Отчёт по лабораторной работе №2

Дисцпоина-Ахитектура компьютеров и операционные системы.Операционные системыю

Дедова Виктория Сергеевна.Нбибд-01-22

Contents

List of Figures

List of Tables

1 Цель работы

Изучить идеологию и применение средств контроля версий.Освоить умения по работе с git.

2 Теоретическое введение

Здесь описываются теоретические аспекты, связанные с выполнением работы.

Например, в табл. 2.1 приведено краткое описание стандартных каталогов Unix.

Table 2.1: Описание некоторых каталогов файловой системы GNU Linux

Имя каталога	Описание каталога
`/`	Корневая директория, содержащая всю файловую
`/bin`	Основные системные утилиты, необходимые как в однопользовательском режиме, так и при обычной работе всем пользователям
`/etc`	Общесистемные конфигурационные файлы и файлы конфигурации установленных программ
`/home`	Содержит домашние директории пользователей, которые, в свою очередь, содержат персональные настройки и данные пользователя
`/media`	Точки монтирования для сменных носителей
`/root`	Домашняя директория пользователя `root`
`/tmp`	Временные файлы
`/usr`	Вторичная иерархия для данных пользователя

Более подробно об Unix см. в [1–6].

3 Выполнение лабораторной работы

Базовая настройка git

Первичная настройка параметров git

Зададим имя и email владельца репозитория: git config –global user.name “Name Surname” git config –global user.email “work@mail” Настроим utf-8 в выводе сообщений git: git config –global core.quotepath false Настройте верификацию и подписание коммитов git. Зададим имя начальной ветки (будем называть её master): git config –global init.defaultBranch master

Создайте ключи ssh

по алгоритму rsa с ключём размером 4096 бит: ssh-keygen -t rsa -b 4096 по алгоритму ed25519: ssh-keygen -t ed25519

Создайте ключи pgp

Генерируем ключ gpg –full-generate-key

Из предложенных опций выбираем: тип RSA and RSA; размер 4096; выберите срок действия; значение по умолчанию — 0 (срок действия не истекает никогда). GPG запросит личную информацию, которая сохранится в ключе: Имя (не менее 5 символов). Адрес электронной почты. При вводе email убедитесь, что он соответствует адресу, используемому на GitHub. Комментарий. Можно ввести что угодно или нажать клавишу ввода, чтобы оставить это поле пустым.

1 2 3

Добавление PGP ключа в GitHub Выводим список ключей и копируем отпечаток приватного ключа: gpg –list-secret-keys –keyid-format LONG Отпечаток ключа — это последовательность байтов, используемая для идентификации

более длинного, по сравнению с самим отпечатком ключа. Формат строки: sec Алгоритм/Отпечаток_ключа Дата_создания [Флаги] [Годен_до] ID_ключа Скопируйте ваш сгенерированный PGP ключ в буфер обмена: gpg –armor –export | xclip -sel clip Перейдите в настройки GitHub (https://github.com/settings/keys), нажмите на кнопку New GPG key и вставьте полученный ключ в поле ввода. Настройка автоматических подписей коммитов git Используя введёный email, укажите Git применять его при подписи коммитов: git config –global user.signingkey git config –global commit.gpgsign true git config –global gpg.program $(which gpg2) Настройка gh Для начала необходимо авторизоваться gh auth login Утилита задаст несколько наводящих вопросов Авторизоваться можно через броузер.

4 5 Также у вас поросят ввести свою кодовую фразу,которую вы писали при создании. 6

Контрольные вопросы: 1.Что такое системы контроля версий (VCS) и для решения каких задач они предназначаются? Система контроля версий (VCS) — это система, регистрирующая изменения в одном или нескольких файлах с тем, чтобы в дальнейшем была возможность вернуться к определённым старым версиям этих файлов. Для примеров в этой книге мы будем использовать исходные коды программ, но на самом деле под версионный контроль можно поместить файлы практически любого типа. Если вы графический или веб-дизайнер и хотели бы хранить каждую верспю изображения или макета - а этого вам наверняка хочется - то пользоваться системой контроля версий будет очень мудрым решением. даёт возможность возвращать отдельные файлы к прежнему виду, возвращать к прежнему состоянию весь проект, просматривать происходящие со временем изменения, определять, кто последним вносил измепения во внезапно переставший работать модуль, кто и когда внёс в код какую-то ошибку, и многое другое. Вообще, если, пользуясь, вы всё непортите или потеряете файлы, всё можно будет легко восстановить. Вдобавок, накладные расходы за всё, что вы получаете, будут очень маленькими 2.Объясните следующие понятия VCS и их отношения:

хранилище, commit, исторня, рабочая копия. Хранилище-система, которая обеспечивает хранение всех существовавших вариантов файлов Со-фиксация изменений История-список предыдущих ревизий Рабочая копия-копия другой ветки Команде commit можно передать сообщение, описывающее изменения в ревизии. Она также записывает идентификатор пользователя, текущее время и временную зону, плюс список измененных файлов и их содержимого. Сообщение, описывающее изменения, определяется через опцию -т, или — message, Можно также вводить сообщения, состоящие из пескольких строк; в большинстве оболочек вы можете сделать это оставив открытую кавычку в конце строки. commit -m “добавлен первый файл. 3.Что представляют собой и чем отличаются пентрализованные п депентрализованные VCS? Приведите примеры VCS каждого вида. Системы контроля версий. Централизованная система контроля версий Subversion и децентрализованная спетема контроля версий Mercurial. Существуют СКВ цептрализованные, в которых имеется одиш репозиторийї, в который собираются измепения со всех рабочих копий разработчиков. и децентрализованые. когда репозиториея много, п они могут обмениваться изменениями между собой. Централизованные СКВ - репозиторииї олин. У кажлого разработчика своя рабочая копия Время от времени разработчик может затягивать к себе в рабочую копию новье пзменения из репозитория, шли проталкивать своп изменения из своей рабочей копии в репозиторий. Прочне особенности централизованных СКВ зависят от реализации 4.Опишите действия с VCS при единолишой работе с храпилищем. Традиционные системы управления версиями используют централизованную модель, когда имеется елиное хранилище документов, уцравляемое специальным сервером. который и ниполняет большую часть функций по управленто верспями. Пользователь. работающий с локументами, должен сначала получить нужную ему версию документа из хранила; обычно создаётся локальная копия документа, т. н. «рабочая копия». Может быть получена последняя версия или лобия из предыдущих, которая может быть

выбрана по номеру верези или дате создания, шогда и по другим признакам. После того, как в локумент виссены нужные изменения, новая версия помещается в хранштлишо. В отличие от простого сохранения файла. прельдушая версия не стирается, а тоже остается в хранилище и может быть отуда получена в любое время. Сервер может использовать т. н дельта-компрессто такой способ хранения документов, при котором сохраляются. только изменения между последовательными версиями, что позволяет уменьшить объём хранимых данных. Поскольку обычно наболее востребованной является последняя версия файла, система может при сохранении новой версии сохранять её целиком. заменяя в хранилище последшою рапее сохранённую версшо на разницу между этой и последпей верспей. Некоторые системны (например. ClearCase) поддерживают сохранение версий обоих вилов: большинство версиі сохраняется в виде дельт, но периодически (по спелиальной команде администратора) выполияется сокрацение версий всех файлов в полпом виде: такой подход обеспечивает максимально полое восстановление петорий в случае повоеждения репозитория только изменения между последовательными версиями, что позволяет уменьшить объём хранимых данных. Поскольку обычно наиболее востребованной является последняя версия файла, система может при сохранении новой версии сохранять её целиком, заменяя в хранилище последнюю ранее сохранённую версию на разницу между этой и последней версией. Некоторые системы (например. ClearCase) поддерживают сохранение версий обонх видов: большинство версий сохраняется в виде дельт, но периодически (по специальной команде администратора) выполняется сохранение версий всех файлов в полном виде; такой подход обеспечивает максимально полное восстановление пстории в случае повреждения репозитория. 5.Опишите порядок работы с общим хранилищем VCS. Традиционные системы управления версиями используют централизованную модель, когда имеется единое хранилище документов, управляемое специальным сервером, который и выполняет большую часть

функций по управлению версиями. Пользователь, работающий с документами, должен сначала получить нужную ему версию документа из хранилища; обычно создается локальная копия документа, т. н. «рабочая копия». Может быть получена последняя версия или любая из предыдущих, которая может быть выбрана по номеру версии или дате создания, иногда и по другим признакам. После того, как в документ внесены нужные изменения, новая версия помещается в хранилище. В отличие от простого сохранения файла, предыдущая верспя не стирается, а тоже остаётся в хранилище и может быть оттуда получена в любое время. Сервер может использовать т. н. дельт компрессию - такой способ хранения документов, при котором сохраняются только изменения между последовательными версиями, что позволяет уменьшить обьём хранимых данных. Поскольку обычно наиболее востребованной является последияя версия файла, система может при сохранении повой версии сохранять её целиком, заменяя в хранилище последнюю ранее сохраненную версию на разницу между этой и последней версней. Некоторые системы (например, ClearCase) поддерживают сохранение версий обоих видов: большинство версий сохраняется в виде дельт, но периодически (по специальной команде администратора) выполняется сохранение версий всех файлов в полном виде; такой подход обеспечивает максимально полное восстановление истории в случае повреждения репозитория. 6.Каковы основные задачи, решаемые инструментальным средством git? Устанавливает единственную новую команду, git. Все возможности предоставляются

