Технологии разработки программного обеспечения

Старший преподаватель Кафедры вычислительных систем
Елизавета Ивановна Токмàшева

email: eliz_tokmasheva@sibguti.ru

2024, 1 курс, 2 семестр

О чём курс

Система контроля версий — git

Система автоматической сборки — make

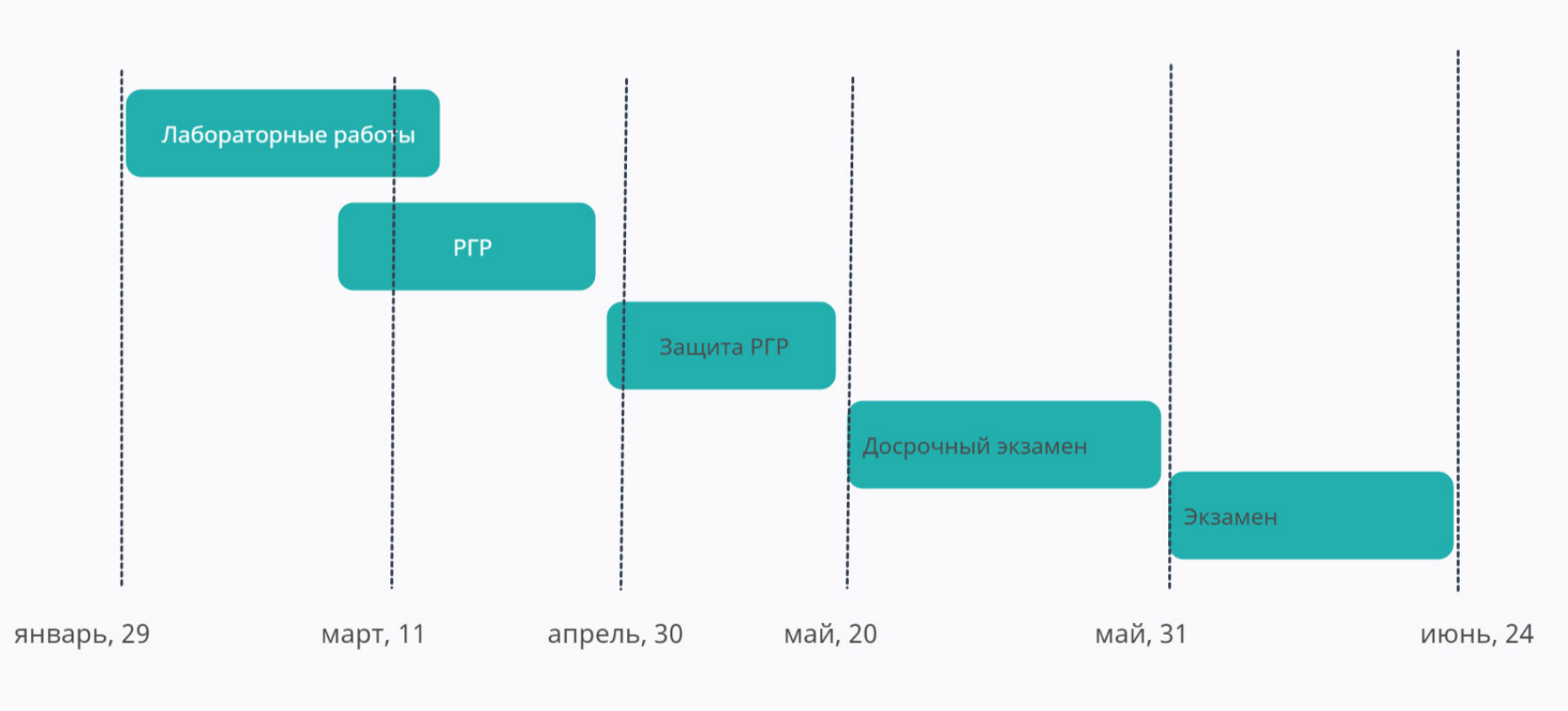
Unit-тестирование

Непрерывная интеграция (*Continuous Integration, CI*)

