



--distributed-even-if-your-workflow-isnt

- [About](#)
- [Documentation](#)
  - [Reference](#)
  - [Book](#)
  - [Videos](#)
  - [External Links](#)
- [Blog](#)
- [Downloads](#)
  - [GUI Clients](#)
  - [Logos](#)
- [Community](#)

---

This book is translated into [Deutsch](#), [简体中文](#), [正體中文](#), [Français](#), [日本語](#), [Nederlands](#), [Русский](#), [한국어](#), [Português \(Brasil\)](#) and [Čeština](#).

Partial translations available in [Arabic](#), [Español](#), [Indonesian](#), [Italiano](#), [Suomi](#), [Македонски](#), [Polski](#) and [Türkçe](#).

Translations started for [Azərbaycan dili](#), [Беларуская](#), [Català](#), [Esperanto](#), [Español \(Nicaragua\)](#), [فارسی](#), [हिन्दी](#), [Magyar](#), [Norwegian Bokmål](#), [Română](#), [Српски](#), [ภาษาไทย](#), [Tiếng Việt](#), [Українська](#) and [Ўзбекча](#).

---

The source of this book is [hosted on GitHub](#).  
Patches, suggestions and comments are welcome.

[Chapters ▾](#)

## 1. [1. Введение](#)

- 1. 1.1 [О контроле версий](#)
- 2. 1.2 [Краткая история Git](#)
- 3. 1.3 [Основы Git](#)
- 4. 1.4 [Установка Git](#)
- 5. 1.5 [Первоначальная настройка Git](#)
- 6. 1.6 [Как получить помощь?](#)
- 7. 1.7 [Итоги](#)

## 2. [2. Основы Git](#)

- 1. 2.1 [Создание Git-репозитория](#)
- 2. 2.2 [Запись изменений в репозиторий](#)
- 3. 2.3 [Просмотр истории коммитов](#)
- 4. 2.4 [Отмена изменений](#)
- 5. 2.5 [Работа с удалёнными репозиториями](#)
- 6. 2.6 [Работа с метками](#)

7. 2.7 [Полезные советы](#)

8. 2.8 [Итоги](#)

### 3. [3. Ветвление в Git](#)

1. 3.1 [Что такое ветка?](#)

2. 3.2 [Основы ветвления и слияния](#)

3. 3.3 [Управление ветками](#)

4. 3.4 [Приёмы работы с ветками](#)

5. 3.5 [Удалённые ветки](#)

6. 3.6 [Перемещение](#)

7. 3.7 [Итоги](#)

### 1. [4. Git на сервере](#)

1. 4.1 [Протоколы](#)

2. 4.2 [Настройка Git на сервере](#)

3. 4.3 [Создание открытого SSH-ключа](#)

4. 4.4 [Настраиваем сервер](#)

5. 4.5 [Открытый доступ](#)

6. 4.6 [GitWeb](#)

7. 4.7 [Gitis](#)

8. 4.8 [Gitolite](#)

9. 4.9 [Git-демон](#)

10. 4.10 [Git-хостинг](#)

11. 4.11 [Итоги](#)

### 2. [5. Распределённый Git](#)

1. 5.1 [Распределённые рабочие процессы](#)

2. 5.2 [Содействие проекту](#)

3. 5.3 [Сопровождение проекта](#)

4. 5.4 [Итоги](#)

### 3. [6. Инструменты Git](#)

1. 6.1 [Выбор ревизии](#)

2. 6.2 [Интерактивное индексирование](#)

3. 6.3 [Прятанье](#)

4. 6.4 [Перезапись истории](#)

5. 6.5 [Отладка с помощью Git](#)

6. 6.6 [Подмодули](#)

7. 6.7 [Слияние поддеревьев](#)

8. 6.8 [Итоги](#)

### 1. [7. Настройка Git](#)

1. 7.1 [Конфигурирование Git](#)

2. 7.2 [Git-атрибуты](#)

3. 7.3 [Перехватчики в Git](#)

4. 7.4 [Пример навязывания политики с помощью Git](#)

5. 7.5 [Итоги](#)

## 2. [8. Git и другие системы контроля версий](#)

1. 8.1 [Git и Subversion](#)

2. 8.2 [Миграция на Git](#)

3. 8.3 [Итоги](#)

## 3. [9. Git изнутри](#)

1. 9.1 [Сантехника и фарфор](#)

2. 9.2 [Объекты в Git](#)

3. 9.3 [Ссылки в Git](#)

4. 9.4 [Раск-файлы](#)

5. 9.5 [Спецификации ссылок](#)

6. 9.6 [Протоколы передачи](#)

7. 9.7 [Обслуживание и восстановление данных](#)

8. 9.8 [Итоги](#)

1st Edition

# 1.1 Введение - О контроле версий

## О контроле версий

Что такое контроль версий, и зачем он вам нужен? Система контроля версий (СКВ) — это система, регистрирующая изменения в одном или нескольких файлах с тем, чтобы в дальнейшем была возможность вернуться к определённым старым версиям этих файлов. Для примеров в этой книге мы будем использовать исходные коды программ, но на самом деле под версионный контроль можно поместить файлы практически любого типа.

