**Введение**

Цель работы – поближе познакомиться с оболочкой bash и используемыми в повседневной жизни it-специалистами утилитами. Данная курсовая работа была сделана в операционной системе Linux. В данной работе мы подробно рассмотрим команды категории scripting.

**Linux** **–** семейство [Unix-подобных операционных систем](https://ru.wikipedia.org/wiki/Unix-подобная_операционная_система) на базе [ядра Linux](https://ru.wikipedia.org/wiki/Ядро_Linux), включающих тот или иной набор утилит и программ проекта [GNU](https://ru.wikipedia.org/wiki/GNU), и, возможно, другие компоненты. Традиционно системами Linux считаются только те, которые включают в качестве компонентов основные программы [проекта GNU](https://ru.wikipedia.org/wiki/Проект_GNU), такие как [bash](https://ru.wikipedia.org/wiki/Bash), [gcc](https://ru.wikipedia.org/wiki/Gcc), [glibc](https://ru.wikipedia.org/wiki/Glibc), [coreutils](https://ru.wikipedia.org/wiki/Coreutils), [GNOME](https://ru.wikipedia.org/wiki/GNOME) и ряд других, в связи с чем часто всё семейство иногда идентифицируется как ***GNU/Linux***.

**Bash** – усовершенствованная и модернизированная вариация [командной оболочки](https://ru.wikipedia.org/wiki/Командная_оболочка_UNIX) [Bourne shell](https://ru.wikipedia.org/wiki/Bourne_shell). Одна из наиболее популярных современных разновидностей командной оболочки [UNIX](https://ru.wikipedia.org/wiki/UNIX). Особенно популярна в среде [Linux](https://ru.wikipedia.org/wiki/Linux), где она часто используется в качестве предустановленной командной оболочки. Bash — это командный процессор, работающий, как правило, в интерактивном режиме в текстовом окне. Bash также может читать команды из файла, который называется [*скриптом*](https://ru.wikipedia.org/wiki/Скрипт) (или *сценарием*). Как и все Unix-оболочки, он поддерживает автодополнение имён файлов и директорий, подстановку вывода результата команд, переменные, контроль за порядком выполнения, операторы ветвления и цикла. Ключевые слова, синтаксис и другие основные особенности языка были заимствованы из sh. Оболочка в первую очередь предназначена для вызова команд и программ Linux. Таким образом, она является своего рода интерпретатором команд (сравнимым с команда.com из мира MS-DOS). Оболочка выполняется в любом командном окне, например konsole, gnome-terminal, xterm, и в любой текстовой консоли после входа в систему (логина).

В то же время оболочка является мощным языком программирования, с помощью которого можно автоматизировать рабочие процессы. Особые оболочковые команды позволяют использовать в рамках этой программы переменные, создавать запросы и циклы и т. д.

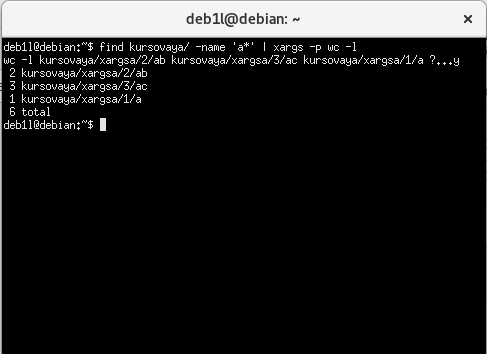
**Scripting**

**Xargs**

Программа берет данные стандартного ввода или из файла, разбивает их в соответствии с указанными параметрами, а затем передает другой программе в качестве аргумента.

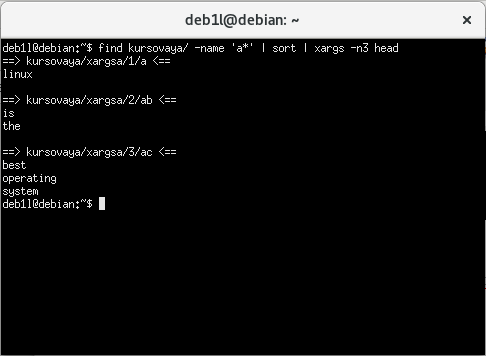
**Задача:** найти файлы с названием ‘a\*’ в заданной директории и вывести количество строк в них, запрашивая подтверждение перед выполнением.

