Использование Emacs совместно с системами контроля версий

Alex Ott

30 января 2008 г.

1 Введение

Emacs, как средство разработки программ, и других сложных вещей имеет в своем составе модули работы с системами контроля версий. Существует два подхода к работе с системами контроля версий используя Emacs. Первый метод — через общий интерфейс, который предоставляется модулем VC и которые знает про самые разные системы контроля версий. Второй метод — использование модулей, написанных специально для конкретной системы контроля версий.

Достоинством первого метода является то, что все операции производятся унифицировано, а переключение между конкретными системами зависит от настройки. К недостаткам этого подхода можно отнести то, что при этом не используются все возможности конкретной системы контроля версий.

В свою очередь, использование специальных модулей для каждой из конкретных систем контроля версий позволяет задействовать в работе все ее возможности. К недостатком этого подхода можно отнести то, что при этом используются разнородные привязки клавиш и функции.

В данной статье мы рассмотрим оба подхода к поддержке систем контроля версий. В начале рассказ пойдет о пакете VC, а в следующих за ним разделах будут рассмотрены модули поддержки конкретных систем контроля версий.

2 Пакет VC

Пакет VC предоставляет пользователю унифицированный интерфейс к различным системам контроля версий. В настоящее время существуют интерфейсы к системам RCS, CVS, MetaCVS, Subversion, GNU Arch, SCCS, Darcs, SourceSafe, ClearCase, Git и Mercurial (некоторые из них доступны только в версии GNU Emacs 22.0, репозитории GNU Emacs или с сайтов авторов). К тому же не все модули имеют поддержку всех команд, которые предоставляются пакетом VC, например, модуль vc-arch не поддерживает команду `vc-cancel-version`, и некоторые другие.

2.1 Основные понятия и подходы

На развитие пакета VC очень сильное влияние оказала система контроля версий RCS, поэтому некоторые концепции напрямую связаны с принципами использования данного пакета.

Когда файл находится под контролем системы контроля версий, то говорят, что файл «зарегистрирован»в системе контроля версий. Каждому файлу соответствует «главный»файл, который содержит данные и историю изменений. «Главному»файлу может соответствовать «рабочий»файл, в который можно вносить изменения. В некоторых системах контроля версий, вы должны «блокировать»рабочий файл, перед внесением изменений. В других системах контроля версий, вы можете вносить изменения в собственные рабочие версии файлов, и должны явно подтверждать изменения в центральный репозиторий. Для систем контроля версий, в которых требуются блокировка файлов, для внесения изменений необходимо перевести файл из состояния «только для чтения», в режим доступности для записи. Плюс, в таких системах контроля версий, обычно только один пользователь может одновременно вносить изменения в файлы, а остальные пользователи должны ожидать окончания работы данного пользователя. Это в первую очередь касается системы контроля версий RCS.

При работе с пакетом VC некоторая информация отображается в строке состояния. Информация представляется в следующем виде — сначала идет сокращенное название системы контроля версий, контролирующей данный файл, затем идет символ, описывающий состояние файла, а затем отображается номер версии данного файла. Если символ, описывающий состояние файла равен `-`, то файл не заблокирован или не изменен, а если он заблокирован или изменен, то будет использоваться символ `:`. В том случае, если файл был заблокирован другим пользователем, то поле статуса будет выглядеть как `:user:`.

2.2 Работа с пакетом

2.2.1 Основная цепочка действий

Основной командой в пакете VC является `vc-next-action`, которая выполняет следующую логическую команду для данной системы контроля версий. В зависимости от используемой системы контроля версий, порядок команд в цепочке действий может отличаться.

Для систем, которые требуют блокировки файла (это RCS, SCCS, и CVS при соответствующей настройке), порядок действий следующий:

1. если файл не зарегистрирован, то он регистрируется;
2. если файл зарегистрирован и не заблокирован, то команда вытягивает свежий файл, и блокирует его;
3. если файл был заблокирован пользователем, но не изменялся, то выполняется обновление буфера до новой версии;
4. если файл изменялся, то эта команда приводит к появлению буфера для ввода описания изменения, а затем подтверждает изменения;

5. если файл был заблокирован кем-то другим, то VC старается получить блокировку.

Для систем не поддерживающих концепцию блокировок, порядок выполнения действия следующий:

1. если файл не зарегистрирован, то он регистрируется, но изменения не подтверждаются;

2. если файл уже был добавлен, но не подтвержден, то это изменение подтверждается;

3. если файл был изменен, а «главный»файл нет, то ваши изменения подтверждаются после ввода описания изменений;

4. если файл в репозитории был изменен, то пользователю выдадут запрос на слияние изменений в ваш рабочий файл.

Пользователь может задавать префиксные аргументы для команды `vc-next-action`. Использование префиксного аргумента изменяет поведение команды, что позволяет изменять номер версии под которой файл будет зарегистрирован, номер версии, которую необходимо получить, или изменить имя используемой системы контроля версий, что часто полезно, если файл был зарегистрирован в нескольких системах контроля версий.

Ввод описания изменения производится в специальном буфере, который называется `*VC-Log*`. В нем вы можете ввести сообщение, и подтвердить изменения используя сочетание клавиш **C–c C–c**, находясь в этом буфере. Чтобы не подтверждать изменения, просто покиньте это буфер без использования **C–c C–c**. А если вы хотите использовать одно и то же сообщение для нескольких файлов, то вы можете воспользоваться сочетаниями клавиш **M–n**, **M–p**, **M–s** и **M–r**, которые работают так же как и в минибуфере, позволяя перемещаться по истории сообщений.

2.2.2 Работа с изменениями

Пакет VC позволяет пользователю просматривать изменения внесенные в процессе работы с файлами. Для получения списка изменений между вашей рабочей версией и последней подтвержденной версией, пользователь может воспользоваться командой `vc-diff` (сочетание клавиш **C–x v =**). При использовании данной команды с префиксным аргументом, пользователь может получить список изменений между двумя произвольными версиями. Команда `vc-version-other-window` (сочетание **C–x v ~**) позволяет получить произвольную версию файла, и показать ее в отдельном буфере. А некоторые версии систем контроля версий позволяют получать аннотированные версии файлов, находящихся под управлением системы контроля версий. Эту задачу реализует команда `vc-annotate`, которая запускается сочетанием **C–x v g**. Аннотированная версия позволяет понять как вносились изменения, и просматривать изменения внесенные в конкретных версиях.

Вы можете отменить внесенные изменения с помощью команд, определенных пакетом. Пакет определяет две команды отмены изменений — `vc-revert-buffer` (сочетание **C–x v u**), которая возвращает файл связанный с буфером к версии, на которой он был основан, а команда `vc-cancel-version` (сочетание **C–x v c**) отменяет последнее подтвержденное изменение. Для последней команды, вы можете не откатывать изменения

для текущего буфера, так что вы можете внести новые изменения, основанные на промежуточных результатах. После выполнения команды `vc-revert-buffer` файл остается не заблокированным, так что для внесения изменений, вас необходимо снова заблокировать его.

2.2.3 Работа с файлами

Регистрация файла в системе контроля версий производится с помощью команды `vc-register` (сочетание клавиш **C–x v i**). Выбор системы контроля версий в которой файл будет зарегистрирован, зависит от нескольких параметров. Если в данном каталоге есть файлы, зарегистрированные в какой-либо системе контроля версий, то используется именно она. Если в каталоге находятся файлы зарегистрированные в нескольких системах контроля версий, то используется та, которая находится первой в списке `vc-handled-backends`. А если нет ни одного зарегистрированного файла в данном каталоге, то используется первая запись из списка `vc-handled-backends`. Так что вы зарегистрируете файл в CVS в пустом каталоге только явно указав использовать эту систему контроля версий :-)

По умолчанию, новый файл получает номер версии равный 1.1. Вы можете использовать другое число, если вы используете числовой аргумент при выполнении данной команды. В том случае, если переменная `vc-initial-comment` имеет не пустое значение, то данная команда запросит у пользователя комментарий, который будет использован для описания файла.

Пакет VC поддерживает переименование зарегистрированных файлов. Для этой цели вы можете использовать команду `vc-rename-file`, которая переименовывает рабочий файл, а также «главный»файл. К тому же эта команда, обновляет все снимки, чтобы они ссылались на новое имя файла. Но вы не можете использовать эту команду для заблокированных файлов.

Для получения информации о текущем файле вы можете использовать команду `vc-print-log`, которая привязана к сочетанию клавиш **C–x v l**. Данная команда отображает историю изменений, включая текст комментариев к этим изменениям.

2.2.4 Использование VC совместно с Dired

Команды описанные в предыдущих разделах в первую очередь предназначены для работы с отдельными файлами. Но для больших проектов, часто необходимо работать с множеством файлов — просматривать какие файлы изменились, список изменений и т.п. Для реализации этих действий, пакет VC поддерживает специальный режим — `vc-dired-mode`. Для создания буфера, использующего такой режим, используется команда `vc-directory` (сочетание **C–x v d**). Буфер созданный такой командой почти не отличается от обычного буфера Dired. В буфер помещается рекурсивный список файлов, зарегистрированных в системе контроля версий.

При работе с буфером dired, данные могут отображаться в сокращенном или полном режиме. В сокращенном режиме показываются только измененные или не обновленные файлы. В полном режиме, показываются все файлы зарегистрированные в системе контроля версий. Пользователь может переключаться между полным и сокращенным режимом с помощью сочетания клавиш **v t**. При показе данных в буфере, отображается

имя пользователя (для систем поддерживающих блокировки) или статус файла (для остальных систем).

В буфере с `vc-dired-mode` действуют все команды dired, за исключением команды **v**, которая используется в качестве префиксной для команд пакета VC. Команды VC те же самые, что и описанные в прочих разделах, только не используется префикс **C–x**, и они могут выполняться к множеству помеченных файлов.

2.2.5 Ветви версий (branches)

Как и другие пакеты для интеграции с системами контроля версий, пакет VC поддерживает работу с разными ветвями версий. Пакет поддерживает создание ветвей, переключение между ними, а также слияние изменений между разными ветвями. Для переключения на другую ветвь разработки пользователю необходимо лишь задать префиксный аргумент для команды `vc-next-action`. Пользователь может указывать не конкретную версию, а лишь номер ветви. Следующая работа будет производиться именно в этой ветви.

Создание новой версии производится аналогичным образом — необходимо лишь при подтверждении изменений с помощью команды `vc-next-action` указать новый номер версии используя префиксный аргумент. Если пользователь укажет номер для не существующей ветви, то она будет создана. После создания ветви, все остальные действия будут производиться в рамках этой ветви. Для работы с другой веткой, пользователь должен явно переключиться на нее.

Для слияния изменений используется команда `vc-merge`, которая привязана к сочетанию клавиш **C–x v m**. Данная команда запрашивает у вас откуда необходимо брать изменения, и применяет их к вашей рабочей версии. Если вы ничего не ввели в ответ на запрос, то будут взяты изменения из ветви с которой вы работаете, если они были подтверждены за время работы с вашим файлом. Если во время работы возникают конфликты, то пакет оповещает пользователя об этом, и помечает конфликты в файле, или даже может запустить Ediff для их разрешения. Вы можете воспользоваться командой `vc-resolve-conflicts` для разрешения возникших конфликтов. Она запускает новую сессию Ediff, где вы можете просматривать возникшие конфликты, и применять или отвергать предлагаемые изменения.

2.2.6 Взаимодействие с удаленными репозиториями

Некоторые системы контроля версий поддерживают концепцию центрального репозитория, который хранит «главные»файлы, а также историю изменений. Репозиторий может быть локальным, или располагаться на удаленном сервере. VC автоматически распознает тот случай, когда репозиторий располагается на удаленном сервере, и принимает некоторые действия, которые позволяют более комфортно работать с данными, особенно в тех случаях, когда связь является медленной и нестабильной.

При первом редактировании файла, создается локальная копия изменяемого файла. Это позволяет выполнять команды получения списка изменений, или отмены внесенных изменений, без какого-либо сетевого взаимодействия. Создаваемая копия файла имеет имя `FILE.~VERSION.~`. При подтверждении изменений в репозиторий, локальная копия удаляется. Пользователи также могут создавать резервные копии вручную, используя

команду `vc-version-other-window`, разница между резервными копиями созданными автоматически и вручную, лишь в лишнем знаке точка перед знаком `~`. Многие команды VC могут использовать любую из резервных копий.

2.2.7 Работа со «снимками»

«Снимок»(snapshot) — это именованный набор версий файлов, который может рассматриваться как отдельная единица. Некоторые системы контроля версий сами поддерживают концепцию «снимков», тогда пакет VC использует эти возможности. Для прочих систем контроля версий, пакет VC сам реализует поддержку «снимков». В этом случае, эти «снимки»видны только при использовании пакета VC.

Для работы со «снимками»пакет VC определяет две команды. Команда `vc-create-snapshot` (сочетание **C–x v s**) создает «снимок»на базе версий файлов находящихся в текущем каталоге, или ниже. Команда `vc-retrieve-snapshot` (сочетание клавиш **C–x v r**) позволяет пользователю получить «снимок»с заданным именем.

Многие команды пакета VC позволяют использовать имена «снимков»в качестве аргументов вместо номеров версий файлов. Это особенно полезно просмотре списка изменений между релизами программ.

2.2.8 Прочие команды

Обычно, для файлов находящихся под управлением системы контроля версий, для каждого файла ведется отдельный список изменений. Но в проектах GNU списки изменений ведутся для всего проекта, или для отдельных каталогов. Для организации взаимодействия между этими двумя подходами, пакет VC определяет отдельную команду `vc-update-change-log` (она привязана к сочетанию **C–x v a**), которая обновляет общий список изменений на основании списка изменений для отдельных файлов. При использовании префиксов, данная команда позволяет обновить общий список изменений, только изменениями для отдельных файлов.

Вы можете вставлять в файлы специальные идентификационные заголовки. Эти заголовки зависят от конкретной используемой системы контроля версий, и в некоторых системах, они используются для служебных целей. Для вставки такого заголовка, вы можете использовать команду `vc-insert-headers` (сочетание клавиш **C–x v h**). VC автоматически выбирает заголовок подходящий для системы контроля версий. Заголовки хранятся в переменных вида `vc-BACKEND-header`, так что вы можете изменять их значения, для вставки собственных заголовков.

2.3 Настройка

Используя стандартные средства настройки Emacs пользователь может настроить поведение выполняемых команд. Для этого необходимо лишь выполнить `M-x customize-group vc`. Часть опций применяется ко всем поддерживаемым системам контроля версий, часть применяется только к конкретным реализациям. Полное описание основных переменных, используемых для настройки пакета VC вы можете найти в руководстве по GNU Emacs.

2.4 Дополнительная информация

Некоторые из подсистем, обеспечивающих работу пакета VC с разными системами контроля версий доступны только с сайтов их авторов, и пока не включены в поставку GNU Emacs или XEmacs. Вы можете найти эти модули по адресам: vc-darcs[1], vc-vss[2], vc-clearcase[3].

Много информации относящейся к пакету VC, а также к другим модулям вы можете найти на сайте Emacs WiKi[4]. Кроме этого, подробное описание пакета вы можете найти в руководстве по Emacs.

3 Пакет DVC

С распространением распределенных систем контроля версий (distributed version control systems), появилась необходимость в реализации их поддержки в Emacs. Для поддержки конктретных реализаций распределнных систем контроля версий были созданы отдельные пакеты, которые реализовали поддержку специфической системы, с учетом разницы в идеологии и командах.

Однако как упоминалось ранее, для поддержки большинства централизованных систем контроля версий существует пакет VC, описанный в разделе Пакет VC, поэтому была начата разработка пакета DVC, который должен реализовать общее ядро для поддержки различных распределнных систем контроля версий. Пакет DVC во многом является продолжением разработки пакета xtla, и разрабатывается той же командой разработчиков. Подробно о пакете можно прочитать на сайте проекта[5].

3.1 Архитектура пакета и поддерживаемые системы контроля версий

DVC имеет многоуровневую архитектуру. На самом верхнем уровне находятся функции обеспечивающие автоматическое определение системы контроля версий и выполнение функций, общих для всех систем контроля версий. Уровнем ниже находится функции, специфические для каждой из систем контроля версий. И на самом нижнем уровне находятся функции, используемые подсистемами, отвечающими за работу с конкретными системами контроля версий.

В настоящее время DVC поддерживает следующие системы контроля версий:

[1] http://www.emacswiki.org/elisp/vc-darcs.el
[2] http://www.chezmarshall.freeserve.co.uk/elisp/vcvss.html
[3] http://www.fukt.hk-r.se/~flognat/vc/
[4] http://www.emacswiki.org/cgi-bin/wiki/CategoryVersionControl
[5] http://download.gna.org/dvc/

название в dvc	система контроля версий
xhg	mercurial (он же hg)
xgit	git
baz	GNU Arch (tla) и Bazaar 1.x
bzr	Bazaar-NG
xmtn	Monotone
xdarcs	Darcs

Не все системы контроля версий имеют полную поддержку в DVC. На сайте[1] вы можете посмотреть таблицу, в которой перечислены поддерживаемые для каждой из систем функции.

3.2 Установка пакета

Установка пакета достаточно проста - необходимо скачать дистрибутив с сайта проекта, распаковать его, и установить с помощью станлартной последовательности команд:

```
./configure
make
make install
```

Для того, чтобы воспользоваться функциями пакета необходимо добавить в файл инициализации следующие строки:

```
(add-to-list 'load-path "path_to_installed_package")
(require 'dvc-autoloads)
```

выполнение которых приведет к регистрации функций автозагрузки частей пакета при первом обращении к ним. После выполнения этих команд можно выполнить настройку пакета (см. раздел Настройка), но это не обязательно, поскольку настроек по умолчанию обычно хватает для начала работы.

3.3 Начало работы с пакетом

DVC можно пользоваться используя общее знание о соответствующей системе контроля версий - для всех основных команд системы контроля версий имеются соответствующие функции DVC, начинающиеся с префикса **dvc-**[2].