Если вы графический или веб-дизайнер и хотели бы хранить каждую версию изображения или макета — а этого вам наверняка хочется — то пользоваться системой контроля версий будет очень мудрым решением. СКВ даёт возможность возвращать отдельные файлы к прежнему виду, возвращать к прежнему состоянию весь проект, просматривать происходящие со временем изменения, определять, кто последним вносил изменения во внезапно переставший работать модуль, кто и когда внёс в код какую-то ошибку, и многое другое. Вообще, если, пользуясь СКВ, вы всё испортите или потеряете файлы, всё можно будет легко восстановить. Вдобавок, накладные расходы за всё, что вы получаете, будут очень маленькими.

## [Локальные системы контроля версий](#)

Многие предпочитают контролировать версии, просто копируя файлы в другой каталог (как правило добавляя текущую дату к названию каталога). Такой подход очень распространён, потому что прост, но он и чаще даёт сбои. Очень легко забыть, что ты не в том каталоге, и случайно изменить не тот файл, либо скопировать файлы не туда, куда хотел, и затереть нужные файлы.

Чтобы решить эту проблему, программисты уже давно разработали локальные СКВ с простой базой данных, в которой хранятся все изменения нужных файлов (см. рисунок 1-1).

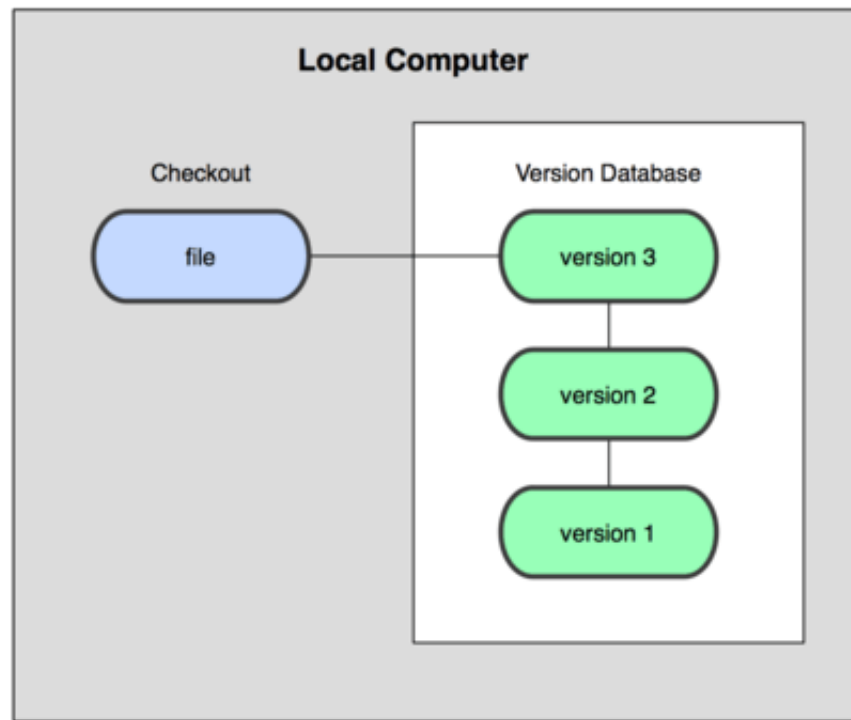


Рисунок 1-1. Схема локальной СКВ.

Одной из наиболее популярных СКВ такого типа является rcs, которая до сих пор устанавливается на многие компьютеры. Даже в современной операционной системе Mac OS X утилита rcs устанавливается вместе с Developer Tools. Эта утилита основана на работе с наборами патчей между парами версий (патч — файл, описывающий различие между файлами), которые хранятся в специальном формате на диске. Это позволяет пересоздать любой файл на любой момент времени, последовательно накладывая патчи.