Создание программного продукта — РГР

Оформление кода — code-style

Двигаемся по курсу



Выход на досрочный экзамен:
сданные лабораторные + РГР,
рекомендация преподавателя практики

Не сданные лабораторные — дополнительный вопрос на экзамене по каждой лабораторной

Общение на одном языке. Терминология, основные понятия

Система контроля версий (СКВ)

Локальный репозиторий

Удалённый репозиторий

Коммит, создание коммита, сохранение изменений

Проиндексировать файл/индексация изменений

Состояние файла

Внесённое изменение

Просмотр истории изменений

Обновление локального репозитория

Обновление удаленного репозитория

Ветка, основная ветка

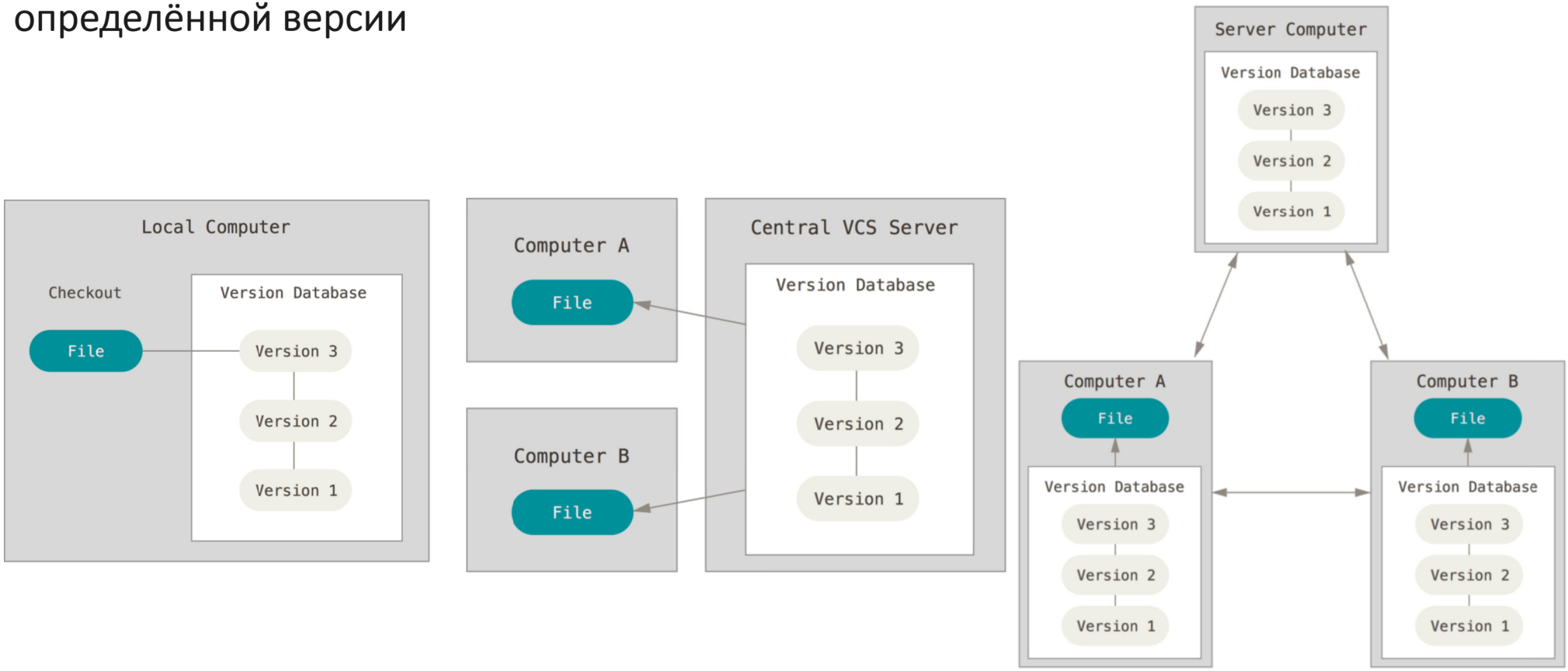
Staged-область

Staged-состояние

Рабочая область, working tree

Системы контроля версий. Какие они?

Системы контроля версий (СКВ) - это система, записывающая изменения в файл или набор файлов в течение времени и позволяющая вернуться позже к определённой версии



Локальная СКВ

Централизованная СКВ

Распределенная СКВ

https://git-scm.com/book/ru/v2/

Локальные системы контроля версий

RCS (Revision Control System, Система контроля ревизий)

Разработана в начале 1980-х годов Вальтером Тичи (Walter F. Tichy).

Особенности:

- Система позволяет хранить версии только одного файла
- Управление несколькими файлами выполняется вручную
- Отслеживаемые файлы не могут одновременно изменяться несколькими пользователями
- Нет поддержки сети

Основные команды:

check-out извлечь файл и поместить в рабочий каталог для чтения

```
$ co <filename.txt>
```

check-in добавить изменения файла

```
$ ci <filename.txt>
```


https://www.gnu.org/software/rcs/manual/rcs.html

Централизованные системы контроля версий

CVS (Concurrent Version System, Система совместных версий)

Разработана в 1986 году Диком Груном.

Особенности:

- Использует RCS формат для хранения файлов
- Позволяет управлять группой файлов расположенных в директориях
- Использует клиент-серверную архитектуру. Информация о файлах хранится на сервере
- Нет возможности сохранять версии директорий
- Перемещение или переименование файлов не подтверждено контролю версий

https://www.gnu.org/software/trans-coord/manual/cvs/html_node/
https://dickgrune.com/Programs/CVS.orig/

Централизованные системы контроля версий

CVS (Concurrent Version System, Система совместных версий)

Основные команды :

Копировать модуль в рабочую директорию

```
$ cvs checkout path-in-repo
```

Создание коммита измененного файла

```
$ cvs commit-m  "Some changes"
```

Обновить рабочую копию, объединив зафиксированные изменения в центральном репозитории

```
$ cvs update
```


https://www.gnu.org/software/trans-coord/manual/cvs/html_node/
https://dickgrune.com/Programs/CVS.orig/

Централизованные системы контроля версий

SVN (Subversion) свободная централизованная система управления версиями.
Официально выпущена в 2004 году компанией Collabnet Inc., в настоящее время поддерживается Apache Software Foundation.

Особенности:

- Атомарное внесение изменений (commit). В случае неудачной обработки внесения коммита изменения не будет применены
- Переименование, копирование и перемещение файлов сохраняет всю историю изменений
- Эффективное хранение изменений для бинарных файлов

Основные команды SVN во много похожи на команды CVN

https://svnbook.red-bean.com/

Распределенные системы контроля версий: git

Git разработан в 2005 году Линусом Торвальдсом (создатель Linux).

Особенности:

- Распределенная система. Все копии репозитория являются равными. Разработчики могут обмениваться изменениями без объединения их в основной ветке.
- Поддержка нелинейной разработки
- Скорость работы
- Простота дизайна
- Возможность работать с большими объемами данных за счет использования алгоритма сжатия zlib

Git. Использование всего пяти команд позволит вам...

