# РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

## Факультет физико-математических и естественных наук

### Кафедра прикладной информатики и теории вероятностей

## ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 12

### *дисциплина: Операционные системы*

Студент: Ким Реачна Группа: НПИбд-02-20

Москва 2021г.

### Цель работы:

Изучить основы программирования в оболочке ОС UNIX. Научится писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

### Теоретичексое 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...]

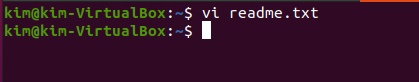
### Выполнение раьоты:

**Задание 1:** Используя команды getopts grep, написать командный файл, который анализирует командную строку с ключами:

-iinputfile — прочитать данные из указанного файла;  
  
-ooutputfile — вывести данные в указанный файл;  
  
-pшаблон — указать шаблон для поиска;  
  
-C — различать большие и малые буквы;  
  
-n — выдавать номера строк.

Для этого сначала мы создаем текстовый файл под названием readme.txt с помощью редактора vi readme.txt , затем вставьте в него какой-нибудь текст *(Рисунок 1-2)*.

*Рисунок 1: Создать файл readme.txt*



*Рисунок 3: Создать файл readme.txt*



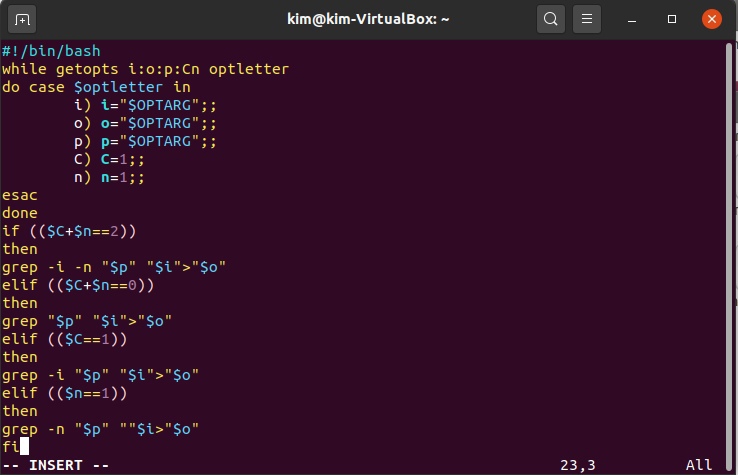
Создайте командный файл с помощью команды vi get.sh*(Рисунок 4)* ,которые будут использованы в нашей работе *(Рисунок )*

*Рисунок 4: Создать командный файл get.sh*



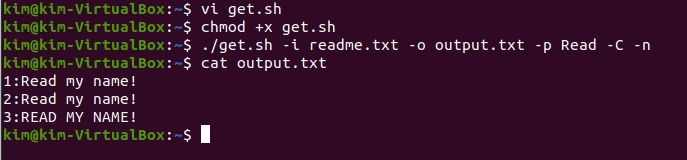
Мы пишем командный файл, используя образец оператора getopts в материале лабораторной работы № 11, а также циклы для распознавания ключа -C и -n *(Рисунок 5)*.

*Рисунок 5: Создать командный файл get.sh*



После создания командного файла мы используем команду chmod +x readme.txt для того, чтобы запросить разрешение на выполнение командного файла для использования в следующей команде. Затем мы вызываем наш командный файл как команду ,выбираем readme.txt файл как файл для чтения и output.txt для записи и поиска параметров стоит слово “Read” , сразу же проверьте работу командного файла , указав оба опции -C и -n. И посмотрите на результат output.txt файл с помощью команды cat *(Рисунок 6)*.

*Рисунок 6: Запустите командный файл*



**Задание 2:** Написать на языке Си программу, которая вводит число и определяет, является ли оно больше нуля, меньше нуля или равно нулю. Затем программа завершается с помощью функции exit(n), передавая информацию в о коде завершения в оболочку. Командный файл должен вызывать эту программу и проанализировав с помощью команды $?, выдать сообщение о том, какое число было введено.

