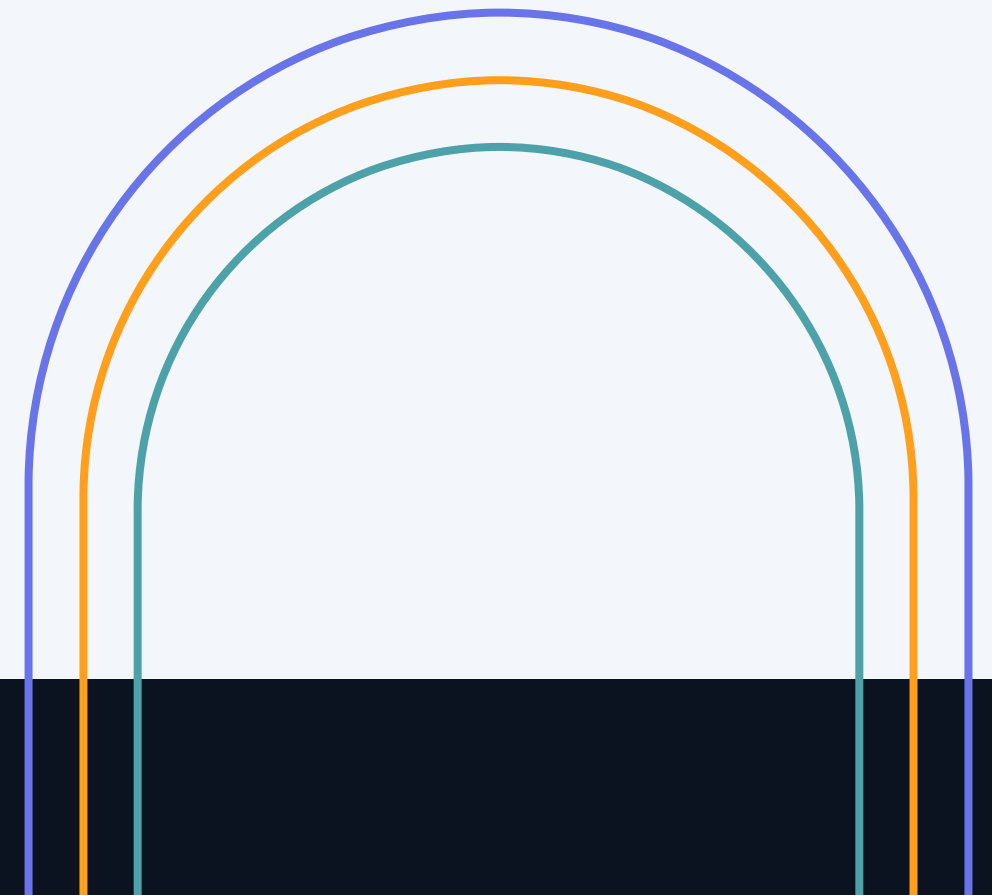




Лекция 2-3. Subversion. Основные сведения

- Общие сведения
- Примеры команд



- 
01. Определение
 02. История создания
 03. Основные понятия
 04. Копирование-изменение-слияние
 05. Блокирование-изменение-разблокирование
 06. ЖЦ проекта с использованием SVN
 07. Примеры команд
 08. Ветвление в Subversion
 09. Конфликты в Subversion

Содержание







ОБЩИЕ СВЕДЕНИЯ



01.



ОПРЕДЕЛЕНИЕ

Subversion –

это свободная система управления версиями с открытым исходным кодом. Согласно классификации из предыдущей лекции, является централизованной

позволяет управлять
файлами и каталогами во
времени

дерево файлов помещается
в центральное хранилище

позволяет восстановить
ранние версии данных,
исследовать историю
изменений данных

Благодаря этому, многие считают данную СКВ своеобразной «машиной времени»

02.



ИСТОРИЯ СОЗДАНИЯ

02. История создания

начало 2000 г

компания CollabNet, Inc решила начать разработку ПО на смену CVS

февраль 2000 г

CollabNet связалась с Карлом Фогелем, который впоследствии придумал название «Subversion» и разработал основные принципы устройства хранилища

31 августа 2001 г

команда прекратила использовать CVS и перешла на Subversion.

Сейчас SVN развивается как Open Source проект, но CollabNet так же оплачивает нескольким ключевым сотрудникам поддержку SVN

03.



ОСНОВНЫЕ ПОНЯТИЯ

03. Основные понятия

Хранилище

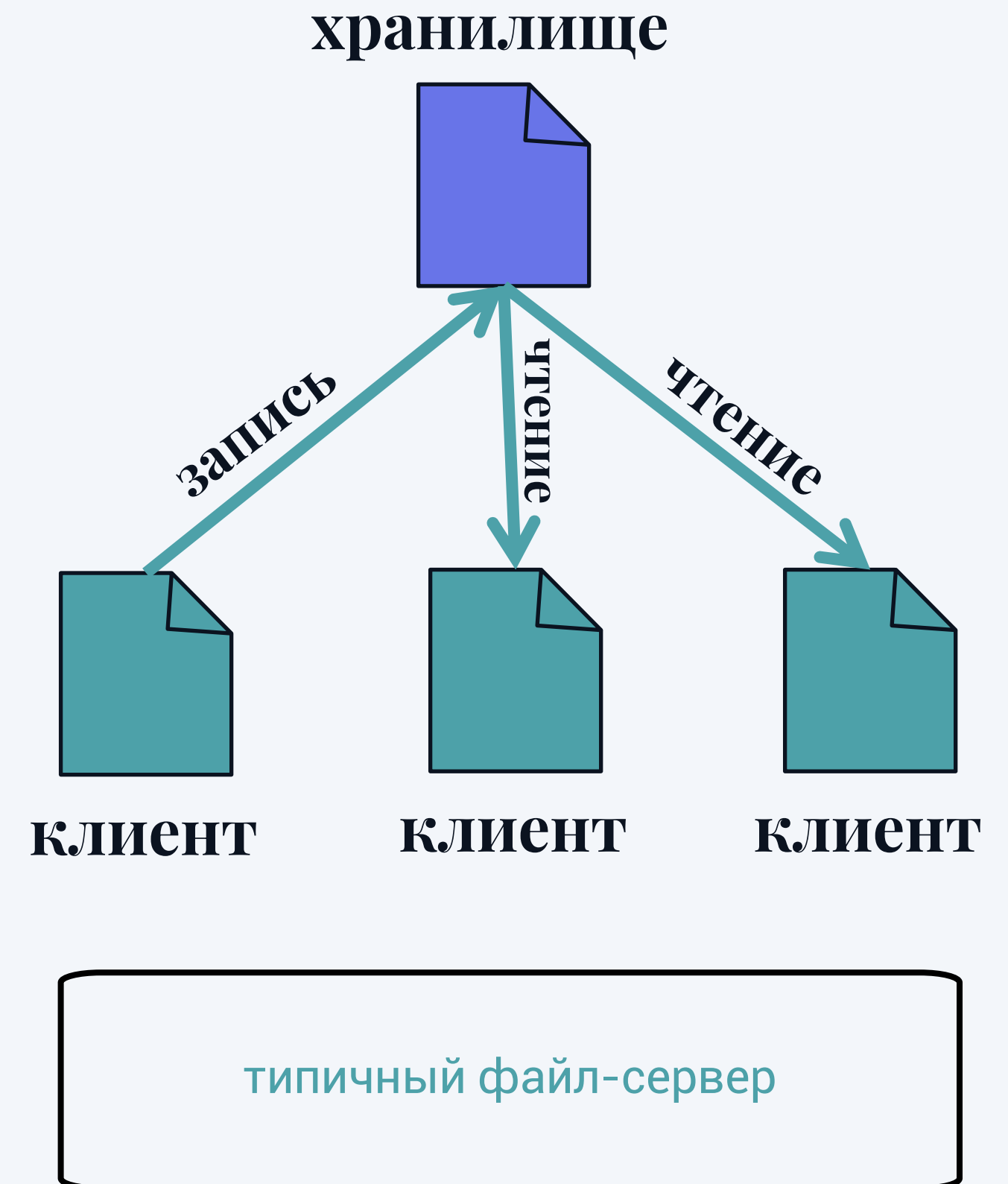
центр хранения данных.
Хранилище хранит информацию в
форме дерева файлов

Дерево файлов

типичное представление файлов и
каталогов

Любое количество клиентов
подключается к хранилищу и читает или
записывает эти файлы.

Записывая данные, клиент делает
информацию доступной для остальных;
Читая данные, клиент получает
информацию от других.



Принцип работы



SVN запоминает каждое внесенное изменение: любое изменение любого файла, равно как изменения в самом дереве каталогов, такие как добавление, удаление и реорганизация файлов и каталогов

При чтении данных из хранилища клиент обычно видит только последнюю версию дерева файлов. Но клиент также имеет возможность просмотреть предыдущие состояния файловой системы.

03. Основные понятия

Рабочая копия

обычное дерево каталогов на локальном компьютере, содержащее набор файлов. SVN **не смешивает** вносимые другими изменения с вашими и **не делает доступными для других** изменения сделанные вами, **пока вы не прикажете сделать это**

Служебный каталог РК

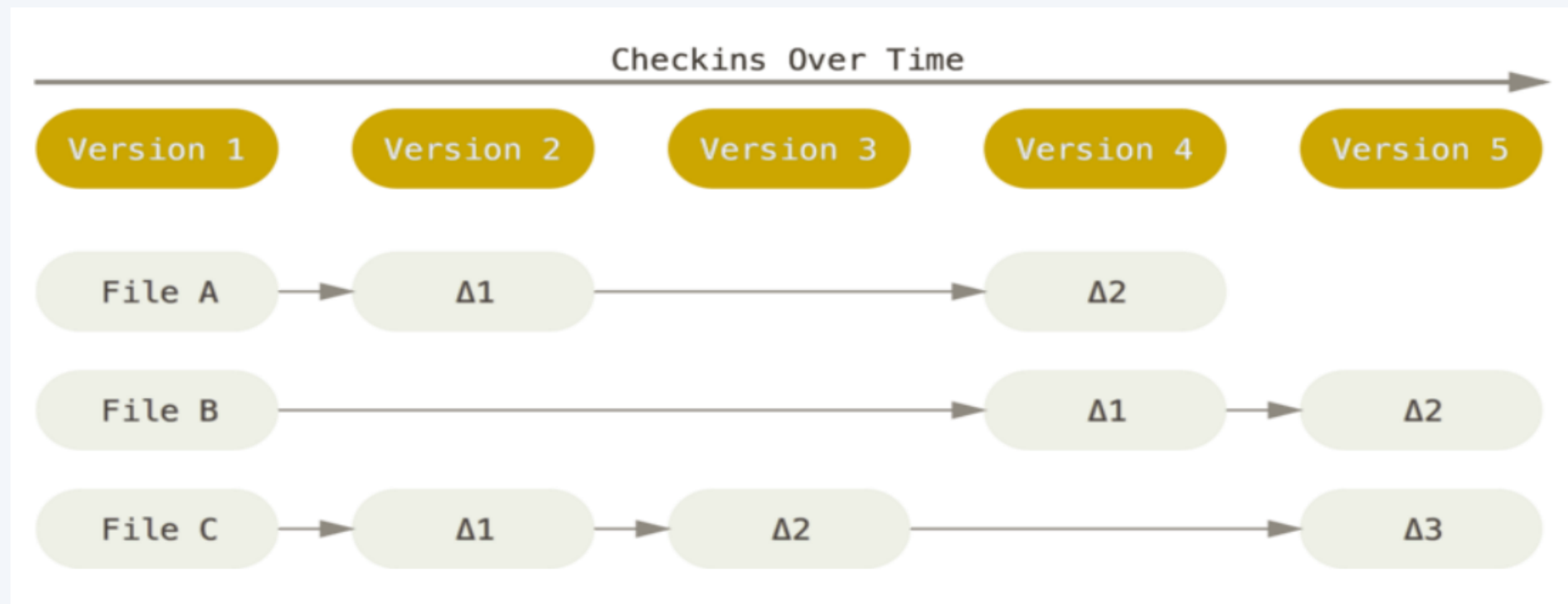
подкаталог с именем .svn, который содержится в РК. Файлы в служебном каталоге помогают SVN определить **какие файлы** РК **содержат неопубликованные изменения** и **какие файлы устарели** по отношению к файлам других участников



Хранение изменений

Дельта-кодирование

способ представления данных в виде разницы (дельты) между последовательными данными вместо самих данных



03. Основные понятия

Рабочая правка

на какой правке основан ваш рабочий файл

Временная метка

когда рабочая копия последний раз обновлялась из хранилища

Информация, хранящаяся в служебном каталоге `.svn/` для каждого файла

СОСТОЯНИЯ

не изменялся и
не устарел

изменялся локально
и не устарел

не изменялся и устарел

файл необходимо
сначала обновить

изменялся
локально и устарел





МОДЕЛИ ВЕРСИОНИРОВАНИЯ



Проблема

предоставить пользователям
возможность совместного использования
информации, при этом не наступая друг
другу на пятки.

Пользователи могут
просто непреднамеренно перезаписать в
хранилище изменения друг друга.



04.

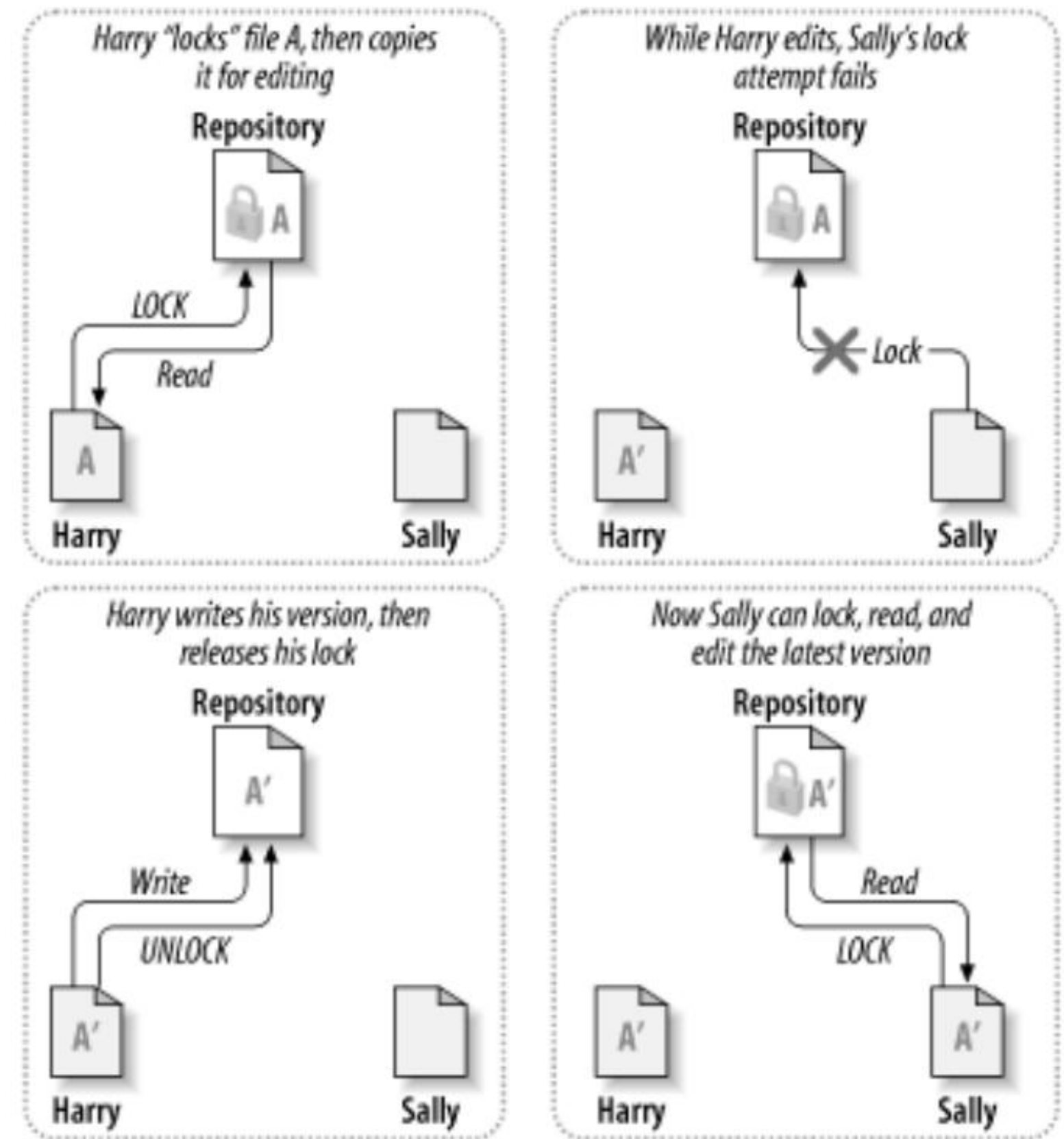


Блокирование — изменение
— разблокирование

04. Блокирование – изменение – разблокирование

Пользователь, решивший редактировать файл, блокирует его, в то время другой пользователь не может его редактировать.

