

Федеральное государственное бюджетное образовательное учреждение  
высшего образования

«РОССИЙСКАЯ АКАДЕМИЯ НАРОДНОГО ХОЗЯЙСТВА И ГОСУДАРСТВЕННОЙ СЛУЖБЫ ПРИ ПРЕЗИДЕНТЕ РОССИЙСКОЙ ФЕДЕРАЦИИ»

КОЛЛЕДЖ МНОГОУРОВНЕВОГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

Отчёт по работе с онлайн Linux.

По дисциплине «Системное программирование»

Специальность «Информационные системы и программирование»

Выполнила студентка ГРУППЫ 32ИС-21 Арутюнян С.К.

Проверил Гусятинер Л.Б.

Москва, 2024

**Bash-скрипты**

**Введение в bash-скрипты**

**12. Глава 12. Bash-скрипты**

* 1. Введение в bash-скрипты<https://metanit.com/os/linux/12.1.php>
  2. Переменные и получение ввода  
     <https://metanit.com/os/linux/12.2.php>
  3. Аргументы скрипта  
     <https://metanit.com/os/linux/12.3.php>
  4. Результат команд и операторов  
     <https://metanit.com/os/linux/12.4.php>
  5. Pattern matching  
     <https://metanit.com/os/linux/12.5.php>
  6. Математические операции  
     <https://metanit.com/os/linux/12.6.php>

Задание.

1. Прочитать
2. Выполнить примеры
3. Если ошибки – задать вопросы преподавателю
4. Поэкспериментировать (осторожно!)
5. Записать команды

Создадим простейший shell-скрипт. Для этого создадим в домашнем каталоге обычный текстовый файл, который назовем hello:



После создания файла hello определим в нем следующее содержимое:



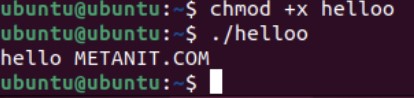
Есть несколько способов выполнить shell-скрипт:

Сделать скрипт исполняемым и запустить как обычную программу

Запустить скрипт как аргумент программы bash

Сделать скрипт исполняемым и запустить как обычную программу

Итак, воспользуемся первым способом и запустим наш скрипт hello как исполняемое приложение



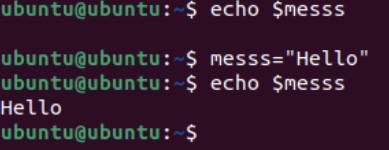
В данном случае команда chmod +x hello делает скрипт исполняемым.

При втором способе передаем скрипт команде bash:

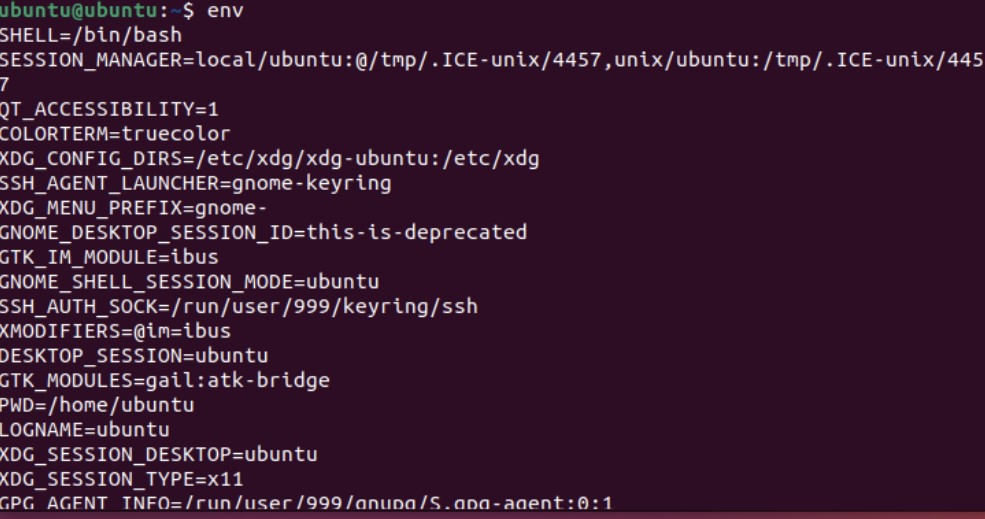


## Переменные и получение ввода

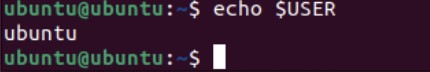
Причем определять переменные и работать с ними мы можем как в shell-скрипте, так и непосредственно в консоли:



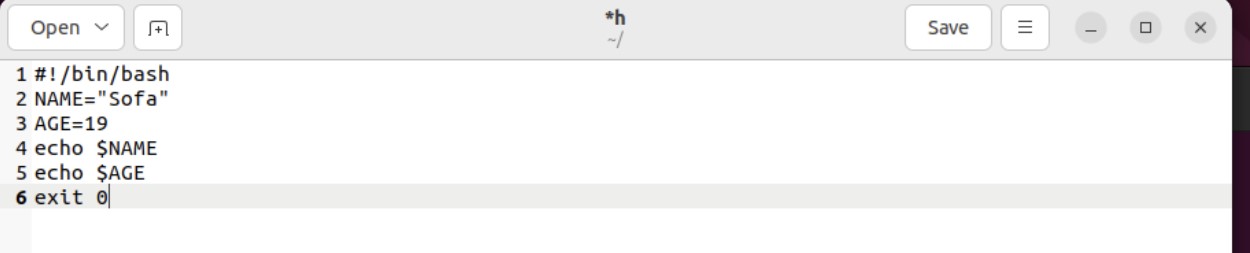
Стоит отметить, что система при загрузке уже устанавливает ряд глобальных переменных - переменных окружения. Эти переменные можно получить с помощью команды env:



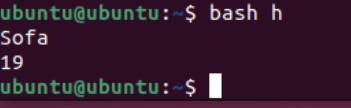
Например, среди переменных устанавливается глобальная переменная USER, которая хранит имя текущего пользователя. Выведем ее значение на консоль:



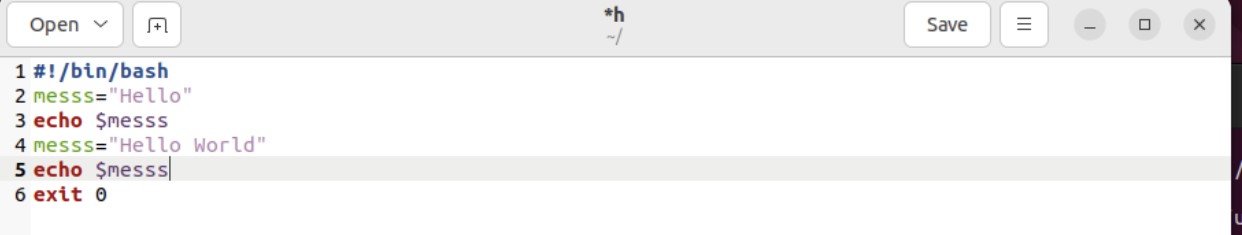
Аналогичным образом мы можем работать с несколькими переменными. Например, определим следующий shell-скрипт:

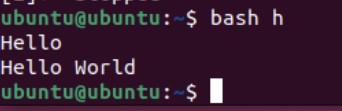


Здесь определяются две переменных: NAME и AGE, и затем их значения выводятся на консоль. Запустим скрипт:



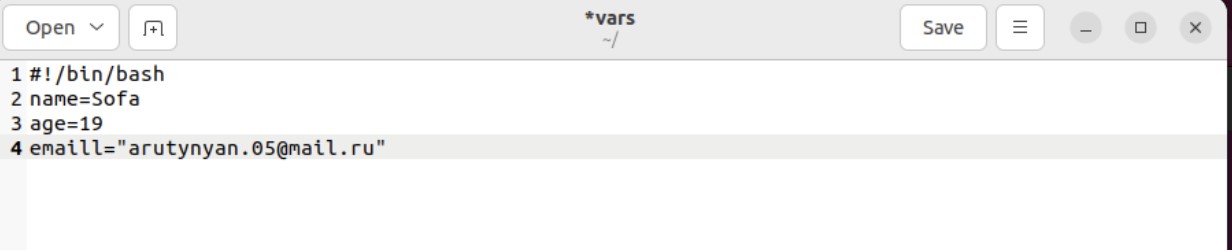
Отличительной особенностью переменных является то, что мы можем менять их значения. Например, определим следующий скрипт:



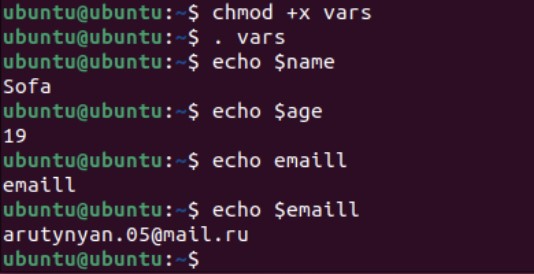


### **Встраивание переменных**

Расспространенной техникой при работе с переменными является их встраивание (sourcing) в текущую оболочку. В этом случае переменные часто определяются в одном файле. Например, определеим в домашнем каталоге файл vars со следующим содержимым:

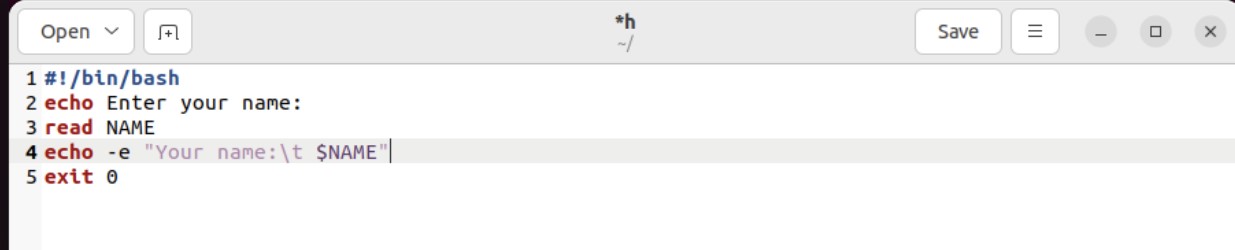


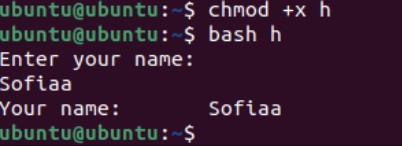
Здесь определены три переменных. Встроим файл в консоль и обратимся к переменным:



**Получение ввода**

С помощью команды read можно получить от пользователя некоторые данные в некоторую переменную. Например, определим следующий скрипт hello:

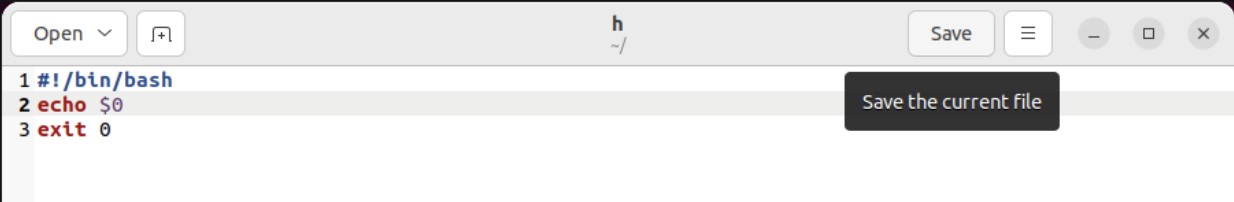


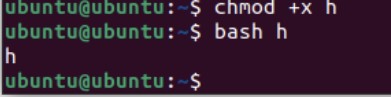


## Аргументы скрипта

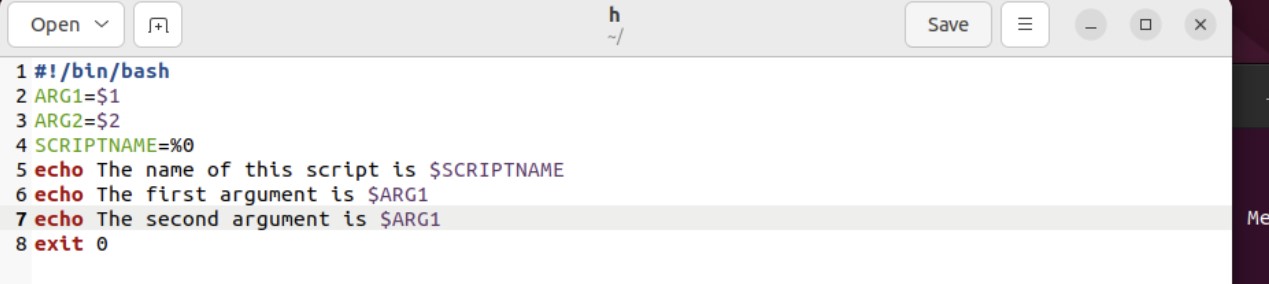
При запуске скрипта ему можно передать некоторые аргументы из командной строки. Внутри скрипта эти аргументы доступны через номера аргументов. Например, первый аргмуент доступен через $1, второй аргумент через - $2 и так далее. Также есть специальный идентификатор - $0, который всегда хранит имя скрипта.

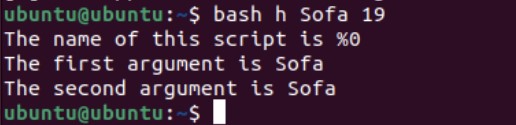
Например, определим следующий скрипт hello:





Подобным образом можно передать и большее количество аргументов. Например:





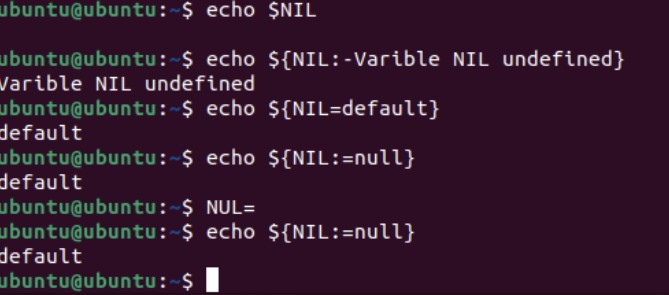
## Результат команд и операторов

Рассмотрим их применение. Начнем с того, что переменная в принципе может быть не установлена. Например:

Здесь переменная NIL не установлена, а команда echo для этой переменной выводит пустоту. Теперь применим операторы подстановки.

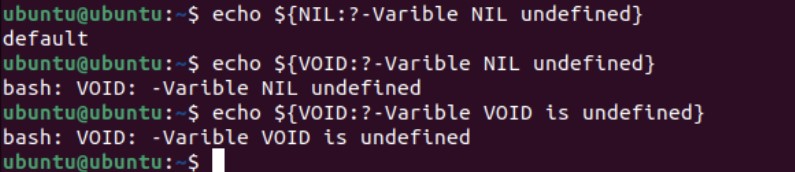
Оператор ${parameter:-value} выводит value, если переменная не установлена:Оператор ${parameter=value} устанавливает для переменной значение value, если эта переменная не установлена:

Оператор ${parameter:=value} присваивает переменной значение value, если переменная в данный момент не имеет значения или вообще не существует. Итак, выше для переменной NIL было установлено значение "default". То есть переменная NIL существует и имеет значение. Применим этот оператор к переменной NIL:

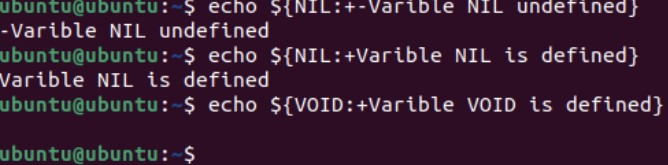


Оператор ${parameter:?value} показывает значение value, если параметр не существует или пуст.



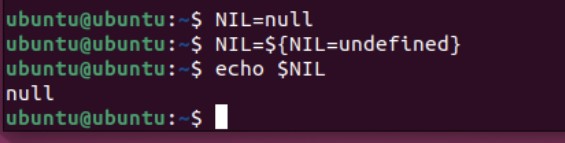


Оператор ${parameter:+value} показывает значение value, если переменная имеет значение:



Здесь проверяются две переменных - NIL и VOID. Поскольку ранее только NIL была определена, то только для нее отображается указанное сообщение.

