

Командная

строка

Что такое командная строка Bash

Командная строка — это **текстовый интерфейс** для работы человека с операционной системой, путём отправки компьютеру команд. Командную строку обычно можно найти в окне терминала.

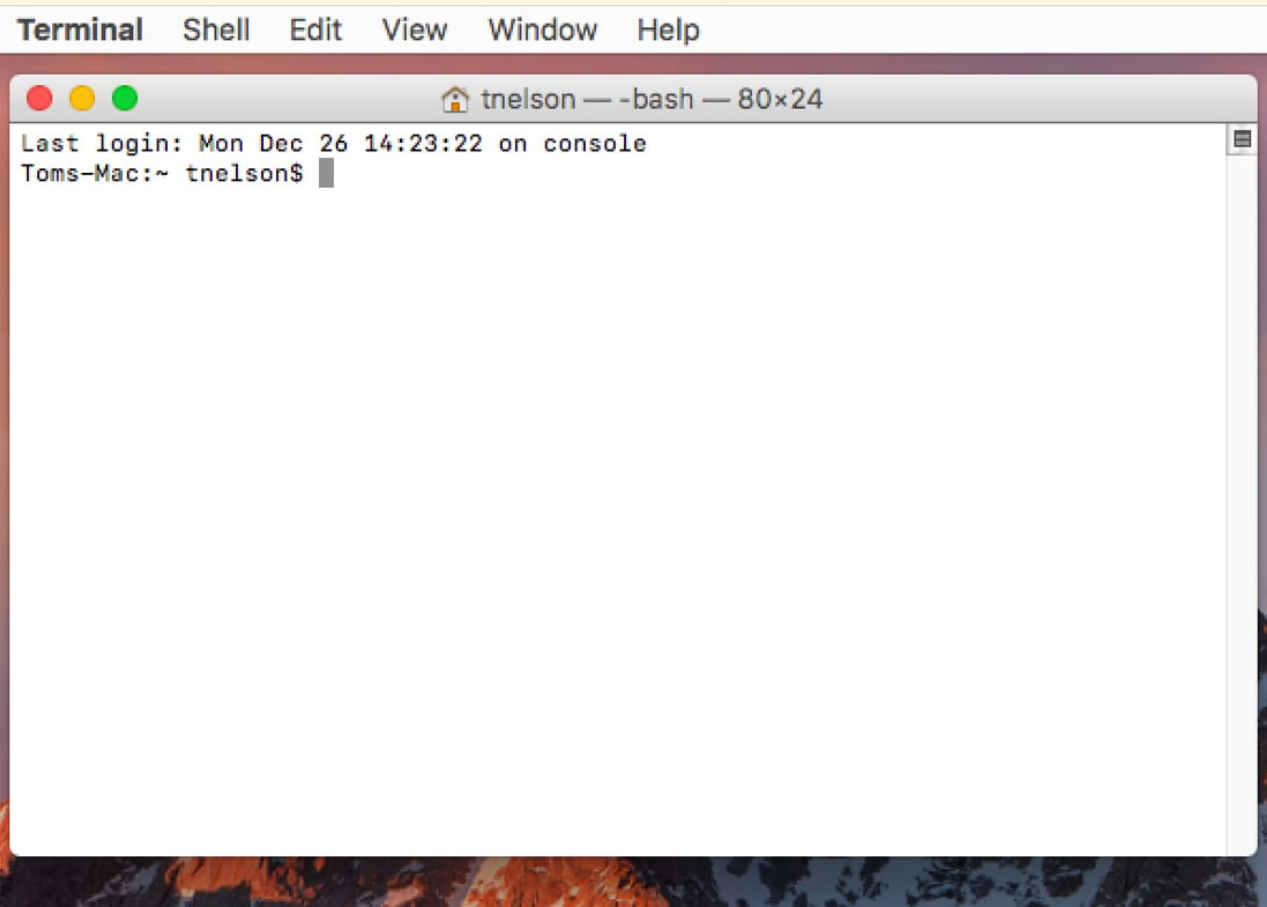
Bash — это **компьютерный язык**; он задаёт слова и правила для составления команд для операционной системы (ОС).