При разблокировке файла следующим участником другой пользователь может забрать изменения предыдущего пользователя и редактировать файл дальше.



Проблемы



Проблемы администрирования

Файл может быть
заблокирован, и забыт быть
разблокированным.
Придётся обращаться к
администратору.
Ведёт к потере времени

Излишняя пошаговость

Если участники проекта хотят
редактировать разные, не
перекрывающиеся
части файла

Ложное чувство безопасности

Если файлы А и В
редактируются двумя людьми,
но эти файлы зависят друг от
друга, изменения всё равно
приведут к поломкам

Замечания



Не стоит возлагать большие надежды на то, что блокирующая система лучше защищена от конфликтов;
на практике блокирование снижает продуктивность как ничто другое

Модель губительна для командной работы, но есть моменты когда блокирование уместно, например для файлов бинарных форматов, таких как графические или звуковые, в них, как правило, невозможно объединить конфликтующие изменения

05.



Копирование — изменение
— слияние

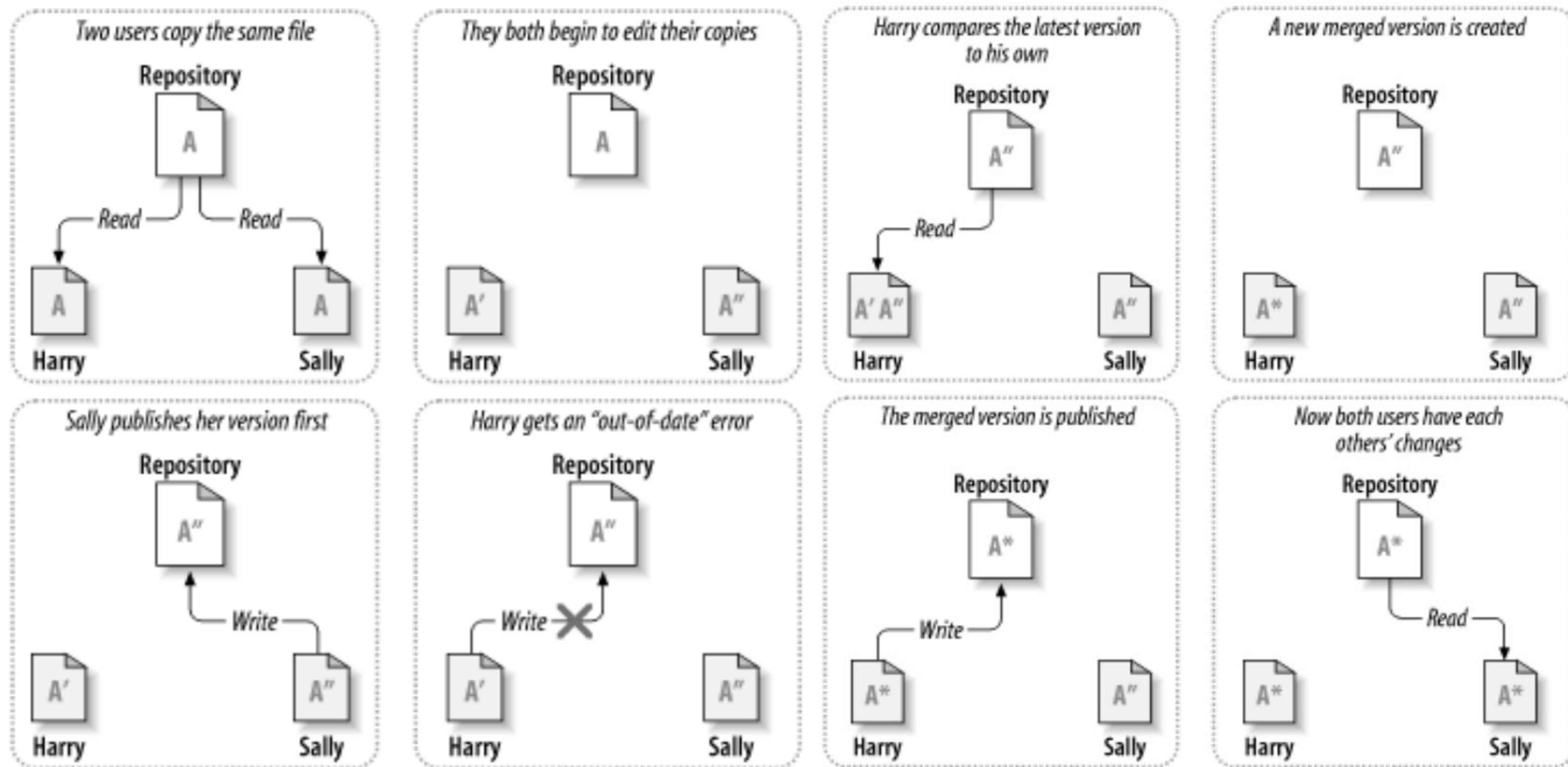


Оба пользователя копируют файл и редактируют его на своём компьютере. Затем один из пользователей отправляет изменения на сервер. Второй пользователь заканчивает свои изменения и тоже хочет отправить файлы на сервер, но не может, т.к. там уже находится обновлённая версия.

Он должен сначала взять изменения с сервера, объединить изменения, разрешить конфликты, если они есть, и затем отправить свои изменения на сервер



05. Копирование – изменение – слияние



Subversion, CVS и другие системы управления версиями
пользуются моделью копирование-изменение-слияние в
качестве альтернативы блокированию

каждый
пользовательский
клиент связывается
с хранилищем
проекта и создаёт
персональную
рабочую копию

пользователи
работают
параллельно,
изменяя свои
личные копии

личные копии
сливаются в новую,
финальную версию

СКВ помогает в
слиянии, но за его
корректное
выполнение
отвечает человек

ПРЕИМУЩЕСТВА

Пользователи могут работать параллельно, не тратя время на ожидание друг друга

При работе над одними и теми же файлами оказывается, что большинство параллельно вносимых изменений совсем не перекрываются, конфликты бывают редко

Время, которое было потрачено на разрешение конфликтов значительно меньше времени отнимаемого блокирующей системой

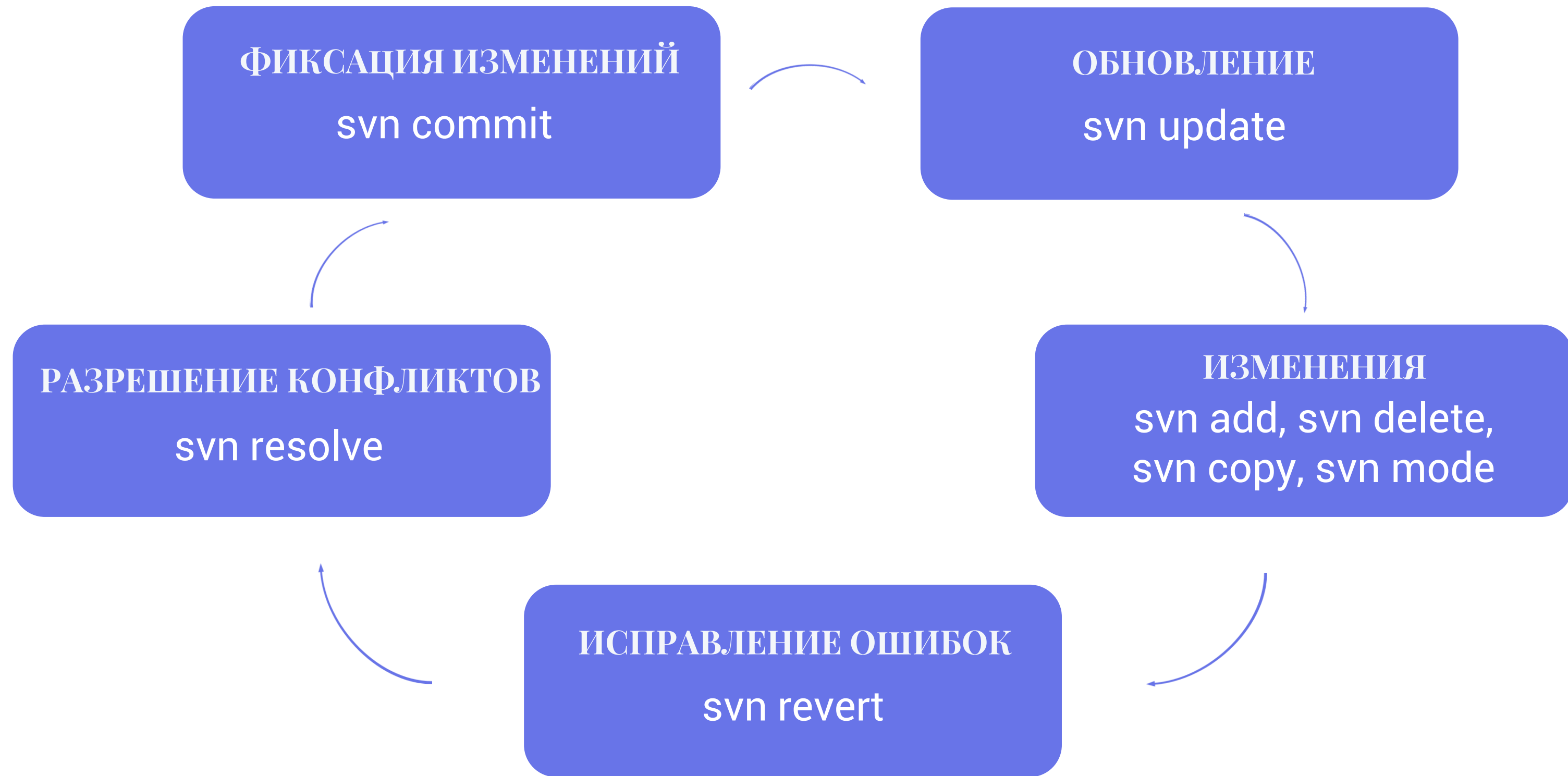
НЕДОСТАТОК

Немного сложнее для понимания, чем первая модель. Но, несмотря на это, отлично работает

06.



ЖИЗНЕННЫЙ ЦИКЛ ПРОЕКТА С ИСПОЛЬЗОВАНИЕМ SVN



07.



ПОЛЕЗНЫЕ КОМАНДЫ

07. Полезные команды

svn help

Клиент для командной строки
Subversion является
самодокументируемым.

```
C:\Users\krosh>svn help
usage: svn <subcommand> [options] [args]
Subversion command-line client.
Type 'svn help <subcommand>' for help on a specific subcommand.
Type 'svn --version' to see the program version and RA modules,
      'svn --version --verbose' to see dependency versions as well,
      'svn --version --quiet' to see just the version number.

Most subcommands take file and/or directory arguments, recursing
on the directories.  If no arguments are supplied to such a
command, it recurses on the current directory (inclusive) by default.

Available subcommands:
  add
  auth
  blame (praise, annotate, ann)
  cat
  changelist (cl)
  checkout (co)
  cleanup
  commit (ci)
  copy (cp)
  delete (del, remove, rm)
  diff (di)
  export
  help (?, h)
  import
```

07. Полезные команды

`svn help <subcommand>`

В любой момент команда покажет описание синтаксиса, параметров и поведения подкоманды `subcommand`

```
C:\Users\krosh>svn help commit
```

```
commit (ci): Send changes from your working copy to the repository.  
usage: commit [PATH...]
```

A log message must be provided, but it can be empty. If it is not given by a `--message` or `--file` option, an editor will be started.

If any targets are (or contain) locked items, those will be unlocked after a successful commit, unless `--no-unlock` is given.

If `--include-externals` is given, also commit file and directory externals reached by recursion. Do not commit externals with a fixed revision.

Valid options:

| | |
|--------------------------------------|--|
| <code>-q [--quiet]</code> | : print nothing, or only summary information |
| <code>-N [--non-recursive]</code> | : obsolete; same as <code>--depth=empty</code> |
| <code>--depth ARG</code> | : limit operation by depth ARG ('empty', 'files', 'immediates', or 'infinity') |
| <code>--targets ARG</code> | : pass contents of file ARG as additional args |
| <code>--no-unlock</code> | : don't unlock the targets |
| <code>-m [--message] ARG</code> | : specify log message ARG |
| <code>-F [--file] ARG</code> | : read log message from file ARG |
| <code>--force-log</code> | : force validity of log message source |
| <code>--editor-cmd ARG</code> | : use ARG as external editor |
| <code>--encoding ARG</code> | : treat value as being in charset encoding ARG |
| <code>--with-revprop ARG</code> | : set revision property ARG in new revision using the name[=value] format |
| <code>--changelist [--cl] ARG</code> | : operate only on members of changelist ARG |
| <code>--keep-changelists</code> | : don't delete changelists after commit |

07. Полезные команды

svnadmin create
/path/to/rep

Создает пустой репозиторий
расположением /path/to/rep

```
C:\Users\krosh>svnadmin create D:\Test_Repository  
  
C:\Users\krosh>
```

| Имя | Дата изменения | Тип |
|-----------------|------------------|-----------------|
| GitLab-Runner | 27.04.2023 18:56 | Папка с файлами |
| Test_Repository | 03.06.2023 21:05 | Папка с файлами |

| Имя | Дата изменения | Тип | Размер |
|------------|------------------|--------------------|--------|
| conf | 03.06.2023 21:05 | Папка с файлами | |
| db | 03.06.2023 21:05 | Папка с файлами | |
| hooks | 03.06.2023 21:05 | Папка с файлами | |
| locks | 03.06.2023 21:05 | Папка с файлами | |
| format | 03.06.2023 21:05 | Файл | 1 КБ |
| README.txt | 03.06.2023 21:05 | Текстовый документ | 1 КБ |

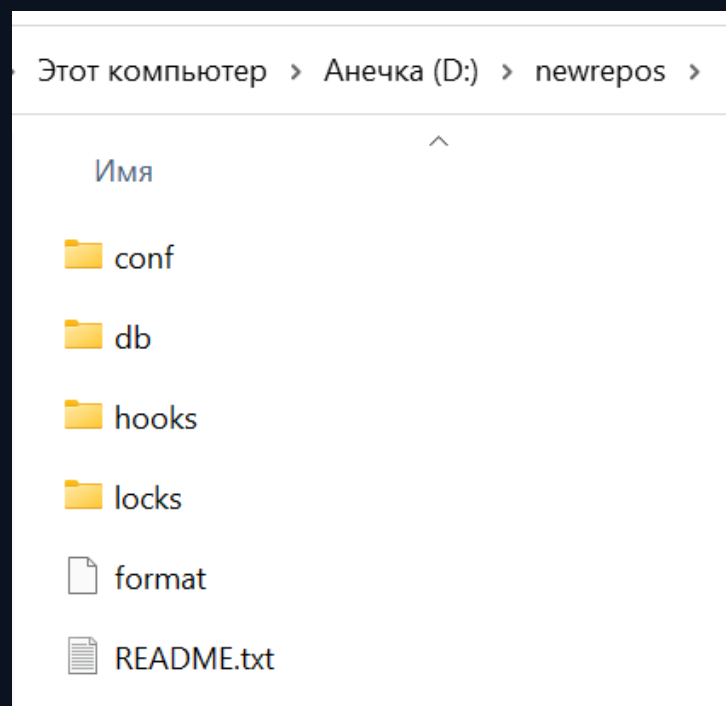
07. Полезные команды

svn import

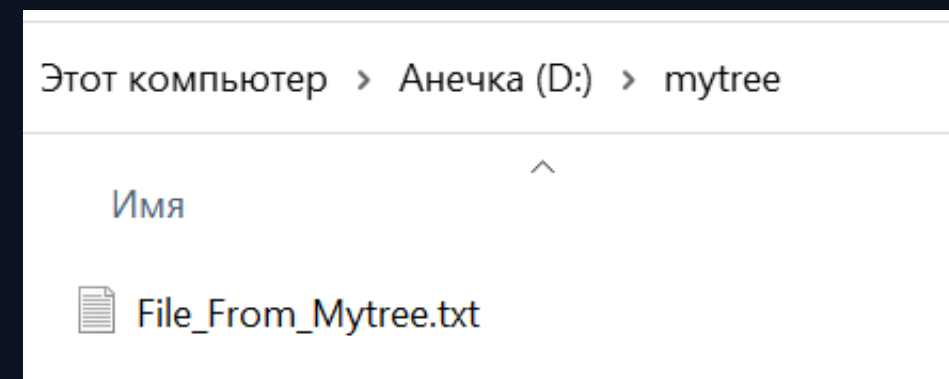
Импортирование нового проекта. это быстрый способ скопировать не версионированное дерево файлов в хранилище, создавая при необходимости промежуточные директории

svnadmin create /usr/local/svn/newrepos

Создание репозитория



Содержимое локальной папки mytree



07. Полезные команды

```
svn import mytree  
file:///newrepos/some/project -m "Mesage"
```

выполняется копирование содержимого директории mytree в директорию some/project хранилища. **Обратите внимание**, что команду необходимо выполнять с ключом **-m**, в котором фиксируется сообщение для лога

Если просматривать содержимое в файловом проводнике, то каталоги не отображаются

Этот компьютер > Анечка (D:) > newrepos

| Имя | Дата изменения |
|------------|-----------------|
| conf | 04.06.2023 8:46 |
| db | 04.06.2023 8:53 |
| hooks | 04.06.2023 8:46 |
| locks | 04.06.2023 8:46 |
| format | 04.06.2023 8:46 |
| README.txt | 04.06.2023 8:46 |

```
PS D:\> svn import mytree file:///newrepos/some/project -m "import project"  
Adding          mytree\File_From_Mytree.txt  
Committing transaction...  
Committed revision 1.
```

07. Полезные команды

`svn list <путь к локальному хранилищу>`

Покажет список файлов, хранящихся в локальном хранилище по адресу `<путь к локальному хранилищу>`

```
PS D:\> svn list file:///newrepos/some/project
File_From_Mytree.txt
PS D:\> |
```

Обратите внимание на то, что после завершения импорта, **оригинальное дерево файлов не конвертируется в рабочую копию.**

Для того, чтобы начать работать необходимо создать новую рабочую копию дерева файлов.

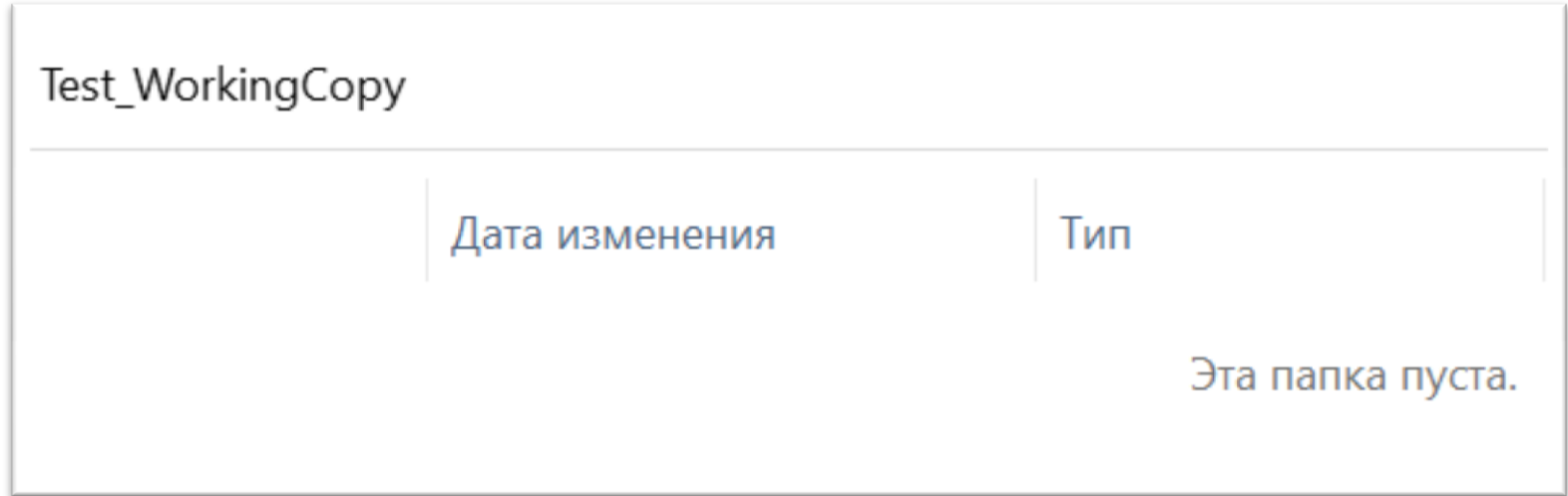
07. Полезные команды

svn checkout <путь к репозиторию>

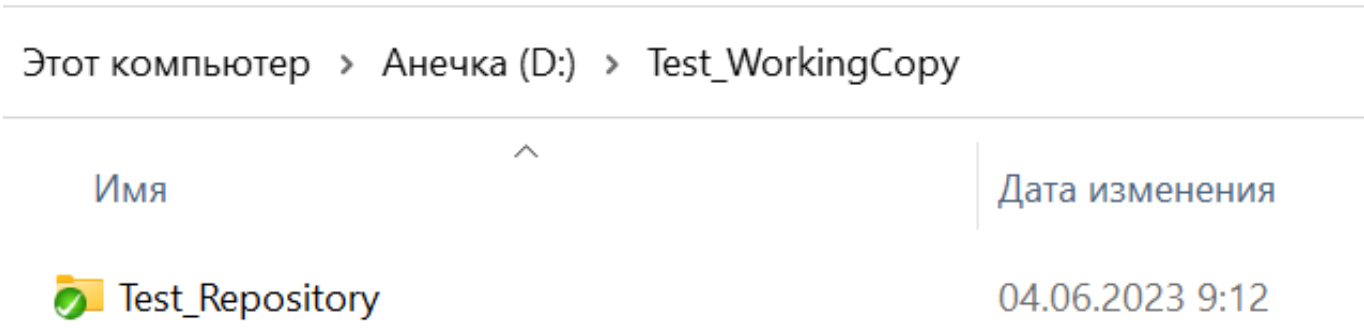
Для того чтобы создать рабочую копию уже существующего репозитория, нужно получить какую-либо из подкаталогов хранилища, лежащего по адресу <путь к репозиторию>

```
PS D:\Test_WorkingCopy> svn checkout file:///D:/Test_Repository
Checked out revision 0.
PS D:\Test_WorkingCopy> |
```

2. Выполняем команду. Обратите внимание на направление слешей.



1. Создаем пустой каталог, в котором будет РК. Внутри этого каталога открываем командную строку



3. Содержимое каталога РК

ПУТИ К ХРАНИЛИЩУ

file:///

прямой доступ к
хранилищу (на
локальном диске)

http://

доступ через протокол
WebDAV (если SVN-сервер
работает через Apache)

https://

тоже что и http://, но с
SSL-шифрованием

svn://

доступ через
собственный протокол к
серверу svnserve

svn+ssh://

тоже что и svn://, но через
SSH-соединение

07. Полезные команды

`svn info` (работает только в рабочих копиях внутри копии репозитория)

Позволяет просматривать информацию о репозитории, такую как URL родительского репозитория, номер текущей ревизии, дату последних изменений

> Test_WorkingCopy > Test_Repository

| Имя | Дата изменения |
|-----|-----------------|
| svn | 04.06.2023 9:12 |

Содержимое РК

```
PS D:\Test_WorkingCopy\Test_Repository> svn info
Path: .
Working Copy Root Path: D:\Test_WorkingCopy\Test_Repository
URL: file:///D:/Test_Repository
Relative URL: ^/
Repository Root: file:///D:/Test_Repository
Repository UUID: e0f8390b-9a96-4648-a314-90df027ed041
Revision: 0
Node Kind: directory
Schedule: normal
Last Changed Rev: 0
Last Changed Date: 2023-06-03 21:05:51 +0300 (Сб, 03 июн 2023)
```

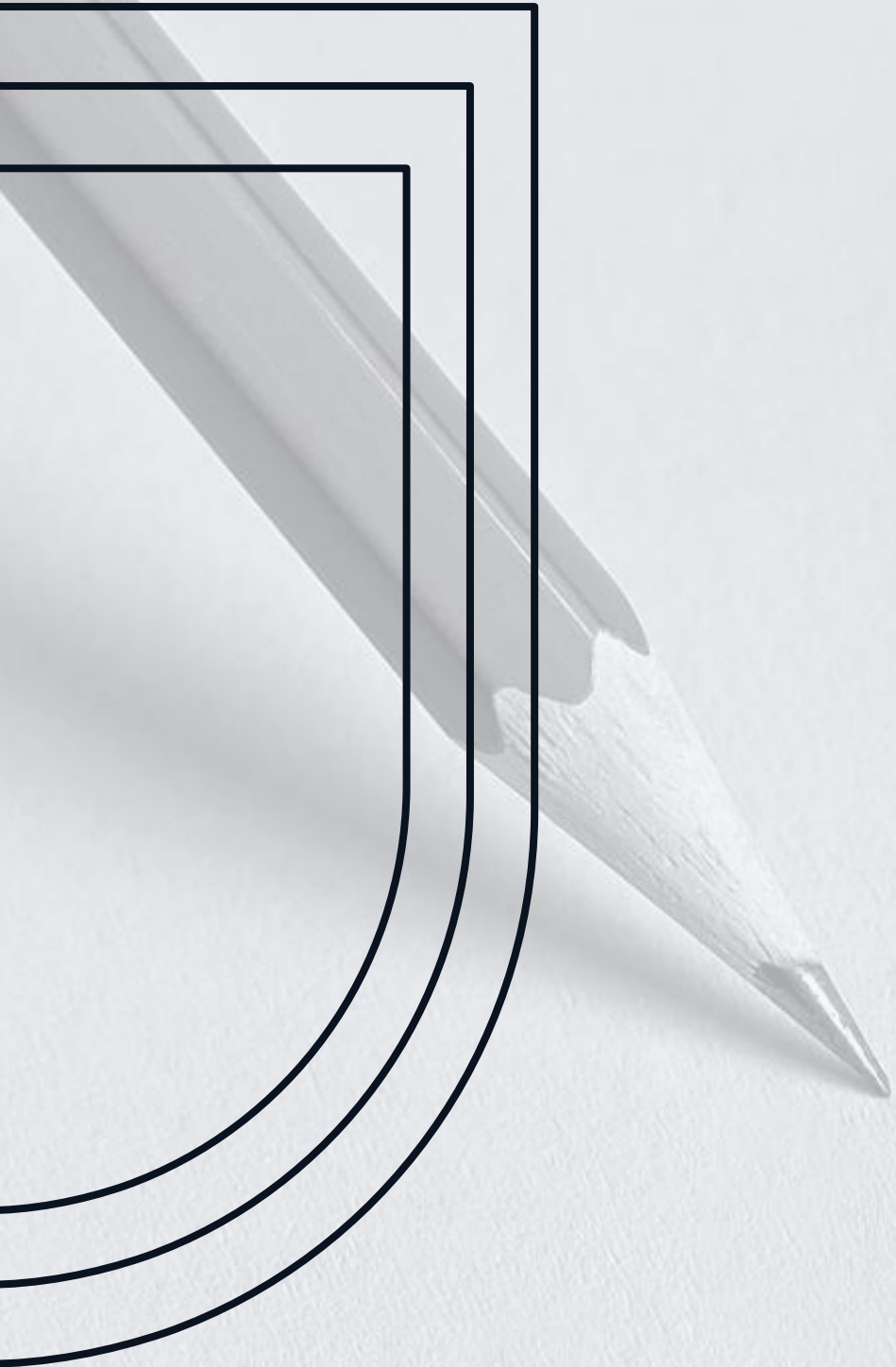
07. Полезные команды

svn update

обновление РК;
необходимо для получения любых изменений,
внесённых с момента последнего обновления другими
разработчиками проекта. Используется для
синхронизации РК с последней правкой в хранилище


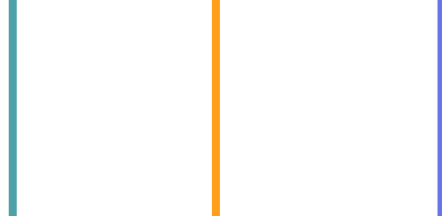
```
PS D:\Test_WorkingCopy\Test_Repository> svn update
Updating '.':
At revision 0.
PS D:\Test_WorkingCopy\Test_Repository> |
```

Когда сервер отправляет
изменения в РК для каждого
элемента выводится латинская
буква — код, определяющий,
какое действие выполнила
Subversion для приведения РК в
актуальное состояние

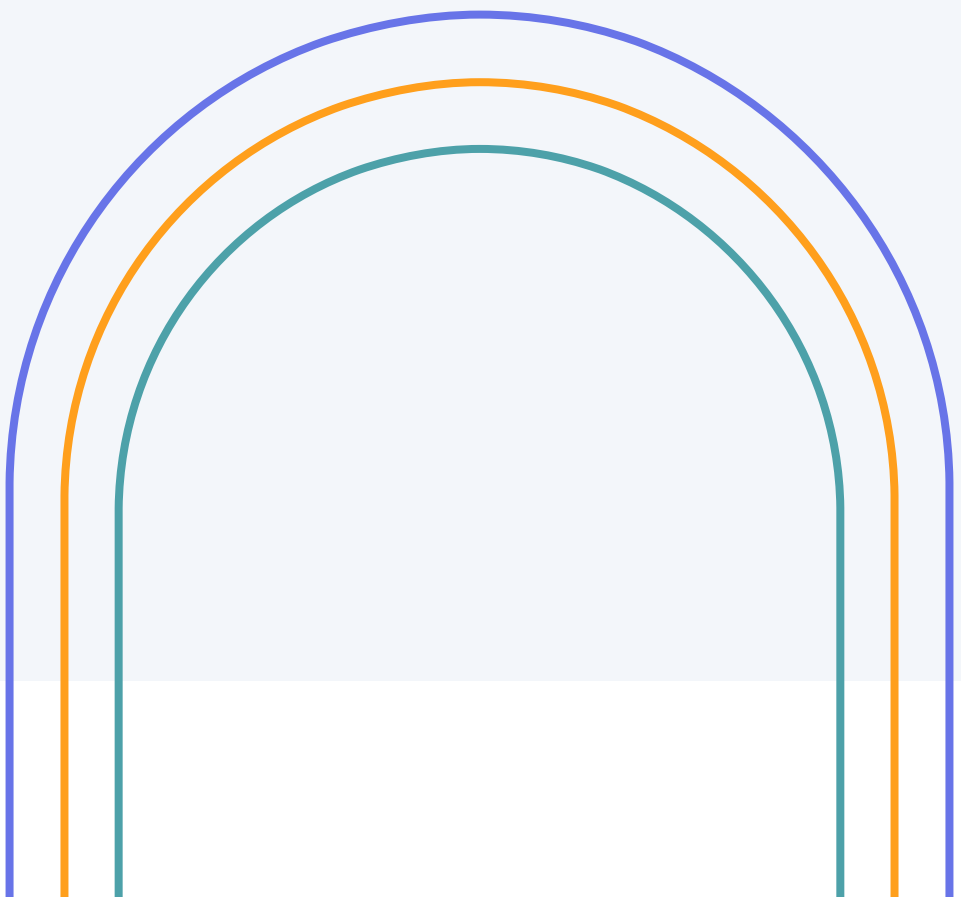


БУКВЕННЫЕ
КОДЫ

| | | |
|---|-------------|--------------------------------------|
| U | Updated | Файл был обновлен |
| A | Added | Файл был добавлен |
| D | Deleted | Файл был удалён |
| R | Replaced | Файл был перемещен |
| G | merGed | Файл был объединён |
| C | Conflicting | Файл конфликтует с локальной версией |



ИЗМЕНЕНИЕ ХРАНИЛИЩА БЕЗ УЧАСТИЯ РАБОЧЕЙ КОПИИ



Существуют случаи использования, которые сразу же фиксируют в хранилище изменения структуры. Это происходит только тогда, когда подкоманда оперирует напрямую с URL вместо рабочей копии.

