Основы программной инженерии

Лабораторная работа 2

черновик
версия 0.6

Целью лабораторной работы является приобретение навыков работы с ветвями (локальными и удаленными) в Git.

В ходе выполнения лабораторной работы студенты должны

1. приобрести навыки работы с ветвями в Git (создание, переключение, слияния);
2. закрепить навыки работы с локальными командами Git и системой ведения проекта GitLab (wiki, issue, merge request).

Задание 1

Выполнение каждого пункта задания фиксируется в отчете. Описывая выполнение очередного пункта задания, старайтесь привести такое описание, прочитав которое неподготовленный человек смог бы его выполнить.

1. Создайте локальный репозиторий.
2. Создайте файл .gitignore. За основу можно взять файл из предыдущей лабораторной работы и, если нужно, изменить его в соответствии с вашими требованиями. Поместите файл под версионный контроль.
3. Скопируйте в рабочую директорию файлы лабораторной работы в соответствии с вашим вариантом. Убедитесь, что программа работает, после чего поместите исходный код под версионный контроль.
4. Создайте отдельную ветвь для исправления ошибки (для этого используется команда git branch). Ветвь должна называться fix. Переключитесь на нее.

```
// Создадим ветвь fix
$ git branch fix

// Переключимся на эту ветвь
$ git checkout fix

Переключено на ветку «fix»

// Посмотрим какие ветви есть
$ git branch

* fix
  master

// Всего две ветви: fix и master. Ветвь fix является активной.
```

Замечание.
Чтобы совместить создание ветви и «переключение» на нее можно было бы воспользоваться командой checkout и параметром “-b”.

ВНИМАНИЕ.
Чтобы избежать ненужных проблем при переключении с одной ветви на другую, переход лучше осуществлять из «чистого» рабочего состояния проекта (т.е. когда команда git status ничего не показывает).

5. Изучите исходный код программы, помещенной под версионный контроль. Добавьте комментарии, описывающие назначение функций программы. Поместите изменения под версионный контроль.
6. Добавьте тест, демонстрирующий наличие ошибки в программе. Поместите изменения под версионный контроль.
7. Создайте issue и опишите ошибку. В ошибке укажите «номер» ревизии, в которой был добавлен тест, демонстрирующий наличие ошибки.
8. Исправьте ошибку. Зафиксируйте изменения (укажите номер issue в комментарии к фиксации).
9. Опишите в issue суть сделанных изменений. Укажите «номер» ревизии, в которой ошибка была исправлена.
10. Переключитесь на ветку master и выполните объединение изменений (для этого используется команда git merge). Произошел ли конфликт при объединении изменений?

```
$ git checkout master

$ git merge fix
```

11. Укажите в issue «номер» ревизии, в которой исправления были перенесены из ветки fix в ветвь master. Закройте issue.
12. Проанализируйте историю изменений в вашем репозитории с помощью команды "git log --oneline --graph --all". Проанализировав изменения, объясните почему не было конфликта при объединении изменений.

Задание 2

Исходными данным для выполнений данного задания является репозиторий.

1. Распакуйте репозиторий.
2. Проанализируйте историю изменений в этом репозитории.
    a. Сколько ветвей в нем есть? Как они называются?
    b. Сколько пользователей работали с этим репозиторием? Какие имена у этих пользователей?
    c. Сколько файлов находится в репозитории? Кто и в какой последовательности вносил изменения в эти файлы?
3. Перенесите изменения из всех ветвей в ветвь master. Если при переносе очередных изменений возникает конфликт, 1) объясните причину этого конфликта, 2) самостоятельно выполните объединение изменений (выполняя объединения изменений, добейтесь расположение абзацев текст в логически правильной последовательности).

Если при слиянии изменений происходит конфликт, вы получите примерно такое сообщение:

```
$ git merge BRANCH_NAME
Автослияние FILE_NAME
КОНФЛИКТ (содержимое): Конфликт слияния в FILE_NAME
Не удалось провести автоматическое слияние; исправьте конфликты и сделайте коммит результата.
```

Git не создал фиксацию автоматически. Он остановил процесс до тех пор, пока вы не разрешите конфликт. Чтобы в любой момент после появления конфликта увидеть, какие файлы не объединены, вы можете запустить git status:

```
$ git status
На ветке master
У вас есть не слитые пути.
  (разрешите конфликты, затем запустите «git commit»)
  (используйте «git merge --abort», чтобы остановить операцию слияния)

Не слитые пути:
  (используйте «git add <файл>…», чтобы пометить разрешение конфликта)

        оба измены:     FILE_NAME

нет изменений добавленных для коммита
(используйте «git add» и/или «git commit -a»)
```

Все, где есть неразрешенные конфликты слияния, перечисляется как неслитое. Git добавляет в конфликтующие файлы стандартные пометки разрешения конфликтов, чтобы вы могли вручную открыть их и разрешить конфликты.

В конфликтующем файле появляется раздел, выглядящий примерно так:

```
<<<<<<< HEAD:FILE_NAME
Blach blach 1
=======
Blach blach 2
>>>>>>> BRANCH_NAME:FILE_NAME
```

Чтобы разрешить конфликт, вам придется либо выбрать один из предлагаемых вариантов решения, либо объединить содержимое по-своему.

Задание 3

В этом задании вам придется работать с вашим удаленным репозиторием как по «Основам программной инженерии», так и по «Программированию на Си». Заранее узнайте адреса этих репозиториев в GitLab.

Задание 3.1

ВНИМАНИЕ.
В этом задании используется удаленный репозиторий по курсу «Программирование на Си»!

Выполните (если еще это не сделали) пункт 6 из лабораторной работы #1 по «Программированию на Си» (создайте ветвь с указанным именем, поместите в нее исходный код решенных задач, отправьте изменения в удаленный репозиторий, создайте merge request).