Команды bash будут использоваться на протяжении всего интенсива. Учить не надо, запомните на практике.

Командная строка:

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

~ означает домашний каталог



Навигация по файловой системе

Одна из самых важных вещей, которые вы можете сделать с помощью Bash — это навигация по файловой системе.

- `pwd` (print working directory) — отображает *текущий каталог*, в котором вы находитесь.
- `ls` (list) — показывает *содержимое текущего каталога*.
- `cd` (change directory) — используется для перехода в другой каталог, например: `cd /home/user/documents` или `cd ..`.

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 в Bash

`cp` ≠ Cp ≠ CP ≠ cP

`mkdir` ≠ mKdir

`ilarion.txt` ≠ Ilarion.txt

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

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

Просмотр и редактирование файлов

Вы также можете просматривать и редактировать файлы с помощью Bash.

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** или создаст файл при отсутствии.


Приколюхи в терминале

Tab  — автодополнение строки за курсором.

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

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)

- *** — для выделения нескольких символов (в том числе \emptyset);
- ?** — для выделения одиночного символа;
- { }** — создания списков элементов или генерации строк с определенными вариациями

`cp *.txt backup/` — копирует все файлы с расширением `.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`.

Git

Локальный git

`git` — это система контроля версий, которая отслеживает и фиксирует изменения в проекте или файле. Git не делает копии (дубликаты) файлов, она только делает снимки состояний файлов и сравнивает их.

- вернуться к предыдущим версиям файла.
- вести четкую историю изменений.
- история изменения проекта практически не занимает памяти на диске.

Git будет использоваться на протяжении всего интенсива.

Службы git-хостинга

GitHub, GitLab и Bitbucket, предоставляют удаленный репозиторий, в котором вы можете хранить код и получать к нему доступ из любого места. В Школе 21 используется свой репозиторий на основе GitLab.

- Совместная работа.
- Видимость проекта для людей всего мира.
- Резервное копирование и восстановление.
- Безопасность (аутентификация, авторизация).

Git команды 1

- `git clone %ссылка%` — клонирование удалённого репозитория на ваш локальный компьютер.
- `git add %файлы%` — добавление изменений в промежуточную область перед их фиксацией в репозитории.
- `git commit` — сохранение изменений в локальном репозитории.
- `git checkout %ветка%` — переключения между ветками и коммитами.
- `git status` — проверки состояния репозитория и просмотра того, какие файлы были изменены.

Git команды 2

- `git push origin %ветка%` — отправка изменений из локального репозитория в удаленный репозиторий.
- `git pull origin %ветка%` — скачивание изменений из удаленного репозитория в локальный репозиторий.
- `git merge %ветка%` — объединение изменений из веток в определённую ветку.

Примеры

Как сделать 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**.