`svn mkdir`

`svn copy`

`svn move`

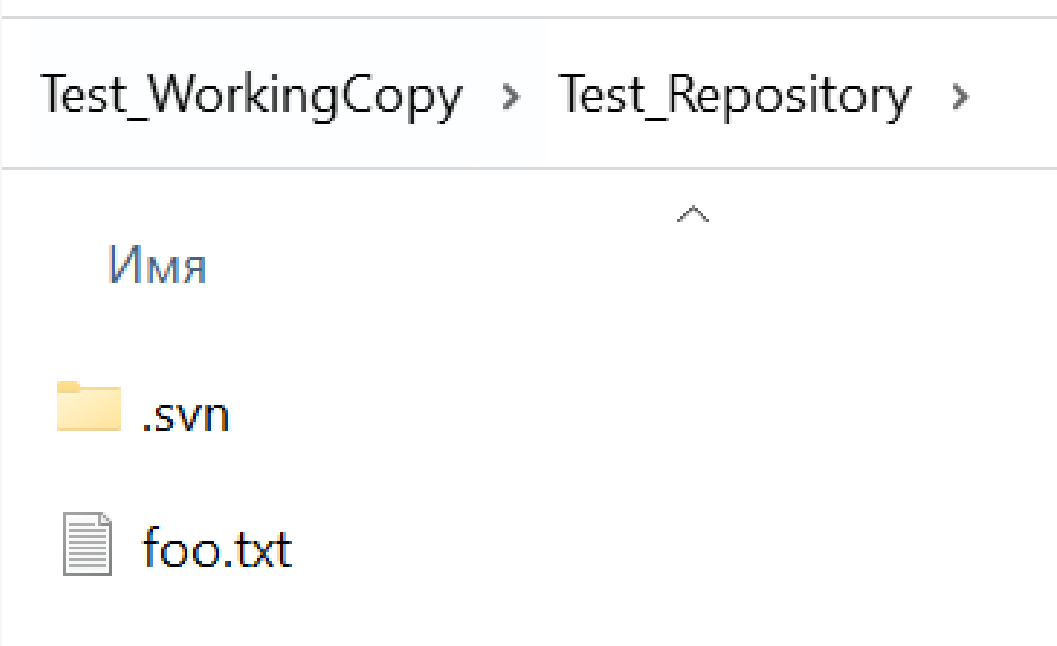
`svn delete`

Когда вы работаете напрямую с URL, любое из приведенных выше действий **приводит к немедленной фиксации**.

Обратите внимание!

Нельзя использовать обычные команды удаления, копирования и перемещения вместо `svn delete`, `svn copy` и `svn move`, так как тогда изменения не будут записаны в `.svn` и при обновлении из репозитория все файлы вернутся обратно

07. Полезные команды

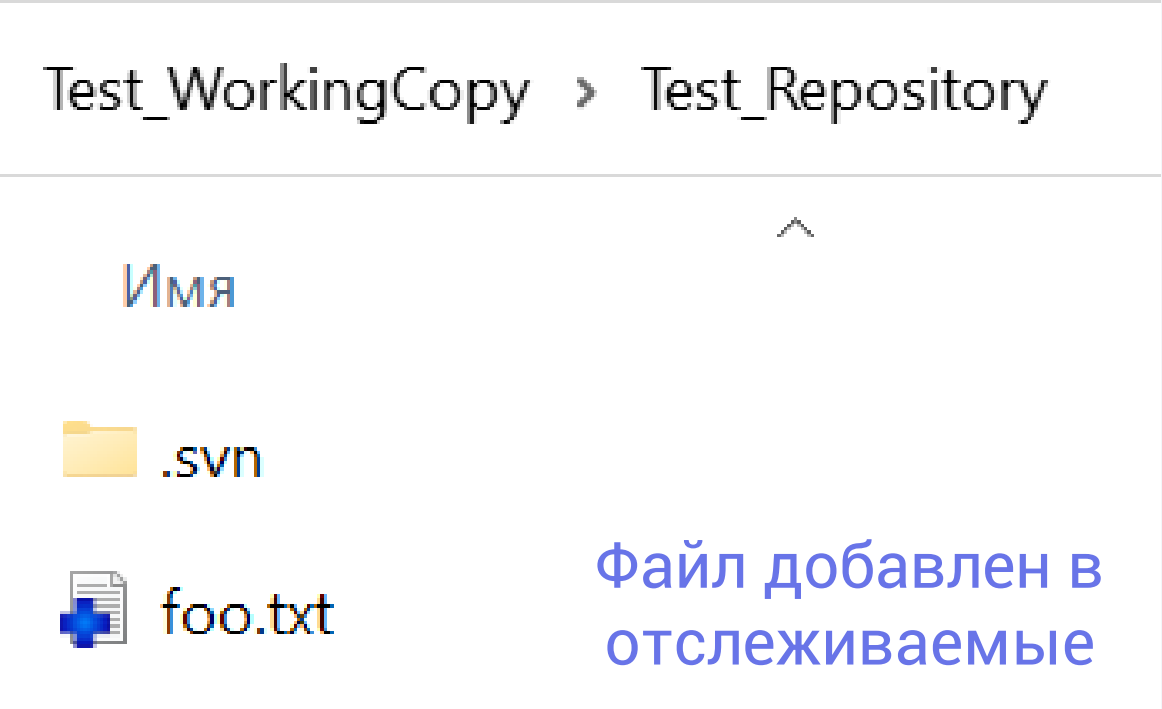


Файл в РК

svn add foo

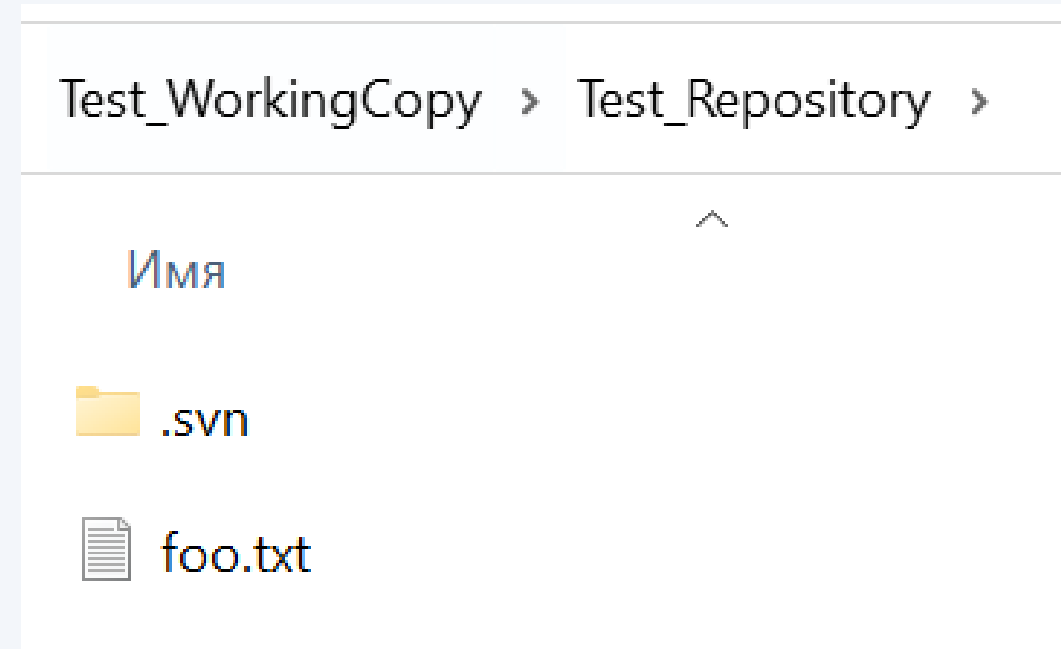
Запланировать файл, директорию или символическую ссылку foo для добавления в Хранилище при следующей фиксации

```
PS D:\Test_WorkingCopy\Test_Repository> svn add foo.txt
A      foo.txt
```



Файл добавлен в отслеживаемые

07. Полезные команды

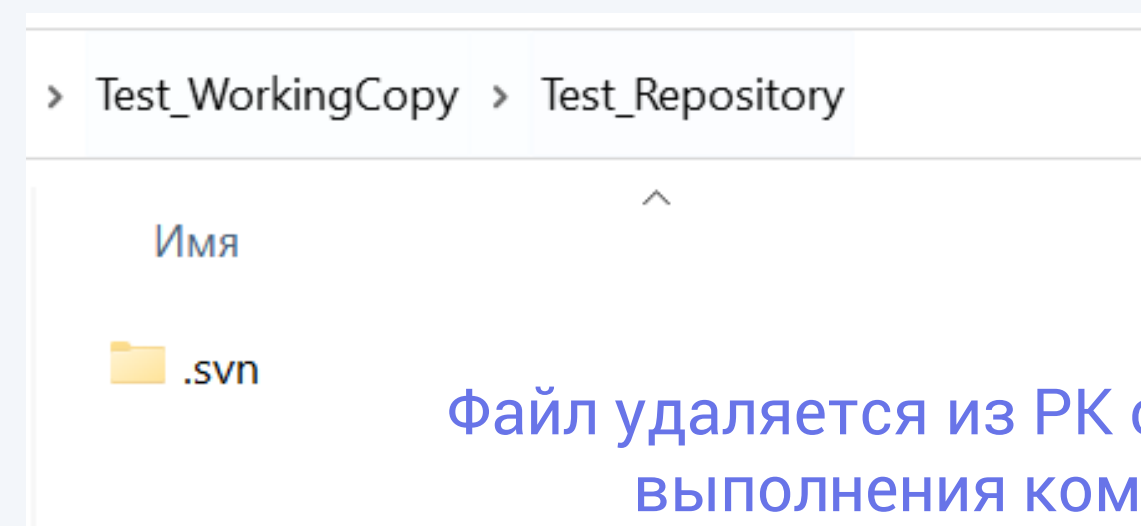


Файл в РК

svn delete foo

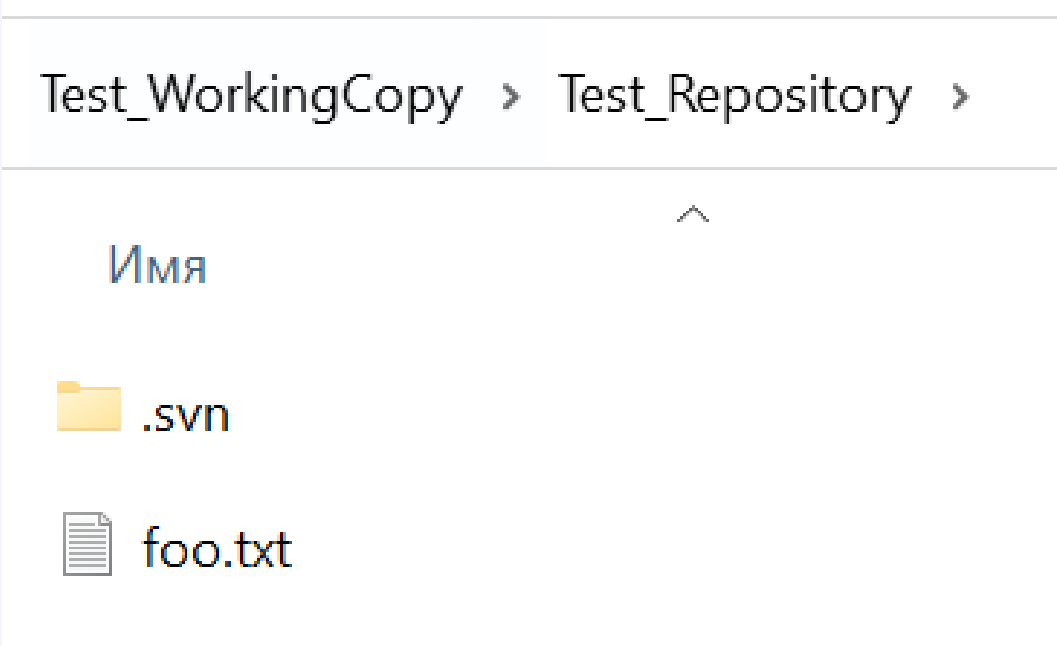
Запланировать удаление из хранилища файла, директории или символьной ссылки foo. Если foo является файлом или ссылкой, он сразу же удаляется из РК. Если foo является директорией, она не удаляется, но Subversion планирует ее удаление. foo будет удалена из РК и хранилища при фиксации изменений.

```
PS D:\Test_WorkingCopy\Test_Repository> svn delete foo.txt
D foo.txt
```



Файл удаляется из РК сразу после выполнения команды

07. Полезные команды

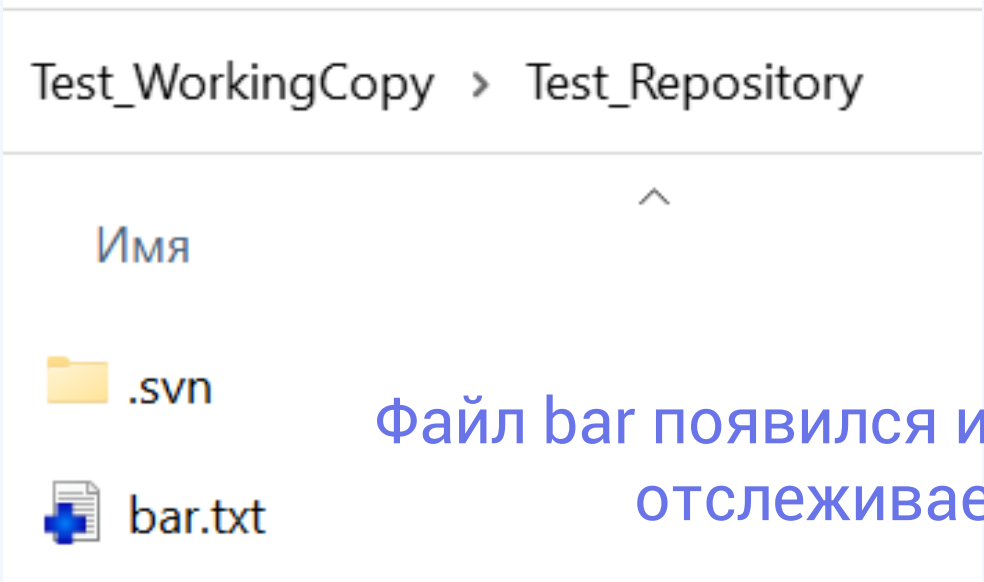


Файл в РК

svn copy foo bar

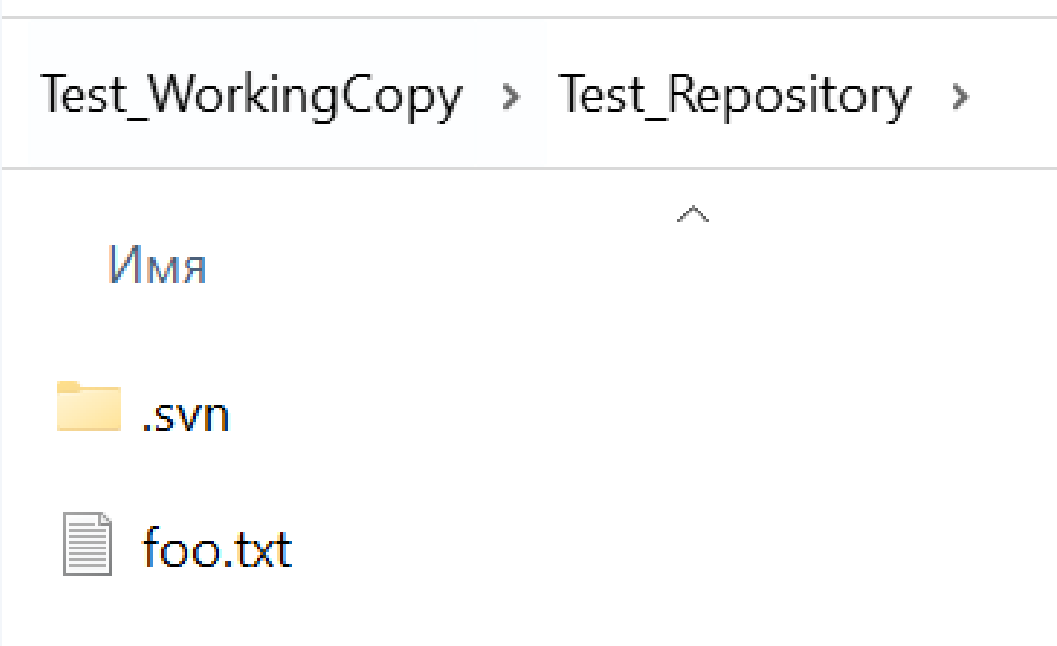
Создать новый элемент bar как копию foo. bar будет автоматически запланирован для добавления. Когда при следующей фиксации bar будет добавлен в хранилище в его истории будет отмечено копирование (то, что первоисточником является foo).
Команда copy не создает промежуточных директорий.

```
PS D:\Test_WorkingCopy\Test_Repository> svn copy foo.txt bar.txt
A      bar.txt
```



Файл bar появился и добавился в отслеживаемые

07. Полезные команды

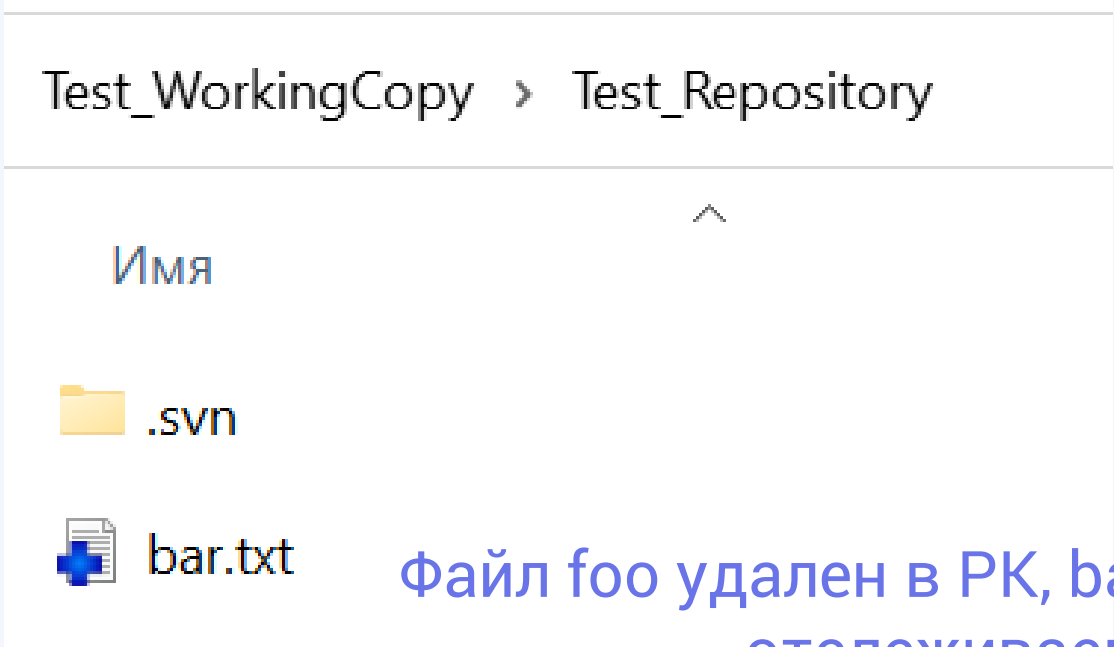


Файл в РК

svn move foo bar

Эта команда полностью аналогична выполнению
svn copy foo bar; svn delete foo, поэтому, bar будет
запланирован для добавления как копия foo, а foo
Будет запланирован для удаления.
svn move не создает промежуточных директорий.

```
PS D:\Test_WorkingCopy\Test_Repository> svn move foo.txt bar.txt
A      bar.txt
D      foo.txt
```

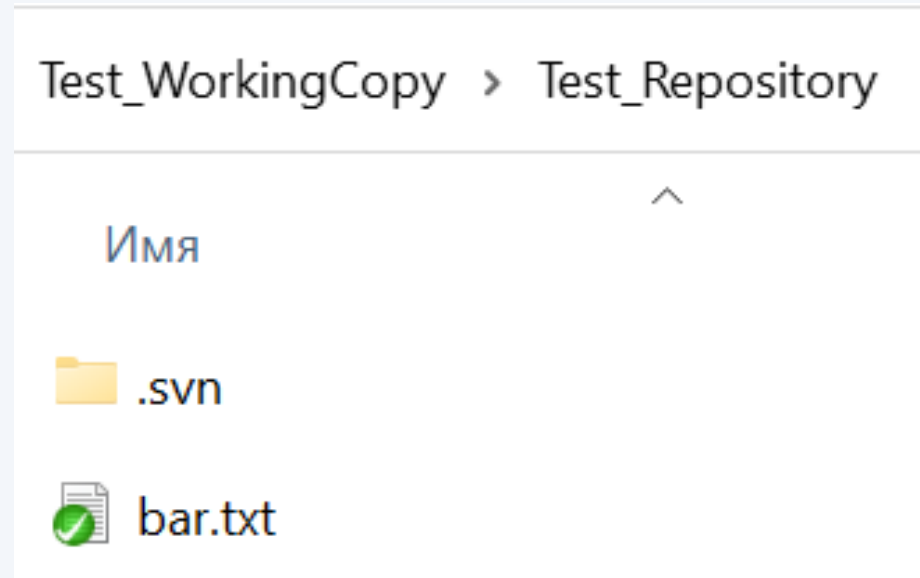


Файл foo удален в РК, bar – добавлен в
отслеживаемые

07. Полезные команды

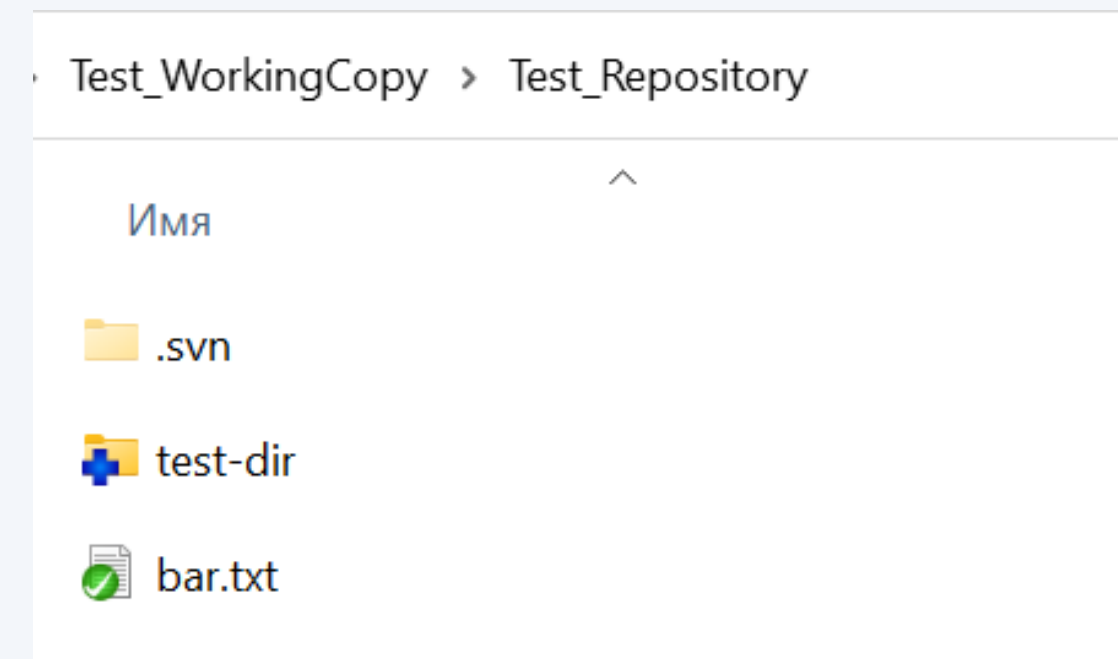
`svn mkdir dir-name`

Эта команда создаёт директорию `dir-name` и добавляет её под контроль версий



Репозиторий

```
PS D:\Test_WorkingCopy\Test_Repository> svn mkdir test-dir
A      test-dir
```

