

## Shell и Bash

[illegible]

## Git и репозиторий

<h3>Локальный git</h3> <p><b>git</b> — это система контроля версий файлов. Позволяет отслеживать изменения в файлах, совместно работать над проектом и автоматически создавать резервные копии. Она широко используется в разработке программного обеспечения.</p> <p>Git работает локально на ваших компьютерах. У него есть три основные системы: рабочая директория, область暂存 и репозиторий. Вы добавляете изменения в область暂存, а затем фиксируете их в репозитории в локальной копии.</p>	<h3>Службы git-хостинга (удалённые репозитории)</h3> <p>Git позволяет хранить удалённые. Удалённые репозитории дают возможность совместно работать над проектом.</p> <p>Популярные платформы для хранения удалённых репозиторий: GitHub, GitLab и Bitbucket.</p> <p>В главе 21 мы рассмотрим ряд онлайн-репозиторий на основе GitLab.</p>	<h3>Git-команды 1</h3> <h4>Запуск проекта</h4> <ul style="list-style-type: none"> <li><b>git clone &lt;url&gt;</b>: клонирует репозиторий и создает новую ветку master (по умолчанию) через из файлов в репозитории.</li> <li><b>git init</b>: создает новую копию в локальном репозитории (файлы).</li> </ul> <h4>Промежуточные команды</h4> <ul style="list-style-type: none"> <li><b>git add</b>: помещает изменения репозитория в промежуточные файлы репозитория.</li> <li><b>git commit</b>: сохраняет конкретные изменения в файлы.</li> </ul>
--	---	---

## Vim

### Что это

Win – это – специальный текстовый редактор, который специально создан для людей, которые работают с текстом.

Для комфортной работы нужно уметь и применять и его возможности.

Понять не приходится на английском. Важно уметь и понимать как он его выдает.

### Сохранение и выход из Win

Для сохранения без выезда из программы использовать команду **Ctrl+S**

Чтобы выйти без сохранения, использовать команду **Ctrl+Q**

Чтобы сохранить, изменить и выйти из Win, использовать команду **Ctrl+S, Ctrl+Q**

### Режимы в win

**Нормальный режим**

- нормальный курсор
- нормальное выделение
- нормальное, вставка

**Режим вставки (режим редактирования текста)**

- выделение и вставка текста

**Режим ввода команд**

- ввод команд (Command window)

### Режимы

- Перевод в нормальный режим **Esc**
- в режиме вставки, или **Ctrl+I**
- в командный режим **Esc**
- **Esc** – вставить, **Ctrl+V** – вставить и выйти

### Навигация по тексту

- перемещает курсор в начало следующего слова, в **Ctrl+Left** - в начало предыдущего слова
- **Ctrl+Right** перемещает в конец слова, **Ctrl+Left** - в конец слова

### Ввод текста

- Вставить перед текущим символом
- Вставить в начало строки
- Вставить в конце текущей строки
- Вставить в конец строки
- Начать новую строку ниже текущей и войти в режим вставки
- Начать новую строку выше текущей и войти в режим вставки

### Манипулирование текстом

Win также предоставляет команды для работы с текстом

- Ctrl+Z** — отменяет текущую строку
- Ctrl+Y** — отменяет отмену, возвращает в исходное состояние
- Ctrl+U** — удаляет символ под курсором
- Чтобы переместить текст используйте **Ctrl+X** для вырезания и **Ctrl+V** для вставки

### Вывод

Win – это – специальный текстовый редактор, который специально создан для людей, которые работают с текстом.

Для комфортной работы нужно уметь и применять и его возможности.

Понять не приходится на английском. Важно уметь и понимать как он его выдает.

# Shell и Bash

## Что такое командная строка

Командная строка (оболочка / shell) — простой текстовый интерфейс для управления операционной системой. Командную строку обычно можно найти в окне терминала.



Shell / Bash

## Командная строка

- имя компьютера (Tom-Mac)
- текущий каталог (~)
- имя пользователя (Inelson)
- приглашение к вводу команды (■)

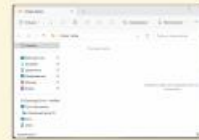
— означает домашний каталог



Shell / Bash

## Что такое Bash

Bash — это интерпретируемый командный язык; он определяет слова и правила для составления команд для операционной системы (ОС).



Shell / Bash

Создание каталогов, файлов, переименование каталогов, переход в каталог

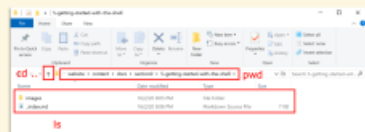
## Навигация по файловой системе

Навигация — одна из самых важных вещей, которую можно выполнить с помощью Bash.

- `pwd` (print working directory): отображает текущий каталог, в котором вы находитесь.
- `ls` (list): показывает содержимое текущего каталога.
  - `ls -l`: показ в длинном формате;
  - `ls -a`: показ скрытых файлов и каталогов.
- `cd` (change directory): используется для перехода в другой каталог, например: `cd /home/user/documents` или `cd ..`.

Shell / Bash

## Параллель интерфейса «Проводника» с командами Bash.



Shell / Bash

## Манипулирование файлами и каталогами

- `cp` (copy): может быть использована для копирования файла, например: `cp file1.txt file2.txt`.
- `mv` (move): может использоваться для перемещения или переименования файла, например: `mv door_1.log door_logs` и `mv quest323.sh9 quest3.sh`.
- `rm` (remove): удаляет файл безвозвратно (нет корзины), например: `rm ai_module_2.sh`.
- `rm -rf`: удаляет каталог и всё его содержимое, например: `rm -rf T01D01`.

Shell / Bash

## Создание файлов и каталогов

- `mkdir` (make directory) — создаёт директорию, например:
  - `mkdir dir1`: 1 директория;
  - `mkdir dir1 dir2 dir3`: несколько директорий;
  - `mkdir -p images/2012/July/Antarctica`: создать путь директорий.
- `touch` — может использоваться для создания файла:
  - `touch file1`: 1 файл.
  - `touch file1 file2 file3`: несколько файлов.

Shell / Bash

## Особенность названий команд

Регистр важен как для команд, так и для названий файлов! (В файловой системе семейства Unix и языках программирования семейства C#)

- `cp` ≠ `Cp` ≠ `CP` ≠ `cP`
- `mkdir` ≠ `mKdIr`
- `T01D01` ≠ `t01d01`
- `printf()` ≠ `Printf()`

Shell / Bash

## Просмотр и редактирование файлов

Команда `cat` (concatenate) отображает содержимое файла, например: `cat file1.txt`.

Команды `nano` или `vi` — это текстовые редакторы для редактирования файлов, например: `nano file1.txt` откроет документ, или создаст при отсутствии.

Shell / Bash

## Приколы в терминале

`Tab` — автодополнение строки за курсором.

`CTRL+L` — очистить окно терминала (`clear`). Скрывает ранее выведенный текст в терминале.

Переход по истории введенных команд.

`↑` — предыдущая введенная команда. Назад в историю командной строки.

`↓` — перемещается по истории в обратном направлении. Вперёд в историю командной строки.

Shell / Bash

## Шаблоны подстановки (wildcard)

Встретив в командной строке шаблон, интерпретатор заменяет его списком из всех имён файлов, соответствующих шаблону.

`*` — произвольная цепочка символов (в том числе `?`);

`?` — один одиночный символ;

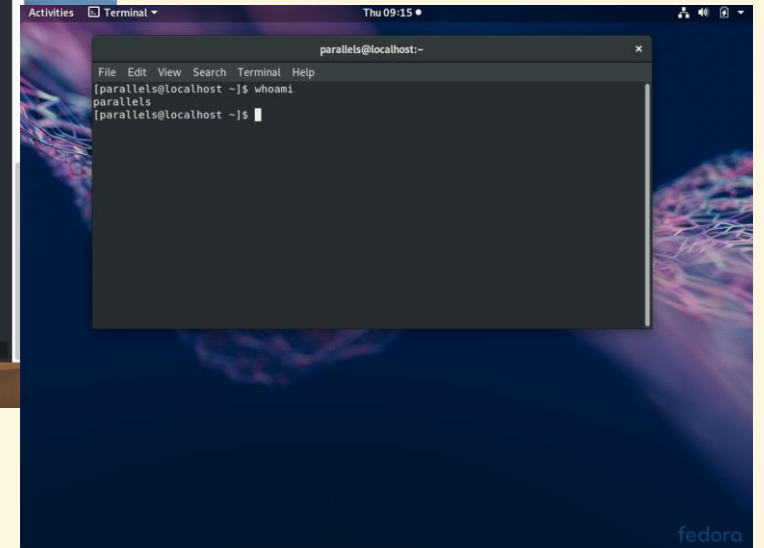
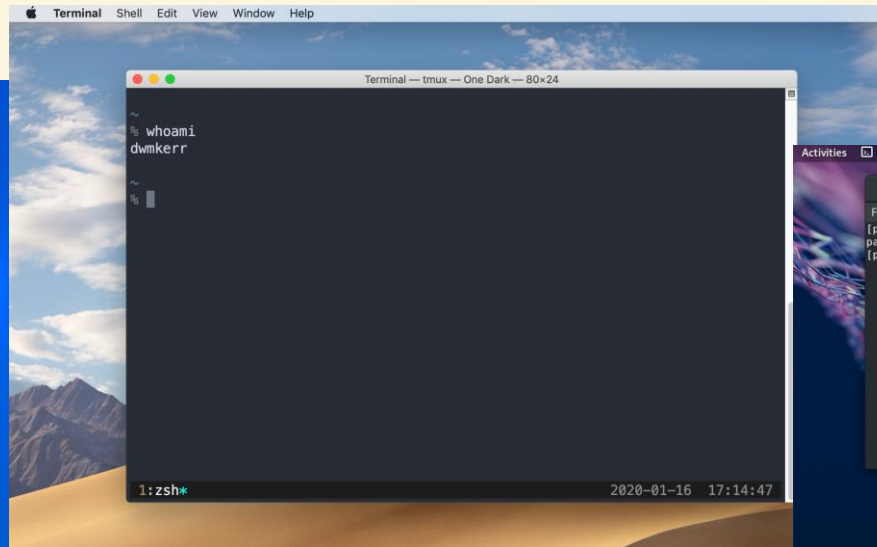
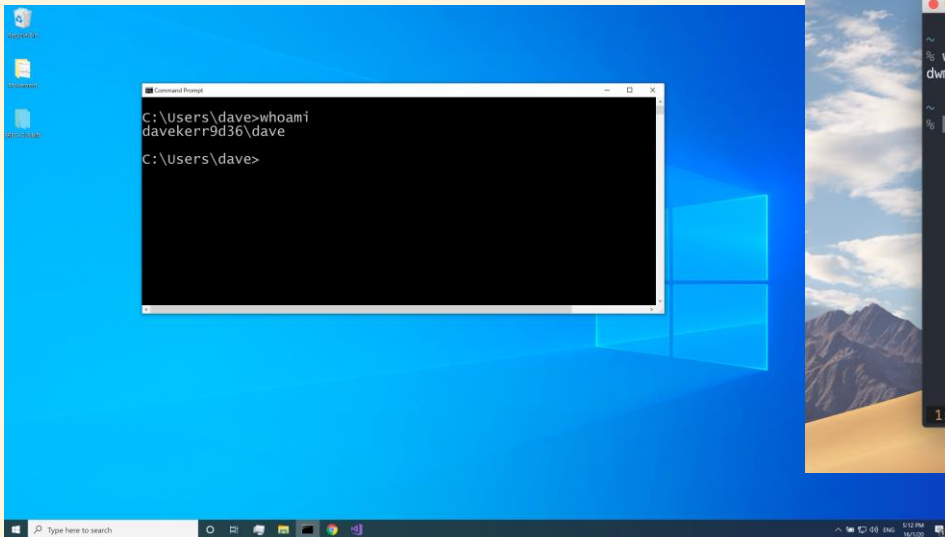
`{}` — создание списка элементов или генерации строк с определенными вариациями.

