# РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

## Факультет физико-математических и естественных наук

### Кафедра прикладной информатики и теории вероятностей

## ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 11

### *дисциплина: Операционные системы*

Студент: Ким Реачна Группа: НПИбд-02-20

Москва 2021г.

### Цель работы:

Изучить основы программирования в оболочке ОС UNIX/Linux. Научиться писать небольшие командные файлы.

### Теоретическое введение:

**Командные процессоры (оболочки):**

Командный процессор (командная оболочка, интерпретатор команд shell) — это программа, позволяющая пользователю взаимодействовать с операционной системой компьютера. В операционных системах типа UNIX/Linux наиболее часто используются следующие реализации командных оболочек:

* *оболочка Борна (Bourne shell или sh)* — стандартная командная оболочка UNIX/Linux, содержащая базовый, но при этом полный набор функций;
* *С-оболочка (или csh)* — надстройка на оболочкой Борна, использующая Сподобный синтаксис команд с возможностью сохранения истории выполнения команд;
* *оболочка Корна (или ksh)* — напоминает оболочку С, но операторы управления программой совместимы с операторами оболочки Борна;
* *BASH* — сокращение от Bourne Again Shell (опять оболочка Борна), в основе своей совмещает свойства оболочек С и Корна (разработка компании Free Software Foundation).

*POSIX* (Portable Operating System Interface for Computer Environments) — набор стандартов описания интерфейсов взаимодействия операционной системы и прикладных программ.

Стандарты POSIX разработаны комитетом IEEE (Institute of Electrical and Electronics Engineers) для обеспечения совместимости различных UNIX/Linux подобных операционных систем и переносимости прикладных программ на уровне исходного кода. POSIX-совместимые оболочки разработаны на базе оболочки Корна.

**Переменные в языке программирования bash:**

Командный процессор bash обеспечивает возможность использования переменных типа строка символов. Имена переменных могут быть выбраны пользователем. Пользователь имеет возможность присвоить переменной значение некоторой строки символов. Например, команда

mark=/usr/andy/bin

присваивает значение строки символов /usr/andy/bin переменной mark типа *строка символов*.

Значение, присвоенное некоторой переменной, может быть впоследствии использовано. Для этого в соответствующем месте командной строки должно быть употреблено имя этой переменной, которому предшествует метасимвол $. Например, команда

mv afile ${mark}

Команда echo в Linux используется для отображения строки текста/строки, которые передаются в качестве аргумента . Это встроенная команда, которая в основном используется в сценариях оболочки и пакетных файлах для вывода текста состояния на экран или в файл.

echo [string]

Команда read принимает ввод с клавиатуры и присваивает его переменной.

read [options] [name...]

### Выполнение работы:

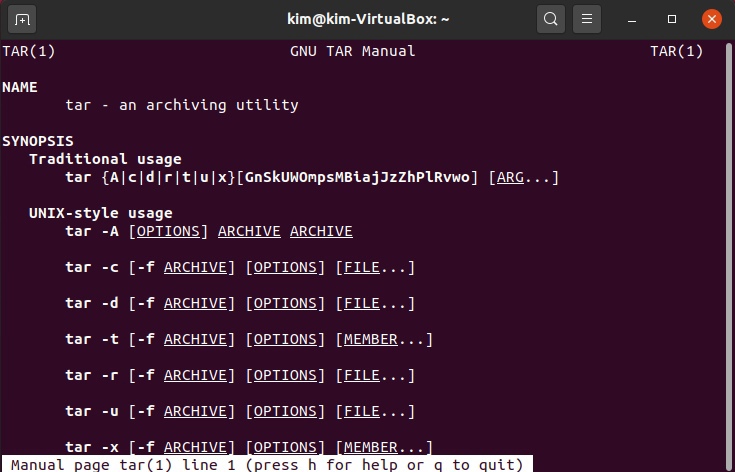
**Задание1:** Написать скрипт, который при запуске будет делать резервную копию самого себя (то есть файла, в котором содержится его исходный код) в другую директорию backup в вашем домашнем каталоге. При этом файл должен архивироваться одним из архиваторов на выбор zip, bzip2 или tar. Способ использования команд архивации необходимо узнать, изучив справку.

* Сначала мы изучаем команду tar с помощью командной man tar *(Рисунок 1-2)*

*Рисунок 1: изучаем команду tar*

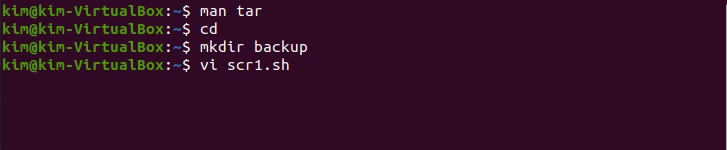


*Рисунок 2: изучаем команду tar*



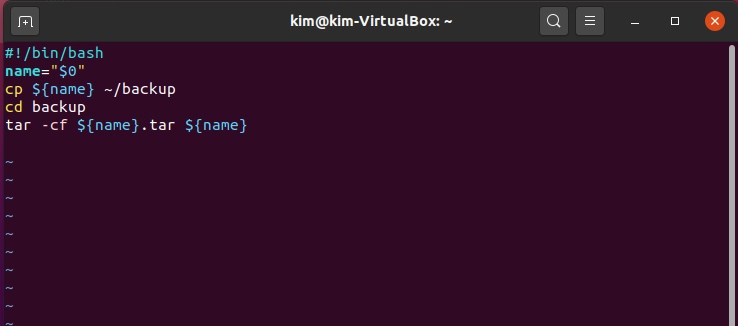
* В этом процессе мы сначала переходим в домашний каталог с помощью команды cd, затем создаем backup каталога в домашнем каталоге с помощью команды mkdir, а затем создаем сценарий, который мы собираемся записать с помощью редактора Vi который называется scr1.sh (*Рисунок 3*)

*Рисунок 3: Создание файла и каталога*



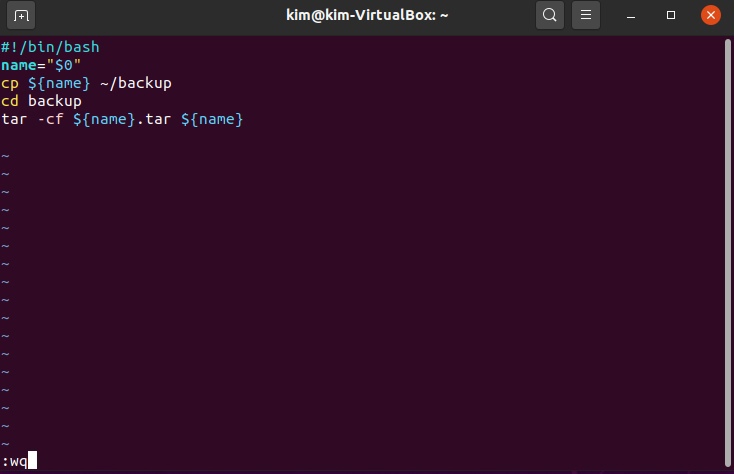
* В редакторе scr1.sh мы пишем командный файл , который мы начнем с #!/bin/bash в первой строке, затем используем команду cp для перемещения или копирования файла в резервную копию каталога, затем меняем каталог на резервную копию, а затем с помощью команды tar объединяем несколько файлов и или каталогов вместе в один файл.(*Рисунок 4*)

*Рисунок 4: Создать командный файл*

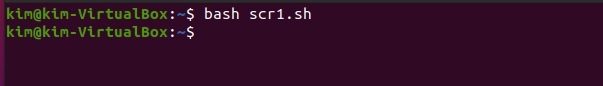


* Затем переключитесь в командный режим с помощью клавиши Esc, затем : затем wq, чтобы записать изменения в файл перед выходом из редактора (*Рисунок 5*), затем он вернется обратно в наш терминал, а затем мы используем команду bash scr1.sh для выполнения команды из файла (*Рисунок 6*)

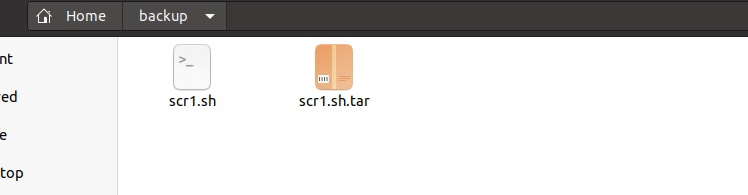
*Рисунок 5: Вставить текст*



*Рисунок 6: Запустите командный файл*



*Рисунок 7: проверка файлов*



Как мы видим, он успешно создает.

**Задание 2:** Написать пример командного файла, обрабатывающего любое произвольное число аргументов командной строки, в том числе превышающее десять. Например, скрипт может последовательно распечатывать значения всех переданных аргументов.

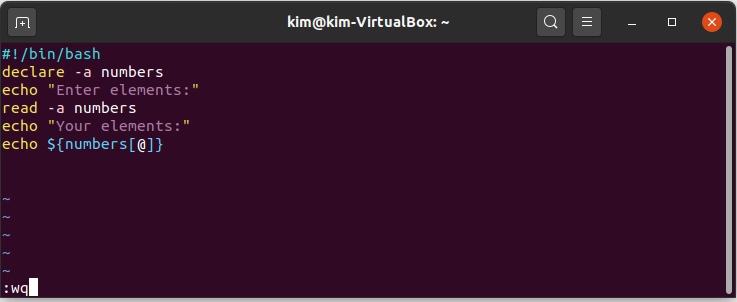
* Для этого сначала мы создаем новый командный файл с помощью редактора vi снова vi scr2.sh и сценарий-это вызов scr2.sh *(Рисунок 8)*

*Рисунок 8: Создать новый командный файл*



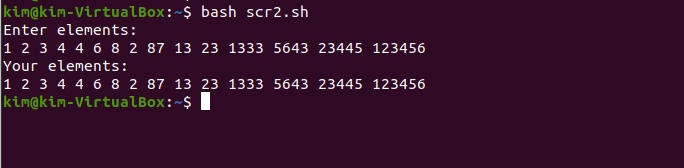
* Для этого мы используем массив, называемый числами. Сначала мы объявляем -a numbers с помощью команды declare, а затем для отображения строковых элементов ввода, которые передали аргумент с помощью команды echo "Enter numbers ". Затем мы используем команду read для чтения элементов массива с клавиатуры, затем снова используем команду echo для отображения строки ваших элементов, а затем элементов массива с помощью echo ${numbers[@]} затем сохраните и закройте файл (*Рисунок 9*).

*Рисунок 9: Вставить командный файл scr2.sh*



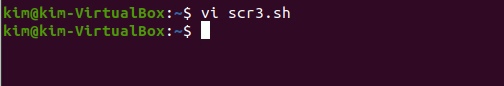
* Мы проверяем нашу работу с помощью команды bash scr2.sh и нажмите клавишу Enter после чего мы сможем ввести любые элементы в наш массив (*Рисунок 10*)

*Рисунок 10: Результаты командного файла scr2.sh*



**Задание 3:** Написать командный файл — аналог команды ls (без использования самой этой команды и команды dir). Требуется, чтобы он выдавал информацию о нужном каталоге и выводил информацию о возможностях доступа к файлам этого каталога.

*Рисунок 11: Создать новый файл scr3.sh*



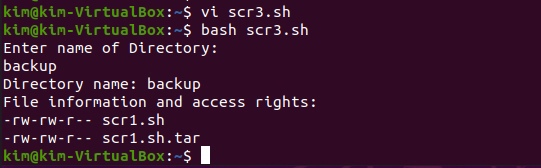
* Пишем текст командного файла. Сначала выведем сообщение о вводе имени каталога, который мы хотим рассмотреть, echo . Команда read позволит нам считать введенную с клавиатуры директорию в переменную name. Выводим имя директории и переходим в заданный каталог: cd ${name} . Выведем строку-сообщение о выводе файлов каталога и прав доступа к ним командой вывода echo. Выведем содержимое текущего катлога командой stat : stat -c '%A %n' \* .Где -с является ключом, который выведет наши файлы построчно, %A - вывод прав доступа в формате, читаемом для человека, а не машины, %n - названия файлов, \* - указывает на текущий каталог *(Рисунок 12)*

*Рисунок 12: Вставить командный файл*



* Здесь мы тестируем нашу работу , сначала используя bash src3.shзатем используйте клавишу Enter ; теперь вы можете увидеть строку ввода имени каталога, в котором вы хотите получить информацию, и их права доступа. (*Рисунок 13*)

*Рисунок 13: Результаты выполнения задания 3*



**Задание 4:** Написать командный файл, который получает в качестве аргумента командной строки формат файла (.txt, .doc, .jpg, .pdf и т.д.)и вычисляет количество таких файлов в указанной директории. Путь к директории также передаётся в виде аргумента командной строки.

*Рисунок 14: Создать новый файл src4.sh*

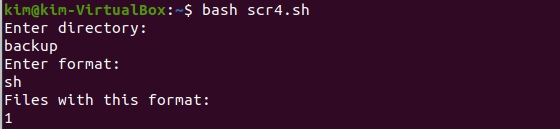


* Давайте напишем сам командный файл. Мы введем обозначения двух переменных: dir, в котором мы запишем рассматриваемый каталог, и format, в котором мы запишем нужный формат файла. Они сопровождаются двумя эхо-выходами, которые информируют пользователя о том, что именно необходимо ввести в данный момент. cd ${dir} - перейдите в нужный каталог. Мы ищем (команда find ) в нем ( " . " - текущий каталог) файлы по имени (- name ), в которых мы найдите введенный формат. Мы используем конвейер для чтения нереализованного вывода и используем команду wc-l для подсчета его строк, файлов, найденных в этом каталоге и соответствующих требованиям.

*Рисунок 15: Вставить командный файл*

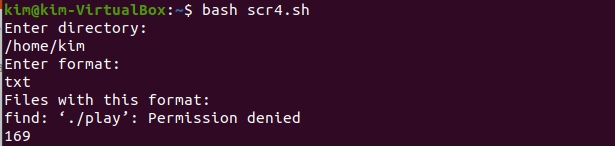


*Рисунок 16: Тестирование и результаты задание 4*



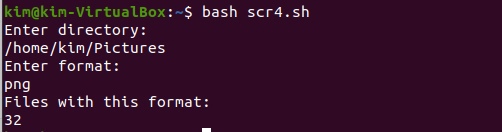
Как мы видим, здесь мы тестируем нашу работу, чтобы найти файлы или каталог с форматом sh.(*Рисунок 16*)

*Рисунок 17: Тестирование и результаты задание 4*



Далее мы ищем файлы и каталоги, которые имеют формат txt

*Рисунок 18: Тестирование и результаты задание 4*



И последнее, что мы тестируем, чтобы найти файлы и каталоги, которые имеют формат png.

### Вывод:

Я изучила основы программирования в оболочке ОС UNIX/Linux. Научилась писать небольшие командные файлы.

## Библиография:

[1]:[Лабораторая №11](https://esystem.rudn.ru/pluginfile.php/1142377/mod_resource/content/2/008-lab_shell_prog_1.pdf)

[2]:[read and echo](https://www.geeksforgeeks.org/read-command-in-linux-with-examples/#:~:text=read%20command%20in%20Linux%20system,the%20number%20of%20bytes%20read.)

[3]:[Команда bash](https://man7.org/linux/man-pages/man1/bash.1.html#:~:text=Bash%20is%20an%20sh%2Dcompatible,input%20or%20from%20a%20file.&text=Bash%20is%20intended%20to%20be,specification%20(IEEE%20Standard%201003.1).)