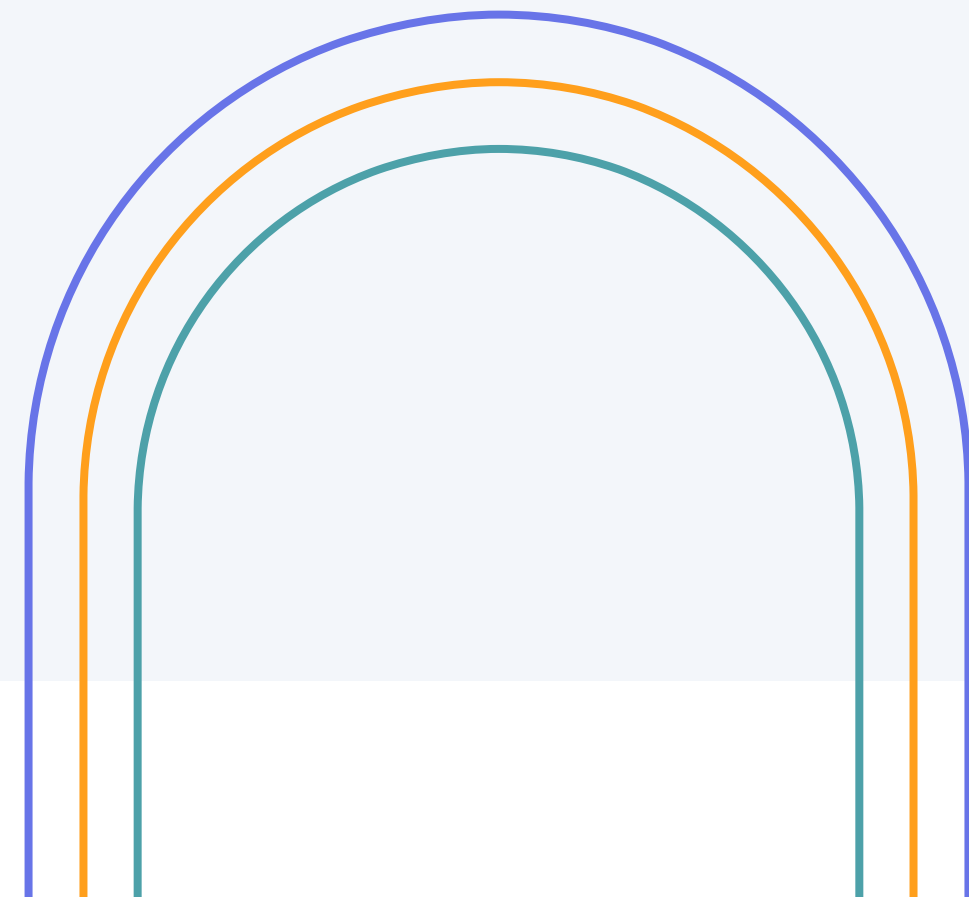


Директория добавлена в
отслеживаемые

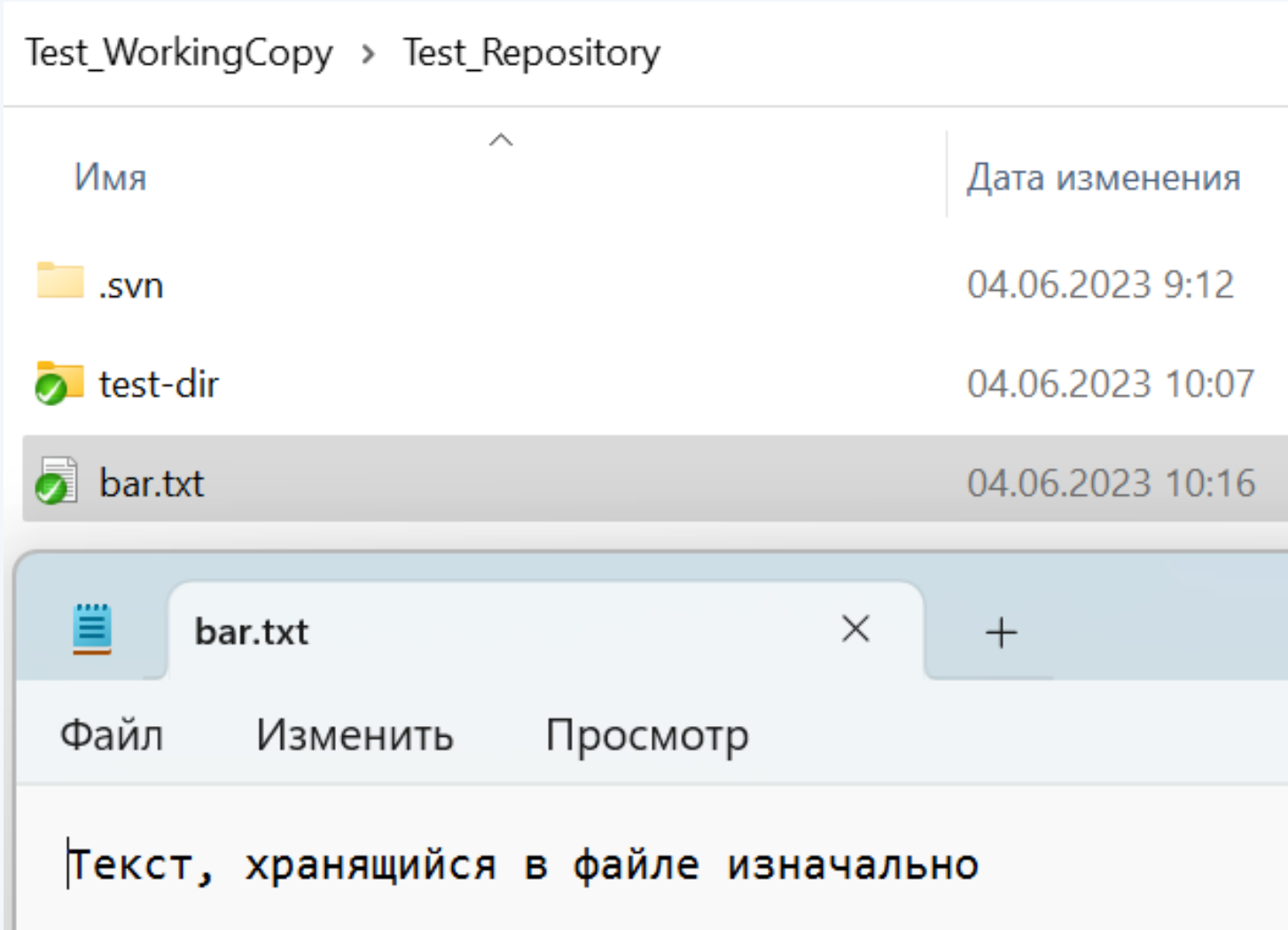




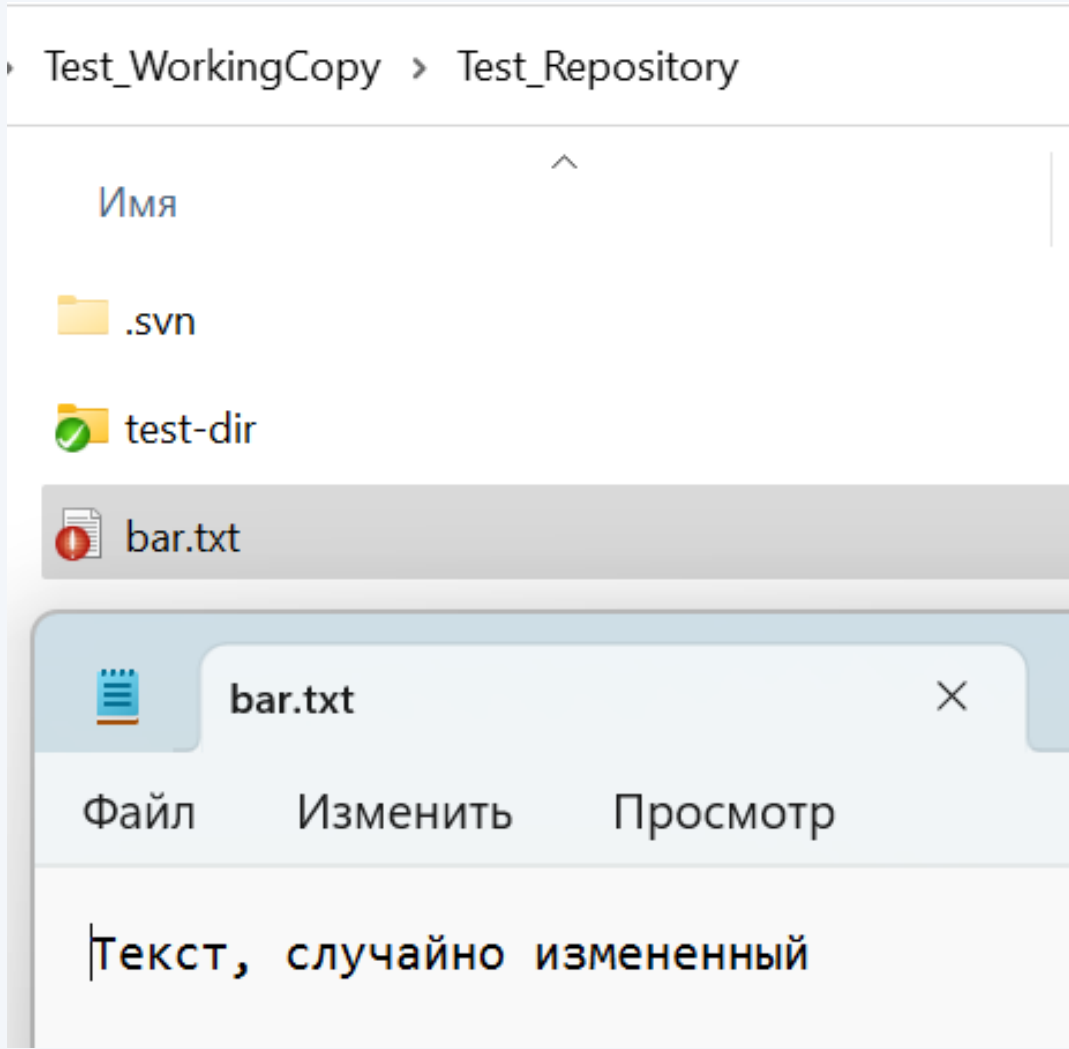
Исправление внесённых изменений



07. Полезные команды

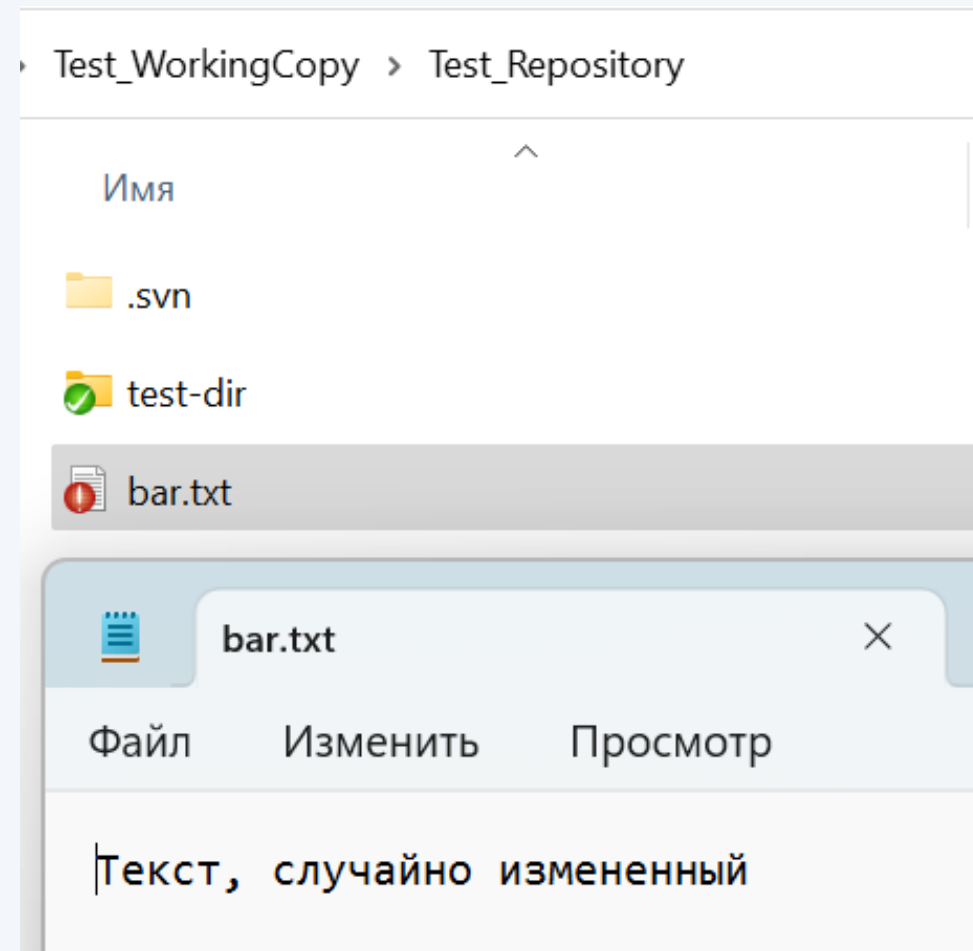


Файл изначальный (он добавлен в репозиторий)



Файл случайно измененный (он находится в РК и не помещался в репозиторий)

07. Полезные команды

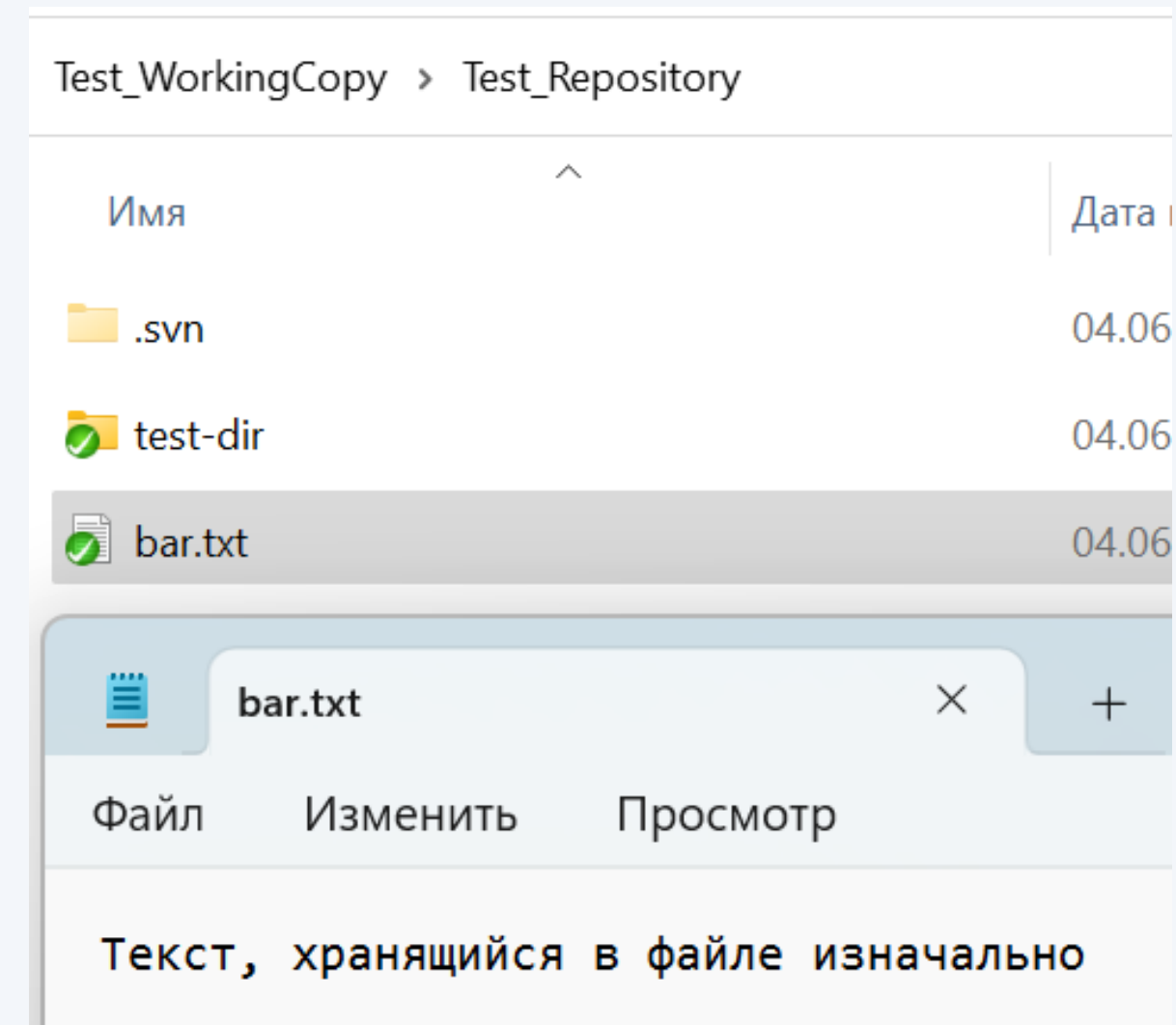


Файл в РК


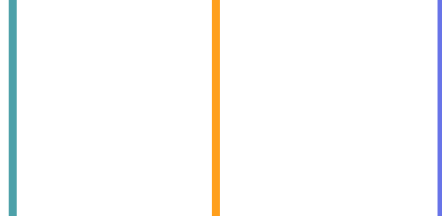
svn revert

возвращает файл в состояние, предшествующее модификации, путём замены файла его кэшированной «первоначальной» копией из .svn области

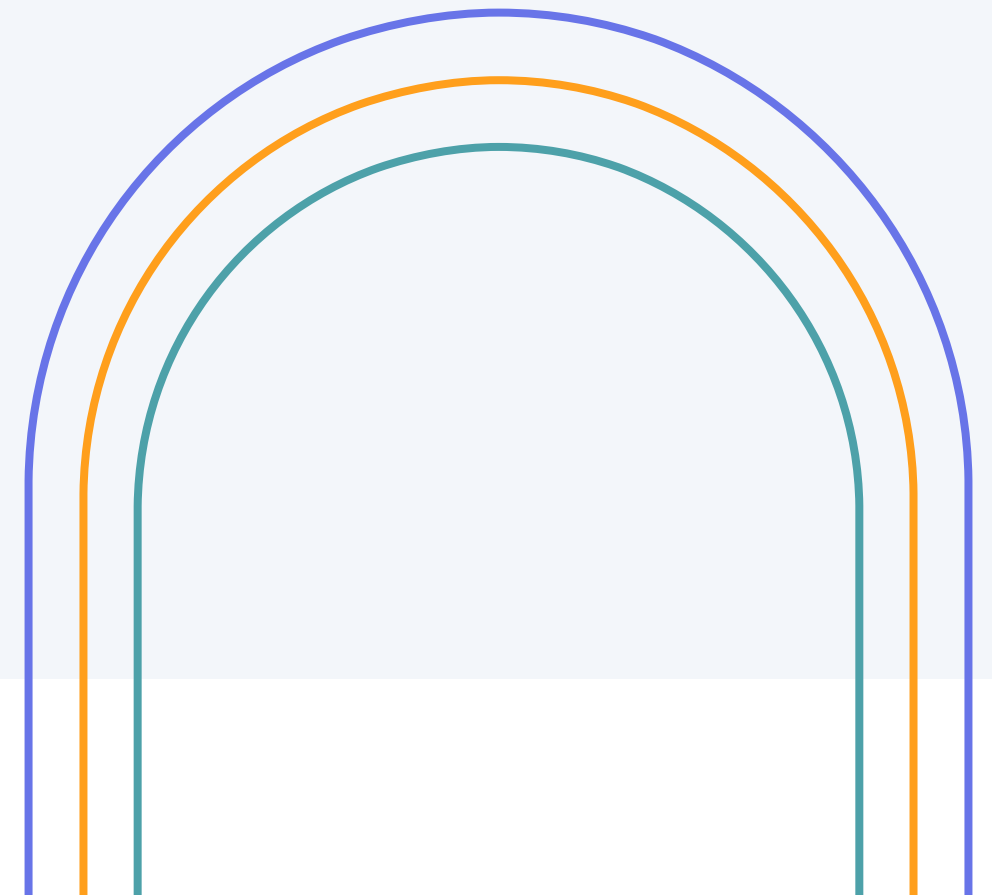
```
PS D:\Test_WorkingCopy\Test_Repository> svn revert bar.txt
Reverted 'bar.txt'
```



Файл изменился на файл, хранящийся в репозитории



Исправление конфликтов (при объединении с чужими изменениями)



07. Полезные команды

```
$ svn update  
U INSTALL  
G README  
C bar.c  
Updated to revision 46.
```

Выполнение команды update показало, что файл, отмеченный C, имеет конфликт

1

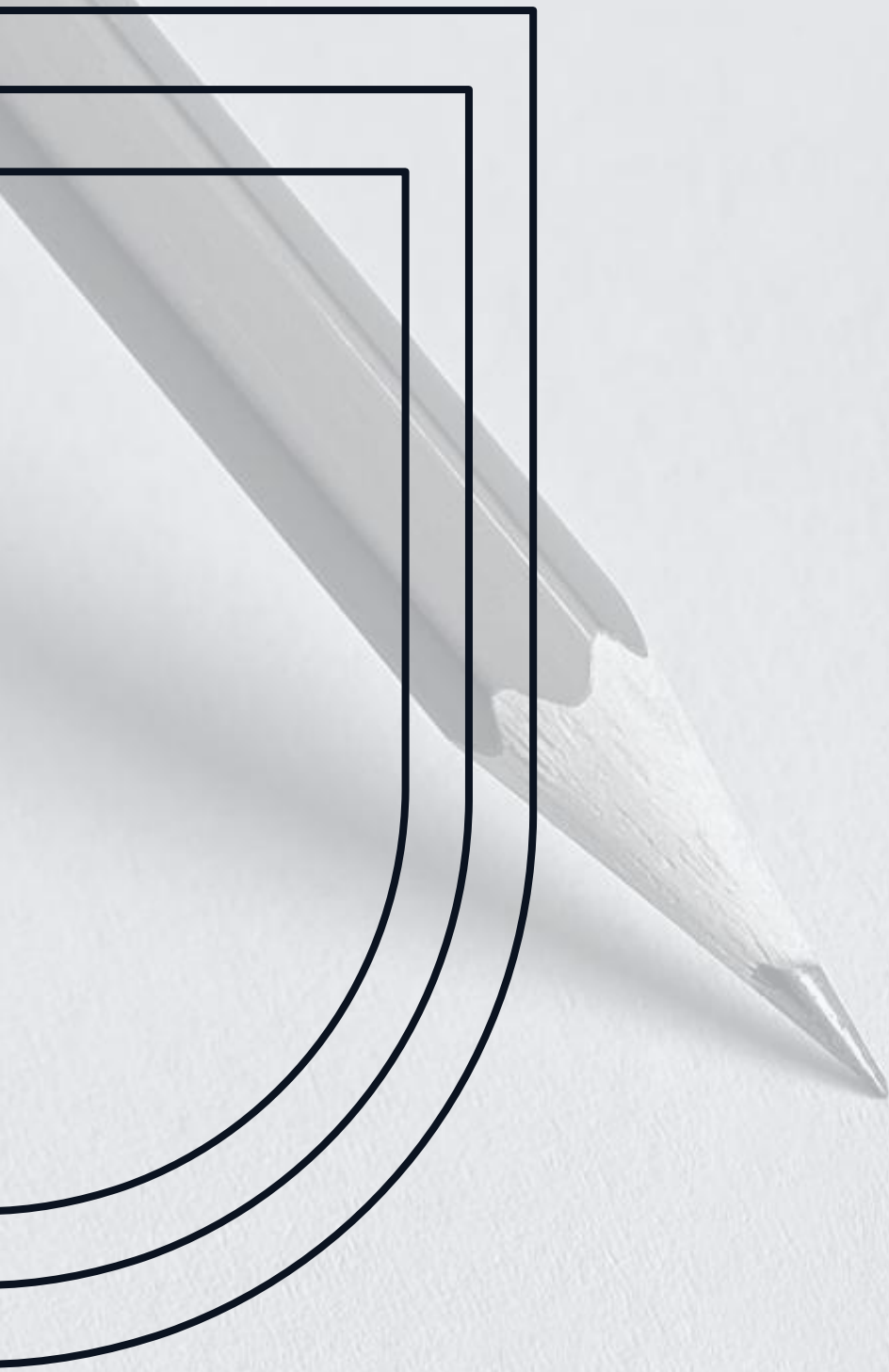
SVN печатает C во время обновления и запоминает, что файл в состоянии конфликта

2

Если SVN считает, что файл объединяемого типа, она помещает маркеры конфликта — специальные текстовые строки которые отделяют «стороны» конфликта — в файл, для того, чтобы визуально показать пересекающиеся области

3

Для каждого конфликтного файла SVN добавляет в РК до трех не версионированных дополнительных файлов



ДОПОЛНИТЕЛЬНЫЕ ФАЙЛЫ

filename.mine

файл в том виде в
каком он был в РК до
обновления

filename.rOLDREV

файл, который был до
внесения изменений

filename.rNEWREV


файл, который SVN-
клиент получил с сервера
при обновлении РК

OLDREV

номер правки файла в
директории .svn

NEWREV

номер правки HEAD
хранилища



ВАРИАНТЫ РАЗРЕШЕНИЯ КОНФЛИКТОВ

Объединить конфликтующий текст
«вручную» (путем анализа и
редактирования маркеров конфликта
в файле)


Скопировать один из временных
файлов поверх своего рабочего файла

Выполнить `svn revert <filename>` для
того, чтобы убрать все локальные
изменения

После того, как конфликт разрешен, нужно поставить в известность Subversion, выполнив **svn resolved**.

