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: Соболевский Денис Андреевич

Группа: НФИбд-02-20

МОСКВА

2021 г.

Цель работы:

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

Теоретическое введение:

В данной лабораторной работе нам предстоит научиться писать командные файлы и использовать их на практике. Для этого нам необходимо ознакимиться с некоторой теорией.

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;

• C-оболочка (или csh) — надстройка на оболочкой Борна, использующая Сподобный синтаксис команд с возможностью сохранения истории выполнения команд;

• оболочка Корна (или ksh) — напоминает оболочку C, но операторы управления программой совместимы с операторами оболочки Борна;

• BASH — сокращение от Bourne Again Shell (опять оболочка Борна), в основе своей совмещает свойства оболочек C 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/Linuxподобных операционных систем и переносимости прикладных программ на уровне исходного кода. POSIX-совместимые оболочки разработаны на базе оболочки Корна. Рассмотрим основные элементы программирования в оболочке bash. В других оболочках большинство команд будет совпадать с описанными ниже.

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.

Использование:

mv afile ${mark}

переместит файл afile из текущего каталога в каталог с абсолютным полным именем /usr/andy/bin.

Использование значения, присвоенного некоторой переменной, называется подстановкой.

Команды read и echo

Команда read позволяет записать значение для переменной с клавиатуры. Она имеет следующий синтаксис:

read

Команда echo выводит текст на экран, если имеет вид:

echo "Some text"

В данном случае она выведет на экран Some text.

С помощью данной команды также можно вывести на экран содержимое, например, переменных:

echo

С прочей теорией и основами языка bash можно ознакомиться в материалах к лабораторной работе №11[1].

Также в ходе выполнения заданий лабораторной работы я столкнулась в необходимости изучения дополнительных натериалов, а именно:

• архивирование файлов в Linux[2]

• использование массивов в bash[3]

• различные способы составления списка содержимого каталога без использования команды ls[4]

• команда find в Linux[5]

• команда wc в Linux[6]

Выполнение работы:

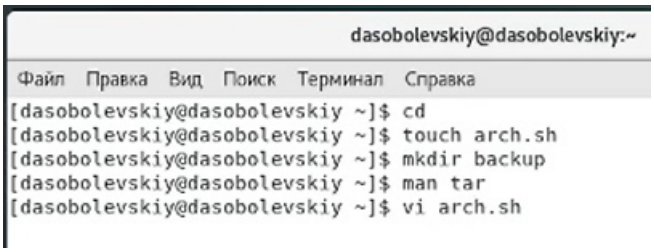
Задание 1

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.

Для этого сначала перейдем в домашний каталог (cd), после чего создадим наш командный файл, который будет называться arch.sh, командой touch. Далее в домашнем каталоге создадим каталог backup командой создания каталогов mkdir, в нем мы будем создавать резервные копии и архивы с командными файлами (рисунок 1).

Чтобы понять структуру архивации файлов и создания архивов воспользуемся справкой man tar и изучим команду tar, которая позволит нам создать архив (рисунок 1). Для создания командных файлов будем использовать текстовой редактор vi. Открываем с его помощью будущий командный файл arch.sh (vi arch.sh) (рисунок 1).

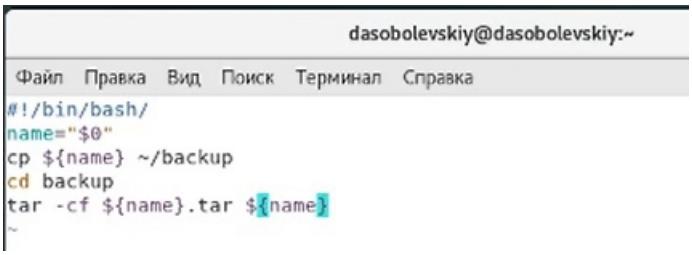
Рисунок 1:



Пишем сам командный файл. Для того, чтобы система распознавала его как командный, в первой строке прописываем #!/bin/bash (рисунок 2).