|  |
| --- |
| find kursovaya/ -name ‘a\*’ | xargs -p wc –l !!! |



**Задача:** найти файлы с названием ‘a\*’ в заданной директории, отсортировать по имени и вывести первые три строки.

|  |
| --- |
| find kursovaya/ -name ‘a\*’ | sort | xargs -n3 head |



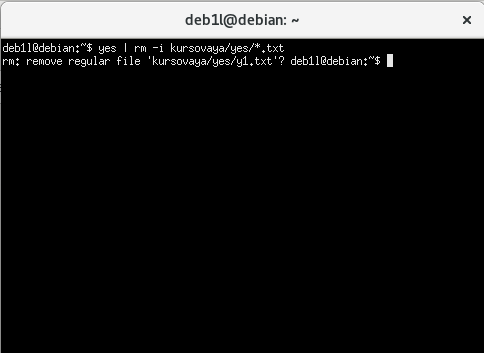
**Yes**

Команда **yes** многократно выводит строку STRING разделяемую символом новой строки (newline) на стандартный вывод, до тех пор, пока не будет принудительно завершена. Если строка не задана, выводится символ **y** и символ новой строки. Может использоваться для ответа “yes” командам, запрашивающим подтверждение на выполнение. Пример этого указан далее. Такая конструкция использовалась ранее, когда не было ключа –f, игнорирующего запросы и ошибки.

**Задача:** удалить файлы из заданной директории.

rm удаляет заданные файлы, каждый раз запрашивая подтверждение

|  |
| --- |
| yes | rm -i kursovaya/yes/\*.txt |



|  |
| --- |
| yes ab |



**Watch**

С заданной периодичностью запускает другие программы, указанные в качестве аргументов.

-d – подсвечивает изменения (при выводе на экран большого кол-ва данных, сразу видны те, которые изменились)

-n – позволяет указать время обновления в секундах

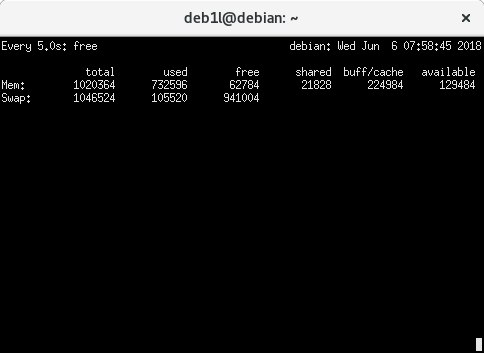
**Задача:** пронаблюдать за расходом оперативной памяти в режиме реального времени, подсвечивая изменения.

|  |
| --- |
| watch -d free |



**Задача:** пронаблюдать за расходом оперативной памяти в режиме реального времени с частотой обновления данных 5 с.

|  |
| --- |
| watch –n 5 free |



**Export**

Экспорт переменных и функций текущего процесса в дочерний процесс.

-p – выводит список всех имен, экспортированных в текущей оболочке

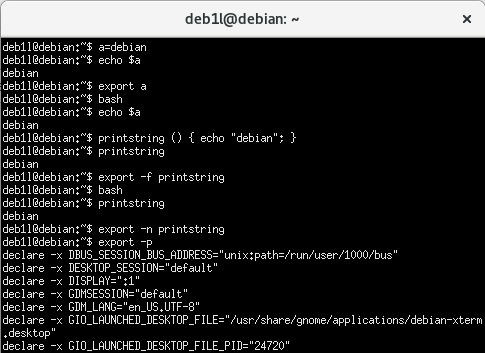
-n – удаляет имена из списка экспорта

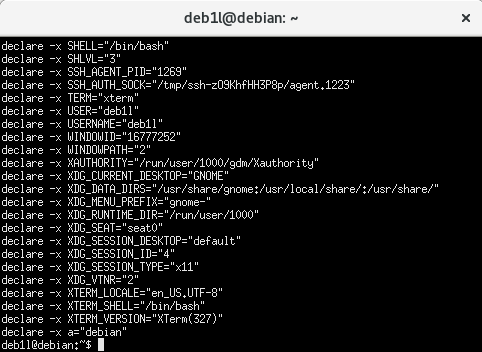
-f – имена экспортируются как функции

**Задача:** создать переменную a, экспортировать ее, создать функцию printstring и также экспортировать, проверить работоспособность вышеуказанных имен в дочерних процессах, затем удалить функцию и вывести на экран все экспортированные имена, дабы убедиться в том, что функция удалена, а переменная все еще существует.