Эта команда удалит три временных файла и Subversion больше не будет считать, что файл находится в состоянии конфликта.





ВАРИАНТЫ РАЗРЕШЕНИЯ КОНФЛИКТОВ

Объединить конфликтующий текст
«вручную» (путем анализа и
редактирования маркеров конфликта
в файле)

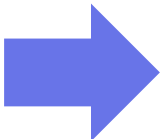
Скопировать один из временных
файлов поверх своего рабочего файла

Выполнить `svn revert <filename>` для
того, чтобы убрать все локальные
изменения

07. Полезные команды

<<<<<<< *имя файла*
ваши изменения
=====
результат автоматического слияния с репозиторием
>>>>>>> *ревизия*

Конфликтный файл




Необходимо разрешить конфликты
вручную редактированием данного
файла или через сторонние
приложения



svn resolved sandwich.txt
svn commit -m "Go ahead and use my
sandwich, discarding Sally's edits."

Пример последовательности команд

После слияния изменений,
необходимо выполнить svn resolved,
а после можно фиксировать
изменений



ВАРИАНТЫ РАЗРЕШЕНИЯ КОНФЛИКТОВ

Объединить конфликтующий текст
«вручную» (путем анализа и
редактирования маркеров конфликта
в файле)

Скопировать один из временных
файлов поверх своего рабочего файла

Выполнить `svn revert <filename>` для
того, чтобы убрать все локальные
изменения

07. Полезные команды

```
<<<<<<< имя файла  
ваши изменения  
=====  
результат автоматического слияния с репозиторием  
>>>>>>> ревизия
```

Конфликтный файл



Если был получен конфликт, то
вместо анализа можно отбросить
изменения и начать сначала, просто
отменив изменения



Обратите внимание!
Когда вы возвращаете файл к
предыдущему состоянию, вам не нужно
выполнять `svn resolved`

```
svn revert sandwich.txt  
Reverted 'sandwich.txt'
```

Пример последовательности команд

АНАЛИЗ ИЗМЕНЕНИЙ

ЦЕЛИ

составление более
аккуратного лог-
сообщения

обнаружение
непреднамеренного
изменения файла

пересмотр и проверка
изменений перед их
публикацией

svn status

найти изменённые файлы
рабочей копии

svn diff

найти изменённые файлы
рабочей копии

svn revert

отменить некоторые из
файлов

07. Полезные команды

svn status

Находит все сделанные файловые и структурные изменения. Под объектом подразумевается файл, директория или ссылка.

```
PS D:\Test_WorkingCopy\Test_Repository> svn status
M      bar.txt
```

| | | |
|----|-------------|---|
| ‘’ | | Без модификаций |
| A | Added | Объект запланирован для добавления |
| D | Deleted | Объект запланирован для удаления |
| M | Modified | Объект был изменён |
| R | Replaced | Объект был заменён внутри рабочей копии |
| C | Conflicting | Конфликтующий объект |

07. Полезные команды

svn status

Находит все сделанные файловые и структурные изменения. Под объектом подразумевается файл, директория или ссылка.

```
PS D:\Test_WorkingCopy\Test_Repository> svn status
M      bar.txt
```

| | | |
|---|----------|------------------------------------|
| X | eXternal | Объект был включён внешне |
| I | Ignored | Объект был проигнорирован |
| ? | | Объект не под версионным контролем |
| ! | | Объект не найден |
| ~ | | Объект поменял свой тип |

svn diff

Показывает, как именно были изменены элементы. В результате выведутся изменения файлов в виде единого формата представления различий. При этом удалённые строки предваряются -, а добавленные строки предваряются +.

```
PS D:\Test_WorkingCopy\Test_Repository> svn diff
Index: bar.txt
=====
--- bar.txt      (revision 12)
+++ bar.txt      (working copy)
@@ -1,1 @@
-
\ No newline at end of file
+
\ No newline at end of file
PS D:\Test_WorkingCopy\Test_Repository> |
```

ФИКСАЦИЯ ИЗМЕНЕНИЙ

07. Полезные команды

`svn commit`

отправляет все изменения в хранилище (фиксирует изменения)

При фиксации изменений необходимо указать описывающие ваши изменения лог-сообщение. Лог-сообщение будет присоединено к созданной правке.

```
PS D:\Test_WorkingCopy\Test_Repository> svn commit -m "Changing bar.txt"
Sending          bar.txt
Transmitting file data .done
Committing transaction...
Committed revision 13.
```

Однако, если вы составляли лог сообщения во время работы, можно указать Subversion взять лог сообщение из файла, передавая имя файла в параметре `--file`

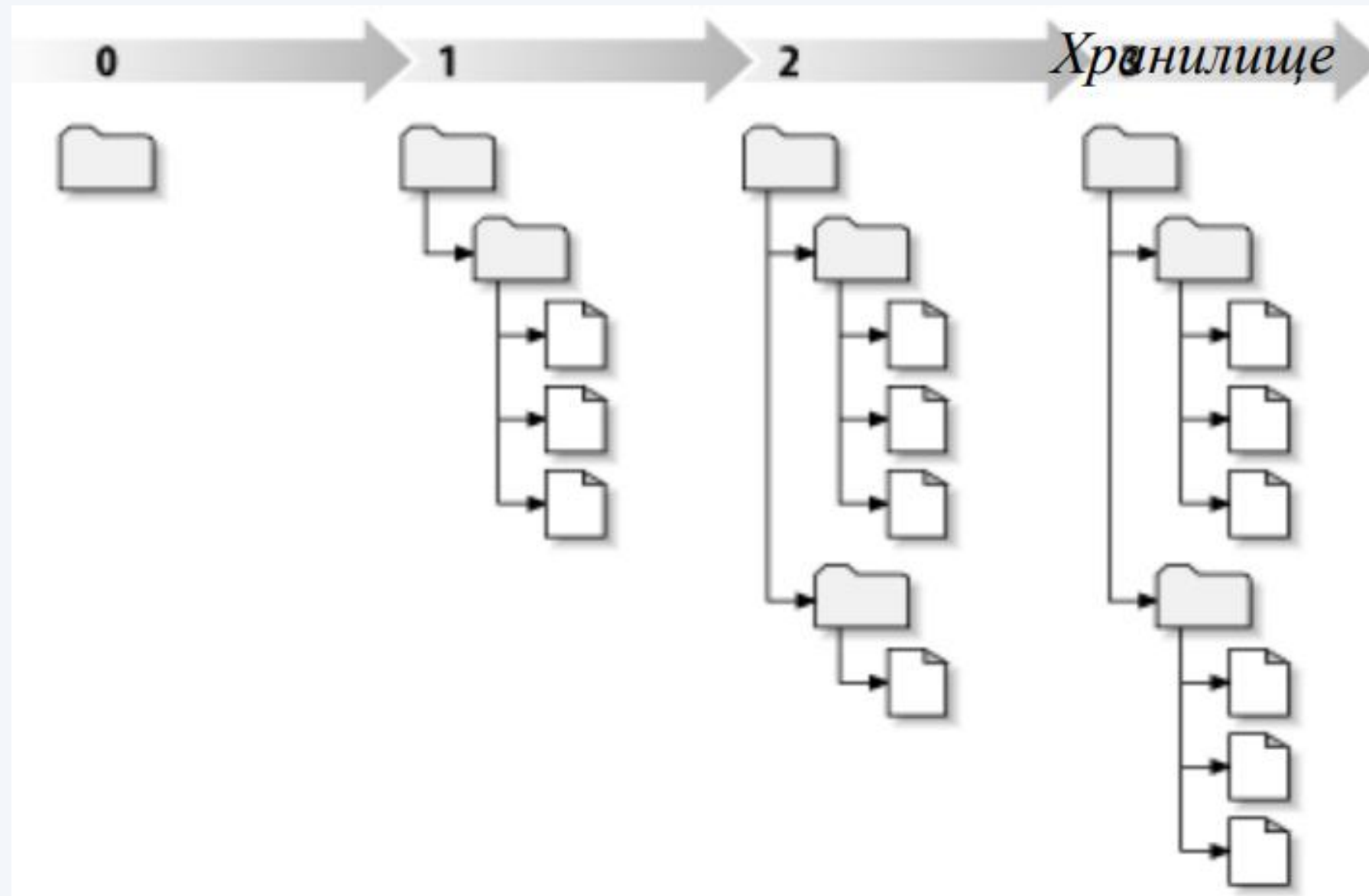
`svn commit --file logmsg.txt`

svn commit



Хранилище, в целом, не знает и не заботится о смысле ваших изменений; оно только контролирует, что бы никто не изменил те же файлы что и вы. Если это все-таки случилось вся фиксация будет отклонена с сообщением информирующим вас, что один или несколько файлов устарели

```
$ svn commit --message "Add another rule"  
Sending  
rules.txt  
svn: Commit failed (details follow):  
svn: Out of date: 'rules.txt' in transaction 'g'
```



Каждый раз, когда происходит фиксация, создаётся новое состояние файловой системы, которое называется **правка**. Каждая правка получает уникальный номер, на единицу больший номера предыдущей правки.

Начальная правка только что созданного хранилища получает номер 0 и не содержит ничего, кроме пустого корневого каталога.

ПРОСМОТР ИСТОРИИ ИЗМЕНЕНИЙ

07. Полезные команды

svn log

Показывает вам развернутую информацию: лог сообщения, присоединенные к правкам, с указанной датой изменений и их автором, а также измененные в правке пути файлов

```
PS D:\Test_WorkingCopy\Test_Repository> svn log
-----
r12 | krosh | 2023-06-04 10:16:53 +0300 (Вс, 04 июн 2023) | 1 line
aaa11
-----
r11 | krosh | 2023-06-04 10:13:33 +0300 (Вс, 04 июн 2023) | 1 line
aaa1
-----
r10 | krosh | 2023-06-04 10:12:19 +0300 (Вс, 04 июн 2023) | 1 line
aaa
```

Обратите внимание на то, что по умолчанию лог сообщения выводятся в обратном хронологическом порядке.

Если вам нужно увидеть другой диапазон правок в заранее определенном порядке или только одну правку, укажите параметр `--revision` (`-r`): `svn log --revision 5:19`

ДРУГИЕ ПОЛЕЗНЫЕ КОМАНДЫ

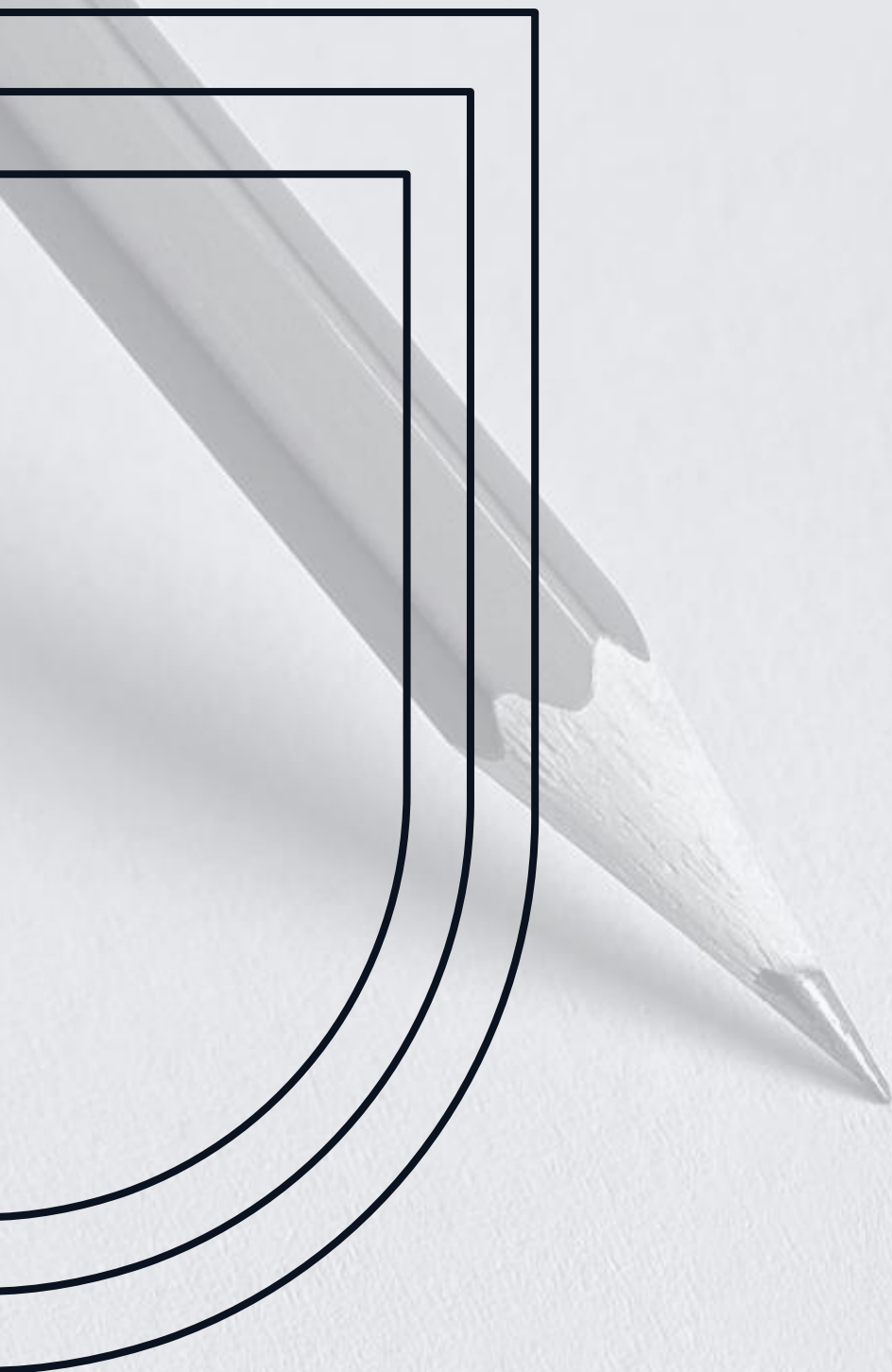
07. Полезные команды

svn cleanup

выполняет поиск и выполнение незавершенных
лог файлов, удаляя по ходу выполнения
блокировки в рабочей копии

Если Subversion когда-нибудь говорила вам о
том, что часть рабочей копии «заблокирована»
тогда вам нужно запустить эту команду





КЛЮЧЕВЫЕ СЛОВА ПРАВOK

HEAD

Последняя (или «самая новая») правка хранилища

BASE

Номер правки элемента рабочей копии

COMMITTED

Правка, в которой элемент последний раз редактировался

PREV

Правка, предшествующая последней правке, в которой элемент был изменен


```
$ svn diff --revision PREV:COMMITTED foo.c
```

```
# показать последнее изменение зафиксированное для foo.c
```

```
$ svn log --revision HEAD
```

```
# показать лог для последней фиксации хранилища
```

```
$ svn diff --revision HEAD
```

```
# сравнить ваш рабочий файл (с учетом локальных изменений)
```

```
# с последней правкой в хранилище
```

```
$ svn diff --revision BASE:HEAD foo.c
```

```
# сравнить ваш «исходный» foo.c (без учета локальных
```

```
# изменений) с последней версией в хранилище
```

```
$ svn log --revision BASE:HEAD
```

```
# показать все логи фиксаций со времени вашего последнего обновления
```

**КЛЮЧЕВЫЕ
СЛОВА ПРАВOK**

ИГНОРИРОВАНИЕ ФАЙЛОВ

07. Полезные команды

Игнорирование

процесс, позволяющий хранить файлы в репозитории не под версионным контролем. Subversion не предлагает добавить «заигнорированные» файлы в следующий КОММИТ.

