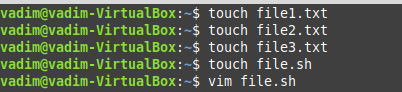
## Сценарии Bash

### Задания к работе

1. Вывести количество файлов в домашнем каталоге, которые заканчиваются на .txt. Создайте несколько таких файлов для тестирования.

Создадим файлы:



Сам скрипт:



Листинг:

#!/bin/bash

find ~ -maxdepth 1 -type f -name "\*.txt" | wc -l

Где:

Команда find ищет объекты (файлы, папки)

~ указывает, что точка поиска будет – домашний каталог

-maxdepth 1 указывает глубину поиска (то есть ищем только в текущем каталоге)

-type f указывает тип искомого объекта – то есть файлы

-name имя (либо маска в кавычках) по которой ищем объекты.

\*.txt указывает, что имя файла должно начинаться с любых символов, а заканчиваться на .txt

| (вертикальная черта) передает вывод команды find на ввод команды wc

Wc считает количество байт, символов, строк переданного ей многострочного потока данных (обычно файл)

Опция –l указывает, что считать будем только строки, то есть количество файлов с расширением .txt в домашнем каталоге, так как каждый файл будет выводиться в отдельной строке в результате выполнения предыдущей команды

Проверим работу скрипта:



1. Вывести текущие переменные окружения в отсортированном по алфавиту порядке.

Сам скрипт:



Листинг:

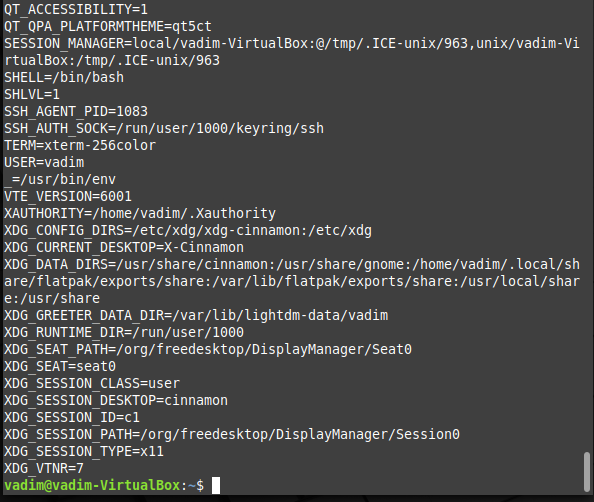
#!/bin/bash

env | sort

Где:

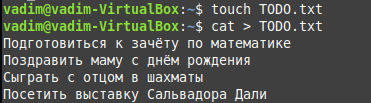
команда env, выполненная без параметров, выведет все переменные окружения и передаст поток вывода команде sort, которая также без параметров (по умолчанию) отсортирует строки в алфавитном порядке

Проверим работу скрипта:

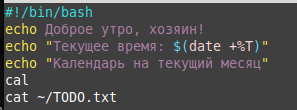


1. Разработать программу "Good morning", которая:
   1. Пожелает пользователю доброго утра.
   2. Выведет текущее время и календарь на текущий месяц.
   3. Выведет список дел из файла TODO домашнего каталога пользователя.

Создадим файл TODO и запишем в него список дел:



Скрипт:



Листинг:

#!/bin/bash

echo Доброе утро, хозяин!

echo "Текущее время: $(date +%T)"

echo "Календарь на текущий месяц"

cal

cat ~/TODO.txt

Где:

echo выведет на экран приветственно сообщение

echo выведет строку в которую передастся результат вывода команды date +%T благодаря тому, что она указана в $() (синтаксис выполнения команды):

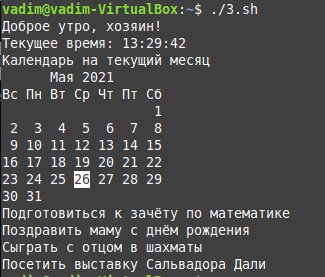
где date команда вывода даты времени + обозначает формат %T, то есть формат вывода времени в виде HH:MM:SS

Далее echo выводит сообщение о текущем календаре на месяц

cal выводит календарь на любой месяц года (текущий по умолчанию, поэтому без параметров)

cat ~/TODO.txt выведет содержимое файла TODO из домашнего каталога на экран

Проверим работу скрипта:



1. Найти и вывести пути до файлов из каталога /usr (включая подкаталоги), размер которых больше 20 Мб. Подсказка: man find.