Основной командой используемой пользователем можно считать команду **dvc-status** (она доступна через глобальное сочетание клавиш **C–x V s**), которая создает буфер со списком измененных файлов с который похож на интерфейс PCL-CVS. Название буфера зависит от того, какая система контроля версий используется для данного репозитория, но в общем виде оно выглядит как `*VCS-status*`, где `VCS` заменяется

[1] http://download.gna.org/dvc/

[2] Кроме того, каждый из модулей, реализующий поддержку конкретной системы контроля версий, определяет собственный набор функций, соответствующих командам этой системы контроля версий и который может быть отличным от команд DVC. Например, модуль реализующий поддержку Mercurial объявляет функции с префиксом **xhg-**, которые могут напрямую быть вызваны пользователем, если для них нет соответствия среди функций DVC.

на название соответствующего backend, реализующего функции работы с конкретной системой контроля версий.

После получения буфера со статусом репозитория, в нем можно производить различные операции. В этом буфере автоматически включается режим **dvc-diff**, для которого определено некоторое количество специальных команд. Интерфейс и управляющие клавиши очень похожи на те, которые применяются в PCL-CVS, так что на освоение пакета DVC уходит очень мало времени.[1] Часть операций также доступна для выполения через меню. DVC создает несколько меню — `DVC-Diff`, `DVC-Buffers` и отдельное меню по названию используемой системы контроля версий, которое предназначено для выполнения команд, относящихся к конкретной системе. `DVC-Buffers` содержит список открытых буферов DVC и используется для операций над этими буферами. Меню `DVC-Diff` в свою очередь используется для доступа к командам, общим для всех систем контроля версий — получение изменений, удаление файлов, отмена изменений и т.п.

Также как и в PCL-CVS, некоторые команды могут выполняться не нат отдельными файлами, а над группами. Для установки пометки используется клавиша **m**, а для снятия пометки — **u**. Кроме того, можно воспользоваться клавишей **backspace**, которая снимает пометку с предыдущего файла.

Наиболее часто при работе с репозиторием используется лишь ограниченное количество команд — просмотреть сделанные изменения и подтвердить сделанные изменения (commit) или наоборот вернуться к предыдущим версиям, добавить или удалить файлы, просмотреть журнал изменений. Для всех этих операций, DVC предоставляет соответствующие команды.

Просмотр файла осуществляется либо с помощью функции **dvc-diff-jump-to-change** (клавиша **RET**), которая открывает файл и переходит к первой изменённой строке, либо с помощью функции **diff-goto-source** (клавиша **o** или средняя кнопка мыши). С помощью команды **dvc-dired-jump** (**C–x C–j**) можно также перейти к нужному файлу в буфере Dired.

Чтобы откатить сделанные изменения можно воспользоваться командой **dvc-revert-files** (клавиша **U** или **C–x V f R**). А вот подтверждение изменений может быть сделано одной из двух команд: **dvc-log-edit** или **dvc-add-log-entry**. **dvc-log-edit** (клавиша **c** или **C–x V c**) открывает буфер, в который вы можете ввести текст, описывающий изменения, и затем выполнить команду **dvc-log-edit-done** (она привязана к привычному сочетанию клавиш **C–c C–c**). Вторая команда — **dvc-add-log-entry** (**t** или **C–x V a**) отличается от первой лишь тем, что сообщение будет выглядеть лучше — в стиле файлов `ChangeLog`. Находясь в буфере статуса, можно сохранить сделанные изменения в виде отдельного файла. Это выполняется с помощью функции **dvc-save-diff** (сочетание клавиш **W s**).

Для просмотра сделанных изменений определено несколько функций. Для непосредственного просмотра изменений из буфера `*VCS-status*` могут использоваться функции **dvc-diff-diff** (клавиша равно) или функция **dvc-diff-ediff** (клавиша **e**). Вторая функция отличается тем, что для просмотра изменений запускает ediff. Для быстрого переключения между буфером с изменениями и соответствующим элементом в буфере

[1] Стоит отметить, что многие команды также доступны вне буфера `*VCS-status*`. Для их выполнения надо к соответствующей клавиатурной команде добавить глобальный префикс **C–x V**. В данном тексте эти команды будут помечаться *глобальные*.

статуса, может использоваться функция **dvc-diff-diff-or-list** (клавиша **j**) — она работает в обоих буферах. А с помощью функции **dvc-diff-view-source** (клавиша **v**) можно открыть отдельное окно с файлом и рассмотреть изменённый участок файла. Для того, чтобы просматривать содержимое буфера с изменениями, не переключаясь в него, определены две функции — **dvc-diff-scroll-down-or-diff** (**M–=**) и **dvc-diff-scroll-up-or-diff** (**M–RET**), которые прокручивают буфер с изменениями вверх и вниз.

Кроме этих функций, определены ещё три функции, которые могут выполняться не из буфера `*VCS-status*` и имеют глобальные привязки клавиш. Функция **dvc-diff** (привязана к **C–x V =**) отображает изменения между текущим и предыдущим состояниями репозитория. Функции **dvc-file-diff** (**C–x V d**) и **dvc-file-ediff** (**C–x V e**) показывают изменения для конкретного файла, используя буфер или ediff, соответственно.

Для просмотра журнала изменений (log) также определено несколько команд, действующих как в буфере статуса, так и привязанных к глобальным сочетаниям клавиш. Так, команда **dvc-log** (**L** или **C–x V L**) показывает список изменений для файла или проекта в сокращённой форме. Для файла, находящегося в текущей позиции буфера статуса, можно посмотреть этот же журнал с помощью команды **dvc-diff-log** (клавиша **l**). Полный же журнал изменений можно посмотреть с помощью команды **dvc-changelog** (**C–x V l**). Все эти команды могут принимать префиксный параметр, определяющий то, сколько последних изменений необходимо показать. По умолчанию показываются все изменения.

3.4 Работа с файлами

Достаточно часто пользователю приходится выполнять и различные работы с файлами — добавлять, удалять их и т.п. DVC определяет несколько команд, которые предназначены для выполнения этих задач. Команда **dvc-add-files** (**a** или **C–x V f a**) добавляет текущий или выбранные файлы в репозиторий. Для удаления выбранных файлов из репозитория служит команда **dvc-remove-files** (клавиши **d** или **r**, или глобальное сочетание **C–x V f D**). Есть ещё одна команда для удаления файлов — **dvc-purge-files** (**C–x V f X**), которая удаляет файлы с диска, не оставляя резервных копий. Поскольку практически все современные системы контроля версий поддерживают операцию переименования файлов, то DVC также предоставляет эту возможность с помощью команды **dvc-rename** (**C–x V f M**).

Для работы с файлами, которые зарегистрированы в системе контроля версий, но не отображены в буфере статуса, пользователь может использовать команду **dvc-inventory** (**C–x V i**), которая отображает список файлов, известных данной системе контроля версий. В этом буфере пользователь может использовать те же команды, и сочетания клавиш, что и в буфере статуса.

Пользователь может управлять списком объектов (файлов и каталогов), которые будут известны системе контроля версий. Обычно, в каждой из систем контроля версий, существует список игнорируемых файлов, который может изменяться пользователем. Для быстрого выполнения этой задачи, определено несколько команд. Команда **dvc-ignore-files** (**# i** или просто **i**) помещает выбранные файлы в список игнорируемых объектов. Команда **dvc-ignore-file-extensions** (**# I**) также изменяет этот список, но туда попадают не полные имена файлов, а только их расширения, что часто бывает очень удобным. Ну а если вам необходимо изменить этот список другим способом — удалить файл из этого

списка, или добавить сложное регулярное выражение, то тут можно воспользоваться командой **dvc-edit-ignore-files** (# **e**), которая откроет список игнорируемых объектов и позволит его отредактировать. При этом синтаксис файла, зависит от используемой системы контроля версий.

3.5 Ветки и сторонние репозитории

Работа с разными репозиториями и ветвями является актуальной для пользователей распределённых систем контроля версий, поэтому DVC представляет достаточный набор возможностей для выполнения этих задач.

Для всех поддерживаемых систем контроля версий имеется общий набор команд, обеспечивающий основную функциональность. Но поскольку DVC является развитием проекта xtla, то для GNU Arch и Bazaar имеется больший набор функций — закладки для репозиториев и т.п. (описание соответствующих функций можно найти в разделе Пакет xtla. Модули для других систем контроля версий также могут предоставлять дополнительные команды.

Общими для всех систем контроля являются следующие команды:

Команда	Сочетание клавиш	Описание
dvc-missing	**M m** или **C–x V m**	показывает список изменений, присутствующих в удалённом репозитории
dvc-merge	**M M** или **C–x V M**	скачивает и применяет изменения из удалённого репозитория
dvc-pull	**M f** или **C–x V F**	скачивает изменения из удалённого репозитория (не применяя их)
dvc-update	**M u** или **C–x V u**	применяет скачанные изменения к текущему репозиторию
dvc-push	**C–x V P**	переносит изменения из текущего репозитория в удалённый репозиторий
dvc-submit-patch	**C–x V p**	отправляет сделанные в текущем репозитории изменения на указанный почтовый адрес
dvc-bookmarks	**C–x V b**	открывает буфер, содержащий закладки, с адресами удалённых репозиториев
dvc-clone	**C–x V C**	клонирует удаленный репозиторий

3.6 Прочие команды

Чтобы покинуть буфер со статусом репозитория, пользователь может выполнить команду **dvc-buffer-quit** (клавиша **q**). А чтобы обновить содержимое буфера, достаточно выполнить команду **dvc-generic-refresh** (клавиша **g**).

Получить справку по использованию пакета можно используя стандартные средства Emacs — находясь в буфере статуса, нажмите **C–h m** и вам будет выдано описание действующего режима, вместе со списком допустимых клавиш, и названий соответствующих команд. Список глобальных сочетаний клавиш, относящихся к пакету, можно получить с помощью сочетания **C–x V C–h**. Кроме того, в состав пакета входят

различные "советы"(tips), с которыми можно ознакомиться с помощью команды **dvc-tips-next-tip**, и в появившемся буфере, можно использовать клавиши **n** и **p** для перехода к следующему или предыдущему совету.

DVC в процессе своей работы открывает некоторое количество буферов, и для того, чтобы было легче перемещаться между ними, предоставляет некоторое количество команд. При просмотре вложенных друг в друга изменений можно воспользоваться командой **dvc-diff-master-buffer** (клавиша ^) для перехода в главный буфер с изменениями. DVC также для каждого буфера устанавливает специальную переменную, указывающую на буфер-партнёр, связанный с текущим буфером общей задачей. Это позволяет быстро переключаться между этими буферами с помощью команды **dvc-buffer-pop-to-partner-buffer** (клавиша **h**).

Аналогичным образом можно просматривать и буфера, используемые DVC в своих целях. Команда **dvc-open-internal-log-buffer** (**B L**) открывает буфер, содержащий все команды выполненные пакетом. А с помощью команды **dvc-show-process-buffer** (**B p**) можно посмотреть на результат выполнения внешних команд, вызываемых пакетом для выполнения тех, или иных действий.

3.7 Настройка

Настройка пакета осуществляется с помощью стандартных механизмов настройки Emacs. Соответствующая группа настройки имеет название **dvc**.

Некоторые режимы, реализуемые пакетом, предоставляют пользователю возможность установки параметров буферов путём запуска специальных хуков. Так например, команда **dvc-status** запускает хук **dvc-diff-mode-hook** (если он определён) при завершении инициализации буфера статуса.

4 Пакет PCL-CVS

Пакет PCL-CVS предоставляет пользователю эффективные средства с системой контроля версий CVS, которая широко распространена в мире свободного программного обеспечения. Кроме стандартных возможностей, которые реализованы пакетом VC, данный пакет предоставляет следующие возможности:

- просмотр состояния файлов для выбранного проекта;
- работа с изменениями для данного проекта;
- работа со всеми, или выбранными объектами вашего проекта.

Данный пакет является частью поставки GNU Emacs 21, и кроме того он доступен в виде пакета для XEmacs. Для других версий Emacs вы можете найти исходные тексты по адресу ftp://flint.cs.yale.edu/pub/monnier/pcl-cvs.

4.1 Основные понятия и принципы работы

Вся работа с пакетом осуществляется в специальном буфере, создаваемом командами PCL-CVS. Этот буфер называется `*cvs*`, и в нем отображается состояние файлов, находящихся в выбранном каталоге, который содержит ваш проект. Пакет работает только с проектами, уже извлеченными из репозитория, так что вам нужно это сделать перед выполнением команд PCL-CVS.

В процессе работы вы можете перемещаться по созданному буферу и выполнять разные команды с выбранными файлами, или файлами на которых находится курсор.

При работе с файлами в проекте, каждому из файлов присваивается определенный статус, и список доступных команд может зависеть от статуса файла.

Данные в буфере отображаются в несколько колонок:

- номер последней версии в репозитории, и кроме этого дополнительный статус файла (необязательно);
- звездочка, показывающая пометку для отмеченного файла;
- статус файла, расшифровка которого будет приведена ниже;
- версия файла, находящегося в выбранном каталоге (необязательно);
- имя файла.

Поле статуса может содержать следующие значения:

Added файл был добавлен, но изменения еще не были подтверждены в репозиторий;

Removed файл был удален, но изменения еще не были подтверждены в репозиторий;

Modified данный файл был изменен. Для этого статуса может быть показан дополнительный статус *merged*, который обозначает что были внесены изменения в репозиторий, и они были совмещены с вашими изменениями безо всяких конфликтов;

Conflict был обнаружен конфликт между вашими изменениями в проекте, и изменениями в репозитории. Обе версии изменений записываются в файл, но при этом сохраняется оригинальный файл, который именуется как `.#FILE.VERSION`. Конфликты также могут возникать по другим причинам, которые вы можете узнать посмотрев на дополнительный статус данного файла. Дополнительный статус может иметь следующие значения: *removed* — вы удалили файл, но кто-то внес в репозиторий новую версию; *added* — вы добавили файл, но кто-то тоже добавил файл в репозиторий; *modified* — вы изменили файл, но кто-то другой удалил этот файл из репозитория.

Unknown файл не зарегистрирован, и не входит в список игнорируемых файлов;

Up-to-date файл соответствует версии в репозитории. Поле дополнительного статуса может сообщить дополнительную информацию о файле: *added* — вы только что добавили файл в репозиторий; *updated* — файл был обновлен в соответствии с данными репозитория; *patched* — аналогично статусу *updated*, но обновление происходило более эффективным методом, это происходит в тех случаях, если вы изменяли файл; *commited* — вы подтвердили изменения в репозиторий;

Need-Update репозиторий содержит более свежую версию, или файл был добавлен в репозиторий и у вас его еще нет;

Need-Merge вы внесли изменения в файл, и кто-то также внес изменения в файл в репозитории, так что необходимо совместить эти изменения;

Missing файл был внезапно удален из вашего каталога, но для него не была выполнена команда **cvs remove**.

4.2 Основные команды и привязки клавиш

Команды, предоставляемые пакетом PCL-CVS имеют префикс **cvs-** и имена, аналогичные подкомандам команды CVS. Для выполнения части команд необходимо наличие буфера `*cvs*`, который создается следующими командами (команды выполняются с помощью сочетания **M**–**x** или через меню `Tools`):

cvs-update выполняет **cvs update** для указанного вами каталога;

cvs-examine выполняет команду **cvs -n update**, которая только проверяет что необходимо сделать, не внося никаких изменений;

cvs-status выполняет команду **cvs status** для указанного вами каталога;

cvs-checkout выполняет команду **cvs checkout** для указанного вами модуля;

cvs-quickdir создает буфер `*cvs*` считывая данные из файлов `CVS/Entries`. Эта команда аналогична `cvs-examine`, но не осуществляет доступа к репозиторию, что иногда очень полезно.

Эти команды могут выполняться также и из буфера `*cvs*` — вы можете использовать **M**–**u** для выполнения `cvs-update`, **M**–**e** для `cvs-examine` и **M**–**s** для `cvs-status`. Кроме этого, вы можете использовать соответствующие команды только для выбранных файлов — **O** (`cvs-mode-update`) для обновления файлов, **e** (`cvs-mode-examine`) для обновления информации о файлах и **s** (`cvs-mode-status`) для получения информации о выбранных файлах.

По умолчанию, эти команды выполняются рекурсивно, но вы можете изменить это поведение, используя флаг `-l` для команд CVS.

4.2.1 Команды перемещения в буфере и пометки файлов

Для перемещения по буферу `*cvs*` используются команды `cvs-mode-next-line` (клавиша **n**) — для перемещения на следующую строку и `cvs-mode-previous-line` (клавиша **p**), которая приводит к переходу на предыдущую строку.

Для работы с пометками используются множество разных команд. Для пометки одного файла используется команда `cvs-mode-mark` (клавиша **m**), а для снятия пометки — команда `cvs-mode-unmark` (клавиша **u**). Для пометки всех файлов используется клавиша **M** (команда `cvs-mode-mark-all-files`), а противоположной ей командой является `cvs-mode-unmark-all-files`, которая привязана к сочетанию клавиш **M**–<**DEL**>. Вы также можете пометить файлы, имена которых подпадают под заданное регулярное

выражение — команда `cvs-mode-mark-matching-files` (клавиша **%**), или которые имеют определенный статус — команда `cvs-mode-mark-on-state` (клавиша **S**).

4.2.2 Добавление, удаление, редактирование и обновление файлов

Добавление файлов производится очень простым способом — просто отметьте все нужные файлы (обычно они имеют статус *Unknown*), и нажмите на клавишу **a** (команда `cvs-mode-add`). Статус файлов будет изменен на *Added*, и затем вы должны подтвердить изменения в репозиторий (см. раздел Работа с изменениями). Вы также можете использовать эту команду и для файлов со статусом *Removed*, что приведет к их восстановлению.

Удаление файлов производится аналогичным образом — вы помечаете файлы, и выполняете команду `cvs-mode-remove-file` (клавиша **r**). При выполнении этой команды у вас будет запрошено подтверждение, и затем файлы будут удалены из каталога. В том случае, если файлы находятся под контролем CVS, то для них также будет выполнена команда **cvs remove**. Как и в предыдущем случае, вам также нужно будет подтвердить изменения в репозиторий.

Обновление файлов производится с помощью команды `cvs-mode-update`, которая привязана к клавише **O**. Это заставляет выполнить команду **cvs update** для файлов имеющих статус *Need-update*.