Глобальное

никак не отражается на репозиториях других пользователей или на центральном хранилище

Локальное

действует внутри репозитория и необходимо, чтобы определённые файлы были не видны всем его пользователям

Например, локальное игнорирование файлов с расширением jpg:

```
svn propset svn:ignore "*.jpg".
```

Для просмотра игнорированных файлов, необходима команда `svn status` с ключом `—no-ignore`.

```
svn status --no-ignore | grep "^I"
```

Вы получите следующий результат:

```
I    myimage.jpg
```

МЕТКИ

07. Полезные команды

Метки

это специальные обозначения, прикреплённые к коммиту. С помощью них удобно искать коммиты, в которых приложение было в определённом состоянии (например, релиз 1.0, релиз 1.2)

Для меток в Subversion нет специальной команды, они выглядят как дешевые копии (cheap copies), или ссылки

```
svn copy http://svn.example.com/project/trunk \  
http://svn.example.com/project/tags/1.0 -m "Release 1.0"
```



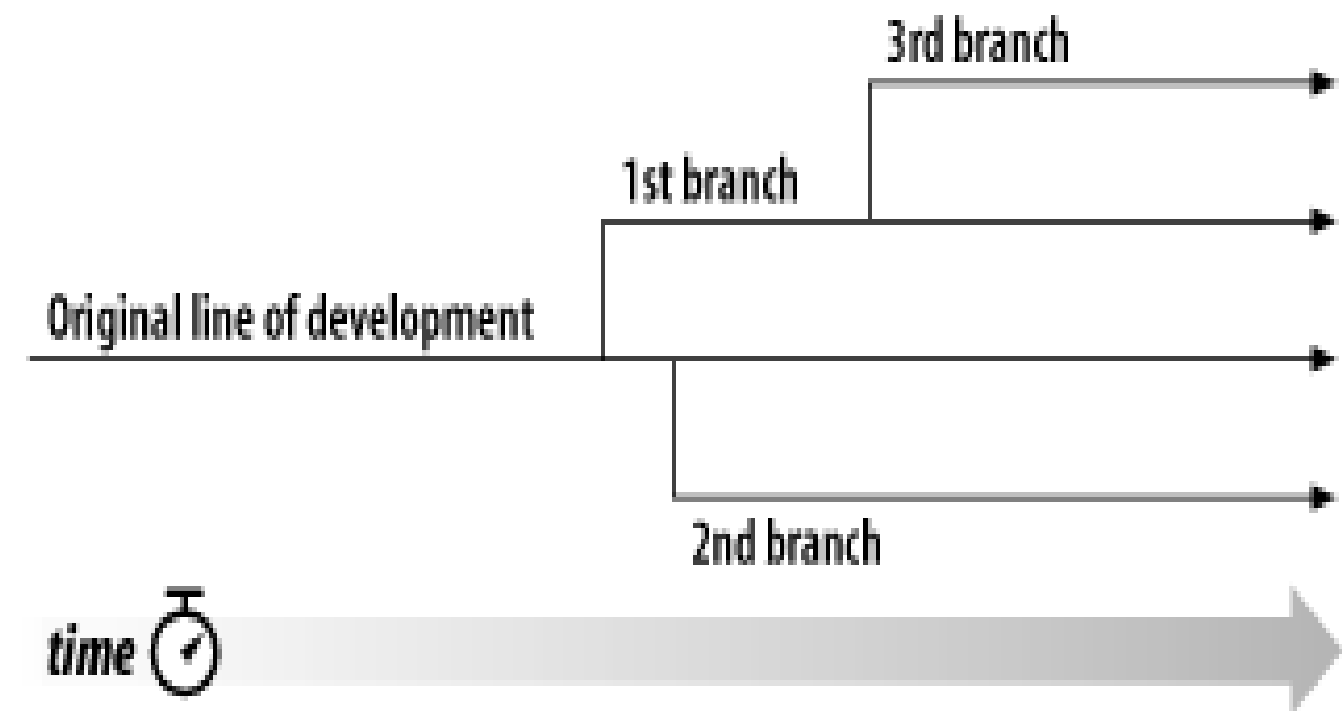
08.



Ветвление в Subversion

Ветка

Ветка - это направления разработки, которое существует независимо от другого направления, однако имеющие с ним общую историю, если заглянуть немного в прошлое. Ветка всегда берет начало как копия чего-либо и движется от этого момента создавая свою собственную историю.



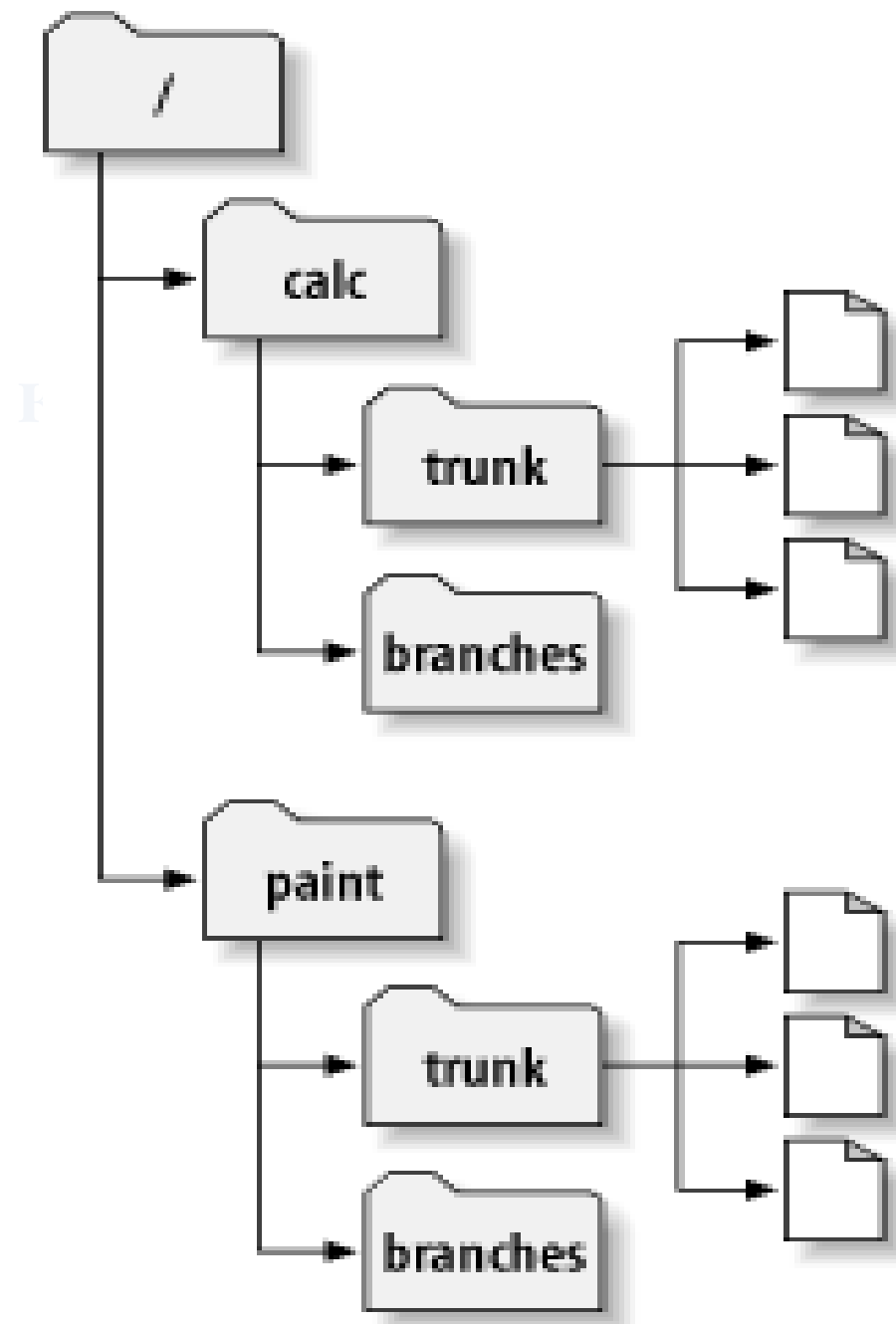
08. Ветвление в Subversion

Использование веток

Рассмотрим пример: вы и ваш соразработчик Салли делите хранилище, содержащее два проекта, paint и calc. Каждая директория проекта содержит поддиректории с названиями trunk и branches.

Недостатки работы в одной ветви

- Во-первых это не очень надежно. Большинство людей предпочитают часто сохранять свою работу в хранилище, на случай если вдруг что-то плохое случится с рабочей копией;
- Во-вторых, это не достаточно гибко. Если вы работаете на разных компьютерах, вам придется вручную копировать изменения взад и вперед, либо делать всю работу на одном компьютере.



Способы создания ветки



С клонированием рабочей копии и созданием ветки

Для начала, создается рабочая копия
корневой директории проекта /calc:

```
$ svn checkout  
http://svn.example.com/repos/calc bigwc
```

Теперь создание копии заключается в
простой передаче двух путей в пределах
рабочей копии команде svn copy:

```
$ cd bigwc  
$ svn copy trunk branches/my-calc-branch  
$ svn status  
A + branches/my-calc-branch
```

Просто с созданием новой ветки, которая вам необходима

кто их сделал

```
$ svn copy  
http://svn.example.com/repos/calc/trunk \  
http://svn.example.com/repos/calc/branch  
es/my-calc-branch \  
-m "Creating a private branch of  
/calc/trunk."  
Committed revision 341.
```



Работа с веткой

После создания ветки проекта, можно создать новую рабочую копию для начала ее использования:

```
$ svn checkout
```

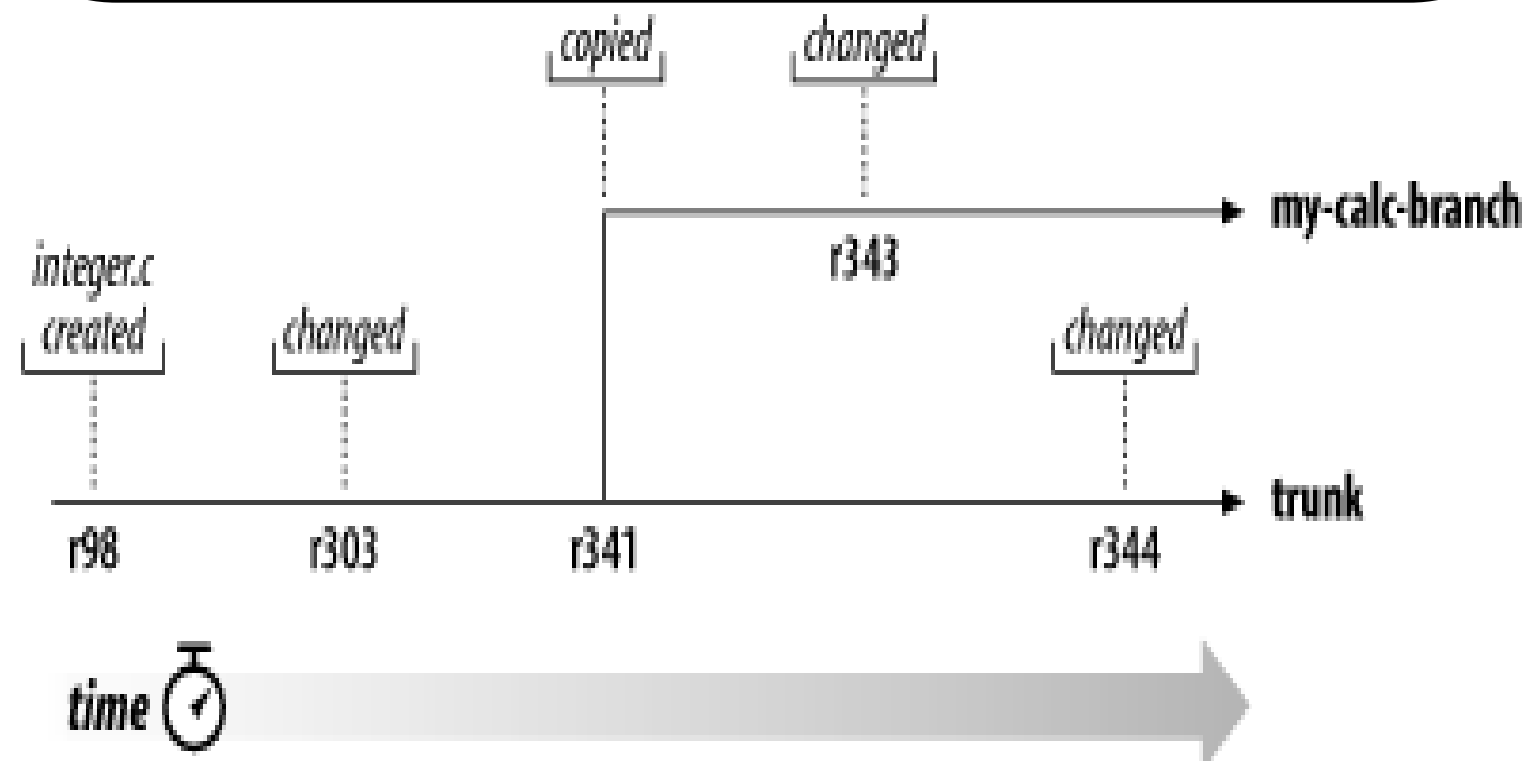
```
http://svn.example.com/repos/calc/branches/my-calc-branch
```

```
A my-calc-branch/Makefile
```

```
A my-calc-branch/integer.c
```

```
A my-calc-branch/button.c
```

```
Checked out revision 341.
```



Ключевые идеи ветвления в SVN

1. В отличие от других систем управления версиями, ветки в Subversion существуют в хранилище не в отдельном измерении, а как обычные нормальные директории файловой системы. Такие директории просто содержат дополнительную информацию о своей истории;
2. Subversion не имеет такого понятия как ветка — есть только копии. При копировании директории результирующая директория становится «веткой» только потому что вы рассматриваете ее таким образом.



Копирование отдельных изменений

Команда `svn merge` способна сравнивать любые два объекта в хранилище и показывать изменения, а также их применять.

```
$ svn merge -r 343:344  
http://svn.example.com/repos/calc/trunk  
U integer.c  
$ svn status  
M integer.c
```

Копирование отдельных изменений

После просмотра результата объединения изменений, можно их как обычно зафиксировать (`svn commit`).

После этого изменения будут внесены в вашу ветку хранилища. В терминах контроля версий такую процедуру копирования изменений между ветками обычно называют портированием изменений (слиянием, “мерджем” (`merge`)).



КЛЮЧЕВЫЕ ПОНЯТИЯ, СТОЯЩИЕ ЗА СЛИЯНИЕМ

Команда `svn merge` принимает три аргумента:

1. Начальное дерево хранилища (как правило, называемое левой частью при сравнении),
2. Конечное дерево хранилища (как правило называемое правой частью при сравнении),
3. Рабочую копию для применения отличий, в виде локальных изменений (как правило, называемую целью слияния).

Когда эти три аргумента указаны, сравниваются два дерева и результирующие различия применяются к целевой рабочей копии в виде локальных изменений. Если результат вас не устраивает, просто отмените (`svn revert`) все сделанные изменения.

Лучше всего её было бы назвать `diff-and-apply`, т.к. она делает именно это.

09.



Конфликты слияния в Subversion



Объединение конфликтов вручную

Конфликтный файл будет выглядеть так:

<<<<<< имя файла

ваши изменения

=====

результат автоматического слияния с
репозиторием

>>>>>> ревизия

Вам необходимо разрешить конфликты вручную редактированием данного файла или через сторонние приложения. Когда вы выполните слияние изменений, выполните `svn resolved` и вы готовы к фиксации изменений:

```
$ svn resolved sandwich.txt
```

И, наконец, зафиксируйте изменения:

```
$ svn commit -m "Go ahead and use my sandwich, discarding Sally's edits."
```

Использование `svn revert`

Если вы получили конфликт и вместо анализа решаете отбросить изменения и начать сначала, просто отмените ваши изменения:

```
$ svn revert sandwich.txt
```

Reverted 'sandwich.txt'

Обратите внимание, что когда вы возвращаете файл к предыдущему состоянию, вам не нужно выполнять `svn resolved`.

Мы рассмотрели необходимые операции для совместной работы с Subversion, теперь перейдём к Git.