- `cp *.txt backup/`: скопирует все файлы с суффиксом `.txt` в директорию `backup`.
- `mv file1.txt file2.txt destination/`: переместит файлы `file1.txt` и `file2.txt` в директорию `destination`.
- `mkdir -p images/documents/media`: создаст три директории: `images`, `documents` и `media`.
- `????`: имена состоящие не менее чем из 3-х символов.

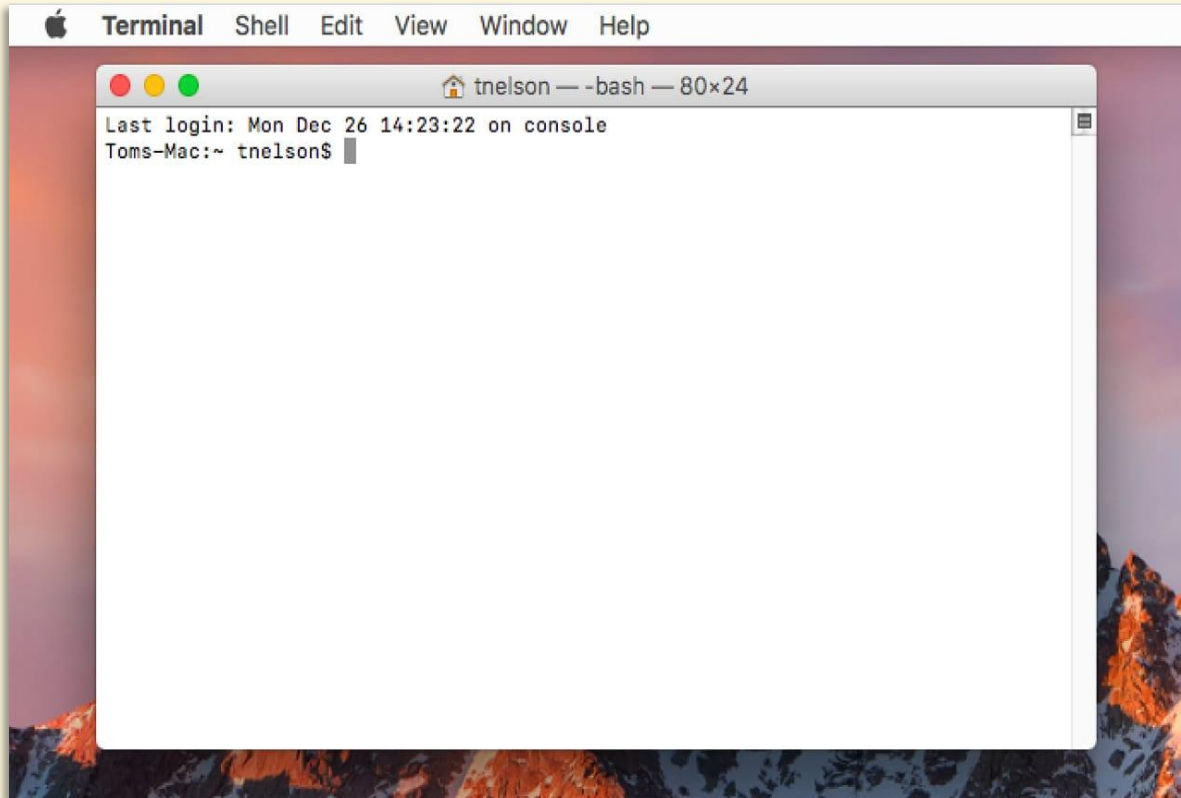
Shell / Bash

# Что такое **командная строка**

Командная строка (*оболочка / shell*) — простой **текстовый интерфейс для управления операционной системой**. Командную строку обычно можно найти в окне *терминала*.



# Командная строка

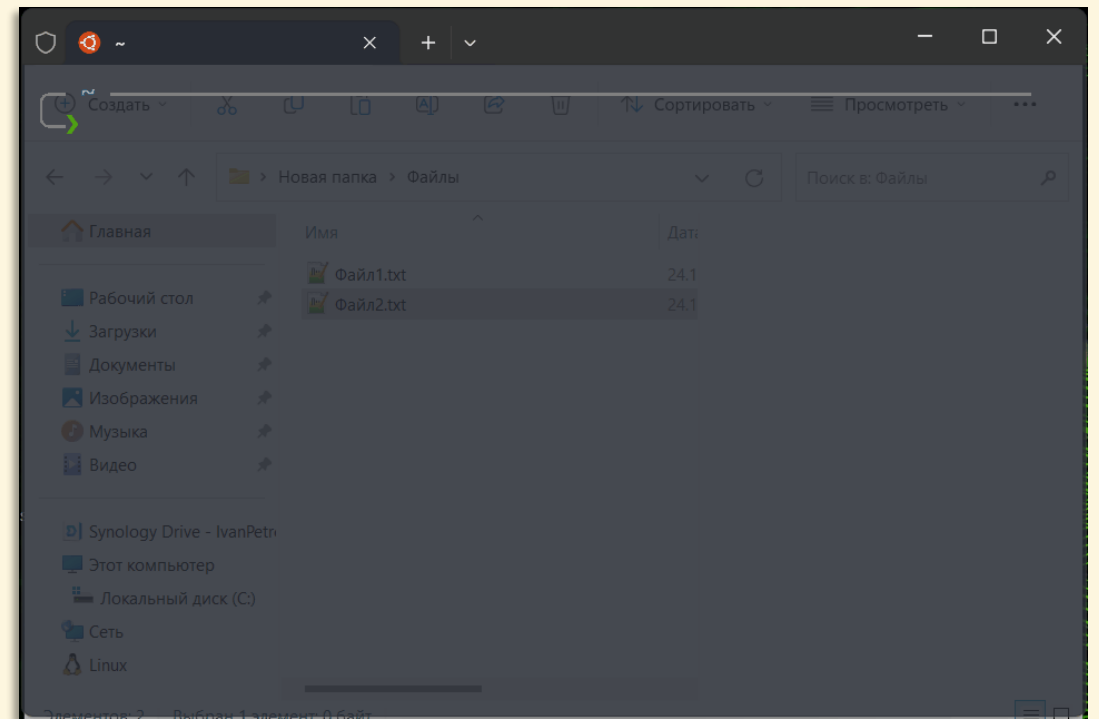
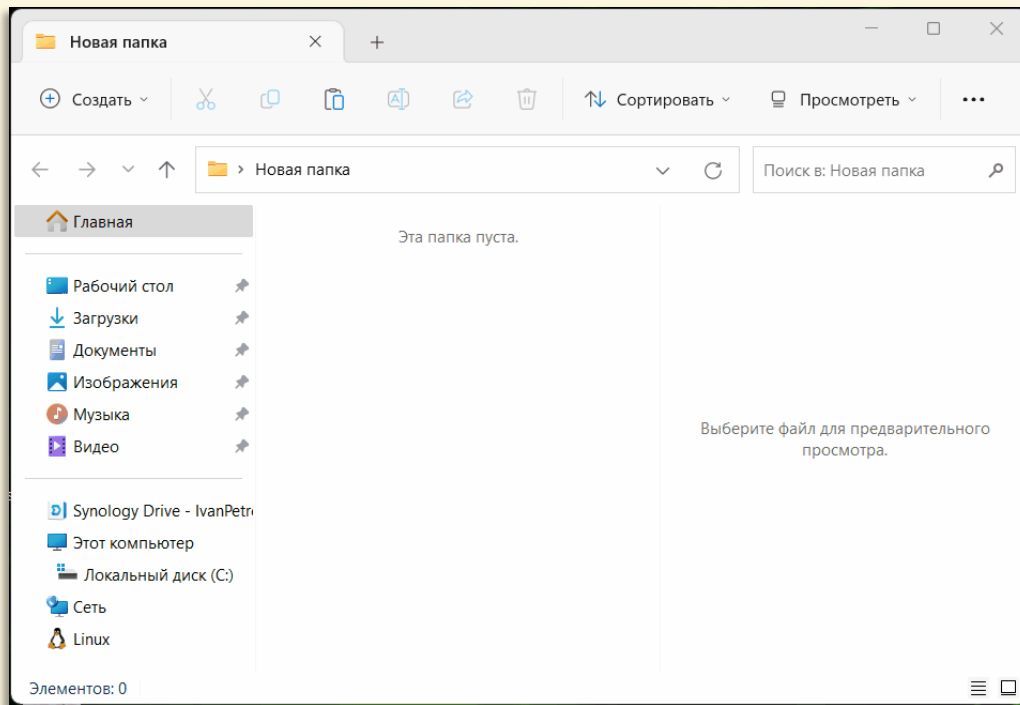


- имя компьютера (Tom-Mac)
- текущий каталог (~)
- имя пользователя (tnelson)
- приглашение к вводу команды (■)

~ означает домашний каталог

# Что такое Bash

Bash — это *интерпретируемый командный язык*; он определяет **слова и правила** для составления команд для *операционной системы (ОС)*.