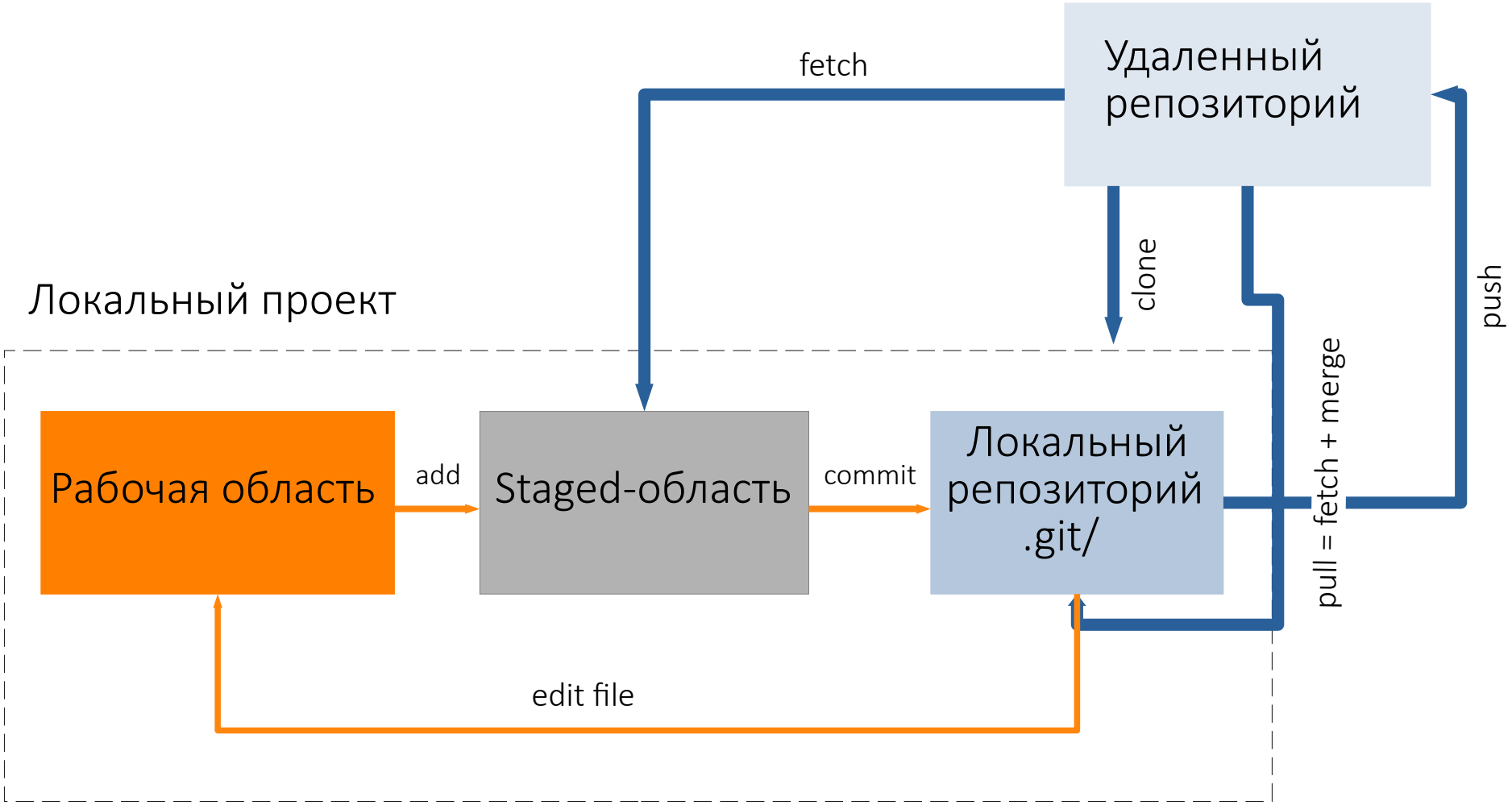
Git

...думать, что вы умеете работать с гитом.

```
$ git add
$ git commit
$ git status
$ git log
$ git push
```

Но так ли это на самом деле?

Git на локальной машине



https://git-scm.com/book/ru/v2

Git на локальной машине

Локальный проект — каталог, где ведется разработка проекта

Рабочая область — пространство, где ведется разработка

Локальный репозиторий — каталог, хранилище всех версий файлов на локальной машине

Staged-область — промежуточное пространство между рабочей областью и локальным репозиторием, для простоты понимания обозначим его как «буфер»

Удалённый репозиторий — хранилище данных на сервере (github, gitlab, etc)