Скрипт:



Листинг:

#!/bin/bash

find /usr -type f -size +20M

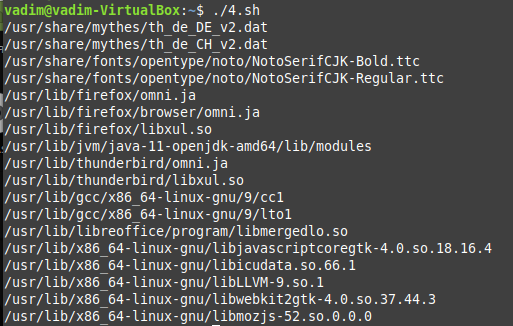
Где:

find ищет файлы в каталоге /usr (и подкаталогах /usr/);

-type f ищет только файлы;

-size указывает, что будем осуществлять поиск по размеру + указывает, что производится поиск файлов, размер которых больше 20 M (мегабайт)

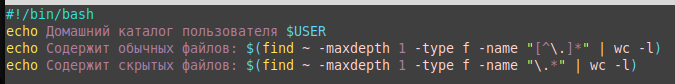
Проверим работу скрипта:



1. Подсчитать количество файлов, количество скрытых файлов в домашнем каталоге текущего пользователя и вывести результат в формате:

Домашний каталог пользователя  
<User>  
содержит обычных файлов:  
XX  
скрытых файлов:  
YY

Скрипт:



Листинг:

#!/bin/bash

echo Домашний каталог пользователя $USER

echo Содержит обычных файлов: $(find ~ -maxdepth 1 -type f -name "[^\.]\*" | wc -l)

echo Содержит скрытых файлов: $(find ~ -maxdepth 1 -type f -name "\.\*" | wc -l)

Где:

$USER содержит в себе имя текущего пользователя

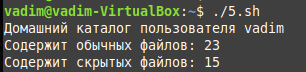
Find –maxdepth 1 –type f найдет файлы в домашнем каталоге с маской имени –name, где ^ обозначает не содержит символ

\. обозначает символ «точка», так как сама по себе точка обозначает любой символ, и имея в виду под точкой именно точку, нужно ее экранировать \. То есть [^\.] обозначает, что первый символ не точка, а далее \* любые символы в имени файла. Так как с точки начинаются имена скрытых файлов

Далее, наоборот, ищем файлы, начинающиеся с точки, так как нам нужно количество скрытых файлов

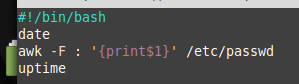
Аналогично заданиям выше, количество считаем при помощи wc -l (Wc считает количество байт, символов, строк переданного ей многострочного потока данных (обычно файл). Опция -l указывает, что считать будем только строки, то есть количество файлов в домашнем каталоге)

Проверим работу скрипта:



1. Вывести на экран дату, время, список зарегистрировавшихся пользователей, uptime системы.

Скрипт:



Листинг:

#!/bin/bash

date

awk -F : '{print$1}' /etc/passwd

uptime

Где:

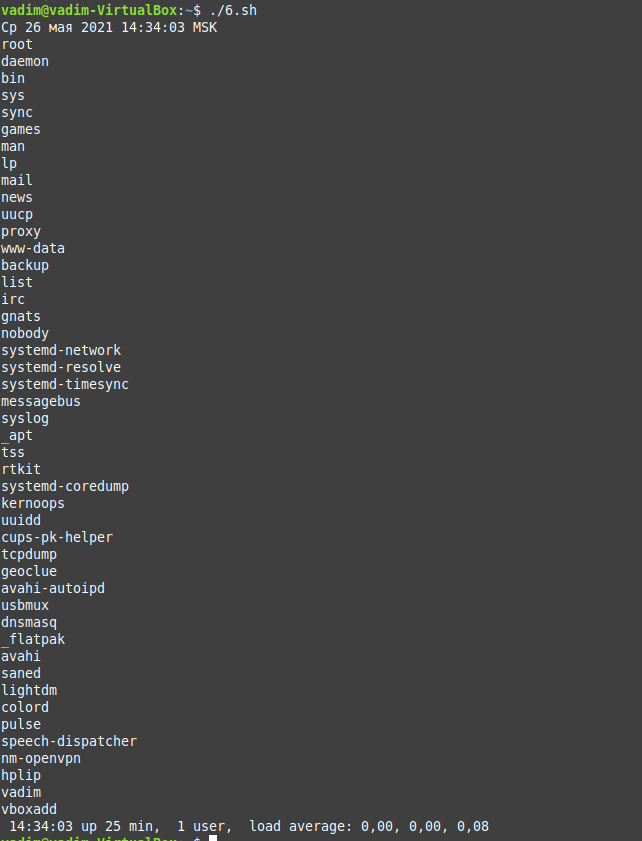
date выводит дату время

awk в качестве разделителя будет использовать : так как мы указали -F (**--field-separator** - разделитель полей, используется для разбиения текста на колонки) и :

‘{print$1}’ указывает, что выведено будет первое поле и обрабатываться будет файл /etc/passwd где хранится список зарегистрированных пользователей.

Uptime выводит текущее время и время прошедшее с момента включения. Также мы могли бы использовать uptime –p (вывод только времемени с момента включения)

Проверим работу скрипта:



7. Вывести количество процессов, запущенных от имени текущего и от имени пользователя root в формате:

Процессов пользователя:  
<User>  
XX  
Процессов пользователя root:  
YY

Скрипт:



Листинг:

#!/bin/bash

echo Процессов пользователя $USER: $(ps -u $USER | wc -l)

echo Процессов пользователя root: $(ps -u root | wc -l)

Где:

ps выведет процессы, а именно с помощью –u процессы указанного конкретного пользователя. Далее мы передаем имя пользователя из переменной $USER

| wc -l считает их количество (количество процессов)

Проверим работу скрипта:



8. Найти и вывести 5 процессов, потребляющих больше всего памяти в системе. Подсказка: man ps

Скрипт:



Листинг:

#!/bin/bash

ps aux --sort -%mem | head -6

Где:

ps aux выводит все процессы всех пользователей

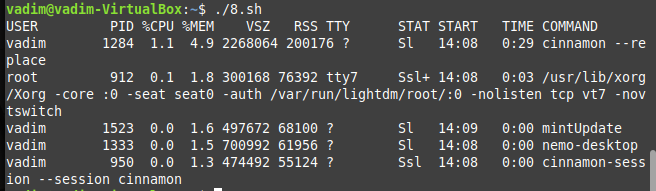
-sort указывает критерий сортировки

Сортируем по полю %mem (RAM)

-%mem указывает сортировку по убыванию ( %mem указывает по возрастанию)

| head -6 выводит только первые 6 строк (1 строка заголовка и 5 строк наиболее потребляющих процессов)

Проверим работу скрипта:



9. Вывести файлы и каталоги из домашнего каталога пользователя, упорядочив их по размеру. Подсказка: использовать команды du и sort.

Скрипт:



Листинг:

#!/bin/bash

du -d 1 -a ~ | sort -k 1 -n

Где:

du выводит размер каталогов (по умолчанию для текущего каталога вместе с всей вложенностью)

-d указывает макс вложенность, 1 - только в текущей папке

-a выводит размер не только каталогов, но и файлов

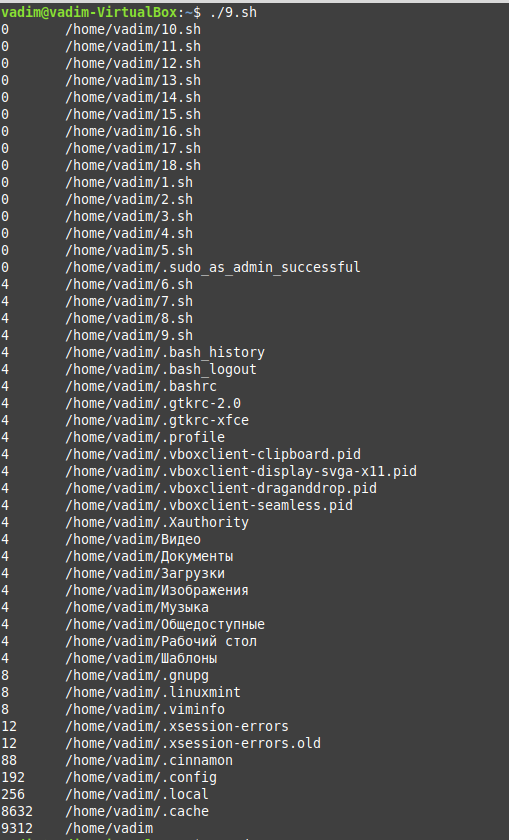
~ это домашняя директория

То есть du –d 1 –a ~ выведет размер файлов и папок только домашней директории

Sort отсортирует эту информацию –k 1 по первому столбцу (по умолчанию разделитель у нас это пробел)

–n сортирует по числам (размер в числах)

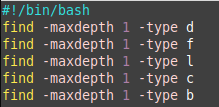
Проверим работу скрипта:



10. Разработать сценарий, который выводит файлы из текущего каталога в следующем порядке:

1. Каталоги.
2. Обычные файлы.
3. Символьные ссылки.
4. Символьные устройства.
5. Блочные устройства.

Скрипт:



Листинг:

#!/bin/bash

find -maxdepth 1 -type d

find -maxdepth 1 -type f

find -maxdepth 1 -type l

find -maxdepth 1 -type c

find -maxdepth 1 -type b

Где:

find –maxdepth 1 обозначает поиск в текущем каталоге без вложенности

type d –directory это директории

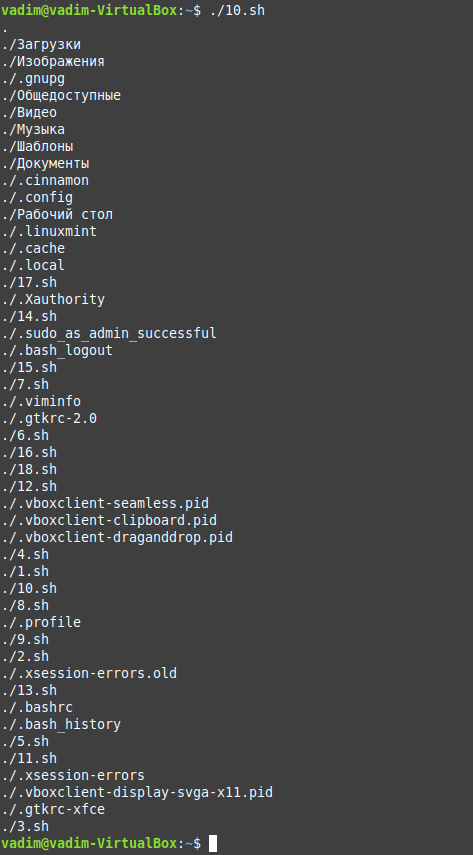
f – regular files это обычные файлы

l – symbolic link это символьные ссылки

c – character special это символьные устройства

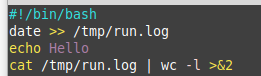
b – block special это блочные устройства

Проверим работу скрипта:



11. Разработать сценарий, который ведёт в файле /tmp/run.log журнал запусков. При каждом запуске сценария в конец журнала должна добавляться строка с датой и временем запуска сценария, в стандартный вывод - фраза "Hello", в stderr - количество предыдущих запусков программы. Убедиться в правильности работы программы и выводе различных сообщений в различные потоки вывода.