И также результат этих операторов можно присваивать переменным:



## Pattern matching

Операторы сопоставления с шаблоном позволяют найти в переменной некоторый шаблон и, если этот этот шаблон найден, изменить переменную. Это следующие операторы:

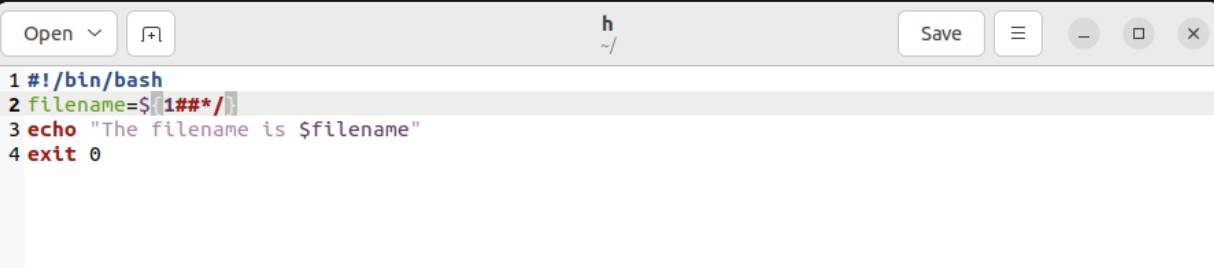
Оператор ## используется для поиска самого длинного соответствия шаблону с начала строки

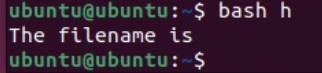
Оператор # используется для поиска самого короткого соответствия шаблону с начала строки

Оператор %% используется для поиска самого длинного соответствия шаблону с конца строки

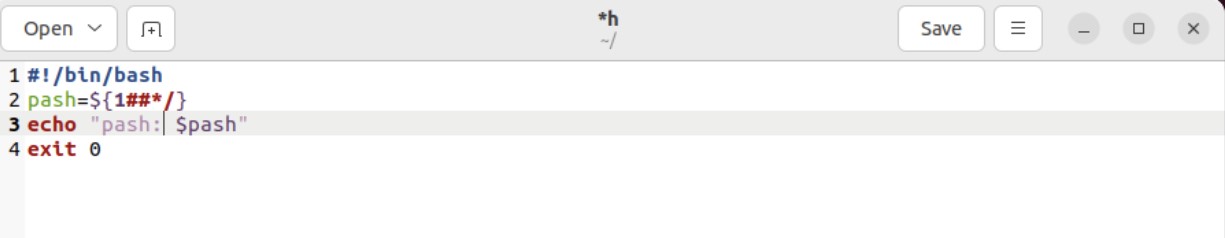
Оператор % используется для поиска самого короткого соответствия шаблону с конца строки

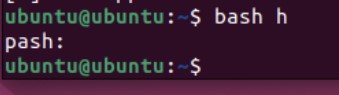
Например, определим в домашнем каталоге следующий скрипт hello:





Используя оператор #, можно найти самое короткое соответствие. Например, изменим скрипт следующим образом:





## Математические операции

## Bash поддерживает выполнение простейших математических операций. В частности:

## +

## Сложение (5 + 4 = 9

## -

## Вычитание (5 - 4 = 1)

## \*

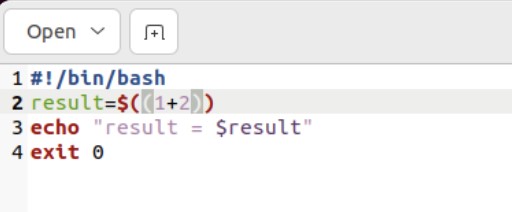
## Умножение (5 \* 4 = 20)

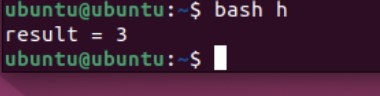
## /

## Деление (5 / 4 = 1)

## %

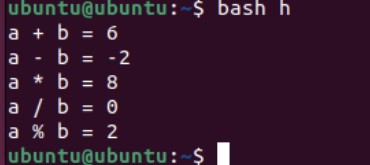
## Получение остатка от деления (5 % 4 = 1)





Теперь используем в скрипте переменные:





Есть также и другой способ определения математических операций - с помощью команды expr:

