git config --global core.editor nano

4. Установить имя ***main*** для вашей ветки по умолчанию

$ git config --global init.defaultBranch main

5. Проверить используемую конфигурацию, можете использовать команду

$ git config --list

Помощь при работе с GIT

Если вам нужна помощь при использовании Git

$ git help <команда>

$ git <команда> --help

$ man git-<команда>

Если вам нужно посмотреть только список опций команды:

$ git add -h

Создание локального Git-репозитория

Вы можете взять локальный каталог, который в настоящее время не находится под контролем, и превратить его в репозиторий **Git**.

- Переходим в каталог проекта, который планируется поставить под версионный контроль **Git:**

$ cd /home/<user>/helloorldProj

а затем выполните команду:

$ git init

- Если вы хотите добавить под версионный контроль существующие файлы (в отличие от пустого каталога), вам стоит добавить их в область индексирования и осуществить первый коммит изменений. Добиться в область индекс. его вы сможете запустив команду **git add**, указав индексируемые файл(-ы), а затем выполнив **git commit**:

$ git add .

$ git commit -m 'Initial project version'

- Основной инструмент, используемый для определения, какие файлы в каком состоянии находятся — это команда:

$ git status

- Создание веток. Чтобы создать ветку (**br1**) и сразу переключиться на неё, можно выполнить команду **git checkout** с параметром **-b**:

$ git checkout -b br1

- Это то же самое:

$ git branch br1

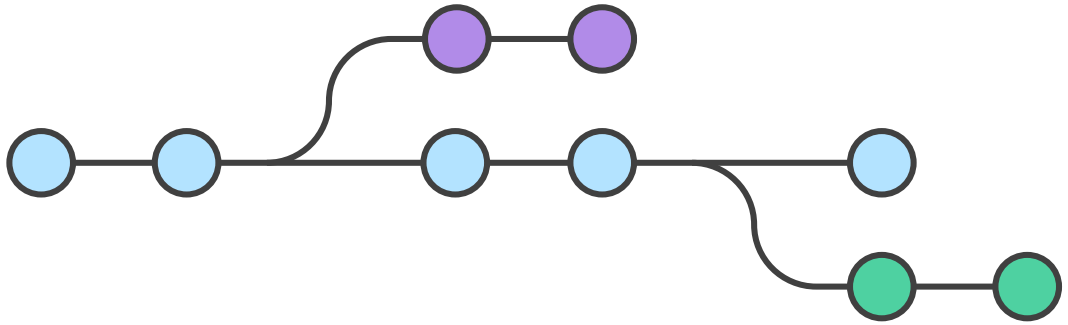
$ git checkout br1

- Переключится на ветку main (master) с ветки br1, можно так:

$ git checkout master

- Удалить ветку br1, если она больше не нужна :

$ git branch -d br1



Клонирование репозитория GITHub

Клонирование существующего репозитория с сервера:

$ git clone https://github.com/libgit2/libgit2

Эта команда создаёт каталог **libgit2**, инициализирует в нём подкаталог **.git**, скачивает все данные для этого репозитория и извлекает рабочую копию последней версии.

Домашняя работа:

1. Добавить в локальный репозиторий файл «readMe.txt» с описанием программы “Hello World!”
2. Регистрируемся на GitHub е. Создаем репозиторий.
3. Изучаем GIT (1 урок): http://learngitbranching.js.org/?locale=ru_RU