Лабораторная работа №4

# Изучение системы контроля версий Git. Особые функции.

Для начала рассмотрим те вещи, которые вам нужно знать про git:

* Git не хранит папки. Он хранит только файлы. Поэтому, например, невозможно положить в git пустую папку – можно только папку с файлами.
* Git не хранит файлы. На самом деле, он хранит объекты, в которых указаны пути к файлам. Так что в git-репозитории действительно есть файлы, но вот сам git (т.е. папка .git, как мы узнали из предыдущей лабораторной) их не хранит.
* Другими словами, git сохраняет в commit содержимое всех файлов (делает слепки содержимого каждого файла и сохраняет в objects). Если файл не менялся, то будет использован старый object. Таким образом, в commit в виде новых объектов попадут только **изменённые файлы**, что позволит хорошо экономить место на диске и даст возможность быстро переключиться на любой commit.
* Ревизии (Revision) не имеют порядкового номера. Разные файлы входят в разные ревизии, поэтому в каждый момент времени каждый файл принадлежит своему времени.
* Редакции (правки, commits) могут идти не по порядку.
* В Git’е, по сути дела, нет веток. *На самом деле,* у нас есть много коммитов, которые образуют граф. Выбираем любой путь от parent-commit к любому child-commit и получаем состояние проекта на этот коммит. Чтобы коммит «запомнить» можно создать на него именованный указатель. Такой именованный указатель и есть ветка (branch). `HEAD` работает по такому же принципу — показывает, где мы есть сейчас. Новые коммиты являются продолжением текущей ветки (туда же куда и смотрит HEAD).

Git – достаточно сложная система. Чтобы облегчить работу программиста, в поставке git есть две вещи, которые должны быть у вас под рукой всегда:

* git status — показывает состояние вашего репозитория (рабочей копии) и где вы находитесь.
* gitk— графическая утилита, которая показывает наш граф. В качестве ключей передаём имена веток или --all, чтобы показать все.

Если вы сделали что-то не так, запутались, не знаете, что происходит — эти две команды вам помогут.

В список полезных команд git также входят:

git diff, а именно:

git diff HEAD~ – показывает разница между текущим и предыдущим коммитом

git diff 0da94be 59ff30c – показывает изменения между двумя коммитами.

git diff --name-only SHA1 SHA2 – показывает только имена файлов, которые изменились.

git diff --name-only HEAD~10 HEAD~5 – показывает только имена файлов, которые изменились, при этом, показывает только те файлы, которые изменились на расстоянии от 10 до 5 коммита до верхушки ветки. Т.е., если было 20 изменений, то это команда покажетизменения, которые были между десятым и пятнадцатым коммитом.

git show – покажет вам информацию о текущем коммите: как, когда, зачем (сообщение) он был сделан.

За более подробной информацией обращайтесь в git help.

# Задание:

1. Пройти игру; ссылку получить у преподавателя ( или <https://github.com/demns/git-game.git> ). Прохождение заключается в поиске решения при помощи командной строки; в отчёте нужно будет указать все этапы, как вы что проходили.
2. Открыть gitk и посмотреть на игру с другой стороны; уметь объяснить преподавателю, что где происходит.
3. \* попробуй выполнить следующие команды и посмотреть на результат:
   1. cherry-pick
   2. git for-each-ref --sort=-committerdate refs/heads/
   3. git log --graph --abbrev-commit --decorate --date=relative --all

P.S. выход из VIM -- :q ☺

# Чтение:

<http://habrahabr.ru/post/174467/>