**Лабораторная работа №1**

**Тема:** «Работа с системой контроля версий GIT.»

**Цель:** закрепить на практике умение работы с системами контроля версий на базе GIT.

**Оборудование:** персональный компьютер (ПК), OS Windows 7, MS Office 2007.

**Порядок выполнения работы:**

1. Ознакомиться с заданием;

2. В соответствии с заданием выполнить лабораторную работу

3. Продемонстрировать проделанную работу преподавателю;

4. Оформить отчет и сдать преподавателю.

**Содержание отчета:**

1. Тема, цель, оборудование;

2. Отчет должен включать:

а) структурную схему основных элементов

б) описание последовательных шагов и пунктов задания с соответствующими скриншотами

3. Выводы.

**Теоретические сведенья**

Системы управления версиями (СУВ) хранят всю историю продуктов разработки: исходных файлов, документации, используемых библиотек и так далее. Главное назначение СУВ — облегчение параллельной работы над приложением:

* Управление версиями позволяет автоматически разрешить конфликты, возникающие, когда одни и те же документы меняются несколькими разработчиками или во всяком случае облегчить нахождение компромисса.
* СУВ помогают вести разработку в нескольких направлениях за счет механизма ветвления / слияния (*branch* / *merge*). Например, при внедрении нового компонента в программную систему новый код выделяется в отдельную ветвь, а после проведения модульных и интеграционных тестов эта ветвь вливается в основную ветвь разработки (*trunk*).
* Использование СУВ значительно упрощает организацию выпусков программной системы: за счет использования меток (*tag*) понятно, какие версии компонентов ПО интегрируются друг с другом.

Выделяют два типа СУВ:

В **централизованных системах** информация о всех версиях компонентов хранится на выделенном сервере в репозитории. Разработчики запрашивают данные из репозитория, создавая рабочие копии (фактически, песочницы), в которых проводят изменения и загружают файлы обратно на сервер.

В **распределенных системах** рабочие копии равнозначны и содержат всю историю правок, информацию о ветвях и так далее. Обычно одна из рабочих копий находится на удаленном хостинге (например, GitHub в случае СУВ Git), однако ее может и не существовать. Есть мнение, что создание распределенных систем управления версиями в начале XXI века сыграло важную роль в развитии программного обеспечения..

**GIT**

В Git'е файлы могут находиться в одном из трёх состояний: зафиксированном, изменённом и подготовленном. "Зафиксированный" значит, что файл уже сохранён в вашей локальной базе. К изменённым относятся файлы, которые поменялись, но ещё не были зафиксированы. Подготовленные файлы — это изменённые файлы, отмеченные для включения в следующий коммит.

Для совершения большинства операций в Git'е необходимы только локальные файлы и ресурсы, т.е. обычно информация с других компьютеров в сети не нужна. Если вы пользовались централизованными системами, где практически на каждую операцию накладывается сетевая задержка, вы, возможно, подумаете, что боги наделили Git неземной силой. Поскольку вся история проекта хранится локально у вас на диске, большинство операций кажутся практически мгновенными.

К примеру, чтобы показать историю проекта, Git'у не нужно скачивать её с сервера, он просто читает её прямо из вашего локального репозитория. Поэтому историю вы увидите практически мгновенно. Если вам нужно просмотреть изменения между текущей версией файла и версией, сделанной месяц назад, Git может взять файл месячной давности и вычислить разницу на месте, вместо того чтобы запрашивать разницу у СКВ-сервера или качать с него старую версию файла и делать локальное сравнение.

Кроме того, работа локально означает, что мало чего нельзя сделать без доступа к Сети или VPN. Если вы в самолёте или в поезде и хотите немного поработать, можно спокойно делать коммиты, а затем отправить их, как только станет доступна сеть. Если вы пришли домой, а VPN-клиент не работает, всё равно можно продолжать работать. Во многих других системах это невозможно или же крайне неудобно. Например, используя Perforce, вы мало что можете сделать без соединения с сервером. Работая с Subversion и CVS, вы можете редактировать файлы, но сохранить изменения в вашу базу данных нельзя (потому что она отключена от репозитория). Вроде ничего серьёзного, но потом вы удивитесь, насколько это меняет дело.

Таким образом, в проектах, использующих Git, есть три части: каталог Git'а (Git directory), рабочий каталог (working directory) и область подготовленных файлов (staging area).

Установить Git в Windows очень просто. У проекта msysGit процедура установки — одна из самых простых. Просто скачайте exe-файл инсталлятора со страницы проекта на GitHub'е и запустите его:

**Задание**

**Часть А**

Необходимо в отчете подробно расписать:

1. процесс установки Git;
2. настройку Git;
3. добавления файлов и их коммит;
4. добавления файлов на GitHub;
5. создание клона существующего репозитория.

**Часть Б**

1. Создать приватный репозиторий;
2. Добавить в репозиторий несколько пользователей;
3. Пользователь должны загрузить по 1 файлу в репозиторий;
4. Проверить журнал изменений репозитория.