Shell / Bash

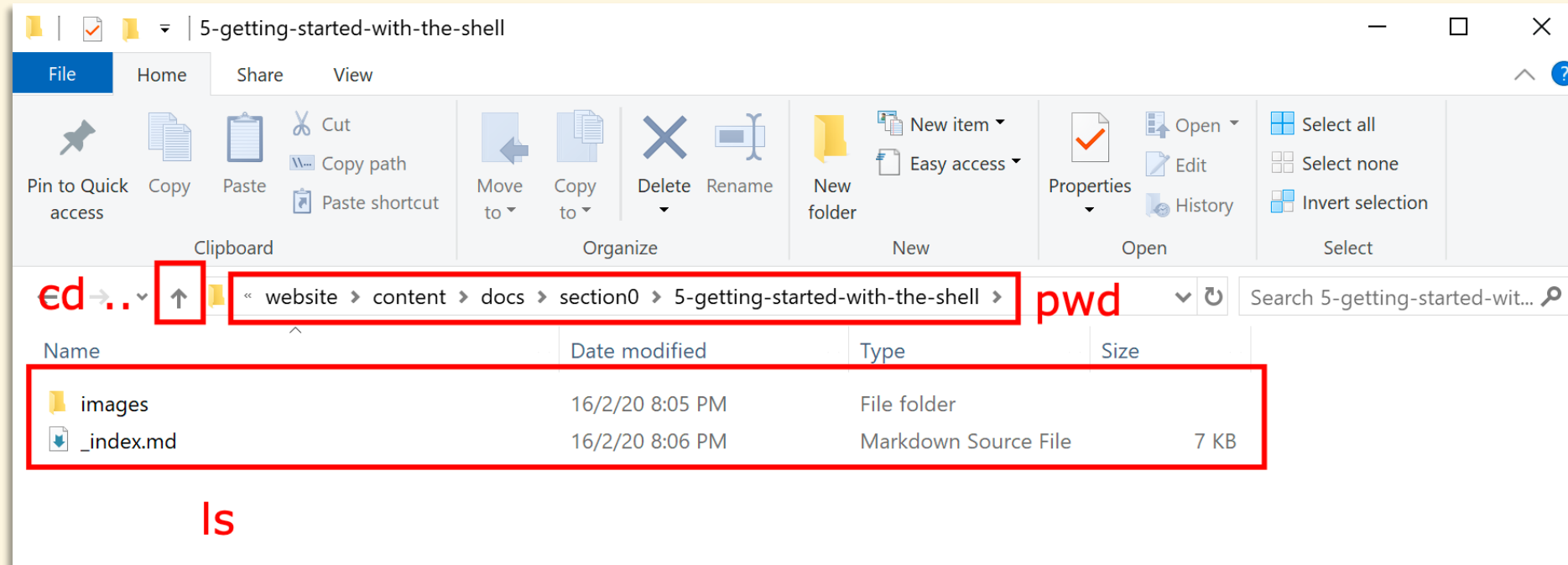
Создание каталога, файлов, перенос в каталог, переход в каталог.

# Навигация по файловой системе

Навигация – одна из самых важных вещей, которую можно выполнить с помощью Bash.

- `pwd` (*print working directory*): отображает *текущий каталог*, в котором вы находитесь.
- `ls` (*list*): показывает *содержимое текущего каталога*.
  - `ls -l`: показ в длинном формате;
  - `ls -a`: показ скрытых файлов и каталогов.
- `cd` (*change directory*): используется для перехода в другой каталог, например: `cd /home/user/documents` или `cd ..`.

# Параллель интерфейса «Проводника» с командами Bash.



# Манипулирование файлами и каталогами

- `cp` (*copy*): может быть использована для копирования файла, например: `cp file1.txt file2.txt`.
- `mv` (*move*): может использоваться для перемещения или переименования файла, например: `mv door_1.log door_logs` и `mv quest323.sh9 quest3.sh`.
- `rm` (*remove*): удаляет файл безвозвратно (нет корзины), например: `rm ai_module_2.sh`.
- `rm -rf`: удаляет каталог и всё его содержимое, например: `rm -rf T01D01`.



# Создание файлов и каталогов

- `mkdir` (make directory) — *создаёт директорию*, например:
  - `mkdir dir1`: 1 директория;
  - `mkdir dir1 dir2 dir3`: *несколько директорий*;
  - `mkdir -p images/2012/July/Antarctica`: *создать путь директорий*.
- `touch` — может использоваться для *создания файла*:
  - `touch file1`: 1 файл.
  - `touch file1 file2 file3`: несколько файлов.

# Особенность названий команд

**Регистр важен как для команд, так и для названий файлов!**

(В файловых системах семейства Unix и языках программирования семейства Си)

- `cp` ≠ `Cp` ≠ `CP` ≠ `cP`
- `mkdir` ≠ `mKdir`
- `T01D01` ≠ `t01d01`
- `printf()` ≠ `Printf()`

# Просмотр и редактирование файлов

Команда `cat` (concatenate) отображает **содержимое файла**, например: `cat file1.txt`.

Команды `nano` или `vi` - это **текстовые редакторы** для редактирования файлов, например: `nano file1.txt` **откроет** документ, или создаст при отсутствии.

# Приколы в терминале

Tab ⇐⇒ — автодополнение строки за курсором.

CTRL + L — очистить окно терминала (*clear*). Скрывает ранее выведенный текст в терминале.

## Переход по истории введённых команд.

↑ — предыдущая введённая команда. Назад в историю командной строки.

↓ — перемещается по истории в обратном направлении. Вперёд в историю командной строки.

# Шаблоны подстановки (wildcard)

Встретив в командной строке *шаблон*, интерпретатор заменяет его списком из всех имён файлов, соответствующих шаблону.

- \* — произвольная цепочка символов (в том числе  $\emptyset$ );
- ? — один одиночный символ;
- { } — создания списков элементов или генерации строк с определенными вариациями.
- `cp *.txt backup/`: скопирует все файлы с суффиксом `.txt` в директорию `backup`.
- `mv file{1,2}.txt destination/`: переместит файлы `file1.txt` и `file2.txt` в директорию `destination`.
- `mkdir {images,documents,media}`: создаст три директории: `images`, `documents` и `media`.
- `???*`: имена состоящий не менее чем из 3-х символов.

# Git и репозиторий

## Локальный git

`git` — это система контроля версий файлов. Позволяет отслеживать изменения в файлах, совместно работать над проектами и восстанавливать предыдущие версии. Она широко используется в разработке программного обеспечения.

