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

Факультет Программной Инженерии и Компьютерной Техники

Лабораторная работа №2
по дисциплине **«Основы программной инженерии»**

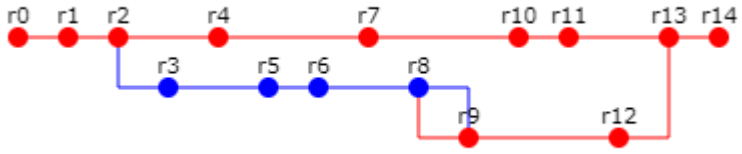
Вариант: **1571**

Преподаватель:
Цопа Евгений Алексеевич

Выполнил: Барсуков Максим
Группа: P3215

Санкт-Петербург, 2024

1. Задание



Сконфигурировать в своём домашнем каталоге репозитории svn и git и загрузить в них начальную ревизию файлов с исходными кодами (в соответствии с выданным вариантом).

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.

2. Реализация с использованием Git

```
#!/bin/bash

# Очистка ненужных данных и backup .git
cp -r .git .rotter
cp .gitignore .rotterignore

rm -rf .git
rm -f .gitignore
echo "- Создан бэкап .git"

find src -type f ! -name '.keep' -delete
echo "- src очищен"


# Создание репозитория и начальная ревизия (r0)
git init
echo "- git init"


# Настройка .git/config
echo -e "\n[merge]\n\ttool = nano" >> .git/config


# Настройка пользователей
git config user.name "red"
git config user.email "red@example.com"
echo "- Пользователь red создан"