В отчете приведите ссылку на созданный merge request.

Задание 3.2

ВНИМАНИЕ.
В этом задании используется удаленный репозиторий по курсу «Основы программной инженерии»!

Целью данного задания является моделирование ситуации, когда вы одновременно работаете над двумя лабораторными работами, а первой принимают лабораторную работу, mege request для которой был создан позже.

В отчете приведите подробные ответы только для пунктов 10 и 11 (имеет смысл привести историю изменений в вашем репозитории и прокомментировать ее).

1. Получите копию удаленного репозитория.
2. Перейдите в рабочую директорию.
3. На основе ветви master создайте ветви lab_02_a и lab_02_b.
4. Переключитесь на ветвь lab_02_a.
5. Добавьте под версионный контроль
    a. текстовый файл lab_02_a.txt, который содержит текст “lab_02_a”;
    b. файл .gitignore для игнорирования **исполняемых** файлов.
6. Отправьте изменения в удаленный репозиторий и создайте merge request (назовем его A).
7. Переключитесь на ветвь lab_02_b.
8. Добавьте под версионный контроль
    a. текстовый файл lab_02_b.txt, который содержит текст “lab_02_b”;
    b. файл .gitignore для игнорирования **объектных** файлов.
9. Отправьте изменения в удаленный репозиторий и создайте merge request (назовем его B).
10. Попросите преподавателя принять merge request B.
11. Проанализируйте состояние merge request A. Объясните, почему возник конфликт.
12. Разрешите конфликт.

Конфликт в ветви lab_02_a может быть разрешен «в лоб»: после того, как merge request B принят и изменения из ветви lab_02_b попали в ветвь master, можно удалить и ветвь lab_02_a (предварительно сохранив все наработки из этой ветви) и merge request A, после чего пересоздать ветвь lab_02_a (на основе сохраненных наработок) и merge request. Такой способ разрешения конфликта использовать НЕЛЬЗЯ.

Задание 3.3

ВНИМАНИЕ.
В этом задании используется удаленный репозиторий по курсу «Основы программной инженерии»!

Целью данного задания является моделирование ситуации, когда состояние удаленного репозитория меняется до того, как вы успели после команды git clone сделать команду git push.

К выполнению этого задания имеет смысл приступать после того, как вы выполнили задание 3.2.

В отчете приведите подробные ответы только для пунктов 11 и 12 (имеет смысл привести историю изменений в вашем репозитории и прокомментировать ее). Распишите последовательность ваших действий во времени (T0: получил копию удаленного репозитория; T1: зафиксировал ... в локальном репозитории; T2: зафиксировал ... в локальном репозитории; T3: отправил в удаленный репозиторий; и т.д.).

1. Получите копию удаленного репозитория.
2. Перейдите в рабочую директорию.
3. На основе ветви master создайте ветвь lab_02_c и переключитесь на нее.
4. Добавьте под версионный контроль текстовый файл. Текстовый файл должен содержать абзац #1 текста, который приводится ниже.
5. Отправьте изменения в удаленный репозиторий.
6. Добавьте в файл абзац #3 текста (т.е. в файле сначала должен быть абзац #1, а за ним абзац #3), зафиксируйте изменения в локальном репозитории, но не отправляйте эти изменения в удаленный репозиторий.
7. Склонируйте удаленный репозиторий в другую директорию (фактически вы получите еще одну копию удаленного репозитория).
8. Переключитесь на ветвь lab_02_c.
9. Добавьте в текстовый файл абзац #2 (т.е. теперь в файле сначала должен быть абзац #1, а за ним абзац #2), выполните фиксацию, после чего отправьте изменения в удаленный репозиторий.
10. Перейдите в первую копию удаленного репозитория.
11. Попытайтесь отправить изменения в удаленный репозиторий. Удается ли сделать это? Почему?
12. Разрешите конфликт.

Если проводить аналогию с вашей деятельностью, то пункты 1 – 6 и 10 – 12 можно рассматривать, например, как работу с удаленным репозиторием дома, а пункты 7 – 9 – как работу с удаленным репозиторием в университете.

Текст для выполнения задания 3.3.

// Абзац #1
Вот оно какое, наше лето,
Лето яркой зеленью одето,
Лето жарким солнышком согрето,
Дышит лето ветерком.

Ля-ля-ля ля-ля-ля,
Ля-ля-ля-ля-ля ля-ля-ля-ля.
Ля-ля-ля ля-ля-ля,
Ля-ля-ля-ля-ля ля-ля!

// Абзац #2
На зеленой солнечной опушке
Прыгают зеленые лягушки,

И танцуют бабочки-подружки,
Расцветает все кругом.

Мы в дороге с песенкой о лете,
Самой лучшей песенкой на свете,
Мы в лесу ежа, быть может, встретим,
Хорошо, что дождь прошел.

Ля-ля-ля ля-ля-ля,
Ля-ля-ля ля-ля-ля-ля.

// Абзац #3
Мы покрыты бронзовым загаром,
Ягоды в лесу горят пожаром.
Лето это жаркое недаром,
Лето — это хорошо!

Ля-ля-ля ля-ля-ля,
Ля-ля-ля-ля-ля ля-ля-ля-ля.
Ля-ля-ля ля-ля-ля,
Ля-ля-ля-ля-ля ля-ля!

Литература

1. Презентации Корниенко В.В., Ломовского И.В.
2. Удачная модель ветвления для Git (https://habrahabr.ru/post/106912/)
3. GitHub Flow: рабочий процесс Гитхаба (https://habrahabr.ru/post/189046/)
4. Pro Git (https://git-scm.com/book/ru/v2), глава 3