Иногда, вам требуется хранить в каталогах проекта, файлы, не зарегистрированные в репозитории. Такие файлы обычно отображаются со статусом *Unknown*, но вы можете сообщить CVS, что эти файлы необходимо игнорировать при выполнении команд, просто перечислив их в файле `.cvsignore`. Для помещения выбранного файла в этот файл, используется команда `cvs-mode-ignore`, которая привязана к клавише **i**. Используя ее, вы можете быстро поместить мешающие файлы в список игнорируемых объектов.

4.2.3 Работа с изменениями

Для подтверждения изменений в репозиторий вам нужно лишь выбрать файлы, изменения для которых вы хотите подтвердить, а зачем нажать **c** (`cvs-mode-commit`) или **C** (`cvs-mode-commit-setup`). Это приведет к возникновению нового буфера с именем `*cvs-commit*`, в котором вы можете ввести описание вносимых изменений. После ввода сообщения, вам необходимо лишь нажать **C–c C–c** и изменения будут внесены в репозиторий. Вы можете прервать этот процесс в любое время, просто не выполняя команду **C–c C–c**. Разница между командами **c** и **C** заключается в том, как они относятся к содержимому буфера `*cvs-commit*`. Первая команда сохраняет предыдущее содержимое буфера, в то время как вторая команда создает буфер заново.

Если вы изменили файл, но не хотите вносить данные изменения в репозиторий, то вы можете использовать команду `cvs-mode-undo-local-changes`, которая привязана к клавише **U**. Эта команда удаляет файл с вашими изменениями, и получает последнюю версию из репозитория.

Вы можете просмотреть изменения внесенные в файл с помощью нескольких команд. Наиболее часто используемой командой является `cvs-mode-diff`, которая вызывается с помощью клавиши = или сочетания **d** =. Эта команда показывает между измененным

файлом и его базовой версией. Кроме этого, также определен набор команд, которые выполняют следующие действия:

cvs-mode-diff-head (d h) показывает изменения между выбранными файлами и головной (HEAD) версией соответствующего файла данной ветви разработки;

cvs-mode-diff-repository (d r) показывает изменения между базовой и головной версиями выбранных файлов в текущей ветке разработки;

cvs-mode-diff-backup (d b) показывает изменения между резервной версией файла и файлом. Эта команда особенно полезна в случаях возникновения конфликтов при слиянии изменений между файлами;

cvs-mode-diff-vendor (d v) показывает изменения между выбранными файлами и головной версией в ветви производителя (vendor branch);

cvs-mode-diff-yesterday (d y) показывает изменения между выбранными файлами и вчерашней головной версией файлов.

Также, для работы с изменениями вы можете использовать утилиты Ediff и Emerge. Команда `cvs-mode-idiff` (привязана к сочетанию клавиш **d e**) запускает Ediff или Emerge (зависит от выбранных настроек), что позволяет вам интерактивно работать с изменениями. При использовании команды `cvs-mode-imerge` (сочетание клавиш **d E**) вы можете выполнить трехстороннее интерактивное слияние изменений, но если уже существует конфликт в выбранных файлах, то их содержимое не будет использоваться при работе, и после завершения работы, все изменения сделанные CVS, будут затерты.

4.2.4 Получение информации о файлах и прочие команды

Для получения информации о файлах может использоваться две команды. Команда `cvs-mode-log` (клавиша **l**) выполняет команду **cvs log** для выбранных файлов и результат отображается в буфере `*cvs-info*`. А команда `cvs-mode-status` (клавиша **s**) выполняет **cvs status** для выбранных файлов и результат отображается в буфере `*cvs-info*`.

Команда `cvs-mode-tag` (клавиша **t**) позволяет вам установить тег на выбранные файлы. По умолчанию, данная команда применяется только к каталогам, но это поведение зависит от настроек клиента.

Иногда вам может понадобиться, чтобы в буфере `*cvs*` не отображались некоторые записи. Вы можете сделать это двумя способами — использовать команду `cvs-mode-remove-handled` (клавиша **x**), которая удаляет из буфера уже обработанные записи, например, файлы со статусом *Up-to-date*, или использовать команду `cvs-mode-acknowledge` (она привязана к сочетанию **C–k**), которая просто удаляет нужные строки. Обработанные записи могут удаляться автоматически, в том случае, если переменная `cvs-auto-remove-handled` имеет значение не равное `nil`.

Чтобы обновить содержимое буфера `*cvs*`, вы можете использовать команду `cvs-mode-revert-buffer`, которая привязана к клавише **g**. А для выхода из буфера `*cvs*` используется команда `cvs-mode-quit`, которая привязана к клавише **q**.

Режим PCL-CVS также определяет несколько дополнительных режимов — для редактирования сообщений для журнала изменений, а также режим для просмотра журнала изменений.

4.3 Настройка

Выполнение команд PCL-CVS зависит от значения некоторых переменных, настройку которых легче всего осуществить с помощью команды `M-x customize-group pcl-cvs`. В эти настройки входит как настройка самих переменных, так и свойств начертаний, который используются для отображения информации в буферах данного режима.

5 Пакет xtla

Пакет xtla значительно расширяет возможности интеграции Emacs с системой контроля версий GNU Arch (он же tla и/или с Bazaar)[1]. Данный пакет, в отличии от модуля vc-arch, предоставляет пользователю полный спектр команд для работы с репозиториями GNU Arch. К основным возможностям пакета относятся:

- Поддержка как с GNU Arch, так и с Bazaar;
- Возможность работы с разными репозиториями, включая регистрацию новых и интерактивный просмотр содержимого зарегистрированных;
- Работа со списками объектов (inventories);
- Интерактивный просмотр изменений, возможность commit'ов, включая commit только выбранных файлов;
- Интерфейс похожий на PCL-CVS;
- Менеджер закладок, позволяющий легко перемещаться между часто используемыми архивами;
- Режимы для редактирования служебных файлов GNU Arch;
- Просмотр отсутствующих изменений;
- Поддержка как GNU Arch (tla), так и Bazaar (baz), учитывая специфические особенности каждой из систем;
- и многое другое.

Пакет xtla в своей работе использует программу **tla** или **baz**, так что она должна быть установлена в вашей системе. На момент написания данной статьи была выпущена версия 1.2 пакета xtla так что в статье будут описаны ее возможности.

[1] В настоящее время ведется разработка пакета dvc, который является развитием идей заложенных в xtla и в котором планируется реализовать поддержку разных распределенных систме контроля версий, в том числе и отличных от xtla. Более подробную информацию вы можете найти на сайте xtla.

5.1 Установка пакета

Для работы с пакетом необходимо наличие нескольких пакетов: ewoc.el, который необходим для работы всего пакета xtla (он поставляется вместе с xtla, и находится в каталоге `contrib`), tree-widget.el, который нужен для работы команды `tla-browse`, и пакет smerge-mode, который не является совсем необходимым, но может быть полезен при разрешении конфликтов.

Скачать пакет можно со страницы проекта[1]. Настройка пакета для установки выполняется с помощью стандартного скрипта **configure**. Если все прошло нормально, то после установки и перезапуска Emacs, часть функций пакета будет помечена как автозагружаемая и вы можете использовать их не используя явных команд загрузки. Однако, вы можете загрузить пакет явно, например так:

```
(require 'xtla-autoloads)
```

5.2 Использование пакета

Команды пакета имеют префиксы `tla-` и `xtla` и доступны через предопределенные ключи (сочетания клавиш), для которых используется префикс **C–x T**. Кроме явного вызова команд через **M–x**, и использования сочетаний клавиш, xtla добавляет несколько пунктов в меню `Tools`, а также создает отдельные меню с именем `Tla-...`, при работе в режимах, специфичных для работы с GNU Arch.

В целом, работу с xtla можно разделить на несколько больших задач:

- Работа с архивами;
- Работа с файлами в выбранном репозитории.

При работе с буферами xtla для них всех определены общие команды: **p** — перемещает к предыдущему объекту, **n** — переходит к следующему объекту, **q** — прекращает работу с данным буфером, **m** и **u** — ставят и снимают пометку с текущего объекта.

5.2.1 Работа с архивами

Для работы с архивами используется команда `tla-archives` (или сочетание клавиш **C–x T A**). При ее вызове, пользователю будет показан список зарегистрированных архивов, по которому он сможет перемещаться, и просматривать их содержимое.

При работе в данном режиме, в буфере действует специальный режим, для которого определено некоторое количество команд — регистрация архивов, работа с закладками и т.п. Наиболее часто используемые команды вынесены в отдельное меню, которое называется `Archives`. Эти же команды доступны через набор определенных клавиш, список наиболее часто используемых клавиш и команд приведен ниже (в скобках приведено название команды, вызываемой указанным сочетанием клавиш):

a a (команда `tla-make-archive`) — создать новый архив;

[1] https://gna.org/projects/xtla-el

e (команда `tla-archive-edit-archive-location`) — редактировать расположение выбранного архива;

s (команда `tla-archive-synchronize-archive`) — синхронизировать текущий архив;

a m (команда `tla-archive-mirror-archive`) — создать зеркало текущего архива;

a r (команда `tla-register-archive`) — зарегистрировать архив;

r (команда `tla-archive-unregister-archive`) — удалить регистрацию архива;

***** (команда `tla-archive-select-default`) — сделать текущий архив архивом, используемым по умолчанию;

RET (команда `tla-archive-list-categories`) — отобразить категории для текущего архива;

o (команда `tla-archive-browse-archive`) — интерактивно показать текущий архив;

g (команда `tla-archives`) — обновить список архивов;

b (команда `tla-bookmarks-add`) — добавить закладку для текущего архива;

Полный список клавиш, определенных для данного режима, вы можете получить с помощью сочетания клавиш **C–h m**.

Кроме использования `tla-archives` вы можете использовать команду `tla-browse` для просмотра списка зарегистрированных архивов и веток разработки, в более красивом виде. Вы можете видеть результат выполнения на рисунке Результат выполнения tla-browse.

5.2.2 Использование закладок

Закладки сильно облегчают работу разработчика в том случае, если он работает с большим количеством репозиториев. Для наиболее часто используемых версий, разработчик может определить закладки и быстро переходить к нужным архивам и веткам разработки. Для работы с закладками существует специальная команда — `tla-bookmarks`, которая вызывается сочетанием клавиш **C–x T b**. После ее выполнения будет создан буфер с именем `*tla-bookmarks*`, который вы можете использовать для работы с закладками. Для удобства работы также создается меню `Tla-Bookmarks`, которое содержит часто используемые команды.

Вы можете помечать несколько проектов как партнерские, и использовать затем команды для группы закладок, что особенно удобно в тех случаях, если ведется много веток разработки, и вы хотите быть в курсе разных ветвей разработки. Необходимо лишь пометить нужные проекты и выполнить команду `tla-bookmarks-marked-are-partners` (она привязана к сочетанию клавиш **M–p**).

Кроме этого, вы можете группировать проекты по разным принципам, и также выполнять команды для отмеченных групп. Для работы с группами используется серия команд `tla-...-group-...`.

При работе в буфере `*tla-bookmarks*` в нем определены следующие команды:

*** g** (команда `tla-bookmarks-select-by-group`) — выбрать закладки в группе;

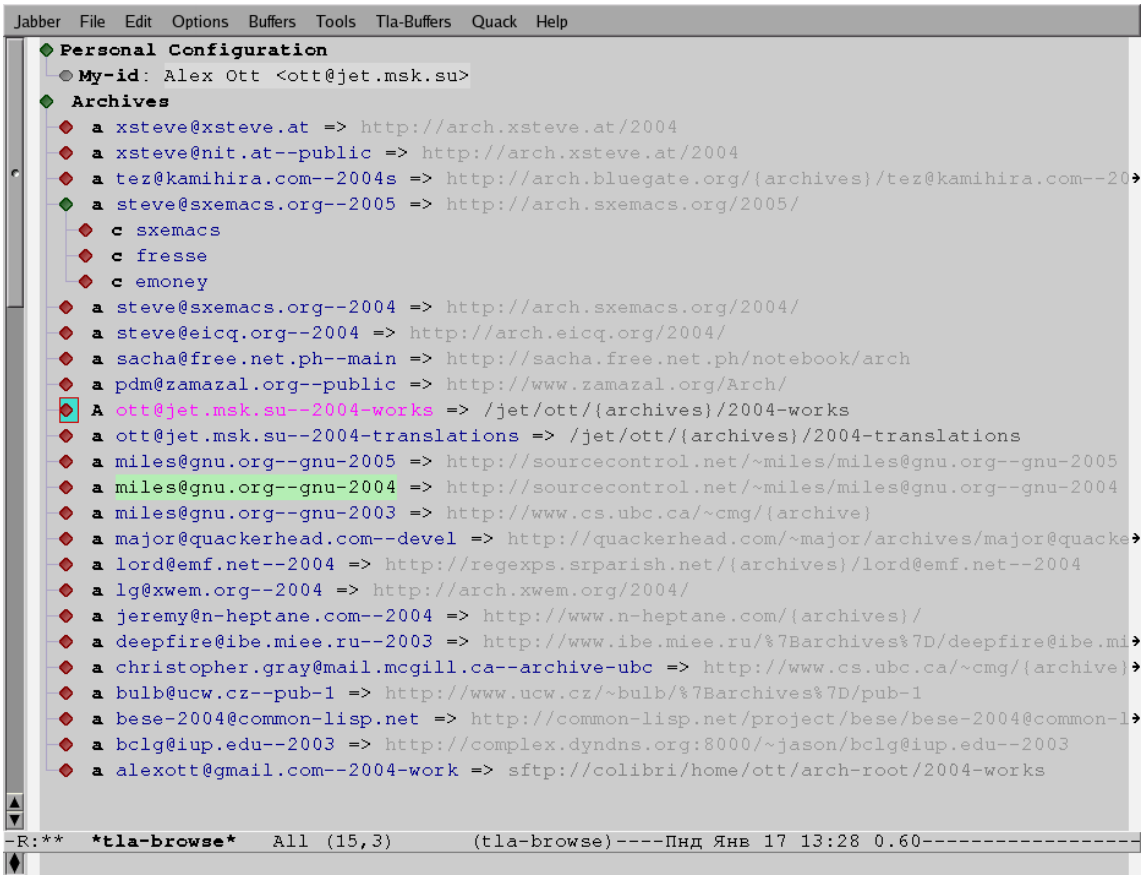

Рис. 1. Результат выполнения tla-browse

*** !** (команда `tla-bookmarks-unmark-all`) — снять пометку со всех закладок;

RET (команда `tla-bookmarks-goto`) — перейти к архиву связанному с данной закладкой;

> (команда `tla-bookmarks-get`) — получить архив для текущей закладки;

= (команда `tla-bookmarks-changes`) — выполнить **tla changes** для текущего дерева;

o (команда `tla-bookmarks-open-tree`) — открыть локальное дерево в буфере dired;

M (команда `tla-bookmarks-missing`) — просмотреть изменения, отсутствующие в вашем дереве;

P (команда `tla-bookmarks-move-up`) — переместить закладку на позицию вверх;

N (команда `tla-bookmarks-move-down`) — переместить закладку на позицию вниз;

i (команда `tla-bookmarks-inventory`) — запустить **tla inventory** для локального дерева;

t (команда `tla-bookmarks-toggle-details`) — переключить степень детализации вывода;

e (команда `tla-bookmarks-edit`) — редактировать текущую закладку;

B L (команда `tla-open-internal-log-buffer`) — открыть внутренний буфер **tla**. В этом буфере хранится список выполненных команд;

B p (команда `tla-show-process-buffer`) — отобразить результаты выполнения последней команды;

Mp (команда `tla-bookmarks-marked-are-partners`) — объявить помеченные закладки как партнеры;

a p (команда `tla-bookmarks-add-partner-interactive`) — добавить партнера к текущей или помеченной закладке;

r p (команда `tla-bookmarks-delete-partner-interactive`) — удалить партнерство с текущей или помеченной закладки;

f w (команда `tla-bookmarks-write-partners-to-file`) — сохранить партнерство для помеченных закладок в файл партнеров;

f r (команда `tla-bookmarks-add-partners-from-file`) — добавить список партнеров из файла партнеров;

a n (команда `tla-bookmarks-add-nickname-interactive`) — добавить имя для текущей закладки;

r n (команда `tla-bookmarks-delete-nickname-interactive`) — удалить имя для текущей закладки;

a t (команда `tla-bookmarks-add-tree-interactive`) — добавить локальное дерево к закладке;

r t (команда `tla-bookmarks-delete-tree-interactive`) — удалить локальное дерево из закладки;

a b (команда `tla-bookmarks-add`) — добавить закладку для архива;

r b (команда `tla-bookmarks-delete`) — удалить текущую закладку;

a g (команда `tla-bookmarks-add-group-interactive`) — добавить группу закладок;

r g (команда `tla-bookmarks-delete-group-interactive`) — удалить группу закладок.

Кроме работы с закладками из буфера `*tla-bookmarks*`, вы можете выполнять действия относящиеся к закладкам из других буферов, например, из буфера для работы с архивами (команда **b**).

5.2.3 Работа с файлами в выбранном репозитории

При работе с выбранным репозиторием пользователь может выполнять самые разные действия — добавлять и удалять файлы, вставлять теги GNU Arch, и многое другое. Для этого выделены отдельные команды. Однако, более удобным способом работы будет использование буфера объектов (inventories).

Команда `tla-inventory` (или сочетание клавиш **C–x T i**) создает буфер `*tla-inventory*`, который содержит список объектов для выбранного каталога. Вы можете видеть результаты выполнения этой команды на рисунке Окно буфера *tla-inventory*. Используя данный буфер пользователь может интерактивно работать с объектами репозитория — добавлять и удалять объекты, указывать их тип, просматривать изменения, и выполнять множество других операций. Для выполнения основных команд может использоваться меню `Inventory`, создаваемое данной командой.

