Федеральное государственное автономное образовательное учреждение высшего образования **«Национальный исследовательский университет ИТМО»**

Факультет Программной Инженерии и Компьютерной Техники

**Лабораторная работа по дисциплине «Методы и средства программной инженерии» №2**

Вариант: 9832

Преподаватель:   
Абузов Ярослав Александрович

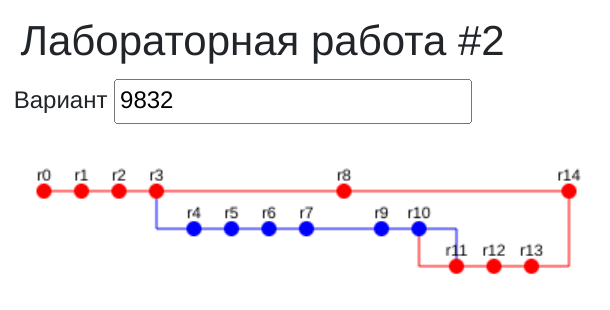
Выполнил: Васильченко Роман

Группа: Р32081

Санкт-Петербург, 2023г

# Задание

Сконфигурировать в своём домашнем каталоге репозитории svn и git и загрузить в них начальную ревизию файлов с исходными кодами



Воспроизвести последовательность команд для систем контроля версий svn и git, осуществляющих операции над исходным кодом, приведённые на блок-схеме.

При составлении последовательности команд необходимо учитывать следующие условия:

Цвет элементов схемы указывает на пользователя, совершившего действие (красный - первый, синий - второй).

Цифры над узлами - номер ревизии. Ревизии создаются последовательно.

Необходимо разрешать конфликты между версиями, если они возникают.

# Инициализация

## Git

#!/bin/bash

# Настройка пользователей

git config --global user.name "User1"

git config --global user.email "user1@example.com"

# Настройка .gitconfig

echo -e "[merge]\n tool = nano" >> .gitconfig

# Создание репозитория и начальная ревизия (r0)

git init

git add .gitconfig .gitignore

touch file.txt

git add file.txt

git commit -m "Initial commit (r0)"

## svn

#!/bin/bash

svn co <URL-адрес репозитория> --username <имя пользователя> --password <пароль>

svn mkdir <URL-адрес директории> -m "Creating trunk"

cd trunk

svn mkdir branches -m "Creating branches directory"

svn mkdir tags -m "Creating tags directory"

# Реализация кода с использованием Git

#!/bin/bash

# Repl stuff

nix-env -iA nixpkgs.nano

# Настройка пользователей

git config --global user.name "User1"

git config --global user.email "user1@example.com"ф

# Настройка .gitconfig

echo -e "[merge]\n tool = nano" >> .gitconfig

# Создание репозитория и начальная ревизия (r0)

rm -rf .git src/

git init

git checkout -b branch1

git add .gitconfig .gitignore

mkdir src

cp -f commits/commit0/\* src/

git add .

git commit -m "Initial commit (r0)"

# Ревизии r1-r3 (пользователь 1)

cp -f commits/commit1/\* src/

git add .

git commit -m "Revision 1 (r1)"

cp -f commits/commit2/\* src/

git add .

git commit -m "Revision 2 (r2)"

cp -f commits/commit3/\* src/

git add .

git commit -m "Revision 3 (r3)"

# Ревизии r4-r7 (пользователь 2)

git checkout -b branch2

cp -f commits/commit4/\* src/

git add .

git commit --author="User2 <user2@example.com>" -m "Revision 4 (r4)"

cp -f commits/commit5/\* src/

git add .

git commit --author="User2 <user2@example.com>" -m "Revision 5 (r5)"

cp -f commits/commit6/\* src/

git add .

git commit --author="User2 <user2@example.com>" -m "Revision 6 (r6)"

cp -f commits/commit7/\* src/

git add .

git commit --author="User2 <user2@example.com>" -m "Revision 7 (r7)"

# Ревизии r8 (пользователь 1)

git checkout branch1

cp -f commits/commit8/\* src/

git add .

git commit -m "Revision 8 (r8)"

# Ревизии r9-r10 (пользователь 2)

git checkout branch2

cp -f commits/commit9/\* src/

git add .

git commit --author="User2 <user2@example.com>" -m "Revision 9 (r9)"

cp -f commits/commit10/\* src/

git add .

git commit --author="User2 <user2@example.com>" -m "Revision 10 (r10)"

# Ревизии r11

git checkout -b branch3

# Создание файла для ревизии r11 с участием обоих пользователей

cp -f commits/commit11/\* src/

git add .