Скрипт:



Листинг:

#!/bin/bash

date >> /tmp/run.log

echo Hello

cat /tmp/run.log | wc -l >&2

Где:

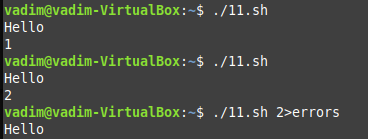
stdout - стандартные потоки ввода-вывода

date выводит дату, время и поток вывода перенаправляется в конец файла /tmp/run.log (если файла нет он создастся)

далее выведем Hello, вывод будет произведен в stdout (по умолчанию)

cat /tmp/run.log выведет содержимое файла, а wc –l посчитает количество строк, и этот вывод перенаправляется с помощью >& во 2 поток то есть stdout

Проверим работу скрипта:



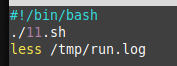
Здесь, 2> перенаправляет второй поток вывода в файл errors, и, соответственно, скрипт выводит только Hello



Файл errors уже содержит число, которое не было выведено. Следовательно, по умолчанию, второй поток вывода так же выводится на экран

12. Разработать сценарий, который запускает сценарий, разработанный в предыдущем задании, а также открывает журнал запусков предыдущего сценария в программе less.

Скрипт:



Листинг:

#!/bin/bash

./11.sh

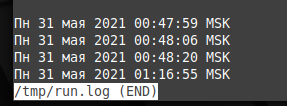
less /tmp/run.log

Где:

./11.sh запускает файл предыдущего скрипта (но только если он в этой же директории)

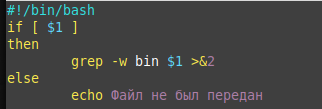
less откроет файл /tmp/run.log

Проверим работу скрипта:



13. Разработать сценарий, который может принимать на стандартный ввод текстовый файл и выводит строки из этого файла, содержащие слово bin целиком в стандартный вывод ошибок.

Скрипт:



Листинг:

#!/bin/bash

if [ $1 ]

then

grep -w bin $1 >&2

else

echo Не был передан файл!

fi

Где:

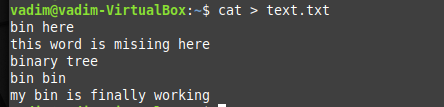
if [ $1 ] проверяет есть ли переменная $1

Если переменная $1 пустая - значит файл не был передан и далее выводится сообщение об этом.

Если непустая, то grep ищет –w слово, то есть символы, обрамленные пробелом, искомое слово – bin, файл поиска хранится в переменной $1, и поток вывода перенаправляется в поток 2 stderr

Проверим работу скрипта:

Перед этим создадим тестовый текстовый файл и запишем в него текст:

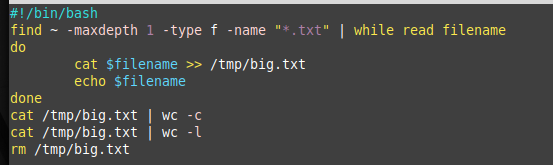


Починить ошибку с Никитой!



14. Создать во временном каталоге /tmp файл big.txt, который будет состоять из всех файлов с расширением txt, находящихся в домашнем каталоге пользователя. Вывести список файлов, из которых он состоит и его размер: в байтах и в строках. По окончании работы удалить файл.