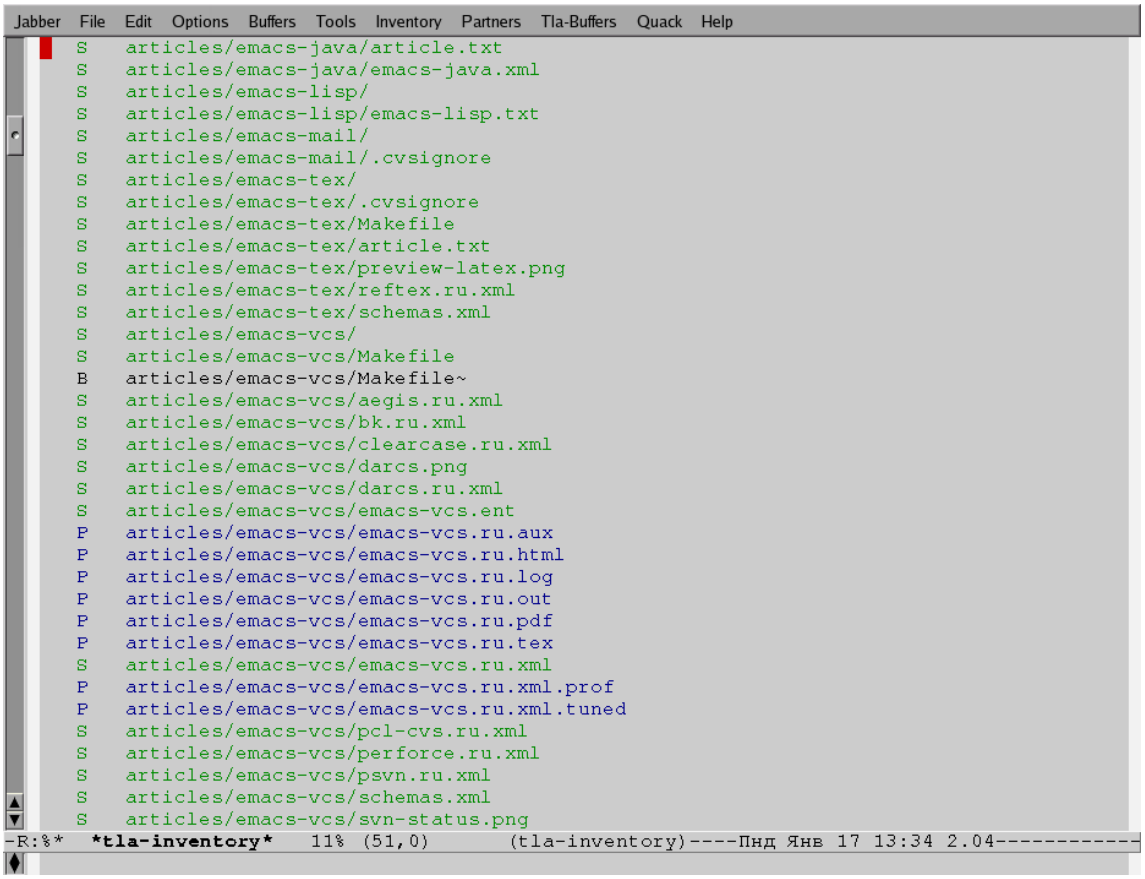

Рис. 2. Окно буфера *tla-inventory*

При работе с буфером `*tla-inventory*` определены следующие команды:

g (команда `tla-generic-refresh`) — обновить содержимое буфера;

U (команда `tla-inventory-revert`) — отменить изменения в текущем файле;

< (команда `tla-inventory-mirror`) — сделать зеркало для текущего архива;

e (команда `tla-inventory-file-ediff`) — запустить Ediff для текущего файла;

= (команда `tla-inventory-changes`) — просмотреть изменения для вашего дерева;

v (команда `tla-inventory-view-file`) — показать текущий файл;

o (команда `tla-inventory-find-file-other-window`) — открыть текущий файл в другом окне;

RET (команда `tla-inventory-find-file`) — открыть текущий файл;

f (команда `tla-inventory-find-file`) — открыть текущий файл;

L (команда `tla-logs`) — запустить **tla logs**;

l (команда `tla-changelog`) — запустить **tla changelog**;

c (команда `tla-inventory-edit-log`) — редактировать лог для текущего дерева;

R (команда `tla-inventory-move`) — переименовать текущий файл;

X (команда `tla-inventory-delete-files`) — удалить локальные файлы;

^ (команда `tla-inventory-parent-directory`) — перейти в родительский каталог в буфере inventory;

r (команда `tla-inventory-remove-files`) — удалить идентификаторы для текущего файла;

a (команда `tla-inventory-add-files`) — добавить текущий файл;

t u (команда `tla-inventory-toggle-unrecognized`) — переключить режим показа не распознанных файлов;

t t (команда `tla-inventory-toggle-tree`) — переключить режим показа корневого дерева;

t b (команда `tla-inventory-toggle-backup`) — переключить режим показа резервных файлов;

t j (команда `tla-inventory-toggle-junk`) — переключить режим показа "мусорных"(junk) файлов;

t p (команда `tla-inventory-toggle-precious`) — переключить режим показа precious файлов;

t s (команда `tla-inventory-toggle-source`) — переключить режим показа файлов исходных текстов;

t ~ (команда `tla-inventory-toggle-all-toggle-variables`) — переключить все переключаемые переменные;

t - (команда `tla-inventory-reset-all-toggle-variables`) — сбросить значения всех переключаемых переменных;

t + (команда `tla-inventory-set-all-toggle-variables`) — установить все переключаемые переменные;

 # u (команда `tla-inventory-add-to-unrecognized`) — пометить текущий или выбранные файлы как не распознанные;

 # p (команда `tla-inventory-add-to-precious`) — пометить текущий или выбранные файлы как precious;

 # ~ (команда `tla-inventory-add-to-backup`) — пометить текущий или выбранные файлы как резервные;

 # b (команда `tla-inventory-add-to-backup`) — пометить текущий или выбранные файлы как резервные;

 # j (команда `tla-inventory-add-to-junk`) — пометить текущий или выбранные файлы как "мусорные";

 # e (команда `tla-inventory-add-to-exclude`) — пометить текущий или выбранные файлы как исключения;

 # x (команда `tla-inventory-add-to-exclude`) — пометить текущий или выбранные файлы как исключения;

 # V (команда `tla-inventory-set-tree-version`) — задать версию для локального дерева;

 # M (команда `tla-inventory-set-id-tagging-method`) — задать метод идентификации;

 # . (команда `tla-edit-.arch-inventory-file`) — редактировать файл `.arch-inventory`;

 # = (команда `tla-edit-=tagging-method-file`) — редактировать файл `=tagging-method`;

*** !** (команда `tla-inventory-unmark-all`) — снять пометку со всех файлов;

*** u** (команда `tla-inventory-unmark-file`) — снять пометку с файла;

*** m** (команда `tla-inventory-mark-file`) — пометить текущий файл;

w a (команда `tla-save-archive-to-kill-ring`) — сохранить имя архива текущего буфера в список удалений;

W l (команда `tla-tree-lint`) — проверить текущее дерево;

W a (команда `tla-inventory-apply-changeset-from-tgz`) — применить набор изменений из файла `.tgz`;

W A (команда `tla-inventory-apply-changeset`) — применить набор изменений;

W v (команда `tla-show-changeset-from-tgz`) — показать набор изменений из файла `.tgz`;

W V (команда `tla-show-changeset`) — показать набор изменений;

W s (команда `tla-changes-save-as-tgz`) — сохранить изменения в виде файла `.tgz`;

W S (команда `tla-changes-save`) — сохранить набор изменений в каталоге;

W R (команда `tla-inventory-redo`) — применить изменения, отмененные последней операцией отмены;

W U (команда `tla-inventory-undo`) — отменить изменения для всего локального дерева;

B b (команда `tla-bookmarks`) — отображает закладки в отдельном буфере;

B L (команда `tla-open-internal-log-buffer`) — открыть внутренний буфер **tla**. В этом буфере хранится список выполненных команд;

B p (команда `tla-show-process-buffer`) — отобразить результаты выполнения последней команды;

M < (команда `tla-inventory-apply-changeset`) — применить набор изменений;

M u (команда `tla-inventory-update`) — обновить текущее дерево из архива методом update;

M r (команда `tla-inventory-replay`) — обновить текущее дерево из архива методом replay;

M s (команда `tla-inventory-star-merge`) — слить вместе изменений из другого архива;

d > (команда `tla-inventory-delta`) — вычислить список изменений для текущего дерева;

d e (команда `tla-inventory-file-ediff`) — запустить Ediff для текущего файла;

d l (команда `tla-changes-last-revision`) — показать изменения относительно последнего релиза;

d = (команда `tla-inventory-changes`) — показать список изменений для текущего дерева;

d m (команда `tla-inventory-missing`) — показать список отсутствующих изменений.

Пользователь может управлять отображением объектов в буфере, например, скрывая файлы резервных копий (backup), или файлы, заведомо содержащие ненужные данные (junk). Для этого используются команды `tla-inventory-toggle-...`

Работа с изменениями ведется аналогично работе с изменениями в пакете PCL-CVS. Вы можете просмотреть список измененных файлов с помощью команды `tla-changes` (или используя сочетание клавиш **C–x T c**), которая покажет вам список измененных файлов (он отображается в буфере `*tla-changes*`), и сами изменения. С помощью команды `tla-changes-edit-log` (клавиша c в буфере `*tla-changes*` вы можете ввести сообщение, описывающее изменения, и затем установить изменения в репозиторий, используя сочетание клавиш **C–c C–c**. Можно также вносить изменения только для отмеченных файлов, что помогает делать изменения атомарными.

5.3 Настройка

Настройка данного пакета не проста, а очень проста — выполните команду `M-x customize-group xtla` и задайте все необходимые опции. Диапазон настроек очень велик — начиная от расположения команд, необходимых для работы, и до задания параметров отображения данных. Вы также можете задать префиксное сочетание клавиш, отличное от используемого по умолчанию.

Пакет позволяет пользователю задать хуки практически на каждую из реализованных им операций. Это позволит пользователю более тонко контролировать работу с данными в репозитории.

5.4 Дополнительная информация

Немного дополнительной информации вы можете найти на странице GNU Arch WiKi, по адресу http://wiki.gnuarch.org/xtla. И конечно весь сайт GNU Arch WiKi[1] будет вам великолепным подспорьем в работе с GNU Arch.

Для получения полного списка команд, используйте сочетание клавиш **C–x T C–h**. Вы также можете получить справку по конкретной команде **tla**, используя сочетание клавиш **C–x T h** `command`, что приводит к запуску команды **tla command -H**.

6 Пакет PSVN

Пакет PSVN является для Subversion тем же, что и PCL-CVS для CVS. Пакет дает пользователю доступ ко всем основным возможностям, предоставляемым системой Subversion.

6.1 Установка пакета

Установка пакета проста — загрузите с сайта[2] последнюю версию пакета, поместите его в доступный для Emacs каталог, и поместите в файл инициализации следующую команду:

```
(require 'psvn)
```

6.2 Работа с пакетом

Работа с пакетом по принципам, совершенно аналогична работе с пакетом PCL-CVS — вся работа производится в буфере, создаваемом командами пакета. В настоящее время этот буфер создается лишь одной командой — `svn-status`, реализация других команд, производится в настоящее время. Буфер, создаваемый в процессе работы данной команды, называется `*svn-status*`. Пример буфера, создаваемого командой `svn-status`, вы можете увидеть на рисунке Окно буфера *svn-status*. Команды, выполняемые пользователем применяются либо к текущему файлу, либо к помеченным файлам.

[1] http://wiki.gnuarch.org
[2] http://www.xsteve.at/prg/emacs/psvn.el

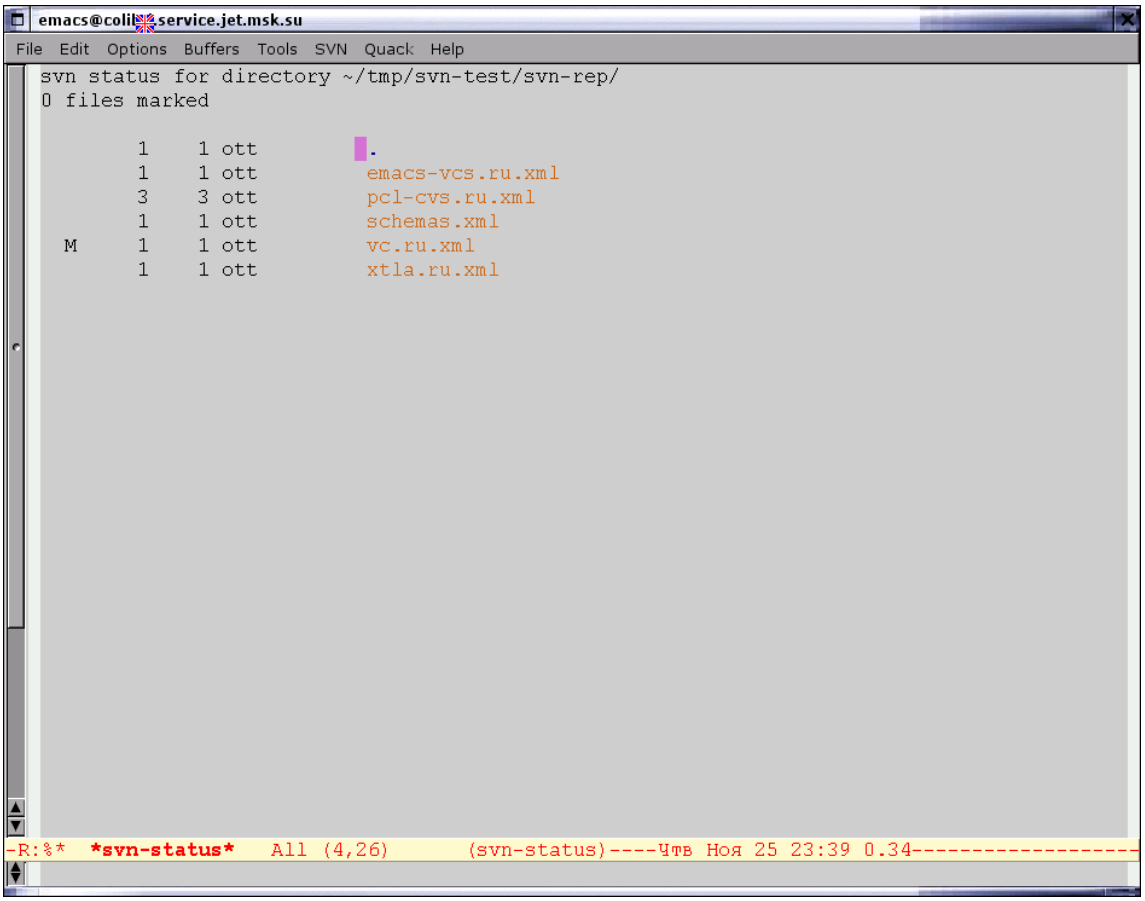

Рис. 3. Окно буфера *svn-status*

Как вы можете видеть на рисунке, информация выдается в виде таблицы. В первой колонке отображается статус объекта, во второй колонке — рабочей версия данного объекта, в третьей — номер версии в которой вносились последние изменения, а в четвертой — имя пользователя, который вносил эти изменения, и в последней — имя объекта. Расшифровку статусов вы можете найти в описании команды **svn status -v**.

Пользователь может выполнять команды, используя предопределенные привязки клавиш, либо может использовать меню `SVN`, появляющееся при работе с буфером `*svn-status*`.

6.2.1 Навигация и работа с пометками

Команды навигации по буферу `*svn-status*` аналогичны командам других пакетов для работы с системами контроля версий — вы можете использовать либо клавишу управления курсором, либо команду `svn-status-previous-line` (привязана к клавише **C**–**p**) для перемещения на предыдущую строку, и команду `svn-status-next-line` (привязана к клавише **C**–**n**) для перехода на следующую строку. Эти привязки несколько отличаются от других режимов, которые для аналогичных команд используют клавиши **n** и **p**.

Для установки пометок на файлы PSVN использует те же самые клавиши, что и остальные режимы — для пометки объекта используется клавиша **m** (команда `svn-s-`

tatus-set-user-mark), а для снятия пометки — клавиша **u** (команда svn-status-unset-user-mark). Кроме этого для снятия всех установленных пометок используется команда svn-status-unset-all-usermarks (сочетания клавиш **M–DEL** или *** !**), а для снятия пометки с файла, находящегося перед курсором, может использоваться команда svn-status-unset-user-mark-backwards, привязанная к клавише **DEL**.

Также определены команды для пометки файлов с разным статусом. Так, для пометки измененных файлов может использоваться команда svn-status-mark-modified (сочетание клавиш *** M**), для пометки добавленных файлов команда svn-status-mark-added (сочетание клавиш *** A**), а для пометки файлов не находящихся под контролем Subversion, команда svn-status-mark-unknown (сочетание клавиш *** ?**).

6.2.2 Добавление, удаление и обновление файлов

Добавление файлов в репозиторий можно производить с помощью нескольких команд. Базовой командой является svn-status-add-file (она привязана к клавише **a**), которая добавляет текущий файл в репозиторий (или выбранные файлы). В качестве дополнения к ней, пакет PSVN определяет команду svn-status-add-file-recursively (клавиша **A**), которая рекурсивно добавляет файлы в репозиторий и команду svn-status-make-directory (клавиша +), которая создает новый каталог в репозитории. Не забудьте, что удаление файлов производится только в вашей рабочей копии репозитория, так что изменения будут внесены только после явного подтверждения.

Удаление файлов производится командой svn-status-rm (клавиша **D** или сочетание **C–d**), что запускает команду **svn rm**. Поскольку Subversion поддерживает явное переименование файлов с сохранением истории файла, то и пакет PSVN имеет в своем арсенале соответствующую команду svn-status-mv, которая привязана к клавише **R**. Как и в предыдущем случае, вам после выполнения данных команд необходимо подтвердить внесенные изменения.

Вы можете выполнить обновление вашего рабочего каталога из репозитория, что часто бывает удобно при одновременной работе нескольких разработчиков над одним проектов. Обновление производится с помощью команды svn-status-update-cmd (клавиша **U**), которая запускает команду **svn update**. Кроме этого, вы можете вытянуть нужную версию файла с помощью команды svn-status-get-specific-revision (клавиша ~). При работе этой команды создается файл F.~REVISION~, который и содержит нужную версию файла. А также вы можете просмотреть состояние и родительского каталога, с помощью команды svn-status-examine-parent, которая привязана к клавише ^.

