

Git

Материал из Википедии — свободной энциклопедии

Git (произн. «гит»^[6]) — распределённая система управления версиями. Проект был создан Линусом Торвалдсом для управления разработкой ядра Linux, первая версия выпущена 7 апреля 2005 года. На сегодняшний день его поддерживает Джуннио Хамано.

Примерами проектов, использующих Git, являются Ядро Linux, Swift, Android, Drupal, Cairo, GNU Core Utilities, Mesa, Wine, Chromium, Compiz Fusion, FlightGear, jQuery, PHP, NASM, MediaWiki, DokuWiki, Qt и некоторые дистрибутивы Linux (см. ниже).

Программа является свободной и выпущена под лицензией GNU GPL версии 2.

Содержание

История

История версий

Возможности

Особенности реализации

Архитектура

Детали реализации в Windows

Сетевые возможности и серверные решения

Преимущества и недостатки

Графические интерфейсы

Фронтенды для Web

Обмен изменениями с другими системами контроля версий

Примечания

См. также

Ссылки

История

Разработка ядра Linux велась на проприетарной системе BitKeeper^[7], которую автор, Ларри Маквой, сам разработчик


Linux, дал по бесплатной лицензии. Разработчики, высококлассные программисты, написали несколько утилит, и для одной Эндрю Триджелл опознал формат передачи данных BitKeeper. В ответ Маквой обвинил разработчиков в нарушении

Git



Тип	<u>распределённая</u> <u>система</u> <u>управления версиями</u> ^{[d][4]}
Разработчик	<u>Линус Торвальдс</u> ^[1] и <u>Джунио</u> <u>Хамано</u>
<u>Написана на</u>	<u>Си</u> , <u>командная оболочка</u> <u>UNIX</u> , <u>Perl</u> , <u>Tcl</u> и <u>bash</u>
<u>Операционная</u> <u>система</u>	<u>кроссплатформенное</u> <u>программное обеспечение</u>
<u>Языки</u> <u>интерфейса</u>	<u>английский</u>
Первый выпуск	<u>7 апреля 2005</u> ^[2]
Последняя версия	<u>2.16.2</u> <u>(15 февраля 2018)</u> ^[3]
Читаемые форматы файлов	<u>git packfile</u> ^[d]
Создаваемые форматы файлов	<u>git packfile</u> ^[d]
<u>Лицензия</u>	<u>GNU GPL 2</u> ^[5]
Сайт	<u>git-scm.com</u> (англ.)



 **Git** на Викискладе

соглашения и отозвал лицензию, и Торвальдс взялся за новую систему: ни одна из открытых систем не позволяла тысячам программистов кооперировать свои усилия (тот же конфликт привёл к написанию Mercurial). Идеология была проста: взять идеологию CVS и перевернуть с ног на голову^[8], и заодно добавить надёжности.

Начальная разработка велась меньше, чем неделю: 3 апреля 2005 года разработка началась, и уже 7 апреля код Git управлялся неготовой системой. 16 июня Linux был переведён на Git, а 25 июля Торвальдс отказался от обязанностей ведущего разработчика.

Торвальдс так саркастически отозвался о выбранном им названии git (что на английском сленге означает «мерзавец»): «Я эгоистичный ублюдок, и поэтому называю все свои проекты в честь себя. Сначала Linux, теперь git.»^{[9][10]}