```

```
git checkout -b branch1

# Новый .gitignore {
echo ".resources" > .gitignore
echo ".rotter" >> .gitignore
echo "commits" >> .gitignore
echo "docs" >> .gitignore
echo "src/.keep" >> .gitignore
echo ".editorconfig" >> .gitignore
echo ".rotterignore" >> .gitignore
echo "git.sh" >> .gitignore
echo "svn.sh" >> .gitignore
echo ".gitattributes" >> .gitignore
echo "LICENSE" >> .gitignore
echo "README.md" >> .gitignore
echo "return-my-git-plz.sh" >> .gitignore
git add .gitignore
echo "- Новый .gitignore создан"
# }

# Ревизия r0 (пользователь 1) {
unzip -o commits/commit0.zip -d src
git add .
git commit -m "Initial commit (r0)"
echo "- Коммит 0 (red)"
# }


# Ревизии r1-r2 (пользователь 1) {
unzip -o commits/commit1.zip -d src
git add .
git commit -m "Revision 1 (r1)"
echo "- Коммит 1 (red)"

unzip -o commits/commit2.zip -d src
git add .
git commit -m "Revision 2 (r2)"
echo "- Коммит 2 (red)"
# }


# Ревизия r3 (пользователь 2) {
git checkout -b branch2

unzip -o commits/commit3.zip -d src
git add .
git commit --author="blue <blue@example.com>" -m "Revision 3 (r3)"
echo "- Коммит 3 (blue)"
# }


# Ревизия r4 (пользователь 1) {
git checkout branch1
```

```
83.
84. unzip -o commits/commit4.zip -d src
85. git add .
86. git commit -m "Revision 4 (r4)"
87. echo "- Коммит 4 (red)"
88. # }
89.
90.
91. # Ревизии r5-r6 (пользователь 2) {
92. git checkout branch2
93.
94. unzip -o commits/commit5.zip -d src
95. git add .
96. git commit --author="blue <blue@example.com>" -m "Revision 5 (r5)"
97. echo "- Коммит 5 (blue)"
98.
99. unzip -o commits/commit6.zip -d src
100.      git add .
101.      git commit --author="blue <blue@example.com>" -m "Revision 6 (r6)"
102.      echo "- Коммит 6 (blue)"
103.      # }
104.
105.
106.      # Ревизия r7 (пользователь 1) {
107.      git checkout branch1
108.
109.      unzip -o commits/commit7.zip -d src
110.      git add .
111.      git commit -m "Revision 7 (r7)"
112.      echo "- Коммит 7 (red)"
113.      # }
114.
115.
116.      # Ревизия r8 (пользователь 2) {
117.      git checkout branch2
118.
119.      unzip -o commits/commit8.zip -d src
120.      git add .
121.      git commit --author="blue <blue@example.com>" -m "Revision 8 (r8)"
122.      echo "- Коммит 8 (blue)"
123.      # }
124.
125.
126.      # Ревизия r9 {
127.      git checkout -b branch3
128.
129.      # Создание файла для ревизии r9 с участием обоих пользователей
130.      unzip -o commits/commit9.zip -d src
131.      git add .
132.      git commit -m "Revision 9 (r9)"
133.      echo "- Коммит 9 (red)"
134.
135.      git checkout branch2
```

```
136.      git merge --ff-only branch3
137.      # }
138.
139.
140.      # Ревизии r10-r11 (пользователь 1) {
141.      git checkout branch1
142.
143.      unzip -o commits/commit10.zip -d src
144.      git add .
145.      git commit -m "Revision 10 (r10)"
146.      echo "- Коммит 10 (red)"
147.
148.      unzip -o commits/commit11.zip -d src
149.      git add .
150.      git commit -m "Revision 11 (r11)"
151.      echo "- Коммит 11 (red)"
152.      # }
153.
154.
155.      # Ревизии r12 {
156.      git checkout branch3
157.
158.      unzip -o commits/commit12.zip -d src
159.      git add .
160.      git commit -m "Revision 12 (r12)"
161.      echo "- Коммит 12 (red)"
162.      # }
163.
164.
165.      # Мердж ревизии r11 с r12 # {
166.      git checkout branch1
167.
168.      # Эта опция сохранит вашу версию файла (ветку, из которой выполняется сл
          ияние) в случае конфликта
169.      # git merge --no-commit branch3 -Xours
170.
171.      # Эта опция сохранит версию файла из ветки, с которой выполняется слияни
          е, в случае конфликта
172.      # git merge --no-commit branch3 -Xtheirs
173.
174.      git merge --no-commit branch3
175.
176.
177.      ##
178.      ## ИСПРАВЛЕНИЕ КОНФЛИКТА ВРУЧНУЮ
179.      ##
180.
181.      git add .
182.      echo "- Слияние r11 и r12"
183.
184.      # Отменяем merge в случае отмены
185.      # git merge --abort
186.
```

```
187.
188.      # Ревизии r13-r14 (пользователь 1) {
189.      unzip -o commits/commit13.zip -d src
190.      git add .
191.      git commit -m "Revision 13 (r13)"
192.      echo "- Коммит 13 (red)"
193.
194.      unzip -o commits/commit14.zip -d src
195.      git add .
196.      git commit -m "Revision 14 (r14)"
197.      echo "- Коммит 14 (red)"
198.      # }
199.
200.      # Вывод графа
201.      git log --graph --abbrev-commit --decorate --
   format=format:'%C(bold blue)%h%C(reset) - %C(bold green)(%ar)%C(reset) %C(white
   )%s%C(reset) %C(dim white)- %an%C(reset)%C(auto)%d%C(reset)' --all
```