Вы можете просматривать и редактировать файлы непосредственно из буфера *svn-status*. Это возможно с помощью команд svn-status-find-files (клавиша **f**), которая открывает файл непосредственно в том же окне, которое и содержит буфер *svn-status*. Для открытия файла в другом окне может использоваться команда svn-status-find-file-other-window (клавиша **o**), вы можете просматривать, а не редактировать файл с помощью команды svn-status-view-file-other-window (клавиша **v**), что бывает очень полезно, если вы хотите лишь посмотреть на файл, не редактируя его. Клавиша **RET** (команда svn-status-find-file-or-examine-directory) играет двоякую роль — она либо открывает файл, либо отображает состояние файлов в соответствующем каталоге.

Хочу заметить, что в настоящее время PSVN не поддерживает команды импорта данных в архив, и получения данных из репозитория. Но работа над поддержкой этих

функций ведется, и может быть они будут реализованы в будущих версиях.

6.2.3 Работа с изменениями

Как уже упоминалось выше, вы должны подтвердить изменения в репозиторий. Подтверждение выполняется с помощью команды `svn-status-commit-file` (клавиша **c**). После подтверждения, ваши изменения будут видны всем людям, работающим с репозиторием.

Кроме подтверждения изменений, иногда необходимо отменить внесенные изменения. Для этого используется команда `svn-status-revert`, которая привязана к клавише **r**. Subversion поддерживает возможность явного удаления статуса наличия конфликтов с файлов, которые имеют такой статус. Для удаления статуса конфликта PSVN предоставляет команду `svn-status-resolved` (клавиша **V**), которая запускает команду **svn resolved**.

Также как и другие пакеты для работы с системами контроля версий, PSVN предоставляет возможность получения списка изменений для выбранных файлов по сравнению с базовой версией. Для этого определено несколько команд. Команда `svn-status-show-svn-diff` (клавиша =), выполняет сравнение с головной версией в репозитории (если она новее чем базовая) или с базовой версией текущего файла. Команда `svn-status-show-svn-diff-for-marked-files` (сочетание **C**–=) может выполнить сравнение сразу нескольких выбранных файлов с их базовыми версиями. А команда `svn-status-ediff-with-revision` (клавиша **E**) использует для сравнения пакет Ediff. Все эти команды могут выполнять сравнение с произвольной версией, если им будет задан префиксный аргумент перед их выполнением.

6.2.4 Работа с метаданными (properties)

Система контроля версий Subversion позволяет пользователю привязывать различные метаданные к файлам и версиям. Кроме этого, существуют метаданные, не привязанные к конкретной версии. Изменения в метаданных распространяются вместе с изменениями, внесенными в обычные файлы. Подробнее о метаданных и работе с ними вы можете прочитать в книге "Version Control with Subversion которая доступна на сайте данной системы.

Пакет PSVN предоставляет набор команд для работы с метаданными. Эти команды доступны как обычным способом, так и через привязки к клавишам. В качестве префиксной клавиши используется клавиша **P**.

Доступные команды можно разделить на две группы — первая группа используется для работы с произвольными метаданными, а вторая группа — для работы с конкретными записями.

К первой группе относятся команды: `svn-status-property-parse` (сочетание **P p**) используется для разбора записи, `svn-status-property-set` (сочетание **P s**) задает значение для конкретной записи, `svn-status-property-delete` (сочетание **P d**) удаляет выбранную запись, `svn-status-property-list` (сочетание **P l**) отображает список определенных для данного объекта записей, `svn-status-property-edit-one-entry` (сочетание **P e**) позволяет изменить запись.

Во вторую группу входят следующие команды: `svn-status-property-edit-svn-ignore` (сочетание **P TAB**) позволяет вам отредактировать список файлов, которые не будут учитываться при Subversion, `svn-status-property-ignore-file` (сочетание **P i**) вносит текущий файл в список игнорируемых объектов, `svn-status-property-ignore-file-extension` (сочетание **P I**) позволяет пополнить список игнорируемых объектов шаблоном для выборки всех файлов с расширением как у текущего, `svn-status-property-set-eol-style` (сочетание **P y**) позволяет задать какой стиль обозначения перевода строк используется для данного объекта, `svn-status-property-set-keyword-list` (сочетание **P y**) позволяет отредактировать список ключевых слов связанных с данным объектом.

6.2.5 Получение информации и прочие команды

Для получения информации о текущем файле может использоваться команда `svn-status-info`, которая привязана к клавише **i**. При ее использовании, в буфере `*svn-process*` будет отображен результат выполнения команды **svn info**. Информация может показаться слишком подробной, и для ее обработки может быть использована команда `svn-status-parse-info` (клавиша **I**), которая разбирает результат предыдущей команды, и приводит его к более компактному виду. Для получения информации об авторе и версии для указанного файла, может использоваться команда `svn-status-blame` (клавиша **b**), а для просмотра логов для выбранных файлов, используется команда `svn-status-show-svn-log` (клавиша **l**).

Обновить содержимое буфера `*svn-status*` вы можете с помощью команды `svn-status-update` (клавиша **g**), а посмотреть на вывод команды **svn** вы можете использовав клавишу **s** (команда `svn-status-show-process-buffer`). Вы также можете управлять тем, объекты с каким статусом будут отображаться в буфере `*svn-status*`. Для скрытия не изменявшихся файлов используется команда `svn-status-toggle-hide-unmodified` (клавиша **_**), а для скрытия незарегистрированных файлов — команда `svn-status-toggle-hide-unknown` (клавиша **?**). Обе эти команды являются переключателями, и могут использоваться как для отключения показа файлов, так и для включения показа.

Для выхода из буфера `*svn-status*` и его удаления используется команда `svn-status-bury-buffer`, которая привязана к клавише **q**, также как и в других пакетах.

6.3 Настройка

Стандартные средства Emacs позволяют пользователю настроить только параметры начертаний, которые используются для отображения данных в буфере. Остальные настройки нужно задавать с помощью команд Emacs Lisp. Но большая часть этих настроек имеет нормальные значения по умолчанию, так что вам скорее всего не понадобиться изменять эти значения.

PSVN также предоставляет несколько хуков, которые могут использоваться для задания параметров создаваемых буферов, или выполнения дополнительных команд. Хук `svn-log-edit-mode-hook` выполняется когда пакет входит в режим `svn-log-edit`, а `svn-log-view-mode-hook` выполняется при вхождении в режим `svn-log-view`.

7 Работа с Darcs

Работа с системой контроля версий Darcs обеспечивается несколькими пакетами, которые несколько отличаются друг от друга по функциональности. Это пакет darcs, доступный с ??? и пакет darcsum, который вы можете загрузить со страницы автора[1]. Кроме этого, существует модуль поддержки Darcs для пакета VC, о котором вы можете прочитать в разделе Пакет VC. Также начата разработка модуля для пакета DVC, который описан в разделе Пакет DVC.

Пакет darcsum имеет больше возможностей чем пакет darcs, и поэтому в дальнейшем наш рассказ будет вестись о нем. Пакет предоставляет пользователю для работы интерфейс, во многом сходный с уже привычным интерфейсом пакета PCL-CVS. Имеются только некоторые отличия, обусловленные самой архитектурой системы Darcs.

7.1 Установка пакета

Установка пакета не отличается от установки других пакетов для Emacs — вам лишь необходимо загрузить с сайта[2] последнюю версию пакета, поместить ее туда, где ее найдет Emacs, и прописать загрузку пакета в вашем файле инициализации, например, с помощью такой вот команды: (require 'darcsum)

7.2 Использование пакета darcsum

Вся работа с пакетом происходит в буфере `*darcs*`, который создается командой `M-x darcsum-whatsnew`. Пример окна создаваемого этой командой вы можете видеть на рисунке Буфер Darcsum. Многие команды, доступные в данном буфере, совпадают по действию с командами пакета PCL-CVS. Команды могут выполняться над текущими изменениями, или над отмеченными. Пометка и ее снятие производится с помощью клавиши **m**, которая выполняет команду `darcsum-toggle-mark`. Перемещение между файлами и изменениями в буфере производится с помощью команд `darcsum-next-line` (клавиша **n**), и `darcsum-previous-line` (клавиша **p**), или с помощью клавиш управления курсором.

Работа с изменениями производится по следующему сценарию: после получения списка измененных файлов, вам нужно отобразить конкретные изменения внесенные в файлы, выбрать нужные изменения, и затем выполнять над ними необходимые команды. Отображение изменений внесенных в файл производится с помощью команды `darcsum-toggle`, которая привязана к клавише **RET**.

Выбранные изменения могут быть сохранены с помощью команды `darcsum-record` (клавиши **c** или **R**), после выполнения которой появится буфер в котором пользователь может ввести имя для данного изменения (первая строка буфера) и длинное описание данного изменения (следующие за первой строки). После ввода описания, с помощью сочетания клавиш **C–c C–c**, изменения будут сохранены.

Изменения могут быть удалены из буфера `*darcs*` с помощью команды `darcsum-remove` (клавиша **r**), или перемещены в другой буфер darcsum, с помощью команды `darc-`

[1] http://www.newartisans.com/johnw/Emacs/darcsum.el
[2] http://www.newartisans.com/johnw/emacs.html

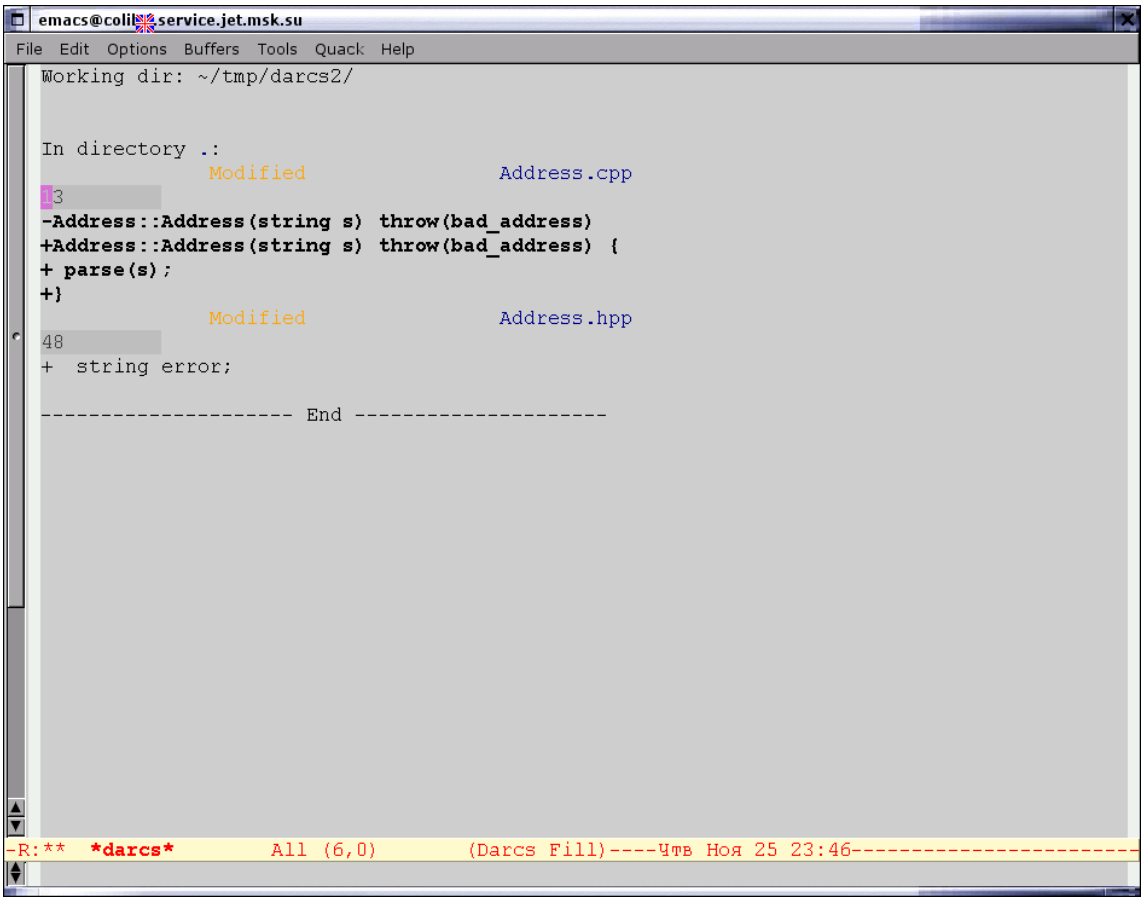

Рис. 4. Буфер Darcsum

`sum-move` (клавиша **M**). Откат к предыдущей версии данных производится с помощью команды `darcsum-revert`, которая привязана к клавише **U**.

Для более удобного просмотра изменений, пакет darcsum определяет набор команд, которые позволяют просмотреть результат выполнения команды **diff** (или ее аналогов) — команда `darcsum-diff` (клавиша =), просмотреть изменения с помощью пакета Ediff — команда `darcsum-ediff` (клавиша **e**), или даже выполнить трехстороннее слияние изменений с помощью команды `darcsum-ediff-merge` (клавиша **E**).

Чтобы обновить содержимое буфера, используется команда `darcsum-redo`, которая заново выполняет команду 'darcs whatsnew'. Для окончания работы с изменениями, используется команда `darcsum-quit`, которая привязана к клавише **q**.

7.3 Настройка

Настройка пакета производится с помощью стандартных возможностей Emacs. Просто выполните команду `M-x customize-group darcsum`, и перед вами появится окно настройки. Вы можете задать имя для исполняемого файла **darcs**, а также изменить начертания шрифтов, которыми отображаются данные в буфере `*darcs*`.

Кроме настройки переменных и параметров начертаний, пакет предоставляет хук `darcsum-mode-hook`, в который вы можете поместить нужные вам команды, которые будут

выполнены после выполнения хуков для режима `text`, от которого унаследован `darcsum-mode`.

8 Поддержка BitKeeper

BitKeeper — это коммерческая реализация системы контроля версий. Однако ее можно использовать и в свободных проектах, без уплаты каких-либо отчислений. Наиболее масштабным проектом, использующим данную систему, является разработка ядра Linux.

8.1 Установка пакета

Для установки пакета bk вам нужно получить его копию с http://www.emacswiki.org/elisp/bk.el, или использовать репозиторий BitKeeper по адресу `bk://bk-emacs.bkbits.net/emacs`. Полученный файл необходимо поместить в доступное для Emacs место и выполнить команду загрузки. Для работы пакета необходимо наличие библиотеки x-migrant, но если ее у вас нет, то вы можете закомментировать ее загрузку в исходном коде пакета.

8.2 Использование пакета

По большей части, работа с пакетом bk похожа на работу с пакетом VC. Так, что если вы знакомы с ним, то привязки клавиш и принципы работы, будут вам знакомы.

Команды пакета могут выполняться через стандартный механизм выполнения команд доступный через сочетание клавиш **M–x**, через глобальные и локальные привязки клавиш, а также через меню `BitKeeper`, доступное при работе в режиме bk-mode.

Для привязки клавиш данный пакет использует два префикса. **C–x v** — это префикс используемый пакетом VC, и так что команды пакета bk перекрывают часть команд VC применимых к конкретным файлам. **C–c b** является глобальным префиксом, и с его помощью применяются команды, которые применимы не только к конкретным файлам.

Следует помнить что только часть команд доступна через глобальные привязки клавиш. Обычно это команды, привязки которых начинаются с префиксов **C–x v** или **C–c b**. Остальные команды доступны только при работе с файлами, которые зарегистрированы в системе BitKeeper.

При работе с файлами, которые находятся под контролем BitKeeper, в строке состояния может отображаться либо значение *BKL*, для случая, когда репозиторий лицензирован в терминах BKL, или *BKCL*, когда используется лицензия BKCL. Для получения более подробной информации об этом, просто наберите `M-x bk-help licensing`.

Регистрация файлов в репозитории производится с помощью команды `bk-register` привязанной к сочетанию клавиш **C–x v i**. Работа с зарегистрированными файлами по принципу похожа на работу с файлами в пакете VC. Получение файла из репозитория для редактирования, и освобождение его и помещение в репозиторий производится с помощью команды `vc-toggle-read-only` (сочетание **C–x C–q**). Вы также можете отменить сделанные вами изменения и вернуться к версии в репозитории с помощью `vc-revert-buffer` (сочетание клавиш **C–x v u**).

Просмотр информации о конкретной ревизии файла производится с помощью команды `bk-print-revision-summary` (сочетание **C–x v l**). Для просмотра истории ревизий в графическом виде, пользователь может использовать команду `bk-revtool` (она привязана к **C–c b r**), которая запускает графический браузер ревизий из поставки BitKeeper. Пользователь может также просмотреть конкретную ревизию файла с помощью команды `bk-print-rev` (сочетание клавиш **C–x v ~**), или аннотированную версию файла, используя команду `bk-annotate`, привязанную к сочетанию клавиш **C–x v g**. Просмотр осуществляется с использованием `view-mode`.

Работа с изменениями в BitKeeper основана на концепции набора изменений (changeset). Для работы с наборами изменений, пакет bk определяет несколько команд. Для внесения изменений, и создания набора изменений, используется команда `bk-citool`, которая привязана к сочетанию клавиш **C–c b c**. Данная команда запускает графическую утилиту из состава BitKeeper для подтверждения изменений в репозиторий.

Для обмена наборами изменений между разными репозиториями, пакет bk определяет две команды. Команда `bk-push` (она привязана к **C–c b .**) используется для передачи набора изменений в родительский репозиторий. А команда `bk-pull` (сочетание **C–c b ,**) используется для втягивания наборов изменений, имеющихся в родительском репозитории. Обе эти команды имеют глобальные привязки клавиш, так что ими можно пользоваться из любого места.

Для получения изменений между вашим файлом и файлом в репозитории, может использоваться команда `bk-diffs`, которая привязана к сочетанию клавиш **C–x v =**. А для рекурсивного выполнения сравнения файлов в каталоге, с их версиями в репозитории, используется команда `bk-recursive-diffs` (сочетание **C–c b =**). Команда `bk-interesting-diffs` (сочетание **C–c b i**) рекурсивно выполняет сравнение для некоторых файлов. К этим файлам относятся, те которые заблокированы и изменены, или те, которые были помещены в репозиторий, но еще не были подтверждены в набор изменений.

