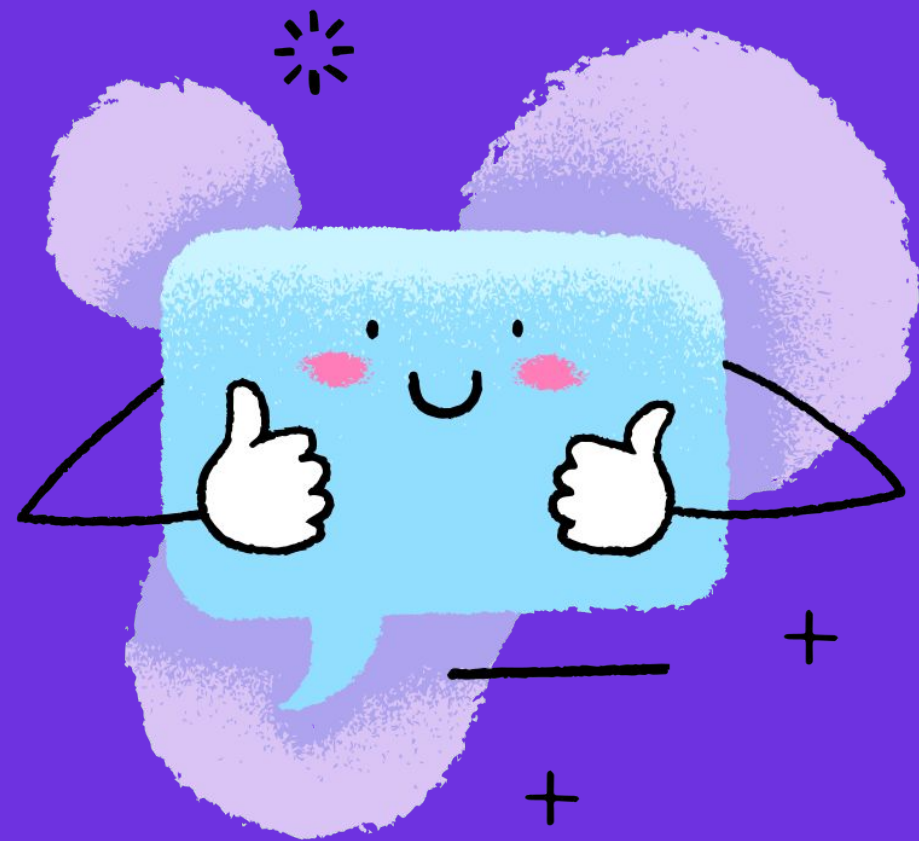


Введение в скрипты `bash`. Планировщик задач `crontab` & `at`

Linux

Вопросы по практической работе и предыдущему уроку



План урока

1. Правила написания скриптов.
Переменные.
2. Условный оператор if и циклы.
3. Циклы for и while.
4. Регулярные выражения и
утилиты для работы с
регулярными выражениями.
5. Планировщики crontab и at.

Правила написания скриптов.

Переменные

Bash — это командный интерпретатор, работающий, как правило, в интерактивном режиме в текстовом окне.

Bash-скрипты — это сценарии командной строки, то есть наборы обычных команд, которые пользователь вводит с клавиатуры. Для автоматизации каких-то рутинных вещей эти команды объединяются в файл-сценарий, который и носит название скрипт.

Правила написания скриптов.

Переменные

shebang (шебанг) — особая запись в начале файла, которая выглядит как сочетание символов `#!` и путь до интерпретатора. Например, `#!/bin/bash` укажет системе, что для выполнения кода после этой строки необходимо использовать `bash`. Запись `#!/usr/bin/perl` укажет, что для выполнения кода нужно использовать `perl`. После этой строки можем писать скрипт.

Комментарии в `bash` определяются символом `#`. Комментарий может быть добавлен в начале строки или встроен в код, например:

- комментарий в начале строки: `# Определяем переменные;`
- комментарий в коде: `echo "text" # Выводим сообщение на экран терминала`

Правила написания скриптов.

Переменные

Переменные необходимы для хранения информации. С ними можно выполнить два действия:

- установить значение переменной;
- прочитать значение переменной.

В `bash` нет строгих различий между типами переменных. С точки зрения командного интерпретатора любая переменная является строкой.

Правила написания скриптов.

