

A thick black L-shaped frame surrounds the text. It starts at the top left, goes right, then down, then right again at the bottom right.

ФУНКЦИОНАЛЬНОЕ И ЛОГИЧЕСКОЕ ПРОГРАММИРОВАНИЕ

Практика-0 Введение

Лукашин Антон


Высшая школа программной инженерии
Институт Компьютерных Наук и Технологий

Обо мне

- Лукашин Антон Андреевич

- Контакты

 an.lukashin@gmail.com

 +7(952)239-16-69

 rinitar

 rinitar

- Опыт работы

- До 2014 года – GGA Software Services
- До 2015 года – EPAM Systems
- По настоящее время – Center of Reactive Programming

План занятий

- Введение в Git
- Практика Prolog
- Знакомство с функциональным программированием в современных языках
 - *Java*
 - *Scala*
 - *Kotlin*
 - *JavaScript*
 - *Python*
 - *Erlang*
 - *Haskell*
 - *C++*

Форма отчетности

- ~~Посещение~~
- Сдача задач происходит через [GitHub](#)
 - *Зарегистрироваться*
 - *Прислать мне свой ник (отправить заявку на Collaboration)*
 - *Сделать ветку `firstname_secondname` от `master`*
 - *Выполнить задания (1 коммит – 1 задание)*
 - *Залить свою ветку в репозиторий*
- По результатам выполнения задачи отписаться в соответствующем разделе wiki

Что будет на занятиях?

- Ничего, если они вам не нужны
- Разбор простых вариантов задач
- Что-то интересное, если простые задачи не интересны

А что если...?

- ... я очень не хочу делать лабы
 - При наличии сертификатов по курсам (CSC, Coursera...) на соответствующую тематику – что-то может быть перезачтено
 - Можно удивить меня интересным проектом в тематике курса
- ... я ничего не сделал к концу семестра
 - Зачета не будет. Решение будет принимать кафедра и Соловьев И.П.
- ... я просто нашел такой же вариант и скопировал решение
 - Если попадетесь – буду «кошмарить» и проверять понимание темы
 - Если нет – ну, я же об этом не узнал...

Введение в Git

- Материалы для изучения:
 - [Git Book](#) главы 1-3
 - [Getting Git Right with Atlassian](#)
 - [Git Handbook by GitHub](#)
- Темы:
 - Основы устройства Git
 - Базовые команды Git
 - Паттерны использования

История Git

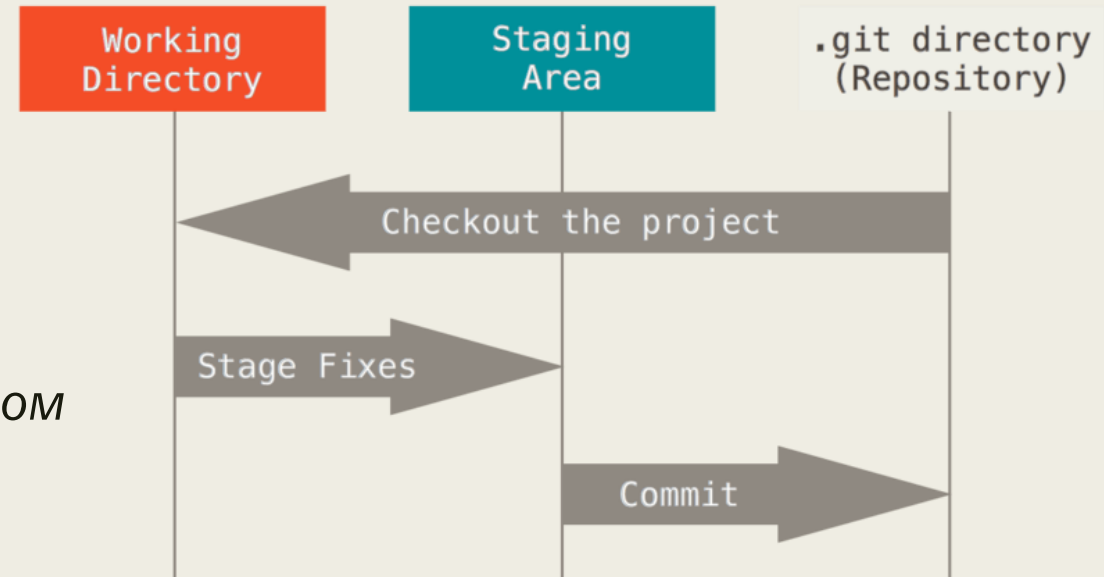
- Ядро Linux — это очень большой открытый проект (~5 млн. строк на 2002, сейчас ~ 16 млн. строк).
 - До 2002 изменения вносились в код путем приёма патчей и архивирования версий
 - В 2002 году проект перешёл на проприетарную и платную РСКВ BitKeeper
- В 2005 году отношения между сообществом разработчиков ядра Linux и компанией разрабатывавшей BitKeeper испортились, и право бесплатного пользования продуктом было отменено. Это подтолкнуло разработчиков Linux (и в частности Линуса Торвальдса,) разработать собственную систему,.
- Основные требования к новой системе были следующими:
 - Скорость
 - Простота дизайна
 - Поддержка нелинейной разработки (тысячи параллельных веток)
 - Полная распределенность
 - Возможность эффективной работы с такими большими проектами как ядро Linux (как по скорости, так и по размеру данных)
- GIT - unpleasant person (неприятный человек)