Для показа списка файлов, которые были заблокированы и изменены, или которые были помещены в репозиторий, но еще не были подтверждены в набор изменений, используется команда `bk-sfiles` (сочетание клавиш **C–c b s**).

Для получения короткого описания пакета и основных команд работы с ним, пользователь может использовать команду `bk-help-overview` (сочетание клавиш **C–c b h**). Пользователь также может получать справку по работе с BitKeeper с помощью следующих команд: команда `bk-help` (сочетание **C–c b m**) отображает справочную страницу BitKeeper, команда `bk-apropos` (сочетание **C–c b a**) осуществляет поиск по документации BitKeeper, а команда `bk-helptool` запускает графическое средство для работы со справочной системой BitKeeper.

8.3 Настраиваем пакет

Для настройки пакета bk можно использовать стандартный механизм настройки Emacs. С помощью команды `M-x customize-group bk`, пользователь может задать пути к исполняемым файлам BitKeeper, аргументы для этих команд, а также сочетание клавиш, которое будет использоваться в качестве префикса для команд пакета.

9 Работа с Aegis

Aegis — система управления конфигурациями, разработанная Peter Miller. Система Aegis имеет много интересных возможностей, что делает ее привлекательной для использования. Одной из таких особенностей является то, что пользователи системы могут иметь разные роли — разработчики, интеграторы и т.п. Как и многие другие системы, Aegis поддерживает атомарную работу с данными, и наборы изменений. Для разработчиков ПО данная система интересна наличием возможностей для автоматической пересборки программ при подтверждении данных, и отказом в приеме, при сбое тестов.

Интеграцию Aegis с Emacs осуществляется с помощью пакета aegis-mode. Для Aegis, в отличии от многих других пакетов, не существует модуля к пакету VC, скорее всего это произошло из-за сильных различий в подходах между RCS, на идеологии которого строится пакет VC и Aegis.

9.1 Установка пакета

Вы можете скачать aegis-mode с сайта автора[1]. Просто разверните пакет туда, где его найдет Emacs, и вставьте команду загрузки в ваш файл инициализации. Например, вот так:

```
(autoload 'aegis-mode "aegis-mode" "Mode for using aegis" t)
```

И потом вы можете в любой момент выполнить команду `M-x aegis-mode`, которая выполнит загрузку пакета. Стоит отметить, что однажды включив aegis-mode, вы не можете его отключить, не прибегнув к перезапуску Emacs.

9.2 Работа с пакетом

Пользователи в Aegis могут иметь разные роли, и модуль интеграции с Emacs пытается поддерживать то же разделение на роли, что и Aegis. Пользователь может изменить роль, в которой он работает, используя меню `Aegis` или выполняя команды `aegis-become-...`. В настоящее время aegis-mode поддерживает четыре роли: администратор, разработчик, интегратор и пользователь проверяющий изменения, и рассказ соответственно пойдет, опираясь на эти роли.

После загрузки пакета, команды определенные в нем, доступны через сочетание клавиш **M–x**, или через меню `Aegis`. В зависимости от роли пользователя, меню может менять список отображаемых команд.

Команды, доступные пользователю имеют префикс `aegis-` за которым следует имя соответствующей команды Aegis. Так что работа с пакетом почти не отличается от обычной работы с Aegis в командной строке.

9.3 Настройка

Пользователь может настроить работу aegis-mode с помощью большого количества переменных и хуков, определенных в файле `aegis-vars.el`. Я коснусь лишь самых

[1] http://acsys.anu.edu.au/~tpot/aegis-mode/

важных из них, а более подробную информацию вы можете найти в указанном файле.

Для каждой из команд Aegis пользователь может указать ключи командной строки, который будут использоваться при выполнении указанной команды. Эти ключи хранятся в переменных имеющих вид `aegis-mode-команда-flags`. Так что вы можете изменять соответствующие переменные, если вам необходимо передать дополнительные ключи какой-то из команд.

Для выполнения определенных команд при переключении между ролями пользователей, пакет использует набор хуков, имена которых имеют вид `aegis-mode-become-имяроли-hook`. С помощью этих хуков, пользователь может задать дополнительную фукнциональность, которая ему требуется при переключении между ролями. Роль, которая будет присвоена текущему пользователю при загрузке пакета, определяется содержимым переменной `aegis-mode-default-hat`.

9.4 Дополнительная информация

Более подробную информацию об Aegis вы можете получить на сайте Aegis[1]. Пакет aegis-mode в достаточной мере отражает идеологию Aegis, так что вам необходимо это знание для успешного использования пакета.

10 Работа с ClearCase

Для системы ClearCase имеется как модуль для пакета VC, так и специализированный модуль — clearcase. Специализированный модуль имеет больше возможностей, и пользователям рекомендуется использовать именно его. Данный пакет поддерживает следующие возможности:

- просмотр и работа со снапшотами;
- режимы для редактирования комментарием и настроек;
- контекстное меню;
- запуск стандартных графических утилит ClearCase;
- выполнение большинства действий напрямую из Emacs.

10.1 Установка пакета

Пакет clearcase доступен для загрузки со страницы автора[2]. Он может использоваться и с GNU Emacs версии 20.4 или выше, и с достаточно свежими версиями XEmacs. Пакет может работать в разных операционных системах — как семейства Windows, так и семейства Unix. Для использования пакета, лишь поместите файлы в доступное для Emacs место, и выполните команду

[1] http://aegis.sourceforge.net
[2] http://members.verizon.net/~vze24fr2/EmacsClearCase

```
(require 'clearcase)
```

Пакет имеет проблемы при совместном использовании со старой версией пакета vc-hooks, в которой используются несовместимые версии пакета tq. Но сейчас это встречается достаточно редко.

10.2 Использование пакета clearcase

Пакет clearcase пытается по возможности сохранять совместимость с VC по части привязок клавиш. Но предоставляемый интерфейс является более мощным, и лучше отражает идеологию ClearCase, чем модуль для пакета VC.

Работа с пакетом может осуществляться через выполнение команд напрямую, используя привязки клавиш, или через меню `ClearCase`, которое появляется при редактировании файлов с использованием вспомогательного режима ClearCase. Кроме выполнения действий прямо из Emacs, пользователь может использовать и внешние программы из состава ClearCase. Сюда относятся утилиты для просмотра дерева версий и работы с проектами, а также менеджер слияний, и прочие утилиты.

Многие операции пакет позволяет производить используя режим dired. Сюда относятся операции просмотр дерева версий, массовая работа с файлами — извлечение и помещение их в репозиторий. Для обработки каталогов, находящихся под управлением ClearCase, при использовании dired, пакет clearcase определяет специальный режим — `clearcase-dired-mode`. Кроме этого, пакет определяет еще несколько вспомогательных режимов, например, режим для редактирования комментариев.

Для своих привязок клавиш, пакет clearcase использует префиксное сочетание клавиш **C–x v**. Для того, чтобы команды clearcase не пересекались с командами пакета VC, пакет clearcase удаляет неиспользуемые привязки, при работе с данными, находящимися под контролем ClearCase.

Команды работы с данными можно разделить на несколько логических групп — работа с файлами, работа с изменениями, использование внешних утилит ClearCase, получение информации, и прочие команды. Практически для каждой команды, которая работает с текущим буфером (они оканчиваются на суффикс `-current-buffer`) существует аналогичная команда, которая работает с текущим файлом в буфере dired. Такие команды имеют суффикс `-dired-file`.

10.2.1 Работа с версиями

Пакет clearcase старается в своей работе поддерживать логику операций, которые предоставляются пакетом VC. Для этого, определены команды `clearcase-next-action-current-buffer` и `clearcase-next-action-dired-files`, которые выполняют следующую операцию в цепочке действий над текущим буфером, или над выбранными файлами. Изменение состояния текущего буфера производится с помощью команды `clearcase-toggle-read-only` (сочетание клавиш **C–x C–q**), что приводит к переводу его либо в состояние доступности для внесения изменений, либо в состояние доступности только для чтения.

Пакет поддерживает массовое выполнение операций получения данных из репозитория и подтверждения данных в репозиторий. Для получения данных из репозитория

могут использоваться команды `clearcase-checkout-current-buffer`, `clearcase-checkout-dired-files` и `clearcase-dired-checkout-current-dir`, которые приводят к получению версий из репозитория для текущего буфера, отмеченных файлов и текущего каталога. А задачи по подтверждению данных производятся с помощью команд, имеющих аналогичные имена — `clearcase-checkin-current-buffer`, `clearcase-checkin-dired-files` и `clearcase-dired-checkin-current-dir`. Отмена получения данных производится с помощью команд `clearcase-uncheckout-current-buffer` (она привязана к сочетаниям **C–x v c** и **C–x v u**), `clearcase-uncheckout-dired-files` (она имеет те же привязки, что и предыдущая команда, но они определены только при работе в режиме `clearcase-dired-mode`) и команды `clearcase-dired-uncheckout-current-dir`.

Регистрация файлов в системе контроля версий ClearCase производится с помощью команды `clearcase-mkelem-current-buffer` (сочетание клавиш **C–x v i**), которая регистрирует текущий буфер, или с помощью команды `clearcase-mkelem-dired-files`, которая регистрирует выбранные файлы.

Обновление данных в текущем буфере, каталоге или файлах в буфере dired производится с помощью команд `clearcase-update-current-buffer`, `clearcase-update-default-directory`, `clearcase-update-dired-files` и `clearcase-update-view`. А получить конкретную версию файла можно с помощью команды `clearcase-version-other-window`, которая привязана к сочетанию клавиш **C–x v ~**.

Создание новой ветки производится с помощью команды `clearcase-mkbrtype`, которая привязана к сочетанию клавиш **C–x v m**.

10.2.2 Работа с изменениями

Команды работы с изменениями можно разделить на две группы. Первая группа для получения списков изменений использует команду **diff** (они имеют название начинающееся с `clearcase-diff`), а вторая группа команд, для этого использует пакет ediff (для них используется префикс `clearcase-ediff`). Ниже перечислены только команды первой группы, команды второй группы имеют аналогичные имена, с учетом вышеуказанных отличий.

- `clearcase-diff-pred-current-buffer` (сочетание клавиш **C–x v =**) и `clearcase-diff-pred-dired-file` — сравнивает текущий буфер или отмеченные файлы с их предыдущими версиями;
- `clearcase-diff-branch-base-current-buffer` и `clearcase-diff-branch-base-dired-file` — сравнивают текущий буфер или отмеченные файлы с базовыми версиями их ветки;
- `clearcase-diff-named-version-current-buffer` и `clearcase-diff-named-version-dired-file` — сравнивают текущий буфер или отмеченные файлы с указанной версией.

10.2.3 Получение информации

Для получения описания файлов используются две команды. Команда `clearcase-describe-current-buffer` (сочетание клавиш **C–x v ?**) используется для получения

описания файла в текущем буфере, а команда `clearcase-describe-dired-file` используется для получения описания файлов, выбранных в буфере dired.

Пользователи также могут получить аннотированную версию текущего буфера или отмеченных файлов с помощью команд `clearcase-annotate-current-buffer` (сочетание клавиш **C–x v g**) и `clearcase-annotate-dired-file`. А для получения истории текущего буфера или файла в буфере dired используются команды `clearcase-list-history-current-buffer` (привязанная к **C–x v l**) и `clearcase-list-history-dired-file`.

Пакет clearcase поддерживает работу со свойствами файлов ClearCase, которые связываются с зарегистрированными файлами. Пользователь может просматривать эти свойства с помощью команды `clearcase-fprop-display-properties`, а также вставлять их в текущий буфер с помощью команды `clearcase-fprop-dump-to-current-buffer`.

10.2.4 Использование внешних утилит ClearCase

Пакет clearcase позволяет использовать множество утилит, входящих в состав ClearCase, не покидая удобную среду Emacs.

- `clearcase-gui-clearexplorer` — запускает **ClearExplorer**;
- `clearcase-gui-deliver` — ;
- `clearcase-gui-merge-manager` — запускает менеджер слияний;
- `clearcase-gui-project-explorer` — запускает утилиту работы с проектами;
- `clearcase-gui-rebase` — запускает ;
- `clearcase-gui-snapshot-view-updater` — ;
- `clearcase-gui-vtree-browser-current-buffer` — запускает браузер дерева проекта для текущего буфера;
- `clearcase-gui-vtree-browser-dired-file` — запускает браузер дерева проекта для файла в dired.

Пакет также определяет набор команд для просмотра изменений с помощью внешних утилит:

- `clearcase-gui-diff-branch-base-current-buffer` — использует графический интерфейс для сравнения данных в текущем буфере;
- `clearcase-gui-diff-branch-base-dired-file` — использует графический интерфейс для сравнения выбранных версий с их базовыми версиями;
- `clearcase-gui-diff-pred-current-buffer` — использует графический интерфейс для сравнения текущего буфера с предыдущей версией;
- `clearcase-gui-diff-pred-dired-file` — использует графический интерфейс для сравнения выбранных версий с их базовыми версиями.

10.2.5 Прочие команды

Редактирование конфигурации ClearCase производится с помощью команд, которые начинаются с `clearcase-edcs`. Для редактирования конфигурации определен специальный режим — `clearcase-edcs-mode`, который включается одноименной командой. Редактирование конфигурации производится с помощью команды `clearcase-edcs-edit`, которая привязана к сочетанию клавиш **C–x v e**. Для сохранения изменений используется команда `clearcase-edcs-save`. Для завершения работы с конфигурацией ClearCase используется команда `clearcase-edcs-finish`.

Пакет clearcase обеспечивает возможность отслеживания состояния самого себя с помощью трассировки выполняемых команд. Для включения и выключения трассировки используются команды `clearcase-enable-tracing` и `clearcase-disable-tracing`. Кроме этого, пользователь может получить данные о внутреннем состоянии пакета с помощью команды `clearcase-dump`. Но эти возможности в основном нужны в тех случаях, когда пакет работает не так как нужно. Если вы нашли ошибку в работе пакета, то вы можете сообщить о ней используя команду `clearcase-submit-bug-report`, которая соберет данные о значениях переменных, версии Emacs и прочую информацию, необходимую для воссоздания ошибки и ее исправления.

Команды `clearcase-integrate` и `clearcase-unintegrate` явно устанавливают или удаляют хуки и команды, которые обеспечивают интеграцию с системой контроля версий ClearCase.

10.3 Настройка

Также как и другие пакеты, пакет clearcase поддерживает стандартный механизм настройки Emacs. Для настройки надо всего лишь выполнить команду `M-x customize-group clearcase`. С его помощью пользователь может задать множество параметров, например, что использовать в качестве программы **diff**, а также другие параметры. Надо лишь внимательно читать описания переменных и сопоставлять их с концепциями ClearCase.

11 Интеграция с Perforce

Perforce — достаточно популярная система контроля версий, используемая во многих проектах по разработке коммерческого программного обеспечения. Ссылки на ресурсы об использовании Perforce вы можете найти в разделе Дополнительная информация. Поддержка Perforce в Emacs осуществляется с помощью пакета p4.

11.1 Установка

Для установки пакета вам нужно скачать исходный текст с сайта проекта[1]. Поместите пакет в место, где его найдет Emacs, и поместите следующую команду в файл инициализации Emacs:

[1] http://p4el.sourceforge.net

```
(require 'p4)
```

Пакет может работать как с GNU Emacs, так и с XEmacs. Для работы пакета необходимо наличие Ediff, и еще нескольких пакетов, которые обычно идут в составе Emacs.

11.2 Работа с пакетом

Работу с пакетом можно логически разделить на несколько групп действий — работа с репозиторием, получение информации о файлах, работа с файлами и ветками, работа с изменениями, настройка параметров и прочие команды. Более подробно, каждая из групп действий будет рассмотрена в следующих разделах.

Команды выполняются через стандартные средства выполнения команд Emacs, используя привязки клавиш или используя меню `P4`, которое появляется после загрузки пакета, и в котором перечислены практически все команды предоставляемые пакетом. Для некоторых команд пакета, пользователь может задать дополнительные аргументы, используя сочетание клавиш **C–u**.

11.2.1 Работа с репозиторием, файлами, заданиями и ветками

Синхронизация локальных файлов с репозиторием производится с помощью команды `p4-get` (сочетание **C–x p G**). Команда позволяет синхронизировать как отдельные файлы, так и все файлы клиента. Пользователь может указать номер версии, с которой будет синхронизироваться клиент.

После получения файлов из репозитория пользователь может открыть его для изменения с помощью команды `p4-edit` (сочетание клавиш **C–x p e**). При выполнении данной команды, сервер уведомляется что файл открывается для изменения и файл переводится из состояния только для чтения, в состояние доступное для изменения. Пользователь может указать номер изменения, к которому будут относиться новые изменения, а также тип файла. Можно открыть файл с новыми параметрами (тип файла или номер изменения), для этого в пакете p4 определена команда `p4-reopen` (**C–x p E**). Кроме этого, пользователь может открыть файл напрямую в репозитории, с помощью команды `p4-depot-find-file`, которая привязана к сочетанию клавиш **C–x p C–f**.

Добавление файла в репозиторий производится с помощью команды `p4-add` (сочетание **C–x p a**). С помощью аргументов, пользователь может указать тип файла — текстовый или двоичный, а также связать новый файл с указанным отложенным изменением. Если тип файла не был указан явно, то он определяется автоматически. Удаление файла из репозитория производится с помощью команды `p4-delete` (сочетание клавиш **C–x p x**). Файл удаляется как из репозитория, так и у клиента. Если файл уже открыт, то он пере-открывается для удаления и помещается в указанное неподтвержденное изменение. Переименование файла производится с помощью команды `p4-rename` (**C–x p m**). Данная команда поддерживает использование шаблонных символов, так что можно одновременно переименовывать множество файлов.

Работа с заданиями (jobs) в Perforce реализуется пакетом p4 с помощью нескольких команд. Команда `p4-jobs` (**C–x p J**) используется для получения списка всех заданий определенных в системе. С помощью аргументов пользователь может сужать список

заданий, список поддерживаемых аргументов вы можете найти в документации на пакет p4. Создание и изменение заданий производится с помощью команды `p4-job` (сочетание клавиш **C–x p j**). С помощью префиксных аргументов, данная команда может удалять задания, а также изменять их. А команда `p4-fix` (сочетание **C–x p X**) используется для связывания задания с номером изменения. Связанные с изменениями задания закрываются в том случае, если изменения подтверждаются.

Спецификации веток могут редактироваться с помощью команды `p4-branch`, которая привязана к сочетанию клавиш **C–x p B**. Пользователь может выполнить слияние изменений между ветками с помощью команды `p4-integ` (**C–x p I**). Можно выполнять слияние как между отдельными файлами, так и между полными ветками. С помощью аргументов, пользователь может указать в какое изменение будет производиться слияние.

Пакет также поддерживает работу с метками. Для получения списка пометок используется команда `p4-labels` (**C–x p L**). Редактирование спецификации метки производится с помощью команды `p4-label` (сочетание клавиш **C–x p L**), а синхронизация меток с клиентом производится с помощью команды `p4-labelsync` (сочетание клавиш **C–x p l**).

11.2.2 Работа с изменениями

Работа с изменениями является важной частью использования системы контроля версий. Для поддержки этого, пакет p4 имеет в своем составе множество команд — для просмотра изменений, подтверждения или отмены изменений.

Получить список изменений, внесенных пользователем можно получить с помощью команды `p4-changes`, которая привязана к **C–x p C**. Эта команда выдает список отложенных и подтвержденных изменений для указанного файла. Если не было передано никаких аргументов, то данная команда выдает список изменений для файлов в текущем каталоге, и каталогах лежащих ниже. Список файлов, которые открыты для отложенного изменения, пользователь может получить с помощью команды `p4-opened` (сочетание клавиш **C–x p o**).

Детальный список изменений, внесенных пользователем можно получить с помощью нескольких команд, однако основной командой является команда `p4-diff` (сочетание клавиш **C–x p =**) — с ее помощью можно получить список изменений между текущим файлом и версией в репозитории. Однако изменения для текущего файла выдаются только в том случае, если этот файл был открыт для редактирования. Если текущий файл не открыт для изменения, то выдается список изменений для всех открытых файлов и их соответствующих версий в репозитории. Кроме этого, задавая ключи командной строки для команды diff, можно изменять выдаваемые данные для получения данных от открытых файлов, файлов отсутствующих клиентов и т.д.

Кроме указанной выше команды, существует еще несколько команд для получения детального списка изменений. Для сравнения двух файлов в репозитории можно использовать команду `p4-diff2` (сочетание клавиш **C–x p d**). Команда `p4-diff-head` выдает изменения между файлами текущего клиента и самыми последними версиями в репозитории. Для сравнения файлов можно использовать пакет Ediff, что и делают две команды — `p4-ediff` и `p4-ediff2`. Первая команда выполняет сравнение текущего файла с его базовой версией, а вторая команда сравнивает между собой два файла в репозитории.

Подтверждение изменения осуществляется с помощью команды `p4-submit` (сочетание

клавиш **C–x p S**). При выполнении этой команды, отложенное изменение и связанные с ним файлы вносятся в репозиторий. В том случае, если перед подтверждением программа не смогла заблокировать все файлы, затрагиваемые изменением, то процесс прекращается и подтверждения не происходит. Пользователь может оповещать других пользователей о вносимых изменениях. Для этого может использоваться команда `p4-notify` (сочетание клавиш **C–x p n**), которая выполняется пользователем. Но кроме этого, может использоваться автоматическое оповещение пользователей при каждом подтверждении изменений. Список пользователей, которые будут оповещены при изменении, задается с помощью команды `p4-set-notify-list`. Оповещение пользователей производится с помощью электронной почты, так что вы должны задать корректный путь к программе sendmail.

Отмена изменений в текущем файле производится с помощью команды `p4-revert`, которая привязана к сочетанию клавиш **C–x p r**. Если аргумент `SHOW-OUTPUT` имеет истинное значение, то результаты выполнения команды будут отображены в буфере `*P4 Output*`. Для обновления изменений в не-открытых файлах используется команда `p4-refresh` (**C–x p R**), так что вам нужно использовать обе команды, если у вас есть открытые и не-открытые файлы в одном каталоге и вы хотите обновить их из репозитория.

11.2.3 Получение информации

Для получения информации о конкретном изменении, может использоваться команда `p4-describe` (она привязана к сочетанию клавиш **C–x p D**). Она отображает описание изменения, его номер, имя пользователя, дату и прочую информацию, включающую в себя список файлов, список конкретных изменений и т.п.

Просмотр истории изменений для текущего файла может быть выполнен с помощью команды `p4-filelog` (сочетание клавиш **C–x p f**). Информация отображается в порядке убывания времени, вплоть до самого начала истории файла, когда он был добавлен в репозиторий.

Чтобы получить файл из репозитория вместе с историей изменения, пользователь может использовать команду `p4-print-with-rev-history` (сочетание **C–x p V**). Для получения только самого файла, без истории, пользователь может использовать команду ‘ p4-print’ (**C–x p p**). Эти две команды не затрагивают список файлов полученных данным клиентом.

Для получения информации о клиенте и сервере может использоваться команда `p4-info` (сочетание **C–x p i**), которая выдает информацию о клиенте — имя пользователя, название клиента и каталог, а также некоторую информацию о сервере.

Чтобы узнать как соотносятся между собой имена локальных файлов и имена соответствующих файлов в репозитории, пользователь может использовать команду `p4-where`, которая привязана к **C–x p w**. А список версий, которые были в последний раз получены, пользователь может получить с помощью команды `p4-have` (**C–x p H**). Получить список файлов в репозитории можно с помощью команды `p4-files` (сочетание клавиш **C–x p F**). Пользователь может получить список всех файлов, или только файлов, совпадающих с заданной маской.

11.2.4 Управление репозиторием

Пакет p4 позволяет управлять репозиторием Perforce не покидая Emacs. Сюда относятся команды управления пользователями, группами пользователей и клиентами.

Создание нового пользователя производится с помощью команды `p4-user` (сочетание клавиш **C–x p u**). Эта же команда используется для редактирования спецификации уже существующего пользователя. Спецификация сохраняется во временный и файл и для ее редактирования запускается редактор, определенный переменной среды `EDITOR`. В спецификацию пользователя входят его имя, полное имя, уровень доступа и другие параметры. Обычно, спецификация пользователя создается автоматически при запуске любой команды, которая может обновить репозиторий. Список существующих пользователей можно получить с помощью команды `p4-users` (сочетание **C–x p U**).

Аналогичным образом организована и работа с группами пользователей. Для получения списков уже существующих групп, может использоваться команда `p4-groups`, а для создания группы пользователей используется команда `p4-group`. Но для создания группы пользователей, текущий пользователь должен обладать специальными правами.

Редактирование спецификации клиента производится с помощью команды `p4-client`, которая привязана к сочетанию клавиш **C–x p c**. В спецификацию клиента входят его имя, описание, список файлов, которые будут видны в репозитории, и другие параметры. По умолчанию, новые клиенты создаются с возможностью просмотра всех файлов в репозитории.

11.2.5 Прочие команды

Сочетание клавиш **C–x p h** запускает команду `p4-help`, которая может использоваться для получения справочной информации о работе с пакетом, или о работе с конкретной командой. Описание привязок клавиш, предоставляемых пакетом p4, можно получить с помощью команды `p4-describe-bindings` (сочетание клавиш **C–x p ?**).

Номер используемой версии пакета p4 пользователь может получить с помощью команды `p4-emacs-version`, которая привязана к сочетанию клавиш **C–x p v**. В том случае, если вы нашли ошибку в работе пакета, то вы можете сообщить о ней с помощью команды `p4-bug-report` (сочетание клавиш **C–x p b**).

В том случае, если сервер Perforce недоступен или вы работаете отключившись от сети, то вы можете отключить проверку версий, при открытии файлов. Включение и отключение этой проверки пользователь может выполнить с помощью команды `p4-toggle-vc-mode` (сочетание клавиш **C–x p t**).

11.3 Настройка

Для работы с пакетом p4, ну и конечно с Perforce, необходимо установить значения переменных среды `P4CLIENT`, `P4USER` и `P4PORT`. Однако вы можете задать нужные значения и с помощью команд пакета. Для этого используются команды `p4-set-client` (сочетание клавиш **C–x p s**) и `p4-set-p4-port` (сочетание **C–x p P**). Получить название клиента, который используется для доступа к репозиторий можно с помощью команды `p4-get-client-name` (сочетание **C–x p g**).

Используя стандартные средства настройки Emacs пользователь может настроить различные параметры пакета, такие как путь к исполняемым файлам Perforce, переменные, влияющие на вывод результатов и многое другое. Группа настройки имеет тоже название что и сам пакет — p4.

11.4 Дополнительная информация

Полное описание команд пакета p4 вы можете найти на сайте проекта[1]. А информацию о Perforce и его использовании вы сможете найти на сайте компании[2]. Кроме этой информации, в интернете существует достаточное количество описаний, которые могут помочь начинающему пользователю в освоении этой системы контроля версий.

12 Интеграция с Microsoft Visual SourceSafe

Microsoft Visual SourceSafe является часто используемой системой контролей версий, в проектах по созданию программного обеспечения для операционных систем фирмы Microsoft. Кроме реализации этой системы для Microsoft Windows, существуют и реализации для других операционных систем. Подробнее об этой системе вы можете прочитать на странице на сайте Microsoft[3].

Интеграция Emacs с Visual SourceSafe обеспечивается пакетом source-safe.el. Он позволяет работать с данной системой контроля версий не покидая Emacs. В следующих разделах будет рассказано о установке, настройке и работе с пакетом.

12.1 Установка пакета source-safe.el

Для установки пакета вам нужно скачать его со страницы автора[4]. Поместите скачанный файл туда, где его найдет Emacs и используйте команды

```
(setq ss-program "S:\\WinNT\\SS.exe"
      ss-project-dirs '(("^D:\\\\OCI\\\\" . "$/PurifyW/")))
(require 'source-safe)
```

для его загрузки. Первые две строки задают расположение исполняемого модуля Visual SourceSafe и пути к каталогам проектов, а третья строка загружает пакет.

12.2 Работа с пакетом

Получение данных из репозитория осуществляется с помощью нескольких команд. Команда `ss-get` извлекает самую последнюю версию текущего файла из репозитория. В том случае, если она запускается из буфера Dired, то обновляется весь текущий каталог и все его подкаталоги. Для извлечения текущего файла из репозитория также может

[1] http://p4el.sf.net
[2] http://www.perforce.com
[3] http://www.microsoft.com/ssafe/
[4] http://emacs.seanm.ca/cvs/lisp/source-safe.el

использоваться команда `ss-checkout`. Команда `ss-lock` позволяет извлечь текущий файл из репозитория, но при этом не заменяет текущую копию данного файла.

Получить список внесенных изменений пользователь может с помощью команды `ss-diff`. Сравнение производится с помощью команды, заданной переменной `ss-diff-program`. В том случае, если задан параметр `non-interactive-p`, то результат выполнения команды отображается в новом буфере, иначе для работы с изменениями, используется Ediff. Подтверждение изменений в репозиторий производится с помощью команды `ss-checkin`.

Пользователь может создать собственную ветку разработки с помощью команды `ss-branch`. По завершении работы с ответвленным файлом, можно слить изменения, используя функцию `ss-merge`, которая обеспечивает трех-стороннее слияние изменений в базовую ветвь. Удалить неиспользуемую ветвь можно с помощью команды `ss-unbranch`, при этом удаляются все внесенные в нее изменения.

Для получения информации, пользователь может использовать несколько команд — команда `ss-history` отображает историю изменений для текущего файла, команда `ss-status` показывает статус текущего файла в репозитории Visual SourceSafe, а команда `ss-locate` позволяет найти расположение текущего файла в репозитории Visual SourceSafe.

К остальным командам пакета относятся команды `ss-help`, которая показывает краткое описание основных команд и переменных пакета, и `ss-submit-bug-report`, которая позволяет отправить разработчикам пакета сообщение о найденной ошибке.

12.3 Настройка

Настройка пакета производится с помощью переопределения соответствующих переменных. Основными переменными настройки являются переменная `ss-program`, которая задает путь к исполняемому модулю Visual SourceSafe и переменная `ss-project-dirs`, которая задает связи между каталогами на диске и именами проектов.

Пользователь также может настроить поведение пакета с помощью других переменных:

ss-username — задает имя пользователя, если значение не задано, то будет использоваться имя текущего пользователя;

ss-password — задает пароль пользователя, если значение не задано, то пароль будет запрошен при первом доступе к репозиторию;

ss-database-alist — список, связывающий именя файлов с базами данных Visual SourceSafe. Эта переменная используется в том случае, если вы одновременно используете несколько баз Visual SourceSafe;

ss-update-email-to, ss-update-email-subject и ss-update-email-body — определяют адрес куда будут отраправляться сообщения о подтверждении данных в репозиторий, а также заголовок сообщения и его тело;

ss-multiple-checkouts-enabled — определяет, можно ли использовать множественное получение данных из репозитория;

ss-confirm-updates — определяет, нужно ли требовать подтверждения при выполенении команды **UPDATE** или нет.

Пакет также определяет два хука — `ss-before-update-hooks` и `ss-after-update-hooks`, которые вызываются до и после выполнения команды **UPDATE** систему Visual SourceSafe.

13 Поддержка Monotone

Monotone — распределённая система контроля версий по концепции схожая с GNU Arch и Darcs. Работа с Monotone из Emacs обеспечивается пакетом monotone.el, который идёт в составе дистрибутива Monotone. Также достаточно полная поддержка Monotone обеспечивается специальным модулем из пакета DVC, который описан в разделе Пакет DVC

13.1 Установка пакета monotone.el

Пакет monotone.el идёт в поставке Monotone. Вы можете найти его в подкаталоге `contrib` дистрибутива Monotone. Пакет тестировался только с GNU Emacs версии 22.1, но наверное будет работать и с другими версиями Emacs Для работы с данным пакетом поместите его туда, где его найдёт Emacs и поместите в файл инициализации следующие команды:

```
(require 'monotone)
(monotone-set-vc-prefix-key "\C-xv")
(setq monotone-passwd-remember t)
```

первая строка загружает сам пакет, вторая строка устанавливает префикс для команд выполняемых с клавиатуры, а третья строка заставляет пакет запоминать введённые пользователем пароли.

13.2 Использование пакета

Доступ к командам пакета может осуществляться разными способами — через меню, с использованием клавиатурных сочетаний и используя именованные команды (все команды имеют префикс **monotone**). Префикс для клавиатурных сочетаний по умолчанию не определён, поэтому его нужно установить при инициализации пакета с помощью функции `monotone-set-vc-prefix-key`, в моем примере файла инициализации она привязывается к префиксу **C–x v**, и далее по тексту будет использоваться именно этот префикс. Команды Monotone выполняются в буфере с именем `*monotone*`.

Регистрация файла в системе контроля версий производится с помощью команды **monotone-vc-register** (сочетание клавиш **C–x v i**). Изменения затем надо будет подтвердить с помощью команды **monotone-vc-commit** для вызова которой можно использовать как сочетание **C–x v q**, так и **C–x v C–q**.

Работа с удалёнными серверами осуществляется с помощью трёх команд: команда **monotone-pull** (сочетание **C–x v p**) — вытягивает изменения с сервера; команда

monotone-push (сочетание **C–x v P**) — помещает изменения на сервер; а команда **monotone-sync** — синхронизирует текущую базу данных с сервером.

Пользователь может получить список сделанных изменений с помощью команды **monotone-vc-diff**, которая привязана к **C–x v =**. Если этой команде передать аргумент, то просмотр изменений будет сделан для всего проекта.

Получить информацию о зарегистрированных объектах можно с помощью нескольких команд. В первую очередь это команда **monotone-vc-status** (сочетание клавиш **C–x v s**), которая выдаёт информацию о текущей ветке проекта. С помощью команды **monotone-vc-print-log** (сочетание **C–x v l**) можно получить лог для текущего буфера (а при задании аргумента для этой команды, можно получить лог для всего проекта). А получить идентификатор текущего объекта можно с помощью команды **monotone-grab-id** (сочетание клавиш **C–x v 6**) — данная команда сохраняет идентификатор объекта в буфере удаления (kill buffer), откуда вы её может вставить с помощью стандартных команд.

В пакете определена одна команда общего назначения — `monotone` (она привязана к сочетанию клавиш **C–x v x**) [1], которая позволяет выполнить любую команду Monotone введённую в мини-буфере.

13.3 Настройка пакета

Настройка пакета осуществляется путём установки некоторого количества переменных. Одну из переменных вы уже видели, это переменная `monotone-passwd-remember`, которая заставляет пакет запоминать введённые пользователем пароли. В том случае, если эта переменная установлена, пароли будут сохраняться в переменной `monotone-passwd-alist`. Кроме этого, используя переменную `monotone-program`вы можете указать имя исполняемого файла Monotone.

Адреса удалённого сервера и коллекций на нем задаются переменными `monotone-server` и `monotone-collection`. Список уже использованных серверов и коллекций хранится в соответствующих переменных: `monotone-server-hist` и `monotone-collection-hist`.

Кроме перечисленных выше переменных, существуют и другие переменные, изменять которые пользователь скорее всего не будет, но знать которые желательно. Истинное значение переменной `monotone-cmd-show` приводит к показу результатов выполнения команд Monotone. С помощью переменной `monotone-program-args-always` можно задать аргументы командной строки, которые всегда будут передаваться при запуске Monotone. Если значение переменной `monotone-msg` не равно `nil`, то в буфер `*Messages*` будут выводиться сообщения пакета monotone.el, что может быть полезным при отладке.

Тонкую настройку работы пакета можно осуществить, используя два хука — `monotone-output-mode-hook` и `monotone-commit-mode-hook`. Первый хук будет вызываться при выводе данных, а второй — при входе в режим редактирования сообщений при подтверждении данных в репозиторий.

[1] Помните, что префикс вы можете задать сами!

14 Работа с Git

Появление и популярность системы контроля версий Git связана с разработкой операционной системы Linux. Первоначально данная система разрабатывалась для обеспечения работы над ядром операционной системы, но затем она стала применяться и в других проектах с открытым исходным кодом. Поскольку над такими проектами работают люди из разных точек земного шара, то и сама система контроля версий имеет идеологию, позволяющую работу в децентрализованной среде[1].

Поддержка Git может осуществляться двумя способами — используя модуль vc-git.el для пакета VC, и используя пакет git.el. Первый пакет обеспечивает работу через стандартные интерфейсы пакета VC, в то время как пакет git.el реализует интерфейс, подобный интерфейсу PCL-CVS, позволяя использовать все возможности Git. Далее в статье речь пойдет о пакете git.el.

14.1 Установка пакета

Установка обоих пакетов очень проста — они оба идут в составе дистрибутива Git и находятся в подкаталоге `contrib/emacs/`. Для компиляции пакета достаточно выполнить команду make, которая также может использоваться для установки пакетов (по умолчанию, пакеты ставятся в каталог `$HOME/share/emacs/site-lisp`, но вы можете изменить это значение передав параметр `emacslispdir` команде make или просто скопировать нужные файлы в каталог, где Emacs сможет найти их.

Для использования пакета git.el, вам нужно поместить в ваш файл инициализации следующую команду:

```
(require 'git)
```

после выполнения которой вам станут доступны все команды пакета.

14.2 Использование пакета git.el

Использование пакета всегда начинается с выполнения команды **git-status**, которая запросит у вас имя каталога в котором необходимо искать дерево Git, и выполнит анализ состояния этого дерева. После анализа, будет открыт буфер с именем `*git-status*`, в котором можно будет выполнять действия с файлами проекта. Пример вывода в буфера вы можете видеть на рисунке Буфер Darcsum (в нижней части вы можете видеть вывод команды **diff**. Для данного буфера определен специальный режим — **git-status-mode**, для которого определенно некоторое количество команд (и соответствующих им комбинаций клавиш, многие из которых совпадают с аналогичными клавишами режима PCL-CVS, что облегчает переход на использование Git). Все команды работают только в данном буфере и не могут использоваться вне его.

Навигация по списку файлов, отображаемому после выполнения команды **git-status** может осуществляться как с помощью клавиш курсора, так и с помощью команд **git-next-file** (клавиши **n** и **SPC**) и **git-prev-file** (клавиша **p**). Для большинства команд можно задать числовой аргумент, который позволит изменить поведение команды. Числовой

[1] Хотя существует возможность использования данной системы и в централизованной среде

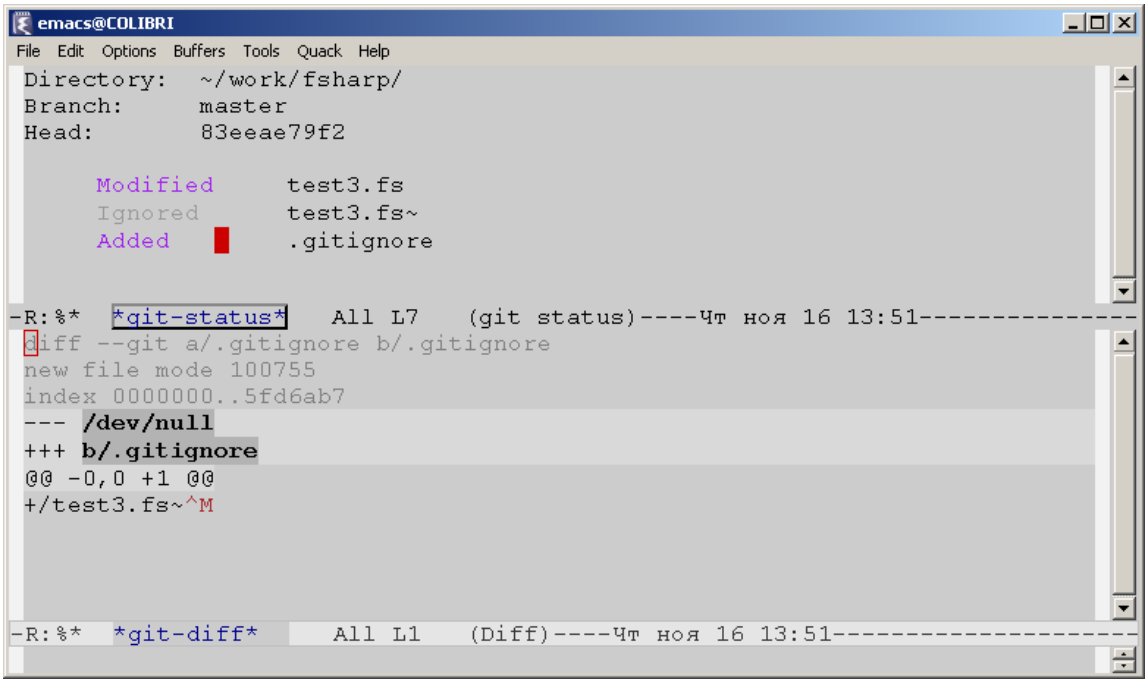

Рис. 5. Буфер *git-status*

аргумент вводится путем набора цифр, а затем вызова нужной команды. Вы также можете использовать отрицательные аргументы, добавив знак минус перед набираемыми цифрами.

По умолчанию, все команды выполняются над файлом, на котором стоит курсор, но также существуют команды для отметки нескольких файлов, над которыми могут выполняться операции. Сюда можно отнести следущие команды и комбинации клавиш: клавиша **m** (команда **git-mark-file**) отмечает один файл и перемещает курсор вниз, а с помощью клавиши **M** (команда **git-mark-all**) вы можете отметить все файлы. Для снятия пометки с одного файла вы можете использовать клавиши **u** (**git-unmark-file**) или **DEL** (**git-unmark-file-up**), только стоит отметить, что первая команда перемещает курсор вниз, а вторая — вверх. Для снятия пометок со всех файлов вы можете использовать команду **git-unmark-all** (сочетание клавиш **M–DEL**). Для инверсии пометок можно использовать команду **git-toggle-all-marks** (клавиша **T**).

Практически также как и в PCL-CVS, открыть текущий файл можно с помощью команды **git-find-file** (клавиша **RET** или **f**). Открыть файл для просмотра можно с помощью клавиши **v** (команда **git-view-file**). В том случае, если при слиянии версий в файле возникли конфликты, то вы можете вызвать команду разрешения конфликтов — **git-resolve-file** (клавиша **R**).

Добавление новых файлов в проект осуществляется с помощью команды **git-add-file** (клавиша **a**), а удаление — командой **git-remove-file** (клавиша **r**). Для того, чтобы избавиться от ненужных файлов, вы можете поместить их в список игнорируемых файлов с помощью команды **git-ignore-file** (**i**). Подтверждение изменений осуществляется с помощью команды **git-commit-file** (клавиша **c**), а отмена изменений в отмеченных файлах — с помощью команды **git-revert-file** (**U**). Сообщения, которые вводятся при подтверждении изменений, можно просмотреть с помощью команды **git-log-file** (клавиша

l).

Клавиша **d** является префиксом для команд, связанных с отображением изменений. Самой важной командой является команда **git-diff-file**, которую можно вызвать нажав = или **d** =. Команда **git-diff-file-base** (**d b**) позволяет выполнить поиск изменений относительно базового файла. Команда **git-diff-file-idiff** (**d e**) позволяет выполнить интерактивный просмотр изменений в текущем файле. А команда **git-find-file-imerge** (**d E**), позволяет открыть текущий файл в режиме интерактивного применения изменений. Остальные команды позволяют выполнить просмотр изменений относительно основной ветки — **git-diff-file-merge-head** (**d h**), относительно версий файлов пользователя до слияния веток — **git-diff-file-mine** (**d m**), и т.п. Полный список команд вы можете получить из справки.

К прочим командам можно отнести команды обновления статуса буфера **git-refresh-status** (клавиша **g**), выхода из данного буфера используется команда **git-status-quit** (клавиша **q**), удаления из списка обработанных файлов **git-remove-handled** (клавиша **x**) и получения справки о работе в данном буфере — **git-help** (клавиши **h** или **?**).

14.3 Настройка пакета

Настройка пакета осуществляется с помощью стандартных средств настройки Emacs. Группа настройки имеет название `git` и позволяет задать и параметры Git и настройки начертаний, используемых при отображении данных.

15 Поддержка Mercurial

Поддержка Mercurial в Emacs реализуется несколькими пакетами. В поставке Mercurial идет пакет mercurial.el. Также поддержка Mercurial может обеспечиваться модулем из пакета DVC, который описан в разделе Пакет DVC. Данный раздел будет полностью посвящен описанию пакета mercurial.el.

15.1 Установка и настройка пакета

Для установки пакета необходимо скопировать его из каталога `contrib` дистрибутива Mercurial в то место, где его сможет найти Emacs, и прописать в файле инициализации следующую команду[1]:

```
(require 'mercurial)
```

15.2 Работа с пакетом

Принципы работы mercurial.el во многом аналогичны работе пакета VC (стоит отметить, что поддержка Mercurial имеется и в VC) и используют те же самые привязки клавиш, что и пакет VC. mercurial.el — вспомогательный режим, который

[1]Самую последнюю версию пакета mercurial.el можно взять из репозитория, ссылка на который указана на странице Emacs WiKi, посвященной Mercurial http://www.emacswiki.org/cgi-bin/wiki/MercurialMode.

добавляет некоторое количество команд для выполнения основных задач при работе с репозиториями.

Часть команд пакета имеет глобальные привязки клавиш (по умолчанию это **C–c h**, но может быть и изменено в настройке), а часть команд доступна только внутри буфера, в котором включен режим `hg-mode`. При этом, для многих команд можно также задать префиксный аргумент, который позволит задать дополнительные параметры в интерактивном режиме. Справку по командам и привязкам клавиш пакета можно получить с помощью команды `hg-help-overview` (**C–c h h**).

Чтобы просмотреть изменения, сделанные в процессе работы, пользователь может использовать либо команду `hg-diff` (**C–x v =**), которая показывает изменения для текущего файла, либо команду `hg-diff-repo` (**C–c h =**), которая показывает изменения для всего репозитория. Обе этих команды, открывают дополнительный буфер, в котором и отображаются изменения. Чтобы закрыть этот буфер используйте клавишу **q**.

Подтверждение изменений осуществляется командой `hg-commit-start`, которая имеет две привязки клавиш: локальную — **C–x v n** и глобальную — **C–c h c**. При выполнении этой команды, mercurial.el создает новый буфер, в котором пользователь может ввести текст сообщения, которое будет записано в журнал изменений. Кроме текста пользователя, в нижней части буфера отображается список файлов, изменения для которых будут подтверждены. По умолчанию, это все файлы, но пользователь может изменить этот список путем перемещения курсора на имя файла и изменения признака выбора (жирный шрифт) с помощью клавиш **SPC** или **RET**, или с помощью средней кнопки мыши. На рисунке Подтверждение изменений в репозиторий mercurial можно увидеть пример работы с этим буфером, при подтверждении изменений.

В буфере, созданном при выполнении команды `hg-commit-start` включается отдельный режим, имеющий название `hg-commit-mode`. Для него определено несколько сочетаний клавиш, которые могут быть использованы пользователем. **C–c c–x**, так же как и во многих других пакетах, производит операцию подтверждения изменений в репозиторий, используя введенный пользователем текст. Сочетание **C–c c–k** прерывает процесс подтверждения изменений. А сочетание **C–x v =** позволяет выполнить просмотр подтверждаемых изменений.

Для отмены сделанных изменений можно также воспользоваться одной из двух команд: `hg-revert-buffer` (**C–x v u**) откатывает изменения только для текущего файла, а команда `hg-revert` (**C–c h U**) проделывает то же самое для всех измененных файлов в репозитории.

Для добавления файла в репозиторий можно воспользоваться командой `hg-add` (**C–c h a**). По умолчанию, она добавляет в репозиторий текущий файл, но если ей задать префиксный аргумент, то она запросит имя файла, который необходимо добавить в репозиторий. Функция `hg-forget` (**C–c h f**) (она еще не реализована полностью) предназначена для отмены добавления файла в репозиторий, если вы еще не выполнили команду подтверждения изменений, что бывает полезно в некоторых случаях.

Чтобы посмотреть статус файлов в репозитории, можно воспользоваться командой `hg-status` (**C–c h s**), но в отличии от других пакетов, пользователю не разрешается выполнять какие либо операции с полученными данными. Для просмотра истории изменений конкретного файла определена команда `hg-log` (**C–x v l**). Аналогичная команда для просмотра истории изменений репозитория называется `hg-log-repo` (**C–**

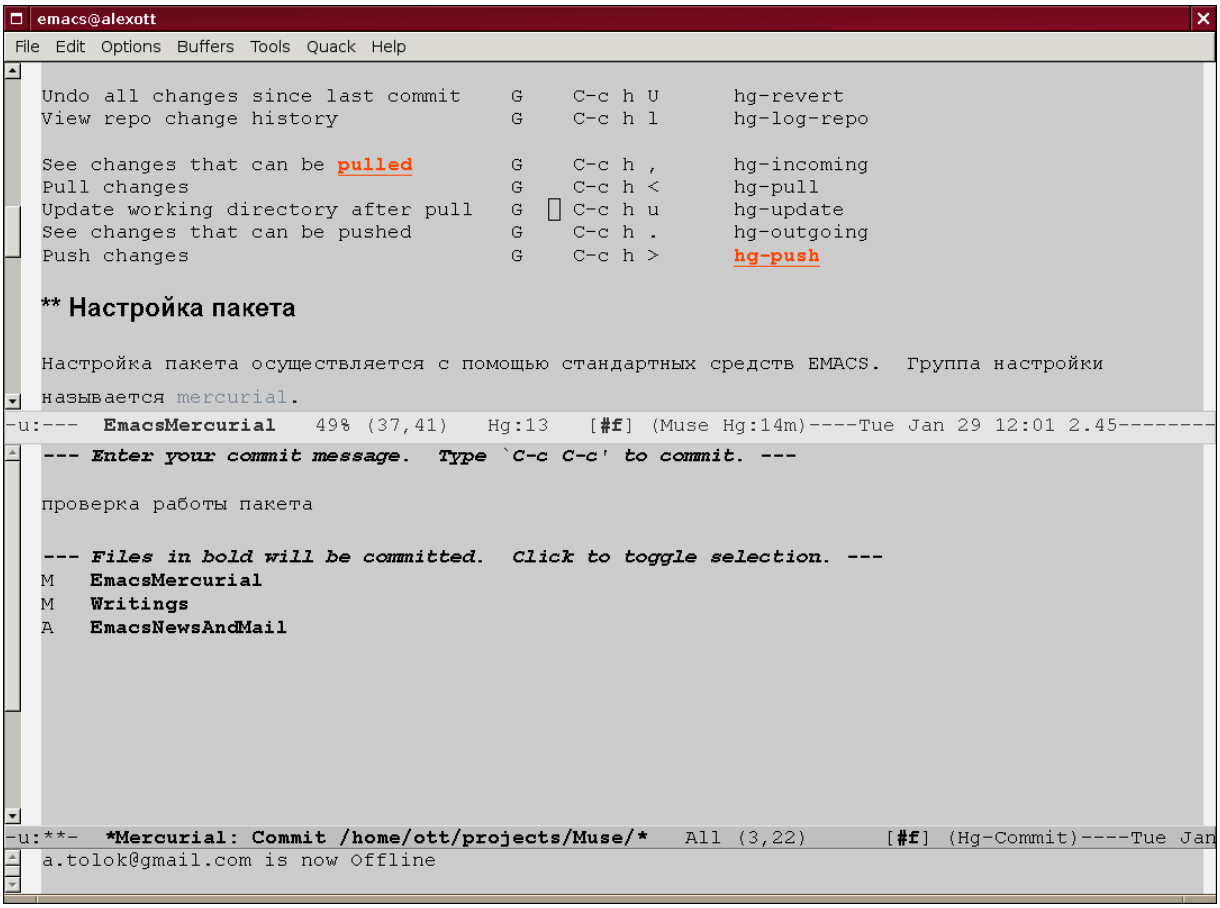

Рис. 6. Подтверждение изменений в репозиторий mercurial

c h l). Также, определена и команда `hg-annotate` (**C–x v a**), которая должна показывать файл, с указанием того, в какой версии что менялось, но пока эта функция полностью не реализована.

Кроме этих базовых операций, пакет также предоставляет набор основных операций для работы с ветками и удаленными репозиториями. Команда `hg-incoming` (**C–c h ,**) позволяет просмотреть список изменений в удаленном репозитории, которые отсутствуют в текущем репозитории, а команды `hg-pull` (**C–c h <**) и hg-update (**C–c h u**), соответственно, скачать изменения, и применить их к текущему репозиторию. Команды `hg-outgoing` (**C–c h .**) и `hg-push` (**C–c h >**) позволяют просмотреть какие изменения присутствуют в текущем репозитории, но отсутствуют в удаленном, и поместить их туда (push).

15.3 Настройка пакета

Настройка пакета осуществляется с помощью стандартных средств EMACS. Группа настройки называется `mercurial`. Пользователь может, например, изменить префикс для глобальных привязок клавиш, используемых некоторыми командами (по умолчанию — **C–c h**).

Пользователь для настройки поведения пакета может воспользоваться несколькими

хуками, которые будут вызываться в определенные моменты работы пакета. `hg-commit-mode-hook` будет вызываться после создания буфера, отображаемого пользователю перед подтверждением изменений. В свою очередь `hg-pre-commit-hook` вызывается после того, как пользователь введет сообщение для журнала изменений, и перед тем, как будет выполнено подтверждение изменений в репозитории. `hg-log-mode-hook` вызывается после создания буфера, заполненного информацией из журнала изменений при вызове команд `hg-log-repo` и `hg-log`. И конечно, пользователь может определить `hg-mode-hook`, который будет выполнен в процессе включения `hg-mode` для данного буфера.

16 Дополнительная информация

В мире существует множество систем контроля версий. Достаточно полный список систем контроля версий вы можете найти в каталоге Google[1].

Кроме этого, статьи о системах контроля версий вы можете найти на сайтах http://better-scm.berlios.de/comparison/, http://wiki.gnuarch.org/moin.cgi/SubVersionAndCvsComparison и http://www.a-a-p.org/tools_version.html.

[1] http://directory.google.com/Top/Computers/Systems/Apple/Macintosh/Development/Tools/Version_Control/