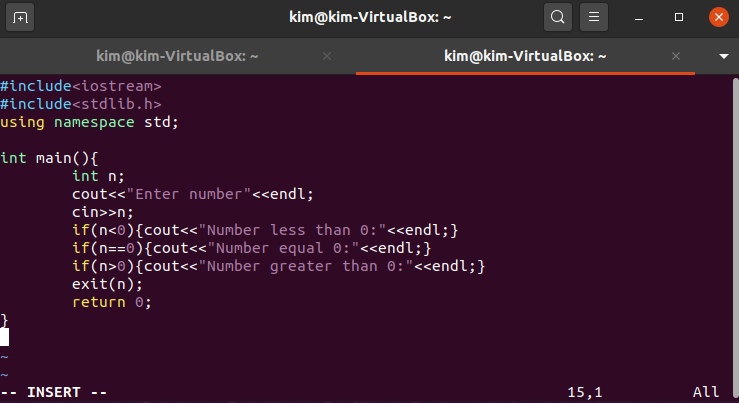
Сначала создайте новый командный файл с именем pro.cpp для записи программы c в файлс помощью редактора команд vi pro.cpp

*Рисунок 7: Создать новый командный файл pro.cpp*



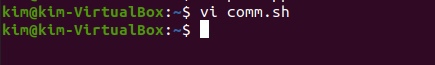
Мы вставляем некоторый стандартный код c++ в командный файл , чтобы вывести число, которое меньше 0, равно 0 и больше 0, и завершить программу с exit(n) *(Рисунок 8)*.

*Рисунок 8: Вставка кода в командный файл*



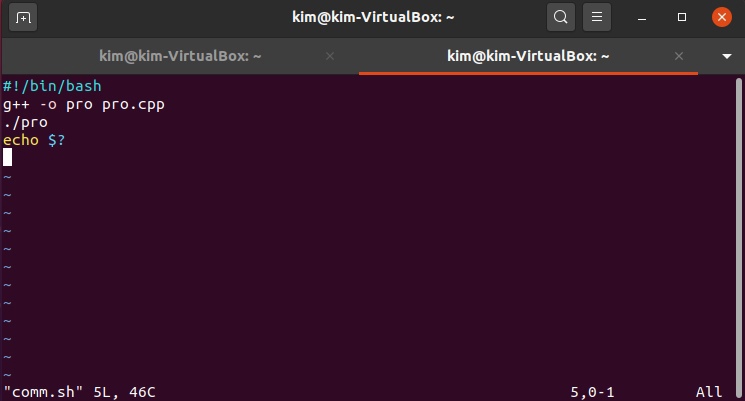
Создайте командный файл с именем comm.sh для выполнения файла pro.cpp в этом процессе работают. *(Рисунок 9)*

*Рисунок 9: создать новый командный файл comm.sh*



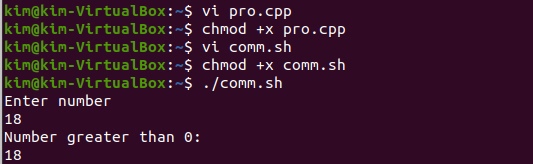
В этом процессе здесь мы вставляем командный файл для выполнения файла pro.cpp , это может быть некоторой проблемой, когда мы используем g++ здесь, поэтому мы можем установить пакет с помощью sudo apt-get install g++ для запуска этой программы. *(Рисунок 10)*

*Рисунок 10: создать новый командный файл comm.sh*



Используйте команду chmod +x для разрешения на выполнение обоих файлов, которые мы только что создали, и давайте проверим работу, которую мы выполняем с помощью этой команды ./comm.sh затем введите число 18( 18 больше 0), и мы увидим результат, что он выводит сообщение больше 0 *(Рисунок 11)*

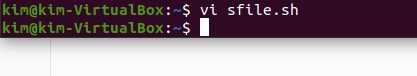
*Рисунок 11: Проверьте работу*



**Задание 3:** Написать командный файл, создающий указанное число файлов, пронумерованных последовательно от 1 до N (например 1.tmp, 2.tmp, 3.tmp,4.tmp и т.д.).Число файлов, которые необходимо создать, передаётся в аргументы командной строки. Этот же командный файл должен уметь удалять все созданные им файлы (если они существуют).

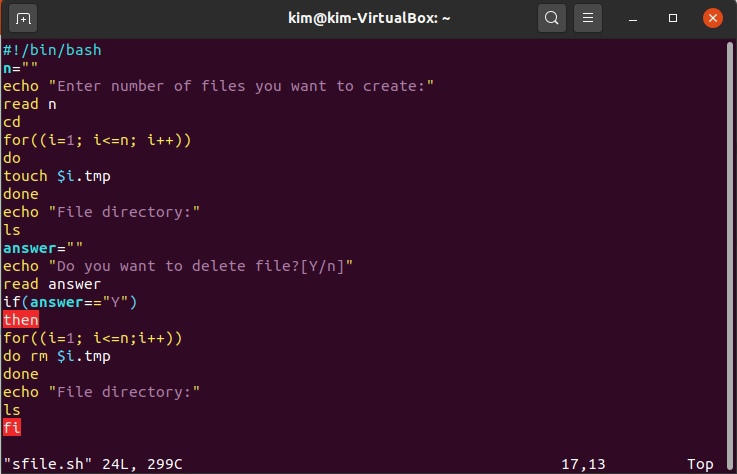
Сначала давайте создадим новый файл вызова sfile.sh с помощью редактора команд vi sfile.sh *(Рисунок 12)*

*Рисунок 12: создадим новый файл sfile.sh*



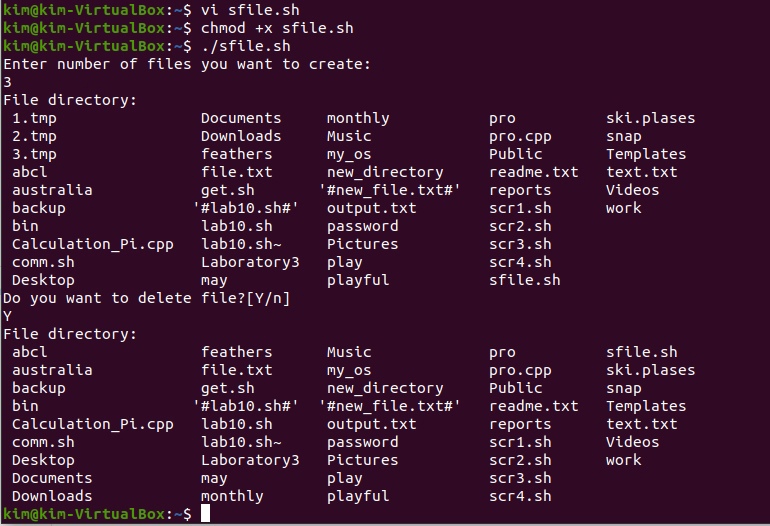
В этом процессе здесь нам потребуется написать код, который мы можем ввести номера файлов, которые мы хотим создать, и удалить их *(Рисунок 13)* с помощью keyborad. Затем мы создаем цикл for для i между 1 и n, который увеличивается на 1, затем с помощью команды коснитесь создать файл, который будет соответствовать значению i. Мы показываем файлы содержимого с помощью команды ls, чтобы убедиться, что файлы созданы. Затем создайте строку, чтобы спросить, хотим ли мы удалить файлы создания с помощью Y/n (Да/нет), затем снова используйте цикл for, если ответ будет Y, то w удалит их с помощью команды rm, затем снова отобразит файлы и посмотрит, успешно ли они удалены. *(Рисунок 13)*

*Рисунок 13: Вставьте код в командный файл*



Затем мы проверяем нашу работу, которую мы только что выполнили в командном файле, чтобы увидеть, работает ли она. Как мы видим, в *(Рисунок 14)* мы вводим номера файлов, которые мы хотим создать, 3 файла, и он отображает 3 новых файла tmp в соответствии со значением i, затем мы снова хотим удалить файлы, которые мы только что создали, поэтому мы вводим букву Y (Да), и снова команда ls показывает, что файлы успешно удалены *(Рисунок 14)*

*Рисунок 14: Проверьте работу*



**Задание 4:** Написать командный файл, который с помощью команды tar запаковывает в архив все файлы в указанной директории. Модифицировать его так, чтобы запаковывались только те файлы, которые были изменены менее недели тому назад (использовать команду find).

Создать новое имя командного файла tar.sh с помощью редактора vi tar.sh *(Рисунок 15)*

*Рисунок 15: Создать новый командный файл tar.sh*

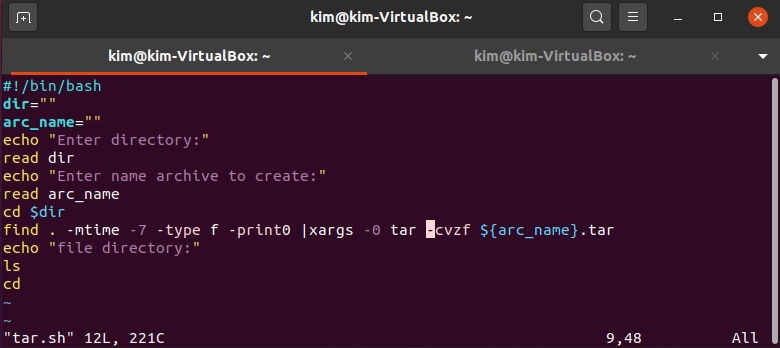


В процессе здесь мы вставляем командный файл в файл, который мы только что создали, чтобы заархивировать все файлы, которые мы создаем (Рисунок 17). Сначала мы введем каталог, в котором мы будем работать, затем введем имя файла архива, который мы хотим создать, затем мы используем команду find, чтобы помочь нам найти подходящие для нас файлы и отобразить файлы, если мы успешно это сделаем. *(Рисунок 16)*