Скрипт:



Листинг:

#!/bin/bash

find ~ -maxdepth 1 -type f -name "\*.txt" | while read filename

do

cat $filename >> /tmp/big.txt

echo $filename

done

cat /tmp/big.txt | wc -c

cat /tmp/big.txt | wc -l

rm /tmp/big.txt

Где:

find ~ -maxdepth 1 -type f -name "\*.txt" найдет все файлы с расширением txt в домашней папке. Передадим вывод в цикл while read который построчно читает в переменную filename

Затем, мы открываем файл cat $filename и дозаписываем его содержимое в файл /tmp/big.txt (если его нет, то он создастся)

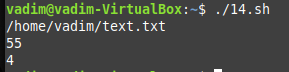
и выводим на экран имя файла echo $filename

далее завершаем цикл done

выводим содержимое созданного файла и передаем это в wc (эта утилита выводит число переводов строк, слов и байт для каждого указанного файла и итоговую строку, если было задано несколько файлов). Здесь –c выводит количество байт, –l количество строк

удаляем файл с помощью команды rm /tmp/big.txt

Проверим работу скрипта:



15. Разработать сценарий, который выводит на экран количество переданных ему аргументов.

Скрипт:



Листинг:

#!/bin/bash

echo $#

Где:

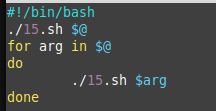
$# - специальный параметр в bash, который дает число позиционных параметров в десятичном формате.

Проверим работу скрипта:



16. Разработать сценарий, который вызывает предыдущий сценарий дважды: первый раз с объединенным полным списком аргументов, второй раз - со списком всех переданных ему аргументов по отдельности.

Скрипт:



Листинг:

#!/bin/bash

./15.sh $@

for arg in $@

do

./15.sh $arg

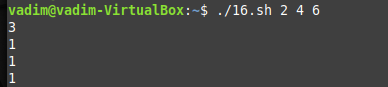
done

Где:

$@ передает одной строкой в ./15.sh все аргументы, переданные в текущий скрипт

В цикле for обрабатываем каждый аргумент, переданный в текущий скрипт, отдельно и по одному передаем в 15 скрипт

Проверим работу скрипта:



17. Разработать сценарий, который вызывает команду grep и принимает следующие аргументы:

* текст, который нужно найти;
* файл, в котором нужно найти этот текст;
* максимальное количество строк, которое нужно вывести на экран.

Вывод команды grep отсортировать и пронумеровать.

Скрипт:



Листинг:

#!/bin/bash

grep $1 $2 | sort | cat -n | head -$3

Где:

grep инструмент для поиска вхождений в строку

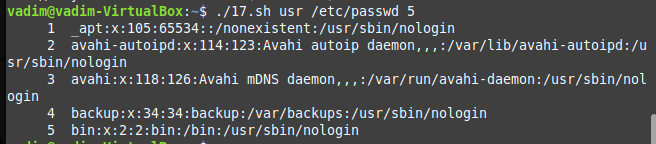
$1 передаст текст, который нужно найти, $2 передаст файл поиска

sort отсортирует по алфавиту

cat -n пронумерует выводимые строки

head -$3 выведет только $3 строк

Проверим работу скрипта:



18. Разработать сценарий, который выводит в одну строку имя пользователя, его домашний каталог, а также количество символов в этих двух переменных. Например: root /root 9. Подсказка: изучите аргументы команды echo, wc, математические вычисления в Bash - $(()).

Скрипт:



Листинг:

#!/bin/bash

echo $USER, $HOME, $(echo -n $USER$HOME | wc -m)

Где:

$USER имя пользователя

$HOME домашний каталог пользователя

Echo $USER$HOME выводит домашний каталог пользователя и имя вместе, при помощи wc – m считаем количество символов. Однако, так как echo добавляет в конец символ перевода строки, этот символ также посчитается, поэтому мы вызываем echo с параметром –n

Проверим работу скрипта:



### Указания к работе

1. Каждое задание следует сделать в отдельном файле и назвать его N.sh, где N - номер задания по методичке.