через подкоманды этой команды. Вы можете просмотреть краткую справку командой help. Некоторые иден грушируются по темам, используйте help topis для списка доступных тем. Одна нз функций системы контроля версий - отслеживать кто сделал изменення. В распределенных системах для этого требуется идентифицировать каждого автора уникально в глобальном плане. Большинство лодей уже имеют такой пдентификатор: email адрес. Вагааг достаточно умен, чтобы автоматически создавать email адрес из текущего

имени и адреса хоста. Основные задачи: создание ветки, размещение веток, просмотр изменений, фиксация изменений. сообщение из текстового редактора, выборочная фиксация, удаление зафиксированных изменений, игнорирование файлов, просмотр истории, статистика ветки, контроль файлов и каталогов, ветвление, объединение веток, публикация ветки 7.Назовите и дайте краткую характеристику командам git. Обновление рабочей копии По мере внесения изменений в проект рабочая копия на компьютере разработчика стареет, расхождение её с основной версией проекта рабочая копия на компьютере разработчика стареет, расхождение её с основной версией проекта увеличивается. Это повьает риск возникновения конфликтных изменений (см. ниже). Поэтому удобно поддерживать рабочую копию в состоянии, максимально близком к текущей основной верспей, для чего разработчик выполняет операцию обновления рабочей копии (update) насколько возможно часто (реальная частота обновлений определяется частотой внесения изменений, зависящей от активности разработки и числа разработчиков, а также временем, затрачиваемым на каждое обновление - если оно велико, разработчик вынужден ограничичать частоту обновлений, чтобы не терять время). Модификация проекта Разработчик модифицирует проект, изменяя вхо дящие в него файлы в рабочей копии в соответствии с проектным заданием. Эта работа производится локально и не требует обращений к серверу VCS. Фиксация изменений Завершив очередной этан работы над заданием. разработчик фикспрует (commit) свон изменения, передавая их на сервер (либо в основную ветвь, если работа над заданием полностью завершена, либо в отдельную ветвь разработки данного задания). VCS может требовать от разработчика перед фиксацией обязательно выполнить обновление рабочей копии. При наличии в системе поддержки отложенных изменений (shelving) изменения могут быть переданы на сервер без фиксации. Если утверждённая политика работы в VCS это позволяет, то фиксация измепений может проводиться не ежедневно, а только по завершении работы над заданием; в этом случае

до завершения работы все связанные с заданием изменения сохраняются только в локальной рабочей копии разработчика. 8.Приведите примеры использования при работе с локальным и удалённым репозиториями. Мы создаем новую ветку выполнив git init в уже созданном каталоге: % mkdir tutorial % cd tutorial % Is-a ./ ../% pwd /home/mbp/work/bzr.test/tutorial % % git init % Is -aE / ./ .git/ % Мы обычно обращаемся к веткам на нашем компьютере просто передав имя каталога содержащего ветку. уг также поддерживает доступ к веткам через http и sftp, например: git log http://bazaar-vcs.org git // git.dev/ git log sftp://bazaarves.org/bzr/bzr.dev/ Установив для дії плагины можно также осуществлять доступ к веткам с использованием гус. Команда statys показывает какие изменения были сделаны в рабочем каталоге с момента последней ревизии: % git status modified: foo bzr status скрывает неинтересные файлы, которые либо не менялись, либо игнорируются. Также команде status могут быть переданы необязательные имена файлов, пли каталогов для проверки. Команда diff показывает изменения в тексте файлов в стандартном формате diff. Вывод этой команды может быть передан другим командам, таким как “patch,” “diffstat,” “filterdiff’ n”colordifP: *% git diff — added file ‘hello.txt’ —- hello.txt 1970-01-01 00:00:00 -0000 + 1+ hello.txt 2005-10-18 14:23:29 +00006.2. Указания к лабораторноі работе 75 @@ -0,0 +1,1 @@ + hello world Команде commit можно передать сообщение описывающее изменения в ревизии. Она также записывает идентификатор пользователя, текущее время и временную зону, плюс список измененных файлов и их содержимого. git commit -m “добавлен первый файл” Если вы передадите список имен файлов, или каталогов после команды commit. то будут зафиксированы только изменения для переданных объектов. Например: bz commit - m “исправления документации” commit.py Если вы сделали какие-либо изменения и не хотите оставлять их, используйте команду evert. что бы вернутся к состоянию предыдущей ревизии. Многие деревья с исходным кодом содержат файлы которые не нужно хранить под контролем версий, например резервные файлы текстового редактора, объектные файлы и*

собранные программы. Вы можете просто не добавлять их, но они всегда будут обнаруживаться как неизвестные. Вы также можете сказать dі игнорировать их добавив их в файл idnoге в корне рабочего дерева. Для получения списка файлов которые игнорируются и соответствующих им шаблонов используйте комапу ignored: % ignored config.h./config.h configure.in- ~ log Команда bz Iод показывает список предыдуших ревизий. Команда log —forward делает тоже самос, но в хронологическом порядке, показывая более поздние ревизии в конце может контролировать файлы и каталоги, отслеживая переименования и упрощая их последующее обьединение: % mkdir sIc % echo ‘int main (’ > src/simple.c % add src added src added sec/simple.‹ % status added: src/ src/simple.c bzr remove удаляет файл из под контроля версий, но может п не удалять рабочую кошю файла2. Это улобно, когда вы добавили не тот файл, или решили, что файл на самом деле не должен быть под контролем версий. % гт -г sc % remove -v hello.txt ? hello.txt % status removed: hello.txt sre/ src/simple.c unknown: hello.txt Часто вместо того что бы начинать свой собственный проект, выхотите предложить изменения для уже готового проекта. Что бы сделать это вам нужно получить копню готовой ветки. Так как эта копия может быть потенциальной новой веткой, Если две ветки разошлись (обе имеют уникальные изменения) тогда теге - это подходящая команда для использования. Объединение автоматически вычислит изменения, которые существуют на объединяемой ветке п отсутствуют в локальной ветке и попытается обьединить их с локальной веткой. git merge URL. 9.то такое и зачем могут быть нужны ветви (branches)? Часто вместо того что бы начинать свой собственный проект, вы хотите предложить изменения для уже готового проекта. Что бы сделать это вам нужно получить копию готовой ветки. Так как эта копия может быть потенциальной повой веткой эта команда называется branch: Управление версиями git branch cd git.dev Эта команда копирует полную исторшю ветки и после этого вы можете делать все операции с ней локально: просматривать журнал, создавать и объединять другие ветки. 10. Как и зачем можно игнорировать некоторые файлы при commit? Нет проблем если шаблон

для игнорирования подходит для файла под контролем версий, или вы добавили файл, который игнорируется. Шаблоны не имеют никакого эффекта на файлы под контролем версий, они только определяют показываются неизвестные файлы, или просто игнорируются. Файл git rignore обычно должен быть под контролем версий, что бы новые кошии ветки видели такие же шаблоны: git add • gitignore git commit -m ” Добавлены шаблоны для игнорирования”. Многие деревья с исходным кодом содержат файлы, которые не нужно хранить под контролем версий, папример, резервные файлы текстового редактора, объектные файлы и собранные программы. Вы можете просто не добавлять их, но они всегда будут обнаруживаться как неизвестные. Вы также можете сказать лг игнорировать их добавив их в файл в корне рабочего дерева. Этот файл содержит список шаблонов файлов, по одному в каждой строчке. Обычное содержимое может быть таким: *о* ~ * tm * ру [со] Если шаблон содержит слеш, то он будет сопоставлен с полным путем начиная от корня рабочего дерева; иначе он сопоставляется только с именем файла. Таким образом пример выше игпорирует файлы с расширением .о во всех подкаталогах, но пример ниже игнорирует только config.h в корне рабочего дерева и HTML файлы в каталоге doc/: /config.h doc/.*html Для получения списка файлов которые игнорируются и соответствующих им шаблонов используйте команду git ignored : $ git ignored config.h/config.h configure.in*~ ~ S

4 Выводы

Я научилася работать с github и создавать в github каталоги и репозитории, освоила основные умения по работе с git.

Список литературы

1. GNU Bash Manual [Electronic resource]. Free Software Foundation, 2016. URL: https://www.gnu.org/software/bash/manual/.
2. Newham C. Learning the bash Shell: Unix Shell Programming. O'Reilly Media, 2005. 354 p.
3. Zarrelli G. Mastering Bash. Packt Publishing, 2017. 502 p.
4. Robbins A. Bash Pocket Reference. O'Reilly Media, 2016. 156 p.
5. Таненбаум Э. Архитектура компьютера. 6-е изд. СПб.: Питер, 2013. 874 p.
6. Таненбаум Э., Бос Х. Современные операционные системы. 4-е изд. СПб.: Питер, 2015. 1120 p.