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

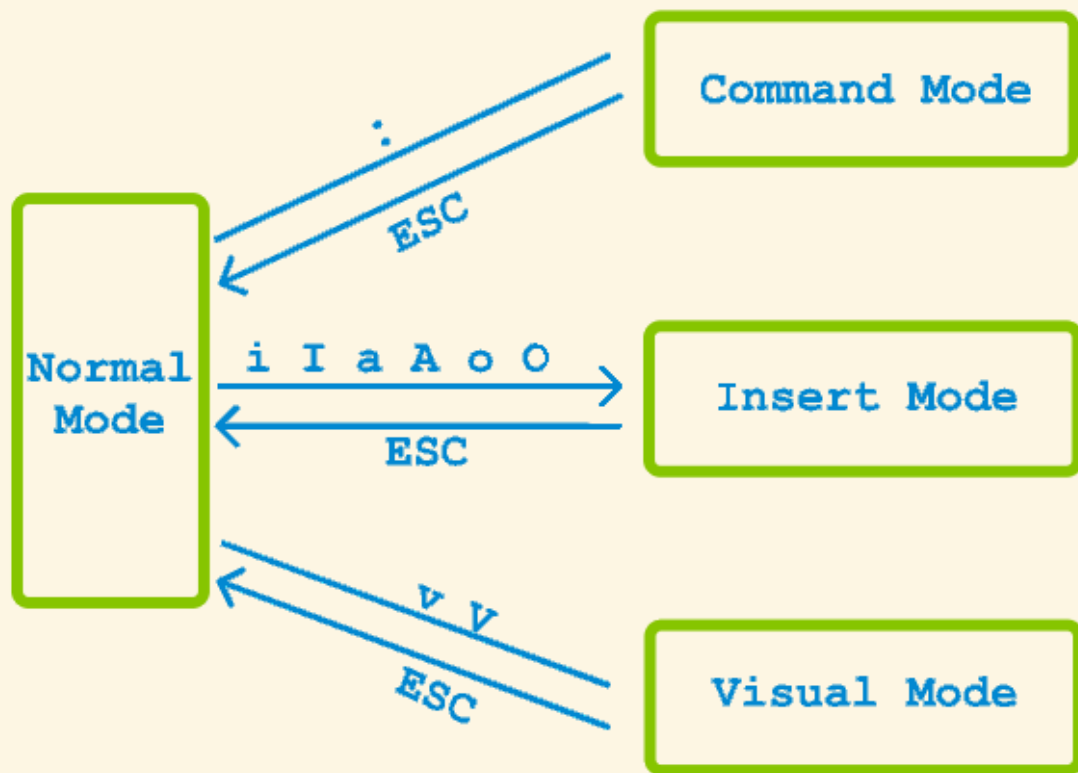
- перемещение курсора
- переключение между режимами
- копирование, вставка

Режим вставки (режим редактирования текста)

- ввод и удаление текста

Режим ввода команд

- ввод команд (сохранить/выйти)



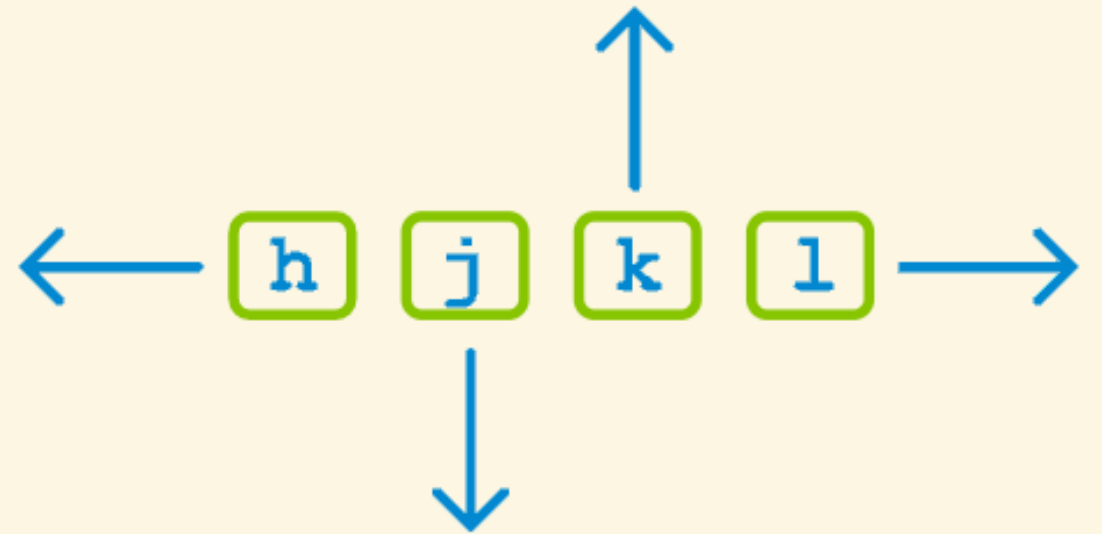
Режимы

- 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 ¹	< un-indent ³	> indent ³	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 ^{1,3}	f find char	g extra ⁶ cmds	c change ^{1,3}	r replace char	l →	/ find	= auto ³ 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 ^{1,3}	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 ⁴	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 ⁵ 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)

Удаchi!

Best of luck!