### Централизованные системы контроля версий

Следующей основной проблемой оказалась необходимость сотрудничать с разработчиками за другими компьютерами. Чтобы решить её, были созданы централизованные системы контроля версий (ЦСКВ). В таких системах, например CVS, Subversion и Perforce, есть центральный сервер, на котором хранятся все файлы под версионным контролем, и ряд клиентов, которые получают копии файлов из него. Много лет это было стандартом для систем контроля версий (см. рис. 1-2).

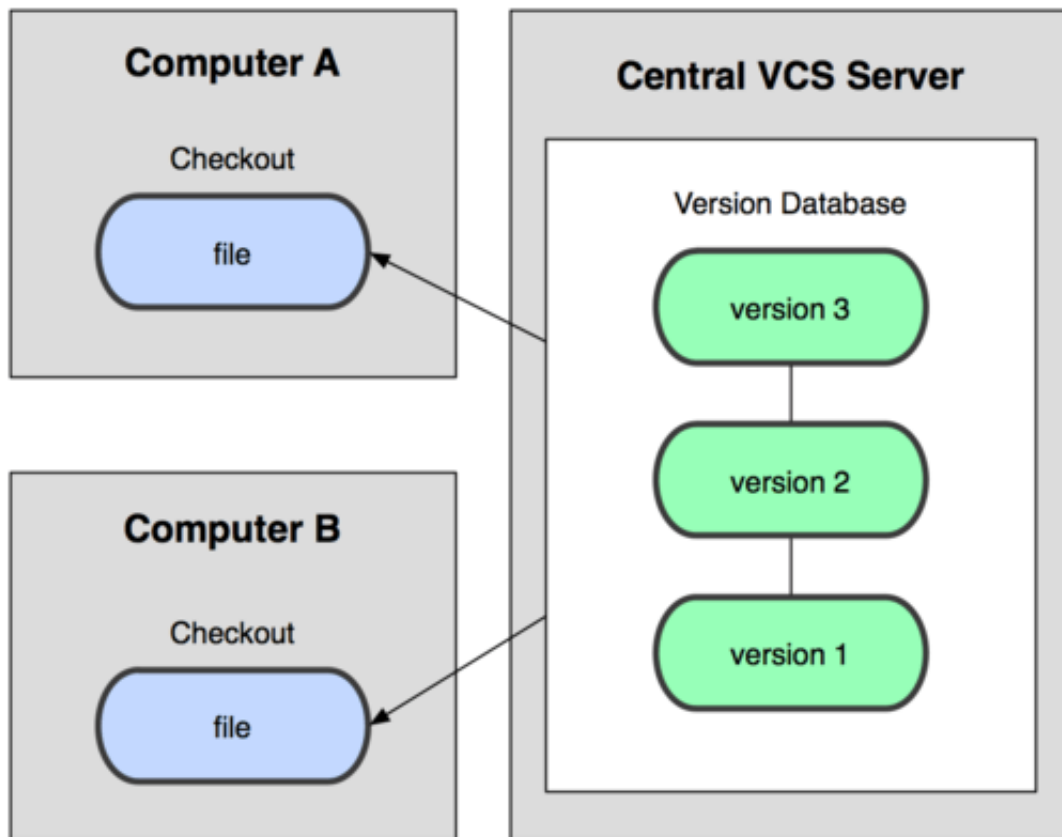


Рисунок 1-2. Схема централизованного контроля версий.

Такой подход имеет множество преимуществ, особенно над локальными СКВ. К примеру, все знают, кто и чем занимается в проекте. У администраторов есть чёткий контроль над тем, кто и что может делать, и, конечно, администрировать ЦСКВ намного легче, чем локальные базы на каждом клиенте.

Однако при таком подходе есть и несколько серьёзных недостатков. Наиболее очевидный — централизованный сервер является уязвимым местом всей системы. Если сервер выключается на час, то в течение часа разработчики не могут взаимодействовать, и никто не может сохранить новой версии своей работы. Если же повреждается диск с центральной базой данных и нет резервной копии, вы теряете абсолютно всё — всю историю проекта, разве что за исключением нескольких рабочих версий, сохранившихся на рабочих машинах пользователей. Локальные системы контроля версий подвержены той же проблеме: если вся история проекта хранится в одном месте, вы рискуете потерять всё.

## Распределённые системы контроля версий

И в этой ситуации в игру вступают распределённые системы контроля версий (РСКВ). В таких системах как Git, Mercurial, Bazaar или Darcs клиенты не просто выгружают последние версии файлов, а полностью копируют весь репозиторий. Поэтому в случае, когда "умирает" сервер, через который шла работа, любой клиентский репозиторий может быть скопирован обратно на сервер, чтобы восстановить базу данных. Каждый раз, когда клиент забирает свежую версию файлов, он создаёт себе полную копию всех данных (см. рисунок 1-3).