мы используем команду find со следующими параметрами:

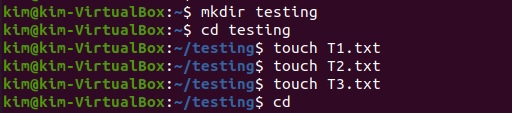
* . - поиск выполняется в текущем каталоге
* -mtime -7 - срок хранения файлов не более 7 дней
* -type -f - поиск файлов без каталога
* -print0 - выведите полное имя файла на стандартный вывод
* c - создание архивного файла
* v - показать ход работы с архивным файлом.
* z - фильтр архива через gzip
* f - имя файла архива.

*Рисунок 16: Создать новый командный файл tar.sh*



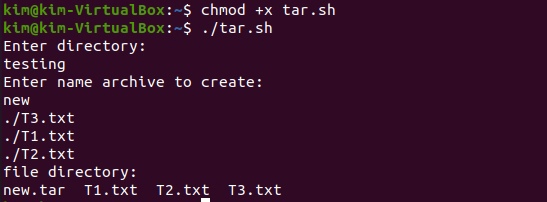
Здесь мы создаем тестирование имени каталога с помощью команды mkdir,а в тестировании каталога мы создаем текстовые файлы с вызовами T1, T2 и T3 с помощью comamnd touch *(Рисунок 17)*

*Рисунок 17: Создание каталога и файлов*



После того , как мы создадим файлы, давайте проверим, правильно ли выполнена работа, поэтому мы запрашиваем разрешение на выполнение командного файла с помощью команды chmod + x и вызываем наш командный файл и вводим каталог, а затем имя файла архива, который мы хотим создать, и с помощью команды ls для отображения результата *(Рисунок 18)*.

*Рисунок 18: Проверьте работу*



### Контрольные вопросы:

1. Она осуществляет синтаксический анализ командной строки, выделяя флаги, и используется для объявления переменных.
2. При генерации имен файлов используют метасимволы:

"\*" - произвольная (возможно пустая) последовательность символов;

“?” - один произвольный символ;

“[…]” - любой из символов, указанных в скобках перечислением и/или с указанием диапазона;

"cat f\*" - выдаст все файлы каталога, начинающиеся с “f”;

“cat *f*” - выдаст все файлы, содержащие “f”;

“cat program.?” выдаст файлы данного каталога с однобуквенными расширениями, скажем “program.c” и “program.o”, но не выдаст “program.com”;

“cat [a-d]*" выдаст файлы, которые начинаются с "a", "b", "c", "d". Аналогичный эффект дадут и команды "cat [abcd]*” и "cat [bdac]\*".

1. for, case, if, while
2. Break, continue
3. Команды ОС UNIX возвращают код завершения, значение которого может быть использовано для принятия решения о дальнейших действиях. Команда test, например, создана специально для использования в командных файлах. Единственная функция этой команды заключается в выработке кода завершения.
4. Означает условие существования файла mani.$s
5. Если речь идет о 2-х параллельных действиях, то это while. когда мы показываем, что сначала делается 1-е действие. потом оно заканчивается при наступлении 2-го действия, применяем until.

### Вывод:

Я изучила основы программирования в оболочке **ОС UNIX**. Научилась писать более сложные командные файлы с использованием логических управляющих *конструкций* и *циклов*.

#### Библиография:

[1]:[Лабораторая работа №12](https://esystem.rudn.ru/pluginfile.php/1142380/mod_resource/content/3/009-lab_shell_prog_2.pdf) [2]:[Команда tar](https://blog.sedicomm.com/2018/11/21/12-prakticheskih-primerov-komandy-xargs-dlya-nachinayushhih-v-linux/) [3]:[tar](https://www.tecmint.com/18-tar-command-examples-in-linux/) [4]:[Compiler g++](https://www.geeksforgeeks.org/compiling-with-g-plus-plus/)