|  |
| --- |
| a=debian |
| export a |
| printstring () { echo “debian”; } |
| export –f printstring |
| bash |
| export –n printstring |
| export –p |

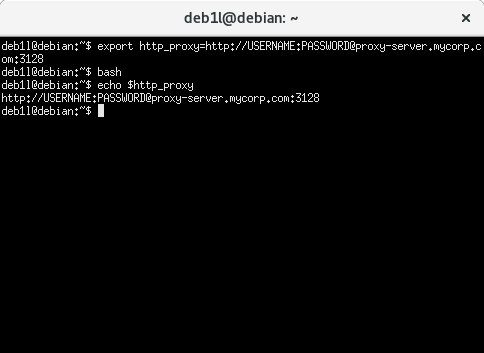
Export http\_proxy=”http://...”





**Задача:** настроить прокси-сервер для работы через него в командной строке.

|  |
| --- |
| export http\_proxy=http://USERNAME:PASSWORD@proxy-server.mycorp.com:3128 |



В переменную окружения записываются все данные прокси-сервера. <http://имя_пользователя:пароль@адрес_сервера:порт>

|  |
| --- |
| export -p |



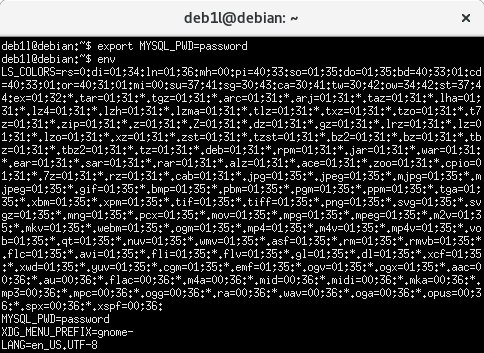
**Env**

Окружение или среда – набор пар переменная=значение, доступный каждому

пользовательскому процессу.

**Задача:** записать MYSQL\_PWD в окружение и вывести окружение на экран.

|  |
| --- |
| export MYSQL\_PWD=password |
| env |



**Seq**

**Задача:** вывести последовательность чисел от 2 до 20 с шагом 2.

|  |
| --- |
| seq 2 2 20 |

seq [от] [шаг] [до]



**Задача:** вывести числа от 2 до 4 с шагом 0.5 в:

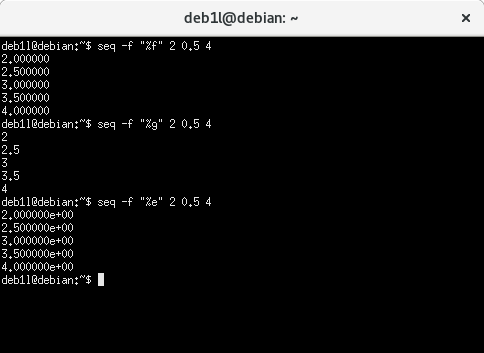
а) формате f

б) формате g

в) формате e

|  |
| --- |
| seq –f “%f” 2 0.5 4 |
| seq –f “%g” 2 0.5 4 |
| seq –f “%e” 2 0.5 4 |

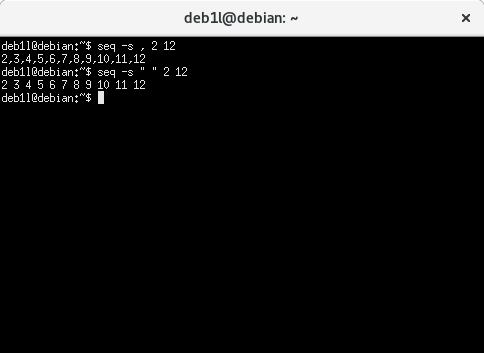
-f – задаёт формат (по умолчанию %g).



**Задача:** вывести числа от 2 до 12, разделив их запятой в первом случае, пробелом - во втором.

|  |
| --- |
| seq –s , 2 12 |
| seq –s “ “ 2 12 |

-s – данный ключ позволяет записывать числа, используя разделяющий символ.



**Задача:** вывести числа от 1 до 20 в столбец равной ширины.

|  |
| --- |
| seq –w 20 |

-w – выравнивает ширину, заполнив пробелы нулями слева.



