📝 Конспект: Основы Bash (Прикладная схема)

1. Как строятся команды

У любой команды в Bash есть три части:

`КОМАНДА [ФЛАГИ] [АРГУМЕНТЫ]`

- **Команда:** Что делаем? (Напр., `ls`, `mkdir`)
- **Флаги (Options):** Как делаем? (Изменяют поведение, обычно с `-`. Напр., `ls -l` — показать подробно).
- **Аргументы:** Над чем делаем? (Цель. Напр., `mkdir "Новая Папка"`).

---

2. Проблема: "Word Splitting" (Деление на слова)

Это **самая важная** концепция для понимания кавычек.

- **Как Bash читает:** Bash *сначала* делит вашу команду на "слова" по пробелам.
- **Пример проблемы:**
  - Вы пишете: `name="Мои Документы"`
  - Затем: `mkdir $name`
  - Bash *сначала* подставляет `$name` и видит: `mkdir Мои Документы`
  - Затем он делит это на "слова": (1: `mkdir`), (2: `Мои`), (3: `Документы`).
  - **Результат:** Создаются *две* папки, а не одна.

---

3. Решение: Кавычки и Экранирование

Кавычки нужны, чтобы сказать Bash: "То, что внутри — это *одно* слово (один аргумент), даже если там есть пробелы".

Тип	Пример	Как работает (Что видит Bash)
"Двойные"	`mkdir "$name"`	`mkdir "Мои Документы"` (1 аргумент). **Переменные ($) работают.**
'Одинарные'	`echo '$name'`	`echo '$name'` (1 аргумент). **Всё буквально.** (Выведет "$name")

\ (Экран)	`mkdir Мои\ Документы`	`mkdir Мои Документы` (1 аргумент). Экранирует *только 1* символ.

Золотое правило: Всегда используйте **двойные кавычки** для переменных: `rm "$filename"`, `echo "$message"`.

---

4. Потоки (Streams) и Перенаправление

У каждой программы есть 3 "трубы": 2 на выход, 1 на вход.

- `stdin (0)`: **Ввод** (Обычно клавиатура)
- `stdout (1)`: **Вывод** (Успешный результат, идёт в терминал)
- `stderr (2)`: **Ошибки** (Текст ошибки, тоже идёт в терминал)

Перенаправление — это управление этими "трубами":

- `команда > файл.txt`
  - Перенаправить `stdout (1)` в файл (файл будет **перезаписан**).
- `команда >> файл.txt`
  - Перенаправить `stdout (1)` в файл (добавить в **конец**).
- `команда < данные.txt`
  - Взять `stdin (0)` из файла (вместо клавиатуры).
- `команда 2> ошибки.txt`
  - Перенаправить *только* ошибки `stderr (2)` в файл.
- `команда > all.txt 2>&1` (или `команда &> all.txt`)
  - **"Перенаправить (2) туда же, куда и (1)"**.
  - *Результат:* И `stdout`, и `stderr` попадают в `all.txt`.

---

5. Конвейер (Pipe |)

Пайп (|) — это особый вид перенаправления. Он соединяет `stdout` одной команды с `stdin` другой.

Схема: `Команда А (stdout)` ➔ `Команда Б (stdin)`

- **Пример:** `ls -l | grep ".txt"`
  1. `ls -l` выполняется, её **вывод** (список файлов) *не* идёт на экран.
  2. Этот вывод "по трубе" (|) подаётся на **ввод** команде `grep`.
  3. `grep` ищет в этом вводе строки, содержащие ".txt".

---

6. Типы команд (Приоритет выполнения)

Когда вы вводите команду, Bash ищет её в строгом порядке:

1. **Alias (Псевдоним):** `alias ll='ls -l'`
    - *Что это?* Простая текстовая замена (ваше сокращение).
2. **Function (Функция):** `my_func() { ... }`
    - *Что это?* Блок кода, загруженный в память Bash.
3. **Builtin (Встроенная):** `cd`, `echo`, `read`
    - *Что это?* Команда, встроенная в сам Bash. Выполняется мгновенно.
4. **Executable (Внешняя):** `ls`, `grep`, `mkdir`
    - *Что это?* Отдельная программа, которую Bash ищет в папках из `$PATH`.

---

7. Glob (Шаблоны) vs. Спецсимволы

Glob (Шаблоны): Используются *только* для поиска **имён файлов**.

- `*` (Звезда): 0 или больше *любых* символов.
    - `*.txt` (все файлы, кончающиеся на .txt)
- `?` (Знак вопроса): Ровно 1 *любой* символ.
    - `photo_0?.jpg` (photo_01.jpg, photo_0A.jpg, но *не* photo_010.jpg)
- `[...]` (Скобки): Ровно 1 символ *из набора*.
    - `file[AB].log` (fileA.log, fileB.log)

Ключевые спецсимволы:

- `$` (Доллар): "Взять значение" (напр., `$PATH`, `$USER`).
- `#` (Решетка): Комментарий (всё до конца строки игнорируется).
- `=` (Равно): Присвоить значение (ВАЖНО: `VAR=5`, без пробелов вокруг `=`).
- `~` (Тильда): Домашний каталог.