# Начало работы

**Управление версиями**

Что такое «управление версиями» и почему оно должно нас интересовать? Система управления версиями, или контроля версий, записывает историю изменения файла или набора файлов, чтобы в будущем была возможность вернуться к конкретной версии. В книге процесс управления версиями будет рассматриваться на фрагментах кода, в реальности же эти операции можно проделывать с файлами практически всех типов.

**Локальные системы контроля версий**

Многие люди в качестве метода контроля версий выбирают простое копирование файлов в другую папку (в лучшем случае такие папки помечаются метками времени). Этот подход крайне популярен в силу своей простоты, но при этом он потрясающе ненадежен. Легко забыть, в какой папке вы работаете, и случайно сделать запись не в тот файл или скопировать вовсе не те файлы, которые вы хотели.

**Централизованные системы контроля версий**

Следующая большая проблема — необходимость сотрудничать с разработчиками других систем. Для ее решения были разработаны централизованные системы контроля версий (Centralized Version Control System, CVCS).

Такая схема имеет много преимуществ, особенно перед локальными системами контроля версий. Например, каждый человек, работающий над проектом, до определенной степени знает, чем занимаются его коллеги. Администраторы могут детально контролировать права допуска прочих сотрудников; администрировать CVCS намного проще, чем управлять локальными базами данных на каждом клиенте.

Однако есть у этой схемы и серьезные недостатки. Самым очевидным является единая точка отказа, представленная центральным сервером. Отключение этого сервера на час означает, что в течение часа любые взаимодействия невозможны.

При повреждении жесткого диска центральной базы данных и отсутствии нужных резервных копий теряется вся информация — вся история разработки проекта за исключением единичных снимков состояния, которые могут остаться на локальных компьютерах пользователей. Впрочем, та же самая проблема характерна и для локальных систем контроля версий — храня историю разработки проекта в одном месте, вы рискуете потерять все.

**Распределенные системы контроля версий**

Именно здесь на первый план выходят распределенные системы контроля версий (Distributed Version Control System, DVCS). В DVCS (к примеру, Git, Mercurial, Bazaar или Darcs) клиенты не просто выгружают последние снимки файлов, они создают полную зеркальную копию репозитория. Соответственно в случае выхода из строя одного из серверов его работоспособность можно восстановить, скопировав один из клиентских репозиториев. Каждая такая выгрузка сопровождается полным резервным копированием всех данных.

**Краткая история Git**

Ядро Linux представляет собой крайне масштабный проект ПО с открытым исходным кодом. В истории поддержки ядра Linux изменения программ долгое время передавались в виде исправлений (patches) и архивированных файлов. В 2002 году для проекта Linux стали использовать собственную систему DVCS, которая называлась BitKeeper.

В 2005 году отношения между сообществом, разрабатывавшим ядро Linux, и коммерческой фирмой, создавшей BitKeeper, были разорваны и бесплатное использование этой системы контроля версий стало невозможным, что побудило сообщество разработчиков Linux (и в частности создателя этой операционной системы Линуса Торвальдса) начать работу над собственным инструментом, взяв за основу некоторые идеи BitKeeper. Вот цели, которые ставились для новой системы:

* быстродействие;
* простое проектное решение;
* мощная поддержка нелинейной разработки (тысячи параллельных ветвей);
* полностью распределенная система;
* возможность эффективной (в плане быстродействия и объема данных) работы с большими проектами, такими как ядро Linux.

С момента своего появления в 2005 году система Git развивалась и совершенствовалась в попытках добиться простоты использования при сохранении изначальных характеристик. Она работает необыкновенно быстро, крайне эффективна для больших проектов и обладает потрясающей ветвящейся системой нелинейной разработки.

**Основы Git**

Система Git представляют собой набор снимков состояния миниатюрной файловой системы. Каждый раз, когда вы создаете новую версию или сохраняете состояние проекта в Git, по сути, делается снимок всех файлов в конкретный момент времени и сохраняется ссылка на этот снимок. Для повышения продуктивности вместо файлов, которые не претерпели изменений, сохраняется всего лишь ссылка на их ранее сохраненные версии. Git воспринимает данные скорее, как поток снимков состояния (stream of snapshots).

Git больше напоминает не простую систему контроля версий, а миниатюрную файловую систему с удивительно мощным инструментарием.

Для осуществления практически всех операций системе Git требуются только локальные файлы и ресурсы — в общем случае информация с других компьютеров сети не нужна. Когда вся история проекта хранится на локальном диске, кажется, что большинство операций выполняется почти мгновенно.

**Целостность Git**

В системе Git для всех данных перед сохранением вычисляется контрольная сумма, по которой они впоследствии ищутся. То есть сохранить содержимое файла или папки таким образом, чтобы система Git об этом не узнала, невозможно. Эта функциональность встроена в Git на самом низком уровне и является неотъемлемым принципом ее работы. Невозможно потерять информацию или повредить файл скрытно от Git.

