Практическое занятие №1
Создание и изучение возможностей репозитория проекта.

Цель работы: ознакомиться с возможностями СКВ Git и научиться создавать репозиторий.

Норма времени: 2 часа.

Оборудование: персональный компьютер, СКВ Git.

Ход работы

Задачи. В ходе практического занятия будет изучено создание репозитория с помощью

Git. Git — это набор консольных утилит, которые отслеживают и фиксируют изменения в файлах (чаще всего речь идет об исходном коде программ, но можно использовать его для любых файлов).

С его помощью можно откатиться на более старую версию вашего проекта, сравнивать, анализировать, сливать изменения и многое другое, т.е. проводить контроль версий (или управление версиями). Существуют различные системы для контроля версий, например SVN, Mercurial, Perforce, CVS, Bitkeeper и др.

Git является распределенным, т.е. не зависит от одного центрального сервера, на котором хранятся файлы. Вместо этого он работает полностью локально, сохраняя данные в папках на жестком диске, которые называются репозиторием. Тем не менее вы можете хранить копию репозитория онлайн, это облегчает работу над одним проектом для нескольких людей. Для этого используются различные сайты, например github и bitbucket.

Практическая часть. Исследование выполняется в несколько этапов.

1. Первоначально необходимо установить Git на компьютер.

Скачайте exe-файл инсталлятора со страницы проекта на GitHub и запустите его: http://msysgit.github.com/. После установки у вас появится как консольная версия (включающая SSH-клиент, который пригодится позднее), так и стандартная графическая.

Git необходимо использовать только из командной оболочки, входящей в состав msysGit, потому что так вы сможете запускать сложные команды, приведенные в примерах. Командная оболочка Windows использует иной синтаксис, из-за чего примеры в ней могут работать некорректно. Для начинающих разработчиков клиент с графическим интерфейсом (например, GitHub Desktop и Sourcetree) будет полезен, но тем не менее знать команды очень важно.

2. После установки нужно добавить настройки. Настроим самые важные опции — имя пользователя и адрес электронной почты.

Откройте терминал и запустите команды:

```
git config --global user.name "My Name"
git config --global user.email myEmail@example.com
1
2
git config --global user.name "My Name"
```

```
git config --global user.email myEmail@example.com
```

Теперь каждое действие будет отмечено именем и почтой. Таким образом, пользователи всегда будут в курсе, кто за какие изменения отвечает, это обеспечивает порядок

3. Создать новый репозиторий. Как мы уже отмечали, Git хранит свои файлы и историю прямо в папке проекта. Чтобы создать новый репозиторий, нужно открыть терминал, зайти в папку проекта и выполнить команду init. Это включит приложение в этой конкретной папке и создаст скрытую директорию .git, где будут храниться история репозитория и настройки.

Создайте на рабочем столе папку под названием git_exercise. Для этого в окне терминала введите:

```
$ mkdir Desktop/git_exercise/
$ cd Desktop/git_exercise/
$ git init
1
2
3
$ mkdir Desktop/git_exercise/
$ cd Desktop/git_exercise/
$ git init
```

Командная строка должна содержать, например, следующее:

```
Initialized empty Git repository in /home/user/Desktop/git_exercise/.git/
1
Initialized empty Git repository in /home/user/Desktop/git_exercise/.git/
```

Это значит, что репозиторий был успешно создан, но пока он пуст. Теперь создайте текстовый файл под названием hello.txt и сохраните его в директории git_exercise.

4. Определить состояние репозитория.

Status — это еще одна важнейшая команда, которая показывает информацию о текущем состоянии репозитория: актуальна ли информация на нем, нет ли чего-то нового, что поменялось и т. д. Запуск git status на нашем недавно созданном репозитории выдаст:

```
$ git status
On branch master
Initial commit
Untracked files:
(use "git add ..." to include in what will be committed)
hello.txt
1
2
```

```
3
4
5
6
$ git status
On branch master
Initial commit
Untracked files:
(use "git add ..." to include in what will be committed) hello.txt
```

Сообщение говорит о том, что файл hello.txt неотслеживаемый. Это значит, что файл новый и система еще не знает, нужно ли следить за изменениями в файле или можно просто игнорировать его. Чтобы начать отслеживать новый файл, нужно его специальным образом объявить.

5. Подготовить файлы. В Git есть концепция области подготовленных файлов. Можно представить ее как холст, на который наносят изменения, нужные в коммите.

Коммит — состояние репозитория в определенный момент времени. Первоначально он пустой, но затем мы добавляем на него файлы (или части файлов, или одиночные строки) командой add и наконец коммитим все нужное в репозиторий (создаем слепок нужного нам состояния) командой commit.

В нашем случае у нас только один файл, так что добавим его:

```
$ git add hello.txt
1
$ git add hello.txt
```

Если нам нужно добавить все, что находится в директории, мы можем использовать:

```
$ git add -A
1
$ git add -A
```

Проверим статус снова, на этот раз мы должны получить другой ответ:

```
$ git status
On branch master
Initial commit
Changes to be committed:
(use "git rm --cached ..." to unstage)
new file: hello.txt
1
2
3
4
```

```
5
6
$ git status
On branch master
Initial commit
Changes to be committed:
(use "git rm --cached ..." to unstage)
new file: hello.txt
```

Файл готов к коммиту. Сообщение о состоянии также говорит о том, какие изменения относительно файла были проведены в области подготовки, в данном случае это новый файл, но файлы могут быть модифицированы или удалены.

6. Зафиксировать изменения (коммит). Чтобы зафиксировать изменения, нужно хотя бы одно изменение в области подготовки (мы только что создали его при помощи git add), после которого мы можем коммитить:

```
$ git commit -m "Initial commit."
1
$ git commit -m "Initial commit."
```

Эта команда создаст новый коммит со всеми изменениями из области подготовки (добавление файла `hello.txt`). Ключ **-m** и сообщение `"Initial commit."` — это созданное пользователем описание всех изменений, включенных в коммит. Считается хорошей практикой делать коммиты часто и всегда писать содержательные комментарии.

Примечание. *После выполнения всех действий следует представить отчет в виде скриншотов всех команд. Результаты этой практической работы будут использованы в качестве данных для практического занятия № 14.*

Контрольные вопросы:

1. Система управления версиями это?
2. На чем основана CVS?
3. Какие системы распределения контроля версий Вы знаете?
4. Охарактеризуйте систему Git.
5. Что такое клонирование репозитория?
6. Что такое синхронизация репозитория?