Git работает **локально** на вашем компьютере. У него есть три основных состояния: рабочая директория, staging area и репозиторий. Вы добавляете изменения в «staging area», а затем фиксируете их в репозитории с помощью коммитов.

git

## Службы git-хостинга (удалённые репозитории)

Git позволяет сотрудничать удаленно. Удалённые репозитории дают возможность совместно работать над проектом.

Популярные платформы для хостинга удаленных репозиторий: GitHub, GitLab и Bitbucket.

В Школе 21 используется свой онлайн-репозиторий на основе GitLab.

git

## Git-команды 1

### Зафиксировать изменения

- `git add <файл>`: добавление изменений в промежуточную область (staging area) перед их фиксацией в репозитории.
- `git commit`: сохранение изменений в **локальном репозитории** (фиксация).

### Просмотр изменений

- `git status`: проверка состояния репозитория и просмотр того, какие файлы редактировались.
- `git diff`: просмотреть конкретные изменения в файлах.

git

## Git-команды 2

### Ветвление и слияние

- `git branch`: показать существующие ветки.
- `git checkout <ветка>`: **переключение между ветками** и коммитами.
- `git merge <ветка>`: объединение изменений из веток в определённую ветку.

### Удалённый репозиторий

- `git clone <ссылка>`: клонирование удалённого репозитория на ваш **локальный компьютер**.
- `git push origin <ветка>`: **отправка изменений** из локального репозитория в удаленный репозиторий.
- `git pull origin <ветка>`: **получение изменений** из удаленного репозитория в локальный репозиторий.

git

## Примеры

Как сделать слияние веток (*merge*)?

1. `git checkout develop`: перейти на целевую ветку (*цель*).
2. `git merge tinydook`: *залить* изменения из ветки tinydook в цель (*develop*).

- `git checkout -b <название новой ветки>`: создать новую ветку от текущей ветки.
- `git add`: добавить в *индексацию* файлы из всех директорий текущей папки.
- `git commit -m "main.c: fix div func"`: сделать коммит и записать комментарий.

git

# Локальный git

`git` — это система контроля версий файлов. Позволяет отслеживать изменения в файлах, совместно работать над проектами и восстанавливать предыдущие версии. Она широко используется в разработке программного обеспечения.

Git работает **локально** на вашем компьютере. У него есть три основных состояния: рабочая директория, staging area и репозиторий. Вы добавляете изменения в «staging area», а затем фиксируете их в репозитории с помощью коммитов.

# Службы git-хостинга (удалённые репозитории)

Git позволяет сотрудничать удаленно. Удалённые репозитории дают возможность совместно работать над проектом.

Популярные платформы для хостинга удаленных репозиторий: GitHub, **GitLab** и Bitbucket.

В Школе 21 используется свой онлайн-репозиторий на основе GitLab.



# Git-команды 1

## Зафиксировать изменения

- `git add <файл>`: добавление изменений в промежуточную область (staging area) перед их фиксацией в репозитории.
- `git commit`: сохранение изменений в **локальном репозитории** (*фиксация*).

## Просмотр изменений

- `git status`: проверка состояния репозитория и просмотр того, какие файлы редактировались.
- `git diff`: просмотреть конкретные изменения в файлах.

# Git-команды 2

## Ветвление и слияние

- `git branch`: показать существующие ветки.
- `git checkout <ветка>`: **переключение между ветками** и коммитами.
- `git merge <ветка>`: объединение изменений из веток в определённую ветку.

## Удалённый репозиторий

- `git clone <ссылка>`: **клонирование удалённого репозитория** на ваш **локальный компьютер**.
- `git push origin <ветка>`: **отправка изменений** из локального репозитория в удаленный репозиторий.
- `git pull origin <ветка>`: **получение изменений** из удаленного репозитория в локальный репозиторий.

# Примеры

Как сделать слияние веток (*merge*)?

1. `git checkout develop`: перейти на целевую ветку (*цель*).
  2. `git merge tinydook`: *залить* изменения из ветки `tinydook` в цель (*develop*).
- `git checkout -b <название новой ветки>`: создать новую ветку от текущей ветки.
  - `git add`: добавить в *индексацию* файлы из всех директорий текущей папки.
  - `git commit -m "main.c: fix div func"`: сделать коммит и записать комментарий.

# Vim

## Что это

Vim — интересный текстовый редактор, который особенно удобен людям, который владеют навыком слепой печати.

Для комфортной работы нужно учиться и привыкать к его устройству

Почти не пригодится на интенсиве. Важно уметь и понимать **как** из него **выйти**.

Vim

## Сохранение и выход из Vim

Для **сохранения без выхода** из программы используйте команду **:w**.

Чтобы **выйти без сохранения**, используйте команду **:q!**.

Чтобы **сохранить изменения и выйти** из Vi, совместите команды **w** и **q**, то есть **:wq**.