Особенности Git

- Слепки, а не дельты
 - *Новая версия файла хранится целиком*
 - *Если файл не изменился, то записывается ссылка*
- Локальные операции
 - *Может работать без внешней сети*
 - *Высокая скорость 90% операций*
- Контроль целостности данных (SHA-1)
- Сжатие данных
 - *Данные хранятся в виде BLOB*
 - *Данные сжимаются (алгоритмы разные: deflate, lzma)*
- Архивирование версий

Состояния файлов

- «Измененный» (changed)
 - *Изменения в рабочей директории*
- «Подготовленный» (added)
 - *Файл в staging area и готов к коммиту*
- «Зафиксированный» (committed)
 - *Файл сохраняется в коммите в локальном репозитории (.git dir)*
- «Спрятанный» (stashed)
 - *Изменения отменяются, сохраняясь в отдельном стеке*



Настройка Git

- Управлять настройками Git позволяет утилита `git config`
- Ключи
 - *--global* локальные настройки пользователя `~/.gitconfig`
 - *--system* глобальные настройки `/etc/gitconfig`
 - Без ключа настройки для текущего репозитория
- Базовые операции
 - `$ git config --global user.name "Great Director"`
 - `$ git config --global user.email greate.dir@lair_of_great_dir.com`
 - `$ git config --global core.editor vim`
 - `$ git config --global merge.tool vimdiff`
 - `$ git config --list`

Детский сад

- git
 - *clone*
 - *status*
 - *add*
 - *commit*
 - *checkout (-b)*
 - *branch*
 - *pull*
 - *push*

Начальная школа

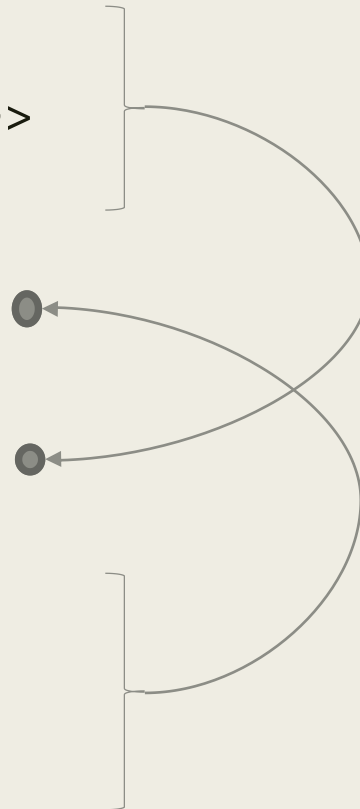
- git
 - *init* (*--bare*)
 - *reset* (*--hard*, *--soft*)
 - *revert*
 - *rebase*
 - *rm* (*--cached*)
 - *merge*
 - *fetch*
 - *stash* (*list*, *pop*, *apply*)
 - *log*
 - *show*
 - *blame*
 - *diff*
 - *tag*

Git flow - branches

- Расширение Feature Branch
- Лучше подходит для больших проектов
- Ветки:
 - *master* - содержит актуальную версию, доставленную на *production* сервер или готовую к доставке. А также тэги версий
 - *development* - содержит актуальную версию для следующего релиза, собранную из *feature branch*
 - *feature branch (<issue-ID>)* – заводится для работы над задачей
 - *release-<date or version>* - содержит в себе релиз-кандидат, который будет смержен в *master* после тестирования
 - *hotfix-<date or version>* - содержит исправления для ветки *master*