Git на локальной машине. Основные команды

- Создание локального репозитория
  $ git init

Создает в текущем каталоге подкаталог с именем .git/

- Добавление (индексация) файлов под версионный контроль
  $ git add <filename>

- Сохранение изменений (создание коммита)
  $ git commit -m «Initial commit»
  (-m, message)

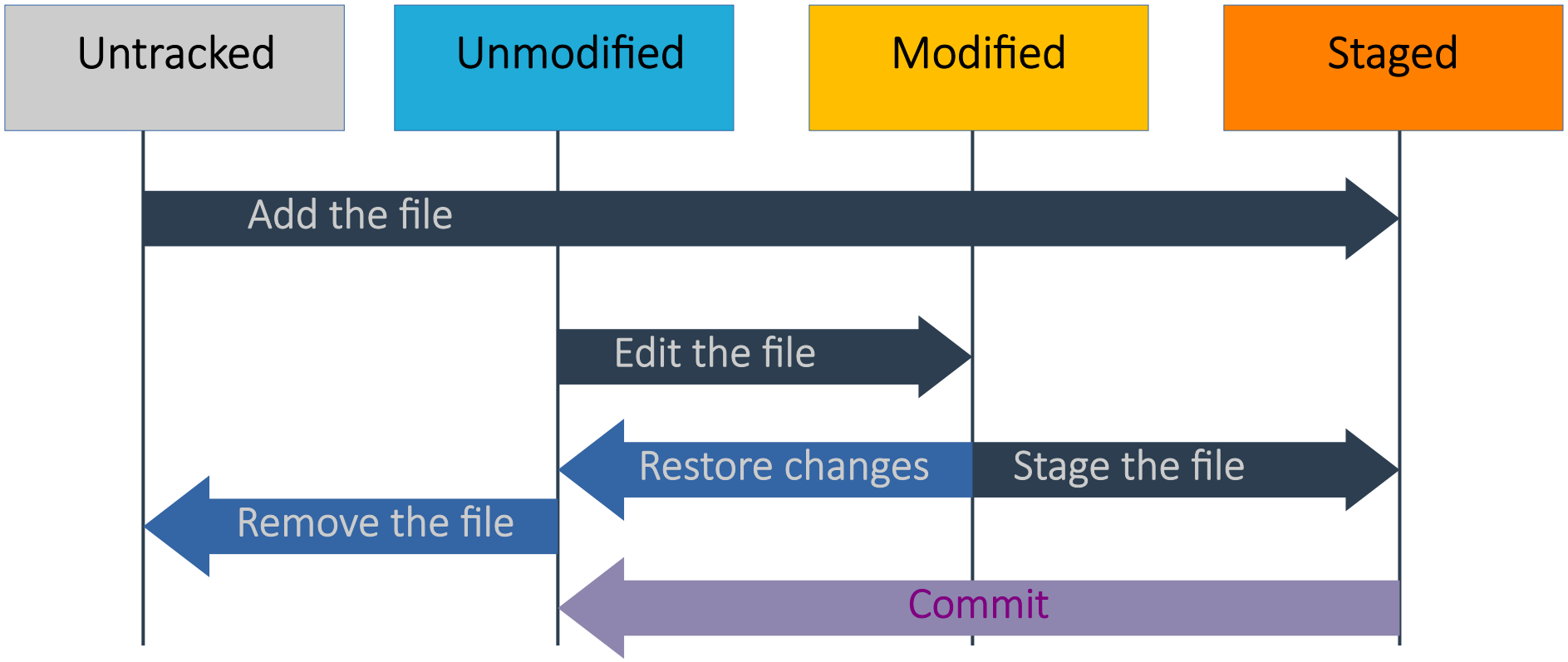
- Просмотр состояний файлов
  $ git status

- Просмотр истории изменений/список коммитов
  $ git log

- Помощь:
  $ git <verb> --help
  $ man git-<verb>

https://git-scm.com/book/ru/v2

Git на локальной машине.
Состояния файлов в репозитории



https://git-scm.com/book/ru/v2

Git на локальной машине.
Состояния файлов в репозитории

Определение состояния файлов:

```
$ git status
```

On branch master

Nothing to commit, working tree clean

После создания файла в каталоге

```
$ git status
```

Untracked files:

use "git add < file> ..." to include in what will be committed)

main.c

nothing added to commit but untracked files present (use "git add" to track)