git rebase branch2

git commit -m "Revision 11 (r11)"

# Ревизии r12-r13 (пользователь 1)

cp -f commits/commit12/\* src/

git add .

git commit -m "Revision 12 (r12)"

# cp -f commits/commit13/\* src/

# git add .

# git commit -m "Revision 13 (r13)"

# Мердж ревизии r8 с r13 (получение r14)

git checkout branch1

git merge branch3 -m "Merging r8 with r13 (r14)"

if [[ $(git status --porcelain | grep "^UU") ]]; then

echo "There are merge conflicts. Please resolve them manually."

while true; do

# Ждем, пока пользователь исправит конфликты вручную

read -p "Press 'c' to continue after resolving conflicts, or 'a' to abort: " input

case $input in

[Cc]\* )

# Продолжаем merge после исправления конфликтов

git add .

git commit

break;;

[Aa]\* )

# Отменяем merge в случае отмены пользователем

git merge --abort

exit;;

\* ) echo "Please answer 'c' to continue or 'a' to abort.";;

esac

done

fi

# # git merge branch3 -Xours -m "Merging r8 with r13 (r14)" # Эта опция сохранит вашу версию файла (ветку, из которой выполняется слияние) в случае конфликта

# # git merge branch3 -Xtheirs -m "Merging r8 with r13 (r14)" # Эта опция сохранит версию файла из ветки, с которой выполняется слияние, в случае конфликта

# Реализация кода с использованием svn

#!/bin/bash

svn co <URL-адрес репозитория> --username <имя пользователя> --password <пароль>

svn mkdir <URL-адрес директории> -m "Creating trunk"

cd trunk

svn mkdir branches -m "Creating branches directory"

svn mkdir tags -m "Creating tags directory"

touch file.txt

svn add file.txt

svn commit -m "Initial commit (r0)" --with-revprop svn:author=User1

svn copy <URL-адрес trunk> <URL-адрес branches/branch1> -m "Creating branch1"

svn switch <URL-адрес branches/branch1>

echo "Change1" > file.txt

svn add file.txt

svn commit -m "Revision 1 (r1)" --with-revprop svn:author=User1

echo "Change2" > file.txt

svn commit -m "Revision 2 (r2)" --with-revprop svn:author=User1

echo "Change3" > file.txt

svn commit -m "Revision 3 (r3)" --with-revprop svn:author=User1

svn copy <URL-адрес branches/branch1> <URL-адрес branches/branch2> -m "Creating branch2"

svn switch <URL-адрес branches/branch2>

echo "Change4" > file.txt

svn commit -m "Revision 4 (r4)" --with-revprop svn:author=User2

echo "Change5" > file.txt

svn commit -m "Revision 5 (r5)" --with-revprop svn:author=User2

echo "Change6" > file.txt

svn commit -m "Revision 6 (r6)" --with-revprop svn:author=User2

echo "Change7" > file.txt

svn commit -m "Revision 7 (r7)" --with-revprop svn:author=User2

svn switch <URL-адрес branches/branch1>

echo "Change8" > file.txt

svn commit -m "Revision 8 (r8)" --with-revprop svn:author=User1

svn switch <URL-адрес branches/branch2>

echo "Change9" > file.txt

svn commit -m "Revision 9 (r9)" --with-revprop svn:author=User2

echo "Change10" > file.txt

svn commit -m "Revision 10 (r10)" --with-revprop svn:author=User2

svn copy <URL-адрес branches/branch2> <URL-адрес branches/branch3> -m "Creating branch3"

svn switch <URL-адрес branches/branch3>

echo "Change11" > file.txt

svn commit -m "Collaborative commit r11 by User1 and User2" --with-revprop svn:author=User1

echo "Change12" > file.txt

svn commit -m "Revision 12 (r12)" --with-revprop svn:author=User2

echo "Change13" > file.txt

svn commit -m "Revision 13 (r13)" --with-revprop svn:author=User2

svn switch <URL-адрес branches/branch1>

svn merge <URL-адрес branches/branch3> -c <номер ревизии> --accept=mine-full -m "Merging r8 with r13 (r14)" --with-revprop svn:author=User1

# Вывод

В процессе выполнения лабораторной работы были настроены репозитории svn и git в домашнем каталоге пользователя, загружены начальные ревизии файлов с исходными кодами, а также выполнены операции над исходным кодом в соответствии с блок-схемой. Были изучены основные команды svn и git, а также способы разрешения конфликтов. Практическая работа позволила лучше понять принципы работы систем контроля версий и их практическое применение в различных сценариях разработки.