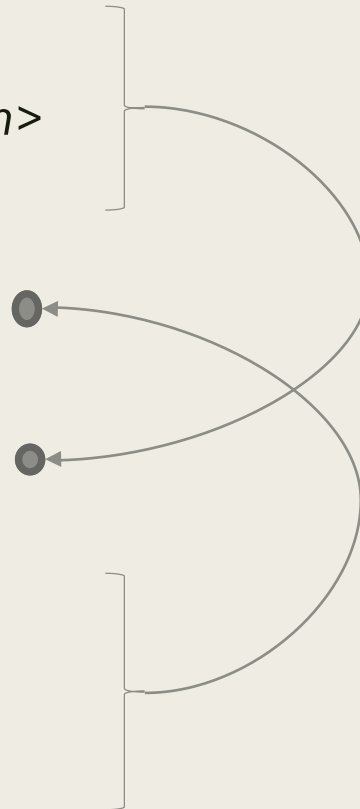
Git flow – daily merge

- День начинается с команд:
 - *git fetch*
 - *git merge origin/<feature_branch>*
 - *git merge origin/development*
- В течение дня:
 - *Commit* изменений (или *stash*)
 - Слияние коммитов (если нужно)
 - Обновление
- День заканчивается командами:
 - *git status*
 - *git add*
 - *git commit*
 - *git push origin/<feature_branch>*



Git flow – daily rebase

- День начинается с команд:
 - *git fetch*
 - *git rebase origin/<feature_branch>*
 - *git rebase origin/development*
- В течение дня:
 - *Commit* изменений (или *stash*)
 - *Слияние коммитов*
 - *Обновление*
- День заканчивается командами:
 - *git status*
 - *git add*
 - *git commit*
 - *git push origin/<feature_branch>*



Git pearls

- Работа с большими репозиториями
- Немного магии
 - *Git log*
 - *Git diff*
 - *Git blame / annotate*
 - *Git grep*
 - *Git checkout*
 - *Git rebase*
 - *Git cherry-pick*

Работа с большими репозиториями

- Проект с большой историей
 - *shallow clone* – `git clone -depth 20, --shallow-since=<date>`
 - `git clone URL --branch branch_name --single-branch`
 - *filter-branch* – `git filter-branch -tree-filter 'rm -rf smth' (needs freeze!)`
- Большое количество бинарных файлов
 - *Не делать так*
 - *Тюнинг* – (`git gc, delta off, zero-compression, core.bigFileThreshold`)
 - *Разбить на submodules*
 - *Использовать расширения (git-bigfiles, git-annex)*

Немного магии - ковыряки

- `git log`
 - `--stat` – количество изменений, файлов, строк
 - `--summary` – переименования и права
 - `<file>` -конкретный файл
 - `--since=«1 day 2 hours»`
 - `<tag>` - начиная с тэга
 - `--pretty=oneline/short/full/format`
- `git diff`
 - `--cached`
 - `-HEAD^` сравнение с предпоследним коммитом
- `git blame` – строки и люди
 - `-L 2,+3 <file> -3` строки начиная со второй
- `git annotate` – строки и коммиты

Немного магии - изменялки

- `git grep`
 - `<smth>` - поиск по проекту
 - `-c <smth>` -число вхождений
 - `-e <smth-1> --and -e <smth-2>`
- `git cherry` – сравнение по коммитам
 - `-pick` – воровство коммита
- `git checkout`
 - `<file>` - вернуть файл к последнему коммиту
 - `HEAD~2 <file>` -вернуть файл на 2 коммита назад

Немного магии – rebase&commit

- `git rebase`
 - *<branch>* - на ветку будет наложена текущая
 - `--continue`
 - `--skip`
 - `--abort`
 - *-i HEAD^4 / hash*
- `git commit`
 - `--amend` – слияние коммитов
 - `--squash` – коммит в предыдущий
 - `--fixup` – коммит в предыдущий но без *log message*

Git 2.*

- `Worktree (git worktree add [-f] [--detach] [-b <new-branch>] <path> [<branch>])` – создает видимость копии-репозитория
- `show-branch --topics <branch>` - коммиты, отсутствующие в ветке
- `commit -cleanup=scissors`
- `tag -sort=version`
- `config -file`
- `--autostash / --no-autostash`
- `stash show / -p , --stat`
- `filter-branch` vs speed and progress bar
- `gitignore` negotiation (exclude single file from ignored dir)

За кадром

- GIT на сервере
- gitolite
- gitosys
- git hooks
- gui tools

Спасибо за внимание!

- Актуальные списки групп
- Планы по посещению
- Контакты представителей
- Координируемся
- Готовность к получению задания – наличие ветки