ВСР 2.3: Использование клиента Git в Терминале (Terminal / Git Bash / Command Prompt)

Цель работы

Исследовать возможности терминального клиента Git, показать наиболее распространённые команды, типичные сценарии работы и особенности инструмента по сравнению с GUI-клиентами.

Среда выполнения

• ОС: Windows / macOS / Linux

• Терминал: Git Bash (Windows), Command Prompt/PowerShell (Windows), Terminal (macOS/Linux)

• Версия Git: актуальная стабильная

Установка Git

• Windows: загрузка установщика с официального сайта Git и установка Git Bash и Git CMD.

• macOS: через Xcode Command Line Tools или Homebrew.

• Linux: через пакетный менеджер (apt, dnf, pacman).

Базовая настройка

```
git config --global user.name "Имя Фамилия"

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

git config --global init.defaultBranch main

git config --global core.autocrlf true   # для Windows
```

Создание локального репозитория

```
mkdir my-project

cd my-project

git init

echo "# My Project" > README.md

git add README.md

git commit -m "feat: init project with README"
```

Подключение к удалённому репозиторию

```
git remote add origin https://github.com/ORG/REPO.git
```

```
git branch -M main

git push -u origin main
```

Клонирование и ежедневный цикл

```
git clone https://github.com/ORG/REPO.git

cd REPO

git pull --rebase

git status

git add .

git commit -m "fix: исправил баг в обработчике"

git push
```

Работа с ветками

```
git switch -c feature/auth-login

# правки...

git add .

git commit -m "feat(auth): добавил форму логина"

git switch main

git pull --rebase

git merge --no-ff feature/auth-login

git push
```

Слияние и разрешение конфликтов

1) Выполнить merge/rebase

2) Открыть конфликтные файлы, вручную оставить нужные изменения

3) Обновить индекс и завершить операцию:

```
git add путь/к/файлу

git merge --continue   # или git rebase --continue
```

Stash (временное сохранение)

```
git stash push -m "быстрый черновик"

git switch main

git pull --rebase
```

```
git switch feature/auth-login

git stash list

git stash pop
```

Теги (версии релизов)

```
git tag -a v1.0.0 -m "Первый релиз"

git push origin v1.0.0
```

Поиск и диагностика

```
git log --oneline --graph --decorate --all

git blame src/app.js

git diff main...feature/auth-login

git bisect start

git bisect bad

git bisect good <хэш-хорошей-версии>
```

Исправление ошибок

• Отмена локальных изменений в файле:

```
git restore --source=HEAD -- файл
```

• Отмена индексации:

```
git reset HEAD файл
```

• Отмена последнего коммита (оставив изменения в рабочей директории):

```
git reset --soft HEAD~1
```

• Восстановление потерянных коммитов:

```
git reflog
```

Продвинутые темы

• Rebase и интерактивный rebase:

```
git rebase -i main  # для чистой истории
```

• Cherry-pick выборочных коммитов:

```
git cherry-pick <hash>
```

• Worktree для параллельной работы с несколькими ветками:

```
git worktree add ../w-main main
```

• Подписи и проверка:

```
git commit -S

git verify-commit
```

Особенности работы в разных терминалах

• Git Bash (Windows): Unix-пути (/), SSH по умолчанию, удобные алиасы и автодополнение.

• Command Prompt/PowerShell: пути вида C:\path, рекомендуется включить UTF-8 (chcp 65001) и установить Git Credential Manager.

• macOS/Linux Terminal: нативная среда UNIX, удобное использование SSH-ключей и скриптов.

Часто используемые алиасы (пример)

```
git config --global alias.st "status -sb"

git config --global alias.co "switch"

git config --global alias.br "branch -v"

git config --global alias.lg "log --graph --oneline --decorate --all"
```

Сравнение: Терминал vs GUI

+ Тонкий контроль, автоматизация, скрипты, одинаково на всех ОС.

− Кривая обучения, требуется аккуратность с командами.

GUI удобен для визуального разрешения конфликтов и ревью, но терминал универсальнее и быстрее в продвинутых сценариях.

Практическая демонстрация (пошаговый сценарий)

1. Клонировать репозиторий и создать ветку фичи.

2. Внести изменения, сделать коммит, запушить в удалённый.

3. Обновить main, выполнить merge и запушить.

```
git clone https://github.com/ORG/REPO.git

cd REPO

git switch -c feature/docs

echo "Обновил README" >> README.md

git add README.md

git commit -m "docs: обновил README"
```

```
git push -u origin feature/docs

git switch main

git pull --rebase

git merge --no-ff feature/docs

git push
```

Типичные ошибки и как их исправлять

• "detached HEAD": создать ветку от текущего состояния:

```
git switch -c hotfix/tmp
```

• Случайный коммит в main:

```
git switch -c fix/move

git reset --hard origin/main

git cherry-pick <hash>...
```

• Конфликты CRLF/LF на Windows: core.autocrlf true и настройка .gitattributes

Заключение

Терминальный клиент Git — мощный, быстрый и кроссплатформенный инструмент. Овладение базовыми и продвинутыми командами позволяет эффективно работать с кодом, автоматизировать рутину и поддерживать чистую историю коммитов.