История версий

Версия	Первоначальная дата выпуска	Последняя версия	Дата выпуска
0.99	2005-07-11	0.99.9n	2005-12-15
1.0	2005-12-21	1.0.13	2006-01-27
1.1	2006-01-08	1.1.6	2006-01-30
1.2	2006-02-12	1.2.6	2006-04-08
1.3	2006-04-18	1.3.3	2006-05-16
1.4	2006-06-10	1.4.4.5	2008-07-16
1.5	2007-02-14	1.5.6.6	2008-12-17
1.6	2008-08-17	1.6.6.3	2010-12-15
1.7	2010-02-13	1.7.12.4	2012-10-17
1.8	2012-10-21	1.8.5.6	2014-12-17
1.9	2014-02-14	1.9.5	2014-12-17
2.0	2014-05-28	2.0.5	2014-12-17
2.1	2014-08-16	2.1.4	2014-12-17
2.2	2014-11-26	2.2.3	2015-09-04
2.3	2015-02-05	2.3.10	2015-09-29
2.4	2015-04-30	2.4.12	2017-05-05
2.5	2015-07-27	2.5.6	2017-05-05
2.6	2015-09-28	2.6.7	2017-05-05
2.7	2015-10-04	2.7.5	2017-05-05
2.8	2016-03-28	2.8.5	2017-05-05
2.9	2016-06-13	2.9.4	2017-05-05
2.10	2016-09-02	2.10.3	2017-05-05
2.11	2016-11-29	2.11.2	2017-05-05
2.12	2017-02-24	2.12.3	2017-05-05
2.13	2017-05-10	2.13.4	2017-08-01
2.14	2017-08-04	2.14.3	2017-10-24
2.15	2017-10-30	2.15.1	2017-10-30
2.16	2018-01-17	2.16.1	2018-01-22
Легенда: Старая версия, не поддерживается Старая поддерживаемая версия Текущая версия Тестовая версия			

Возможности

Система спроектирована как набор программ, специально разработанных с учётом их использования в скриптах. Это позволяет удобно создавать специализированные системы контроля версий на базе Git или пользовательские интерфейсы. Например, Cogito является именно таким примером оболочки к репозиториям Git, а StGit использует Git для управления коллекцией исправлений (патчей).

Git поддерживает быстрое разделение и слияние версий, включает инструменты для визуализации и навигации по нелинейной истории разработки. Как и Darcs, BitKeeper, Mercurial, Bazaar и Monotone, Git предоставляет каждому разработчику локальную копию всей истории разработки, изменения копируются из одного репозитория в другой.

Удалённый доступ к репозиториям Git обеспечивается `git-daemon`, SSH- или HTTP-сервером. TCP-сервис `git-daemon` входит в дистрибутив Git и является наряду с SSH наиболее распространённым и надёжным методом доступа. Метод доступа по HTTP, несмотря на ряд ограничений, очень популярен в контролируемых сетях, потому что позволяет использовать существующие конфигурации сетевых фильтров.

Особенности реализации

Ядро Git представляет собой набор утилит командной строки с параметрами. Все настройки хранятся в текстовых файлах конфигурации. Такая реализация делает Git легко портируемым на любую платформу и даёт возможность легко интегрировать Git в другие системы (в частности, создавать графические git-клиенты с любым желаемым интерфейсом).

Репозиторий Git представляет собой каталог файловой системы, в котором находятся файлы конфигурации репозитория, файлы журналов, хранящие операции, выполняемые над репозиторием, индекс, описывающий расположение файлов, и хранилище, содержащее собственно файлы. Структура хранилища файлов не отражает реальную структуру хранящегося в репозитории файлового дерева, она ориентирована на повышение скорости выполнения операций с репозиторием. Когда ядро обрабатывает команду изменения (неважно, при локальных изменениях или при получении патча от другого узла), оно создаёт в хранилище новые файлы, соответствующие новым состояниям изменённых файлов. Существенно, что никакие операции не изменяют содержимого уже существующих в хранилище файлов.

По умолчанию репозиторий хранится в подкаталоге с названием «.git» в корневом каталоге рабочей копии дерева файлов, хранящегося в репозитории. Любое файловое дерево в системе можно превратить в репозиторий git, отдав команду создания репозитория из корневого каталога этого дерева (или указав корневой каталог в параметрах программы). Репозиторий может быть импортирован с другого узла, доступного по сети. При импорте нового репозитория автоматически создаётся рабочая копия, соответствующая последнему зафиксированному состоянию импортируемого репозитория (то есть не копируются изменения в рабочей копии исходного узла, для которых на том узле не была выполнена команда `commit`).

Архитектура

Нижний уровень git является т. н. файловой системой с адресацией по содержимому (content-addressed file system). Инструмент командной строки git содержит ряд команд по непосредственной манипуляции этим репозиторием на низком уровне. Эти команды не нужны при нормальной работе с git как с системой контроля версий, но нужны для реализации сложных операций (ремонт поврежденного репозитория и т. д.), а также дают возможность создать на базе репозитория git своё приложение.