3. Реализация с использованием SVN

```
1. #!/bin/bash
2.
3. # Создание локального репозитория
4. svnadmin create repo
5. REPO_URL="file://$(pwd)/repo"
6.
7.
8. # Создаём структуру проекта
9. cd repo
10. svn mkdir -m "project structure" $REPO_URL/trunk $REPO_URL/branches
11. cd ..
12.
13. # Создание рабочей копии
14. svn checkout $REPO_URL/trunk/ wc
15. cd wc
16.
17.
18. # Начальная ревизия (пользователь 1) {
19. unzip -o ../commits/commit0.zip -d .
20. svn add *
21. svn commit -m "Initial commit (r0)" --username=red
22. # }
23.
24.
25. # Ревизия r1 (пользователь 1) {
26. # Удаляем файлы из рабочей директории
27. # (если это не сделать, то `... is already under version control`)
28. svn rm *
29. unzip -o ../commits/commit1.zip -d .
30. svn add *
31. svn commit -m "Revision 1 (r1)" --username=red
32. # }
```

```
33.
34.
35.# Ревизия r2 (пользователь 1) {
36.svn rm *
37.unzip -o ../commits/commit2.zip -d .
38.svn add *
39.svn commit -m "Revision 2 (r2)" --username=red
40.# }
41.
42.
43.# Ревизия r3 (пользователь 2) {
44.svn copy $REPO_URL/trunk $REPO_URL/branches/branch2 -m "Creating branch2"
45.svn switch $REPO_URL/branches/branch2
46.
47.svn rm *
48.unzip -o ../commits/commit3.zip -d .
49.svn add *
50.svn commit -m "Revision 3 (r3)" --username=blue
51.# }
52.
53.
54.# Ревизия r4 (пользователь 1) {
55.svn switch $REPO_URL/trunk
56.svn rm *
57.unzip -o ../commits/commit4.zip -d .
58.svn add *
59.svn commit -m "Revision 4 (r4)" --username=red
60.# }
61.
62.
63.# Ревизии r5-r6 (пользователь 2) {
64.svn switch $REPO_URL/branches/branch2
65.
66.svn rm *
67.unzip -o ../commits/commit5.zip -d .
68.svn add *
69.svn commit -m "Revision 5 (r5)" --username=blue
70.
71.svn rm *
72.unzip -o ../commits/commit6.zip -d .
73.svn add *
74.svn commit -m "Revision 6 (r6)" --username=blue
75.# }
76.
77.
78.# Ревизия r7 (пользователь 1) {
79.svn switch $REPO_URL/trunk
80.svn rm *
81.unzip -o ../commits/commit7.zip -d .
82.svn add *
83.svn commit -m "Revision 7 (r7)" --username=red
84.# }
85.
```

```
86.
87. # Ревизия r8 (пользователь 2) {
88. svn switch $REPO_URL/branches/branch2
89.
90. svn rm *
91. unzip -o ../commits/commit8.zip -d .
92. svn add *
93. svn commit -m "Revision 8 (r8)" --username=blue
94. # }
95.
96. svn update
97.
98. # Ревизия r9 (пользователь 1) {
99. svn copy $REPO_URL/branches/branch2 $REPO_URL/branches/branch3 -
    m "Creating branch3"
100.      svn switch $REPO_URL/branches/branch3
101.
102.      svn rm * --force
103.
104.      svn merge $REPO_URL/branches/branch2
105.
106.      unzip -o ../commits/commit9.zip -d .
107.      svn add *
108.      svn commit -m "Revision 9 (r9)" --username=red
109.      # }
110.
111.      svn update
112.
113.      # Ревизии r10-r11 (пользователь 1) {
114.      svn switch $REPO_URL/trunk
115.
116.      svn rm *
117.      unzip -o ../commits/commit10.zip -d .
118.      svn add *
119.      svn commit -m "Revision 10 (r10)" --username=red
120.
121.      svn rm *
122.      unzip -o ../commits/commit11.zip -d .
123.      svn add *
124.      svn commit -m "Revision 11 (r11)" --username=red
125.      # }
126.
127.
128.      # Ревизия r12 (пользователь 1) {
129.      svn switch $REPO_URL/branches/branch3
130.
131.      svn rm *
132.      unzip -o ../commits/commit12.zip -d .
133.      svn add *
134.      svn commit -m "Revision 12 (r12)" --username=red
135.      # }
136.
137.
```

```
138.      svn update
139.
140.      # Мердж ревизии r11 с r12 {
141.      svn switch $REPO_URL/trunk
142.      svn merge $REPO_URL/branches/branch3
143.
144.      nano Lab4.java
145.      svn resolved Lab4.java
146.      # }
147.
148.      # Ревизии r13-r14 (пользователь 1) {
149.      svn rm * --force
150.      unzip -o ../commits/commit13.zip -d .
151.      svn add *
152.      svn commit -m "Revision 13 (r13)" --username=red
153.
154.      svn rm *
155.      unzip -o ../commits/commit14.zip -d .
156.      svn add *
157.      svn commit -m "Revision 14 (r14)" --username=red
158.      # }
159.
160.      svn update
```

4. Вывод

В ходе выполнения лабораторной работы я улучшил свои навыки владения системой контроля версий Git, а также познакомился с Subversion. Во время выполнения 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.