Механизм, которым пользуется Git для вычисления контрольных сумм, называется хешем SHA-1. Это строка из 40 символов, включающая в себя числа в шестнадцатеричной системе (0–9 и a–f) и вычисляемая на основе содержимого файла или структуры папки в Git. Хеш SHA-1 выглядит примерно так:

24b9da6552252987aa493b52f8696cd6d3b00373

Вы будете постоянно наталкиваться на эти хеш-значения, так как Git использует их повсеместно. По сути, Git сохраняет данные в базе не по именам файлов, а по хешу их содержимого.

**Три состояния**

Файлы в Git могут находиться в трех основных состояниях: зафиксированном, модифицированном и индексированном.

Зафиксированное (committed) состояние означает, что данные надежно сохранены в локальной базе. Модифицированное (modified) состояние означает, что изменения уже внесены в файл, но пока не зафиксированы в базе данных. Индексированное (staged) состояние означает, что вы пометили текущую версию модифицированного файла как предназначенную для следующей фиксации.

**Установка Git**

Перед тем как приступить к работе с Git, эту систему следует установить на компьютер. Если она у вас уже установлена, имеет смысл обновить ее до последней версии.

**Установка в Windows**

Установка Git в операционной системе Windows также осуществляется разными способами. Официальный вариант системы доступен на сайте Git. Достаточно зайтина страницу http://git-scm.com/download/win, и загрузка начнется автоматически. Обратите внимание, что этот проект называется Git для Windows (или msysGit) и отличается от Git; дополнительная информация находится на сайте http://msysgit.github.io/.

Другой простой способ — это установка GitHub для Windows. Пакет установки включает в себя версию как с командной строкой, так и с GUI. Он хорошо работает с оболочкой Powershell и обеспечивает надежное кэширование учетных данных и работоспособные настройки CRLF. Подробно мы рассмотрим эти вещи чуть позже, пока же достаточно сказать, что они вам потребуются. Загрузить эту версию можно с сайта <http://windows.github.com>.

Настройки каждого следующего уровня переопределяют настройки предыдущего, то есть конфигурация из .git/config перекроет конфигурацию из /etc/gitconfig.

В системах семейства Windows Git ищет файл .gitconfig в папке $HOME (в большинстве случаев она вложена в папку C:\Users\$USER). Кроме того, ищется файл /etc/gitconfig, но уже относительно корневого каталога MSys, расположение которого вы сами выбираете при установке Git.

# Основы Git

**Создание репозитория в Git**

Есть два подхода к созданию Git-проекта. Можно взять существующий проект или папку и импортировать в Git. А можно клонировать уже существующий репозиторий с другого сервера.

**Инициализация репозитория в существующей папке**

Чтобы начать слежение за существующим проектом, перейдите в папку этого проекта и введите команду

$ git init

В результате в существующей папке появится еще одна папка с именем .git и всеми нужными вам файлами репозитория — это будет основа вашего Git-репозитория. Пока контроль ваших файлов отсутствует

Чтобы начать управление версиями существующих файлов (в противовес пустому каталогу), укажите файлы, за которыми должна следить система, и выполните первую фиксацию изменений. Для этого потребуется несколько команд git add, добавляющих файлы, за которыми вы хотите следить, а затем команда git commit:

$ git add \*.c

$ git add LICENSE

$ git commit -m 'первоначальная версия проекта'

**Клонирование существующего репозитория**

Получение копии существующего репозитория, например проекта, в котором вы хотите принять участие, выполняется командой git clone. Команда git clone по умолчанию забирает все версии всех файлов за всю историю проекта. Это означает, что при повреждении серверного диска практически любой клон на любом из клиентов может использоваться для возвращения сервера в состояние, в котором он пребывал до момента клонирования

Клонирование репозитория осуществляется командой git clone [url].

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

или

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

**Запись изменений в репозиторий**

Итак, у вас есть настоящий Git-репозиторий и некая выгрузка, то есть рабочие копии файлов нашего проекта. Теперь в файлы можно вносить изменения и фиксировать их, как только проект достигнет состояния, которое вы хотели бы сохранить.

Помните, что каждый файл в рабочей папке может пребывать в одном из двух состояний: отслеживаемом и неотслеживаемом. Первый случай — это файлы, входящие в последний снимок системы; они могут быть неизмененными, измененными и подготовленными к фиксации. Второй случай — это все остальные файлы рабочей папки, не вошедшие в последний снимок системы и не проиндексированные для последующей фиксации. После первого клонирования репозитория все файлы оказываются отслеживаемыми и неизмененными, потому что вы просто выгрузили их и пока не отредактировали.