Для каждого объекта в репозитории вычисляется SHA-1-хеш, и именно он становится именем файла, содержащего данный объект в директории `.git/objects`. Для оптимизации работы с файловыми системами, не использующими деревья для директорий, первый байт хеша становится именем поддиректории, а остальные — именем файла в ней, что снижает количество файлов в одной директории (лимитирующий фактор производительности на таких устаревших файловых системах).

Все ссылки на объекты репозитория, включая ссылки на один объект, находящийся внутри другого объекта, являются SHA-1-хешами.

Кроме того, в репозитории существует директория `refs`, которая позволяет задать читаемые человеком имена для каких-то объектов git. В командах git оба вида ссылок — читаемые человеком из `refs`, и нижележащие SHA-1 — полностью взаимозаменяемы.

В классическом обычном сценарии в репозитории git есть три типа объектов — файл, дерево и коммит. Файл есть какая-то версия какого-то пользовательского файла, дерево — совокупность файлов из разных поддиректорий, коммит — дерево и некая дополнительная информация (например, родительский(е) коммит(ы), а также комментарий).

В репозитории иногда производится сборка мусора, во время которой устаревшие файлы заменяются на «дельты» между ними и актуальными файлами (именно так! актуальная версия файла хранится не-инкрементально, инкременты используются только для шагания назад), после чего данные «дельты» складываются в один большой файл, к которому строится индекс. Это снижает требования по месту на диске.

Репозиторий git бывает локальный и удаленный. Локальный репозиторий — это поддиректория .git, создается (в пустом виде) командой `git init` и (в непустом виде с немедленным копированием содержимого родительского удаленного репозитория и простановкой ссылки на родителя) командой `git clone`.

Практически все обычные операции с системой контроля версий, такие, как коммит и слияние, производятся только с локальным репозиторием. Удаленный репозиторий можно только синхронизировать с локальным как «вверх» (`push`), так и «вниз» (`pull`).

Наличие полностью всего репозитория проекта локально у каждого разработчика дает git ряд преимуществ перед SVN. Так, например, все операции, кроме `push` и `pull`, можно осуществлять без наличия Интернет-соединения.

Очень мощной возможностью git являются ветви, реализованные куда более полно, чем в SVN. Создать новую ветвь так же просто, как и совершить коммит. По сути, ветвь git есть не более чем именованная ссылка, указывающая на некий коммит в репозитории (используется поддиректория `refs`). Коммит без создания новой ветви всего лишь передвигает эту ссылку на себя, а коммит с созданием ветви — оставляет старую ссылку на месте, но создает новую на новый коммит, и объявляет её текущей. Заменить локальные девелоперские файлы на набор файлов из иной ветви, тем самым перейдя к работе с ней — так же тривиально.

Также поддерживаются суб-репозитории с синхронизацией текущих ветвей в них.

Команда `push` передает все новые данные (те, которых ещё нет в удаленном репозитории) из локального репозитория в репозиторий удаленный. Для исполнения этой команды необходимо, чтобы удаленный репозиторий не имел новых коммитов в себя от других клиентов, иначе `push` завершается ошибкой, и придется делать `pull` и слияние.

Команда `pull` — обратна команде `push`. В случае, если одна и та же ветвь имеет независимую историю в локальной и в удаленной копии, `pull` немедленно переходит к слиянию.

Слияние в пределах разных файлов осуществляется автоматически (все это поведение настраивается), а в пределах одного файла — стандартным трёхпанельным сравнением файлов. После слияния нужно объявить конфликты как разрешенные.

Результатом всего этого является новое состояние в локальных файлах у того разработчика, что осуществил слияние. Ему нужно немедленно сделать коммит, при этом в данном объекте коммита в репозитории окажется информация о том, что коммит есть результат слияния двух ветвей и имеет два родительских коммита.

Кроме слияния, git поддерживает ещё и `rebase`. Эта операция есть получение набора всех изменений в ветви А, с последующим их «накатом» на ветвь В. В результате ветвь В продвигается до состояния АВ. В отличие от слияния, в истории ветви АВ не останется никаких промежуточных коммитов ветви А (только история ветви В и запись о самом `rebase`, это упрощает интеграцию крупных и очень крупных проектов).

Также git имеет временный локальный индекс файлов. Это — промежуточное хранилище между собственно файлами и очередным коммитом (коммит делается только из этого индекса). С помощью этого индекса осуществляется добавление новых файлов (`git add` добавляет их в индекс, они попадут в следующий коммит), а также коммит НЕ ВСЕХ измененных файлов (коммит делается только тем файлам, которым был сделан `git add`). После `git add` можно редактировать файл далее, получатся три копии одного и того же файла — последняя, в индексе (та, что была на момент `git add`), и в последнем коммите.

Имя ветви по умолчанию: `master`

Имя удаленного репозитория по умолчанию, создаваемое git clone во время типичной операции «взять имеющийся проект с сервера себе на машину»: origin.

Таким образом, в локальном репозитории всегда есть ветвь master, которая есть последний локальный коммит, и ветвь origin/master, которая есть последнее состояние удаленного репозитория на момент завершения исполнения последней команды pull или push.

Команда fetch (частичный pull) — берет с удаленного сервера все изменения в origin/master, и переписывает их в локальный репозиторий, продвигая метку origin/master

Если после этого master и origin/master разошлись в стороны, то необходимо сделать слияние, установив master на результат слияния (команда pull есть fetch+merge). Далее возможно сделать push, отправив результат слияния на сервер и установив на него origin/master

Детали реализации в Windows

В Windows версии (официальная Windows-версия называется mSysGit) используется пакет mSys — эмулятор POSIX-совместимой командной строки над Windows (производный от MinGW, примерный аналог Cygwin).

Под mSys перенесены все необходимые для git библиотеки и инструменты, а также сам git.

При работе с удаленными репозиториями по протоколу SSL будет использоваться хранилище сертификатов из mSys, а не из Windows.

Существует немало графических оболочек для Git, например, TortoiseGit. Все они реализованы через вызовы mSysGit и требуют его установки на машину. Не исключение и SourceTree, решение компании Atlassian, но mSysGit оно содержит внутри себя, что имеет свои плюсы и минусы (так установка в глубокую поддиректорию затрудняет добавление в mSys нужных SSL-сертификатов).

Сетевые возможности и серверные решения

git использует сеть только для операций обмена с удалёнными репозиториями.

Возможно использование следующих протоколов:

- git://, открытый протокол^[11], требующий наличия на сервере запущенного демона^[12] (поставляется вместе с git). Протокол не имеет средств аутентификации пользователей.
- ssh:// Использует аутентификацию пользователей с помощью пар ключей, а также встроенный в UNIX-систему «основной» SSH-сервер (sshd). Со стороны сервера требуется создание аккаунтов, шеллом у которых будет некая команда git.
- http(s)://. Использует внутри себя инструмент curl (для Windows — поставляется вместе с git), и его возможности HTTP-аутентификации, как и его поддержку SSL и сертификатов.

В последнем случае необходимо наличие на сервере некоего ПО веб-приложения, которое будет исполнять роль прослойки между командами git на сервере и HTTP-сервером. Такие прослойки существуют как под Linux, так и под Windows (например, WebGitNet, разработанный на ASP.NET MVC 4).

Кроме поддержки серверной стороны команд push и pull, такие веб-приложения могут также давать доступ только на чтение к репозиторию через веб-браузер.

Преимущества и недостатки

Преимущества и недостатки git по сравнению с централизованными системами управления версиями (такими как, например, Subversion) типичны для любой распределённой системы и описаны в статье «Система управления версиями». Если же сравнивать git с «родственными» ей распределёнными системами, можно отметить, что git изначально

идеологически ориентирован на работу с изменениями, а не с файлами, «единицей обработки» для него является набор изменений, или патч. Эта особенность прослеживается как в структуре самой системы (в частности — в структуре репозитория), так и в принципах построения команд; она отражается на производительности системы в различных вариантах её использования и на достоинствах и недостатках git по сравнению с другими DVCS.

Часто называемые преимущества git перед другими DVCS:

- Высокая производительность.
- Развитые средства интеграции с другими VCS, в частности, с CVS, SVN и Mercurial. Помимо разнонаправленных конвертеров репозитория, имеющиеся в комплекте программные средства позволяют разработчикам использовать git при размещении центрального репозитория в SVN или CVS, кроме того, git может имитировать cvs-сервер, обеспечивая работу через клиентские приложения и поддержку в средах разработки, специально не поддерживающих git.
- Продуманная система команд, позволяющая удобно встраивать git в скрипты.
- Репозитории git могут распространяться и обновляться общесистемными файловыми утилитами архивации и обновления, такими как rsync, благодаря тому, что фиксации изменений и синхронизации не меняют существующие файлы с данными, а только добавляют новые (за исключением некоторых служебных файлов, которые могут быть автоматически обновлены с помощью имеющихся в составе системы утилит). Для раздачи репозитория по сети достаточно любого веб-сервера.

В числе недостатков git обычно называют:

- Отсутствие сквозной нумерации коммитов монотонно непрерывно возрастающими целыми числами. Во многих проектах используется автоматическое получение номера этой версии (например, командой svnversion), построение .H файла на основе этого числа, и далее его использование при создании штампа версии исполняемого файла, некоторых вшитых в него строк и так далее.
- Отсутствие переносимой на другие операционные системы поддержки путей в кодировке Unicode в Microsoft Windows (для версий *msysgit* до 1.8.1). Если путь содержит символы, отличные от ANSI, то их поддержка из командной строки требует специфических настроек, которые не гарантируют правильного отображения файловых имён при использовании тем же репозиторием из других ОС. Одним из способов решения проблемы для git 1.7 является использование специально пропатченного консольного клиента. Другой вариант — использование графических утилит работающих напрямую через API, таких как TortoiseGit.
- Некоторое неудобство для пользователей, переходящих с других VCS. Команды git, ориентированные на наборы изменений, а не на файлы, могут вызвать недоумение у пользователей, привыкших к файл-ориентированным VCS, таким как SVN. Например, команда «add», которая в большинстве систем управления версиями производит добавление файла к проекту, в git подготавливает к фиксации сделанные в файлах изменения. При этом сохраняется не патч, описывающий изменения, а новая версия целевого файла.
- Использование для идентификации ревизий хешей SHA1, что приводит к необходимости оперировать длинными строками вместо коротких номеров версий, как во многих других системах (хотя в командах допускается использование неполных хеш-строк).
- Большие накладные расходы при работе с проектами, в которых делаются многочисленные несвязанные между собой изменения файлов. При работе в таком режиме размеры наборов изменений становятся достаточно велики и происходит быстрый рост объёма репозитория.
- Большие затраты времени, по сравнению с файл-ориентированными системами, на формирование истории конкретного файла, истории правок конкретного пользователя, поиска изменений, относящихся к заданному месту определённого файла.
- Отсутствие отдельной команды переименования/перемещения файла, которая отображалась бы в истории как соответствующее единое действие. Существующий скрипт git mv фактически выполняет переименование, копирование файла и удаление его на старом месте, что требует специального анализа для определения, что в действительности файл был просто перенесён (этот анализ выполняется автоматически командами просмотра истории). Однако, учитывая тот факт, что наличие специальной команды для переименования/перемещения файлов технически не вынуждает пользователя использовать именно её (и, как следствие, в этом случае возможны разрывы в истории), поведение git может считаться преимуществом.
- Система работает только с файлами и их содержимым, и не отслеживает пустые каталоги.

В ряде публикаций, относящихся преимущественно к 2005—2008 годам, можно встретить также нарекания в отношении документации git, отсутствия удобной windows-версии и удобных графических клиентов. В настоящее время эта критика неактуальна: существует версия git на основе MinGW («родная» сборка под Windows), и несколько высококачественных графических клиентов для различных операционных систем, в частности, под Windows имеется клиент TortoiseGit, идеологически очень близкий к широко распространённому TortoiseSVN — клиенту SVN, встраиваемому в оболочку Windows.

