Управляющие символы

< перенаправление ввода

> перенаправление вывода

| конвейер (результат левого как аргумент правому)

history | grep docker



>> добавляют в конец файла, не создают файл, если его нет

> перезаписывает файл, создает, если его нет

Баш скрипт начинается с шабенг строки

# комментраий

#! не комментарий, а шабе

Файлу можно дать права на управление с помощью chmod +x script.sh

./script.sh

Переменные

Без пробелов а=10

Обращаемся к переменной всегда с помощью $

echo $a

echo “a равно $a”

echo “у меня \$9”

$0 – имя скрипта

$1 и так далее – аргумент с которым запущен скрипт

$\* и $@ - все аргументы

$# - количество аргументов

Математические операции производятся в двойных скобочках

a=10

b=20

echo $(($a+$b))

Получение информации, команда read

Команда read умеет считывать ввод откуда угодно (т.е. не только с клавиатуры, а из файла и т.д)

(-p -prompt) read -p тоже что и в пайтоне input(“Текст:”)

(-s -secret) read -s считывает данные и не выводит их. В терминале не отображается то, что пишет пользователь

(-t -timer) read -t таймер, дает время на ввод, после чего продолжается скрипт, если не была введена информация, переменная будет пустой

Конструкция if

Мы пишем ; только если в одной строке несколько команд

&& чтобы задать 2 условия if

В баше необязательно ставить отступы

Если мы пишем одну команду в одной строке, то баш ориентируется на слова и в конструкции if она должна завершаться командой then

В else не требуется then так как это последняя операция в конструкции if

В конце fi

Флаги

-n проверяет наличие аргумента, если там нет данных, то он вернет false

-ea ==

-ne !=

-lt <

-le <=

-gt >

-ge >=