**Sed**

Редактор потока данных для автоматического редактирования текстов.

**Задача:** заменить первые встретившиеся в указанном файле прописные буквы «a» и «b» на заглавные.

|  |
| --- |
| sed –e ‘s/a/A/’ –e ‘s/b/B/’ |

-e – способ выполнения нескольких команд.



**Задача:** заменить прописные гласные буквы на заглавные.

|  |
| --- |
| sed –f “script” “file” |

-f – применяет команды из указанного файла (скрипта).

Скрипт:

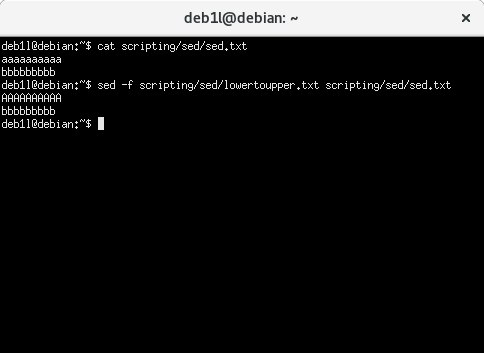
s/a/A/g

s/e/E/g

s/i/I/g

s/o/O/g

s/u/U/g



**Задача:** из указанного файла вывести только строку, содержащую букву «b».

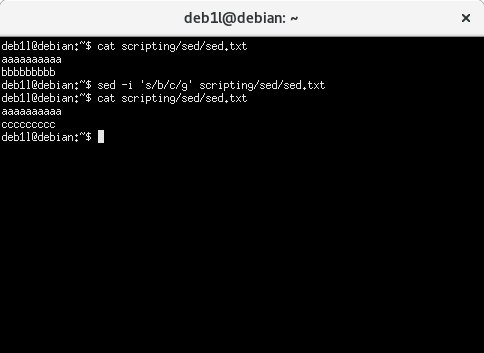
|  |
| --- |
| sed –n ‘s/b/&/p’ |

-n – ничего не выводит на стандартный выход. Можно использовать с модификатором /p для вывода отдельных строк из файла.



**Задача:** заменить все буквы «b» на буквы «c» с заменой исходного файла.

|  |
| --- |
| sed –i ‘s/b/c/g’ |



**Sleep**

Приостанавливает скрипт на указанное время. По умолчанию время указывается в секундах, для минут после числа пишут m, для часов h, для дней d.

**Задача:** приостановить работу терминала на 5 секунд, минут, часов, дней.

|  |
| --- |
| sleep 5 |
| sleep 5m |
| sleep 5h |
| sleep 5d |

**base64**

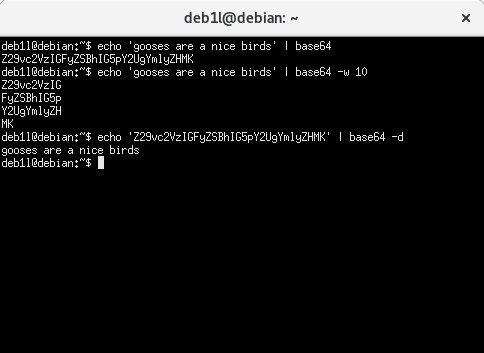
Кодирует или декодирует файл и выводит на стандартный выход

**Задача:** закодировать и декодировать строку, закодированную строку разбить на строки по 10 символов.

|  |
| --- |
| echo ‘gooses are a nice birds’ | base64 |
| echo ‘gooses are a nice birds’ | base64 –w 10 |
| echo ‘Z29vc2VzIGFyZSBhIG5pY2UgYmlyZHMK’ | base64 –d |

-w – переносит закодированную строку через указанное количество символов (по умолчанию 76). Для отключения переноса использовать 0

-d – декодирует данные



**Заключение**

В данной работе мы описали работу некоторых скриптов и научились применять их в работе с bash на Linux.

Список используемой литературы

1. Рафаэль Херцог, Ролан Ма «Настольная книга администратора Debian». (2016)
2. Дэниел Барретт «Карманный путеводитель по Linux» 3-е изд. (2016)
3. Ubuntu Linux Toolbox: 1000+ Commands for Power Users, 2nd Edition by Christopher Negus. (2013)
4. Аоки О. «Справочник по Debian». (2007)
5. Колисниченко Д. «Linux от новичка к профессионалу» 2-ое изд. (2010)