Графические интерфейсы

- [GitKraken](#) — кроссплатформенный бесплатный клиент Git.
- [SmartGit](#) — кроссплатформенный интерфейс для Git на Java.
- [gitk](#) — простая и быстрая программа, написанная на [Tcl/Tk](#), распространяется с самим Git.
- [QGit](#), интерфейс которого написан с использованием [Qt](#), во многом схож с [gitk](#), но несколько отличается набором возможностей. В настоящее время существуют реализации на [Qt3](#) и [Qt4](#).
- [Giggle](#) — вариант на [Gtk+](#).
- [gitg](#) — ещё один интерфейс для [gtk+/GNOME](#)
- [Git Extensions](#) — кроссплатформенный вариант на [.NET](#).
- [TortoiseGit](#) — интерфейс, реализованный как расширение для проводника Windows.
- [SourceTree](#) — бесплатный git клиент для Windows и Mac.
- [Git-cola](#) — кроссплатформенный интерфейс на [Python](#).
- [GitX](#) — оболочка для Mac OS X с интерфейсом [Cocoa](#), интерфейс схож с [gitk](#).
- [Gitti](#) — оболочка для Mac OS X с интерфейсом [Cocoa](#).
- [Gitbox](#) — оболочка для Mac OS X с интерфейсом [Cocoa](#).
- [Github](#)-клиент
- [StGit](#) — написанная на [Python](#) система управления коллекцией патчей (Catalin Marinas)
- [GitTower](#) — коммерческий клиент Git для Mac и Windows.

Фронтенды для Web

Название	Функции	Язык	Активность	Версия	Лицензия
GitWebAdmin	Управление репозиториями.	PHP	да	1.0.1 (15 октября 2016)	MIT
GitLab	Просмотр репозитория и истории изменений, управление репозиториями. Права доступа. Система непрерывного тестирования	Ruby	да	8.8.1 (23 мая 2016)	MIT
Gitblit	Просмотр репозитория и истории изменений, управление репозиториями. Права доступа	Java	да	1.7.1 (23 ноября 2015)	Apache License 2.0
Gerrit	Интеграция с репозиторием для организации совместной инспекции кода	Java	да	2.12.2 (11 марта 2015)	Apache License 2.0
Gitweb	Просмотр репозитория. Может работать как CGI скрипт в веб-сервере	Perl	да	поставляется с git	GPLv2
cgit	Просмотр репозитория и истории изменений	C	да	0.12 (14 января 2016)	GPLv2
ViewGit	Просмотр репозитория и истории изменений	PHP	да	0.0.7 (март 2013)	GNU AGPLv3
GitList	Просмотр репозитория и истории изменений	PHP	нет	июль 2006	New BSD license
git-php	Доработанная версия	PHP	нет	2011	GPLv2 (?)

	другого заброшенного проекта с одноимённым названием.				
<u>wit</u>	???	Python	нет	0.0.4 (сентябрь 2005)	GPLv2
<u>gitarella</u>	Просмотр репозитория и истории изменений	Ruby	нет	0.003 (июль 2006)	GPLv2
<u>Gogs</u>	Просмотр репозитория и истории изменений, управление репозиториями. Права доступа	Go	да	0.9.13 (19 марта 2016)	MIT License

Обмен изменениями с другими системами контроля версий

- CVS — импорт и экспорт, эмуляция CVS-сервера, в стандартной поставке
- Subversion — импорт и экспорт (частично), в стандартной поставке
- .tar.gz, .tar.bz2 (серии версионированных файлов) — импорт и экспорт, в стандартной поставке