Переменные

Переменные bash

Переменные окружения

\$PWD — текущий каталог.
\$ID — покажет имя текущего пользователя и группы, в которых он состоит.
\$PATH — покажет путь до исполняемых файлов

Пользовательские переменные

a=123 присвоит переменной a значение 123

a=\$(ls) присвоит переменной a результат работы команды ls

Специальные переменные

Переменные подстановки: \$0
\$1 ..\$9
\$? — статус выполнения предыдущей команды или скрипта

Условный оператор if и циклы



Условный оператор if и циклы

Операции сравнения (наиболее используемые)

Проверка файлов:

- -e возвращает true (истина), если файл существует (exists);
- -d возвращает true (истина), если каталог существует (directory).

Условный оператор if и циклы

Операции сравнения (наиболее используемые)

Сравнение строк:

- = или == возвращает true (истина), если строки равны;
- != возвращает true (истина), если строки не равны;
- -z возвращает true (истина), если строка пуста;
- -n возвращает true (истина), если строка не пуста.

Условный оператор if и циклы

Операции сравнения (наиболее используемые)

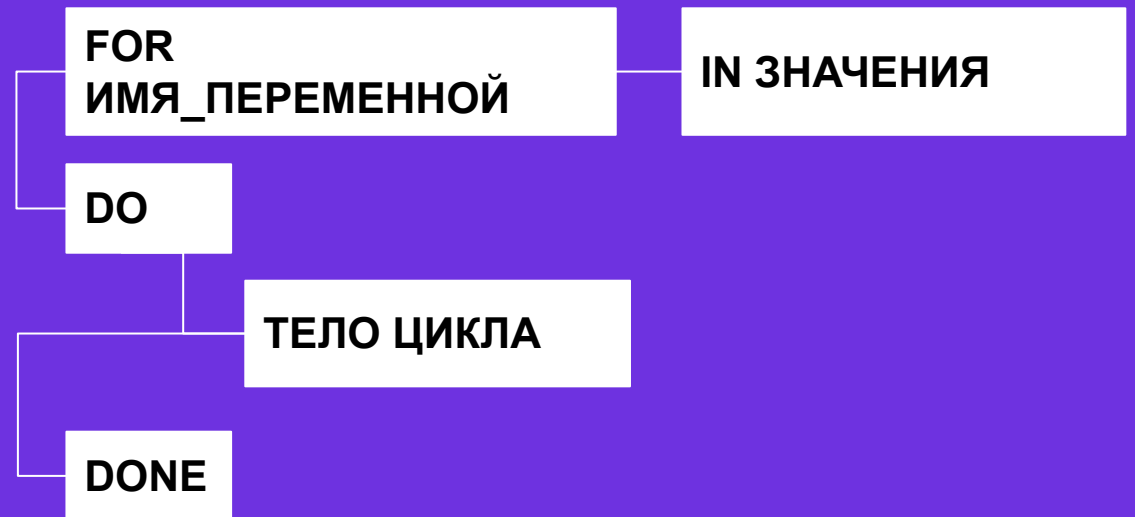
Сравнение целых чисел:

- -eq возвращает true (истина), если числа равны (equals);
- -ne возвращает true (истина), если числа не равны (not equal).

Условный оператор if и ЦИКЛЫ

Цикл — последовательность, которая позволяет выполнить определённый участок кода заданное количество раз.

FOR:

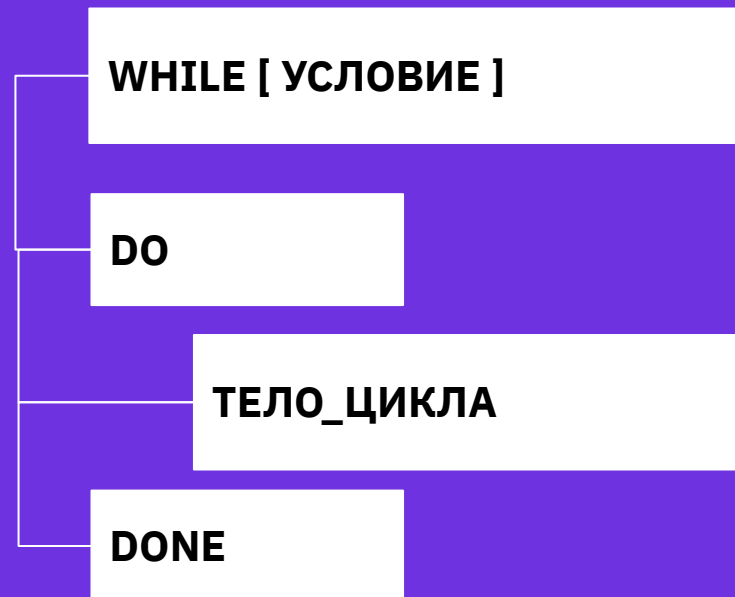


Условный оператор if и циклы

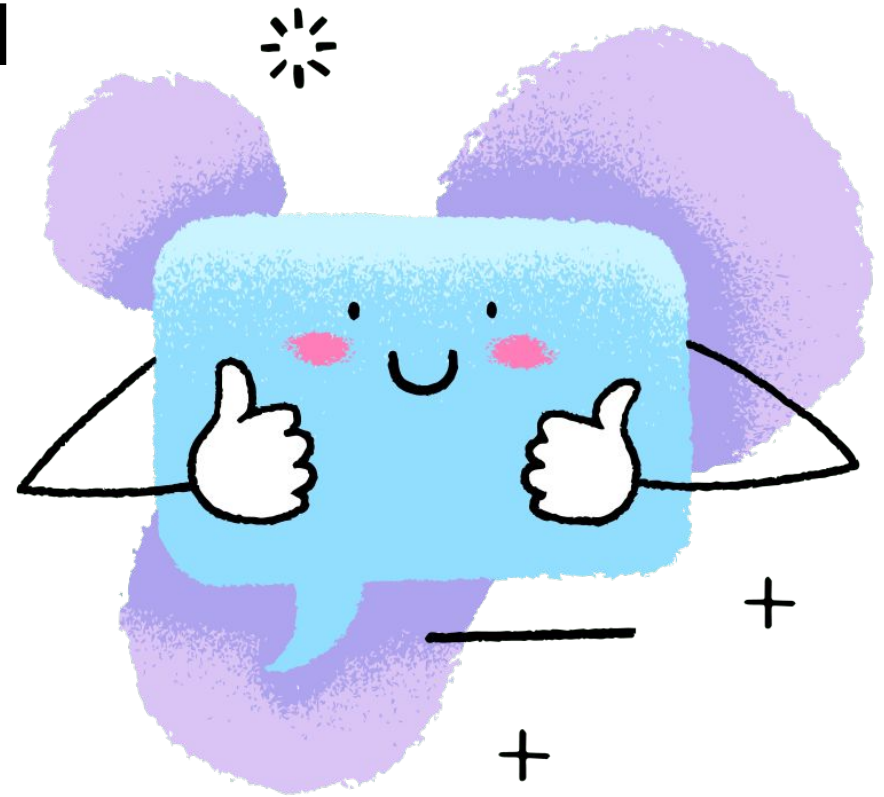
Здесь в качестве [условие] осуществляются операции сравнения и проверки, аналогичные условному оператору if.

Тело_цикла — команды, которые будут выполняться до тех пор, пока условие возвращает true (истина).

WHILE:



Регулярные выражения и утилиты для работы с регулярными выражениями



Регулярные выражения — инструмент, предназначенный для поиска, а также обработки текста по заданному шаблону.

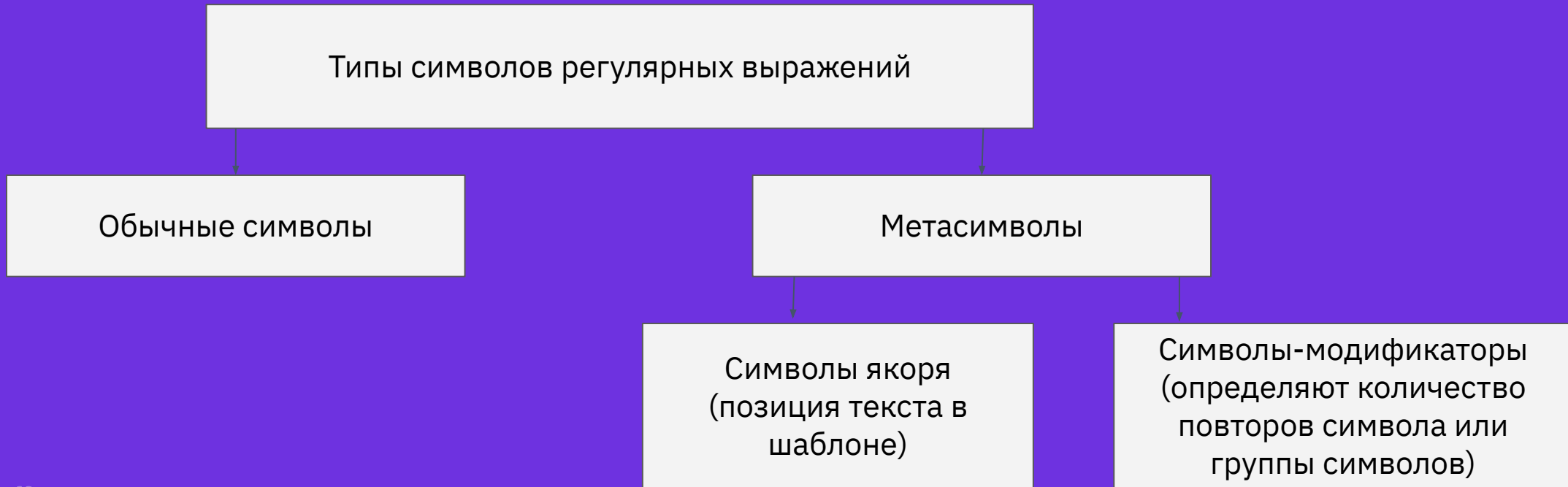
Используя регулярные выражения, мы можем изменять текст, искать строки в файле, фильтровать список файлов согласно каким-то условиям и т. д.

Регулярные выражения — неотъемлемая часть командного интерпретатора `bash`. Они постоянно применяются в работе с командной строкой.

Регулярные выражения и утилиты для работы с регулярными выражениями



Регулярные выражения и утилиты для работы с регулярными выражениями



- \ — с обратной косой черты начинаются буквенные спецсимволы,
- * — указывает, что предыдущий символ может повторяться 0 или больше раз;
- + — указывает, что предыдущий символ должен повториться больше 1 или больше раз;
- ? — предыдущий символ может встречаться 0 или 1 раз;
- {n} — указывает сколько раз (n) нужно повторить предыдущий символ;
- {N,n} — предыдущий символ может повторяться от N до n раз;
- . — любой символ, кроме перевода строки;
- [az] — любой символ, указанный в скобках;
- x|y — символ x или символ y;
- [^az] — любой символ, кроме тех, что указаны в скобках;
- [a-z] — любой символ из указанного диапазона;
- [^a-z] — любой символ, которого нет в диапазоне;
- \b — обозначает границу слова с пробелом;
- \B — означает, что символ должен не быть окончанием слова;
- \d — означает, что символ — цифра;
- \D — нецифровой символ;
- \n — символ перевода строки;
- \s — любой пробельный символ: пробел, табуляция и так далее;
- \S — любой непробельный символ;
- \t — символ табуляции;
- \v — символ вертикальной табуляции;
- \w — любой буквенный символ, включая подчёркивание;
- \W — любой буквенный символ, кроме подчёркивания;
- \uXXX — конкретный указанный символ Unicode.

Основные символы- модификаторы

Регулярные выражения и утилиты для работы с регулярными выражениями

grep находит на вводе целые строки, отвечающие заданному регулярному выражению, и выводит их, если вывод не отменён специальным ключом. Grep работает с регулярными выражениями POSIX (BRE), а также все остальные, включая PCRE в разных режимах. У него есть модификация egrep, которая позволяет расширенный синтаксис (ERE).

Регулярные выражения и утилиты для работы с регулярными выражениями

sed — потоковый текстовый редактор. Позволяет редактировать потоки данных на основе заданных правил. С помощью SED можно провести простые операции по поиску и замене слов в тексте.

Регулярные выражения и утилиты для работы с регулярными выражениями

awk — более мощная, чем SED, утилита для обработки потока данных. С точки зрения AWK данные разбиваются на наборы полей, то есть наборы символов, разделённых разделителем. AWK — это практически полноценный язык программирования, в котором есть свои переменные, операторы выбора и циклы. AWK — родоначальник языка perl.

Cron и at

В ОС Linux есть два типа планировщиков: для выполнения разовых задач (например, перезагрузки сервера ночью) используют планировщик `at`, для выполнения задач с определённой периодичностью используют планировщик `cron`.

Cron и at

at

at позволяет планировать выполнение разовых задач в определённое время без редактирования конфигурационных файлов. Это управляющий демон планировщика atd.

Задача добавляется с использованием следующего синтаксиса: `at -f file_task date_time`. Здесь, используя параметр `-f`, мы говорим at, что задание надо прочитать из скрипта с именем `file`. `Data_time` — дата и время выполнения скрипта.

Cron и at

cron — программа-демон, предназначенная для выполнения заданий в определённое время или через определенные промежутки времени. Список заданий, которые будут выполняться автоматически в указанные моменты времени.

Формат расписания cron

min hour day month weekday user
command

- min (0-59) — минута;
- hour (0-23) — час;
- day (1-31) — день месяца;
- month (1-12) — день месяца;
- weekday (0-6) — день недели, 0 или 7 — воскресенье.

Спасибо!
Каждый день
вы становитесь
лучше :)