Git на локальной машине.
Состояния файлов в репозитории

Определение состояния файлов:

```
$ git status
```

On branch master

Nothing to commit, working tree clean

После создания файла в каталоге

```
$ git status
```

Untracked files:

use "git add < file> ..." to include in what will be committed)

main.c

nothing added to commit but untracked files present (use "git add" to track)

Git на локальной машине.
Добавление файлов под версионный контроль — git add

Начать отслеживать файл:

$ git add <smthg>

```
eliz@LAPTOP-F6OV9FH5:~/ipcalc$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   Makefile

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   Makefile

eliz@LAPTOP-F6OV9FH5:~/ipcalc$ git status -s
MM Makefile
```

Git на локальной машине
Commit — объект или команда?

Коммит – снимок (snapshot) изменений во времени.

Для фиксаций изменений используется команда

```
$ git commit
```

При выполнении команды без опций, откроется редактор по умолчанию в системе, для ввода комментария к коммиту.

ВАЖНО! Основной принцип:
один коммит включает в себя одно изменение.
Изменения в каждом коммите самодостаточны и независимы.
Комментарий отражает внесенное изменение.
Комментарий пишется не для себя, а для третьих лиц.

Git на локальной машине
commit — объект СКВ

Коммит – снимок (snapshot) изменений во времени

5 main.c

```
...  ...   @@ -1,6 +1,7 @@
 1    1     #include <stdio.h>
 2         - int main(int argc, char **argv)
      2    +
      3    + int main()
 3    4     {
 4         - printf("Hello, world!");
      5    +     printf("Hello, world!\n");
 5    6         return 0;
 6    7     }
```

Git на локальной машине
commit — команда git

Для добавления комментария без открывания редактора используется опция-m (message)

$ git commit -m «Add empty function main»

$ git commit -a -m «Add empty function main»

-a,--all автоматическое индексирование всех файлов, которые были изменены или удалены. На новые файлы (неотслеживаемые) действие не распространяется.

```
eliz@LAPTOP-F6OV9FH5:~$ git status
On branch master
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        deleted:    main.c
        modified:   makefile

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .bash_history
        .bash_logout
        .bashrc
```

```
eliz@LAPTOP-F6OV9FH5:~$ git commit -a -m "Delete main.c"
[master 0d56b3c] Delete main.c
 2 files changed, 4 insertions(+), 8 deletions(-)
 delete mode 100644 main.c
eliz@LAPTOP-F6OV9FH5:~$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .bash_history
        .bash_logout
        .bashrc
```

Git на локальной машине
История изменений: log

$ git log

Выводит историю изменений в обратном хронологическом порядке

```
eliz@LAPTOP-F6OV9FH5:~/ctest$ git log
commit cd46041baf3f131fe74032967566c7321adcaa01 (HEAD -> master, origin/master, origin/HEAD)
Author: Bas van den Berg <b.van.den.berg.nl@gmail.com>
Date:   Wed Jan 13 06:21:52 2021 +0100

    copyright: update to 2021

commit cf30ad66e99ce802b36db168d5a484f620db5131
Author: Andrew Eckel <aceckel@gmail.com>
Date:   Wed Sep 16 22:07:13 2020 -0400

    Remove unused uname variable

commit 78a37432bd3c5a972872d2e9239598a751f1f5c6
Author: Andrew Eckel <aceckel@gmail.com>
Date:   Wed Sep 16 22:06:53 2020 -0400

    Compile the example tests with both a C and C++ compiler

commit f8a83d694b2d19d5d2b2945cca37dadad944c99b
Author: Andrew Eckel <aceckel@gmail.com>
Date:   Wed Aug 19 23:09:41 2020 -0400

    Add basic C++ support
```

хэш-суммма коммита