Примечания

- ↑ <https://www.linux.com/blog/10-years-git-interview-git-creator-linus-torvalds>
 - ↑ <https://www.kernel.org/pub/software/scm/git/>
 - ↑ [git.git - The core git plumbing\(https://git.kernel.org/pub/scm/git/git.git/tag/?h=v2.16.2\)](https://git.kernel.org/pub/scm/git/git.git/tag/?h=v2.16.2)
 - ↑ <https://directoryfsf.org/wiki/Git>
 - ↑ <https://github.com/git/git/blob/master/COPYING>
 - ↑ [git \(http://dictionaryreference.com/browse/git?src=2446\)](http://dictionaryreference.com/browse/git?src=2446)
 - ↑ [BitKeeper and Linux: The end of the road?\(https://www.linux.com/news/bitkeeper-and-linux-end-road\)](https://www.linux.com/news/bitkeeper-and-linux-end-road)
 - ↑ [Выступление Торвальдса \(https://www.youtube.com/watch?v=4XpnKHJAok8\)](https://www.youtube.com/watch?v=4XpnKHJAok8)
 - ↑ [GitFaq: Why the 'Git' name\(https://git.wiki.kernel.org/index.php/GitFaq#Why_the_.27Git.27_name.3F\)](https://git.wiki.kernel.org/index.php/GitFaq#Why_the_.27Git.27_name.3F)
 - ↑ [After controversy, Torvalds begins work on 'git'\(http://www.pcworld.idg.com.au/article/12976/after_controversy_torvalds_begins_work_git_/\)](http://www.pcworld.idg.com.au/article/12976/after_controversy_torvalds_begins_work_git_/) PC World.
- Оригинальный текст* (англ.)
- When asked why he called the new software, "git," British slang meaning "a rotten person," he said. "I'm an egotistical bastard, so I name all my projects after myself. First Linux, now git."*
- ↑ [Git - Transfer Protocols\(http://git-scm.com/book/en/Git-Internals-Transfer-Protocols\)](http://git-scm.com/book/en/Git-Internals-Transfer-Protocols)
 - ↑ [Git на сервере - Git-демон\(https://git-scm.com/book/ru/v1/Git-на-сервере-Git-демон\)](https://git-scm.com/book/ru/v1/Git-на-сервере-Git-демон)

См. также

Другие распределённые системы управления версиями:

- Mercurial
- Bazaar
- Darcs

Сервисы, предоставляющие хостинг для git-репозиторияев:

- GitHub
- Codebase
- SourceForge
- Gitorious
- Google Code
- Bitbucket

Ссылки

- [Домашняя страница Git](#)(англ.)
- [Линус Торвальдс о git](#) (рус.) (YouTube)
- [Десять лет Git: интервью с создателем — Линус Торвальдс](#)
- [Интерактивный тур Git How To](#) (рус.)
- [Руководство пользователя GIT](#) (рус.)
- [Git User's Manual](#) (англ.) (также распространяется вместе с исходным текстом программы: `Documentation/user-manual.txt`)
- [Страница Git на kernel.org](#)(англ.)
- [Everyday Git](#) (англ.) («Git на каждый день») — набор из примерно 20 команд (на самом деле их будет около 5-6), которые пригодятся в повседневном использовании системы.
- [Linus Torvalds on Git](#)(англ.) — рассказ [Линуса Торвальдса](#) о git и других [системах контроля версий](#) (YouTube)
- [Randal Schwartz on Git](#)(англ.) — рассказ [Рэндела Шварца](#) о git (YouTube)
- [Contributing with Git](#)(англ.) — Google Talks 27.10.2008 (YouTube)
- [GitCasts.com](#) (англ.) — сайт, посвящённый скринкастам по использованию git.
- [Сравнение Git и Mercurial в FAQ сайта Google Code](#) (англ.)
- [C#-реализация Git](#) — системы контроля версий для .NET и Mono(англ.)
- Учебник [Pro Git](#) на русском языке (Creative Commons Attribution-Non Commercial-Share Alike 3.0 license).
- [Git на пальцах](#)(рус.)
- [Удачная модель ветвления для Git](#) (рус.)
- [Git — наглядная справка](#) (рус.)

Источник — [«https://ru.wikipedia.org/w/index.php?title=Git&oldid=90527441»](https://ru.wikipedia.org/w/index.php?title=Git&oldid=90527441)

Эта страница последний раз была отредактирована 26 января 2018 в 18:19.

Текст доступен по лицензии [Creative Commons Attribution-ShareAlike](#); в отдельных случаях могут действовать дополнительные условия.

Wikipedia® — зарегистрированный товарный знак некоммерческой организации [Wikimedia Foundation, Inc.](#)

[Свяжитесь с нами](#)