Vim

## Режимы в vim

### Нормальный режим

- перемещение курсора
- переключение между режимами
- копирование, вставка

### Режим вставки (режим редактирования текста)

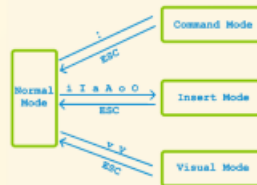
- ввод и удаление текста

### Режим ввода команд

- ввод команд (сохранить/выйти)

Vim

## Режимы



- Переход в **нормальный** режим **Esc**;
- в режим **вставки** **i** или **a**;
- в **командный** режим **:**.
- **:w** — сохранить; **:q** — выйти; **:wq** — сохранить и выйти

Vim

## Навигация по тексту

**w** перемещает курсор к началу следующего слова, а **b** - к началу предыдущего слова

**0** (ноль) перемещается в начало строки, а **\$** - в конец строки.



Vim

## Ввод текста

**i** Вставить перед текущим символом.

**I** Вставить в начало строки.

**a** Вставить после текущего символа.

**A** Вставить в конец строки.

**o** Начать новую строку ниже текущей и войти в режим вставки.

**O** Начать новую строку выше текущей и войти в режим вставки.

Vim

## Манипулирование текстом

Vi также предоставляет команды для работы с текстом

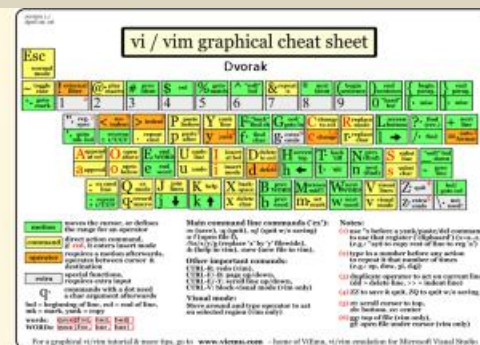
**dd** — удаляет текущую строку

**u** — отменяет последнее изменение, а **Ctrl + r** повторяет последнее изменение.

**x** — удаляет символ под курсором.

Чтобы **скопировать** текст используйте **y** для выделения, **yy**, **h**, **p** для вставки.

Vim



Vim

# Что это

Vim — интересный текстовый редактор, который особенно удобен людям, который владеют навыком слепой печати.

Для комфортной работы нужно учиться и привыкать к его устройству

Почти не пригодится на интенсиве. Важно уметь и понимать **как** из него **выйти**.

# Сохранение и выход из Vim

Для *сохранения без выхода* из программы используйте команду `:w`.

Чтобы *выйти без сохранения*, используйте команду `:q!`.

Чтобы *сохранить изменения и выйти* из Vi, совместите команды `w` и `q`, то есть `:wq`.

# Режимы в vim

## Нормальный режим

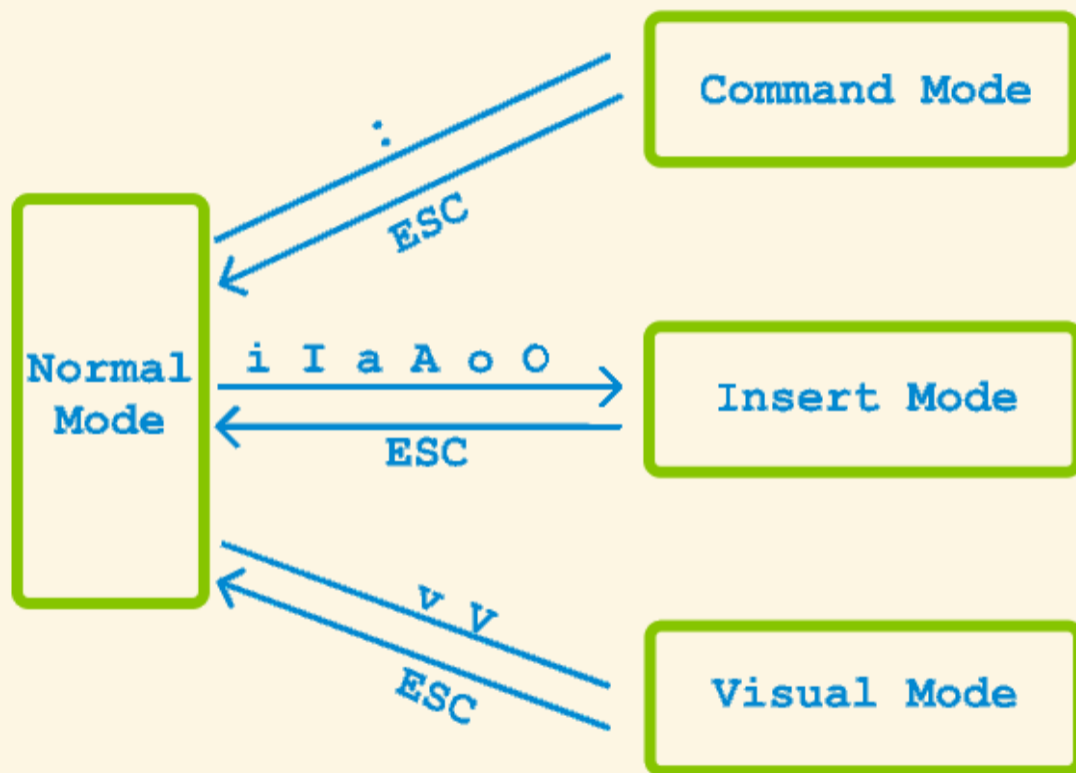
- перемещение курсора
- переключение между режимами
- копирование, вставка

## Режим вставки (режим редактирования текста)

- ввод и удаление текста

## Режим ввода команд

- ввод команд (сохранить/выйти)