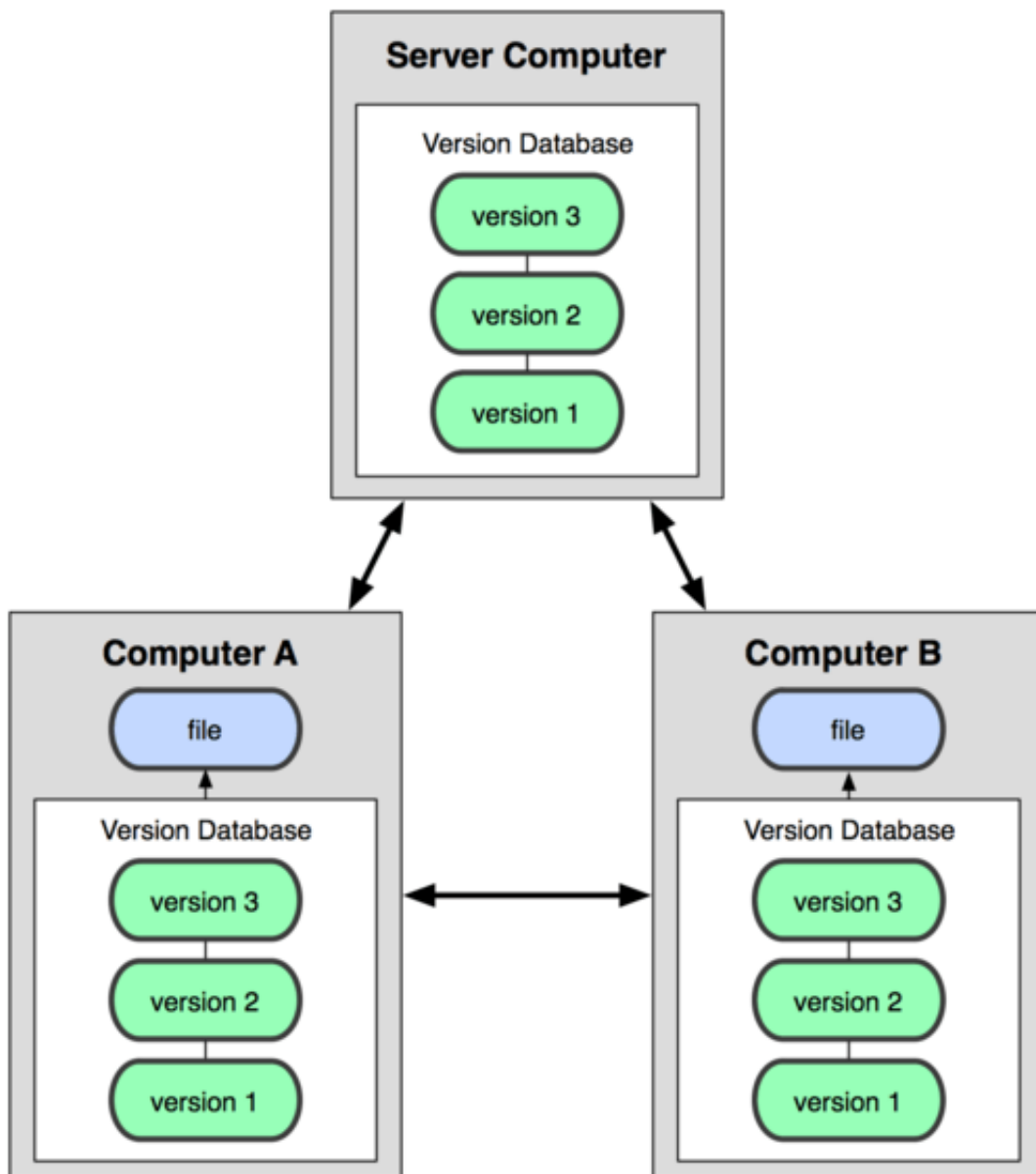


Рисунок 1-3. Схема распределённой системы контроля версий.

Кроме того, в большей части этих систем можно работать с несколькими удалёнными репозиториями, таким образом, можно одновременно работать по-разному с разными группами людей в рамках одного проекта. Так, в одном проекте можно одновременно вести несколько типов рабочих процессов, что невозможно в централизованных системах.

[prev](#) | [next](#)

This [open sourced](#) site is [hosted on GitHub](#).

Patches, suggestions and comments are welcome.

Git is a member of [Software Freedom Conservancy](#)