# Режимы

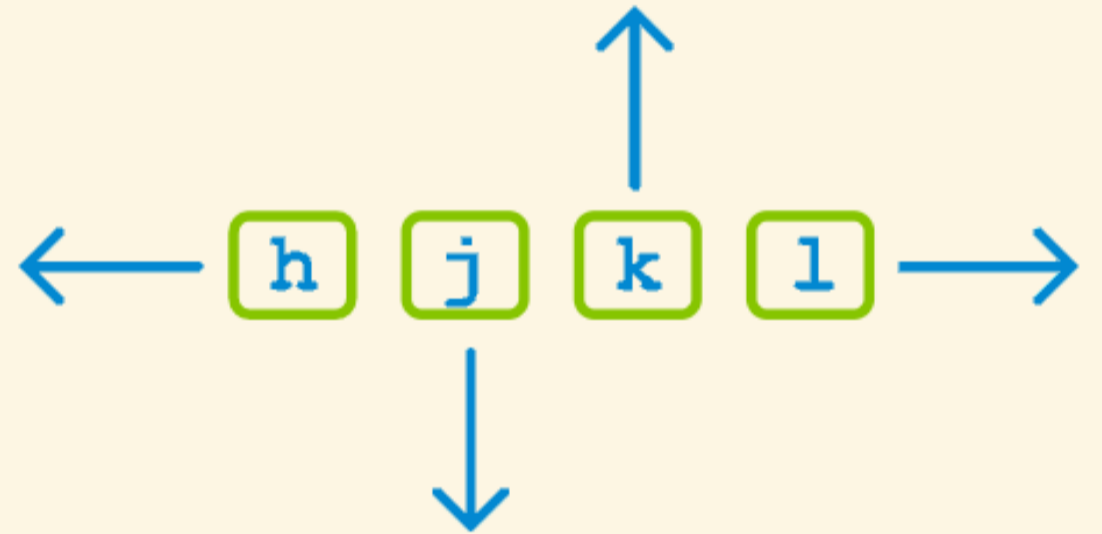
- Переоход в *нормальный* режим `Esc`;
- в режим *вставки* `i` или `a`;
- в *командый* режим `:`.
- `:w` – сохранить; `:q` – выйти; `:wq` — сохранить и выйти



# Навигация по тексту

**w** перемещает курсор к началу следующего слова, а **b** - к началу предыдущего слова

**0** (ноль) перемещается в начало строки, а **\$** - в конец строки.



# Ввод текста

- i** Вставить перед текущим символом.
- I** Вставить в начало строки.
- a** Вставить после текущего символа.
- A** Вставить в конец строки.
- o** Начать новую строку ниже текущей и войти в режим вставки.
- O** Начать новую строку выше текущей и войти в режим вставки.

# Манипулирование текстом

Vi также предоставляет команды для работы с текстом

**dd** — удаляет текущую строку

**u** — отменяет последнее изменение, а **Ctrl + r** повторяет последнее изменение.

**x** — удаляет символ под курсором.

~~Чтобы скопировать текст используйте **y**, для выделения **v**, и **p** для вставки.~~

# vi / vim graphical cheat sheet

**Esc**  
normal mode

Dvorak

~ toggle case	! external filter	@ play macro	# prev ident	\$ eol	% goto match	^ "soft" bol	& repeat :s	* next ident	( begin sentence	) end sentence	{ begin parag.	} end parag.
\ goto mark	1	2	3	4	5	6	7	8	9	0 "hard" bol	[ misc	] misc
" reg. spec <sup>1</sup>	< un-indent <sup>3</sup>	> indent <sup>3</sup>	P paste before	Y yank line	F "back" find ch	G eof/ goto ln	C change to eol	R replace mode	L screen bottom	? find (rev.)	+ next line	
' goto mk. bol	, reverse t/T/f/F	. repeat cmd	p paste after	y yank <sup>1,3</sup>	f find char	g extra <sup>6</sup> cmds	c change <sup>1,3</sup>	r replace char	l →	/ find	= auto <sup>3</sup> format	
A append at eol	O open above	E end WORD	U undo line	I insert at bol	D delete to eol	H screen top	T back 'till	N prev (find)	S subst line	"soft" bol down		
a append	o open below	e end word	u undo	i insert mode	d delete <sup>1,3</sup>	h ←	t 'till	n next (find)	s subst char	- prev line		
. ex cmd line	Q ex mode	J join lines	K help	X back-space	B prev WORD	M screen mid'l	W next WORD	V visual lines	Z quit <sup>4</sup>	bol/ goto col		
. repeat t/T/f/F	q record macro	j ↓	k ↑	x delete char	b prev word	m set mark	w next word	v visual mode	Z extra <sup>5</sup> cmds	\ not used!		

**motion**

moves the cursor, or defines the range for an operator

**command**

direct action command, if **red**, it enters insert mode

**operator**

requires a motion afterwards, operates between cursor & destination

**extra**

special functions, requires extra input

q.

commands with a dot need a char argument afterwards

bol = beginning of line, eol = end of line, mk = mark, yank = copy

words: quux(foo, bar, baz);

WORDS: quux(foo, bar, baz);

**Main command line commands ('ex'):**

:w (save), :q (quit), :q! (quit w/o saving)  
:e f (open file f),  
:%s/x/y/g (replace 'x' by 'y' filewide),  
:h (help in vim), :new (new file in vim),

**Other important comands:**

CTRL-R: redo (vim),  
CTRL-F/-B: page up/down,  
CTRL-E/-Y: scroll line up/down,  
CTRL-V: block-visual mode (vim only)

**Visual mode:**

Move around and type operator to act on selected region (vim only)

**Notes:**

- (1) use "x before a yank/paste/del command to use that register ('clipboard') (x=a..z,\*) (e.g.: "ay\$ to copy rest of line to reg 'a')
- (2) type in a number before any action to repeat it that number of times (e.g.: 2p, d2w, 5i, d4j)
- (3) duplicate operator to act on current line (dd = delete line, >> = indent line)
- (4) ZZ to save & quit, ZQ to quit w/o saving
- (5) zt: scroll cursor to top, zb: bottom, zz: center
- (6) gg: top of file (vim only), gf: open file under cursor (vim only)

# Удачи!

**Best of luck!**