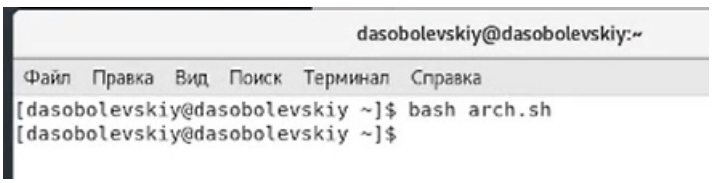
Создаем перменную name, в которой будет содержаться имя данного командного файла. Используем команду tar и ключ -cf, который позволяет нам создать архив и сразу же поместить в него нам командный файл, который мы передаем ссылкой ${name} (рисунок 2).

Рисунок 2:



Теперь протестируем созданный файл. Для того, чтобы запустить его как команду необходимо использовать bash (рисунок 3)

Рисунок 3:

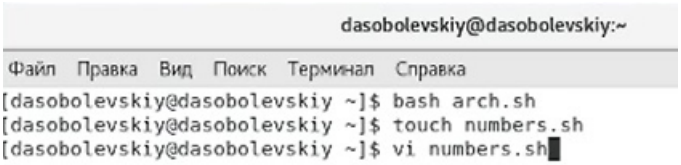


Задание 2

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.

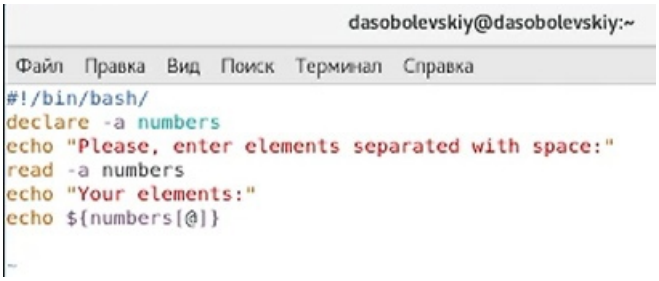
Сперва создадим соответствующий командный файл numbers.sh и сразу откроем его в редакторе (рисунок 4).

Рисунок 4:



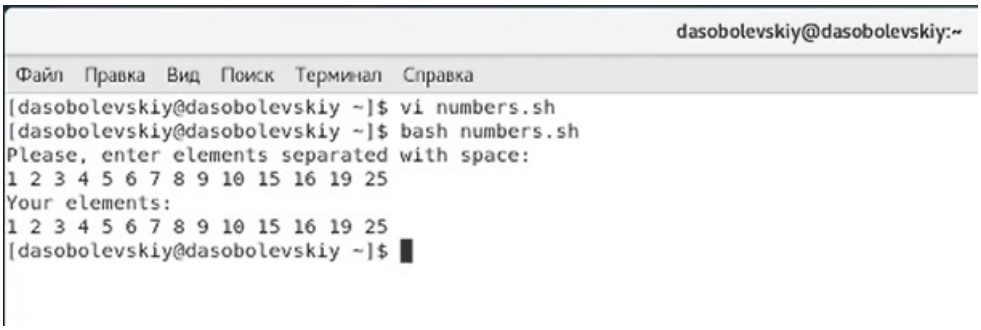
Для того, чтобы вводить неизвестное количество аргументов (даже большее десяти) и обрабатывать их, воспользуемся массивом, который назовем numbers. Сначала объявим его: declare -a numbers. С помощью команды echo выведем на экран сообщение о том, что нужно ввести элементы, притом в качестве разделителя использовать пробелы. Далее командой read -a считываем с клавиатуры элеменьы массива. Выводим строку Your numbers и выводим все элементы массива - echo ${numbers[@]} (@ - все элементы массива) (рисунок 5).

Рисунок 5:



Проверим работу нашего файла. Видим, что он работает исправно и действительно выводит те числа, которые мы ввели (рисунок 6).

Рисунок 6:

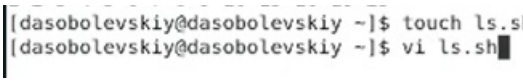


Задание 3

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.

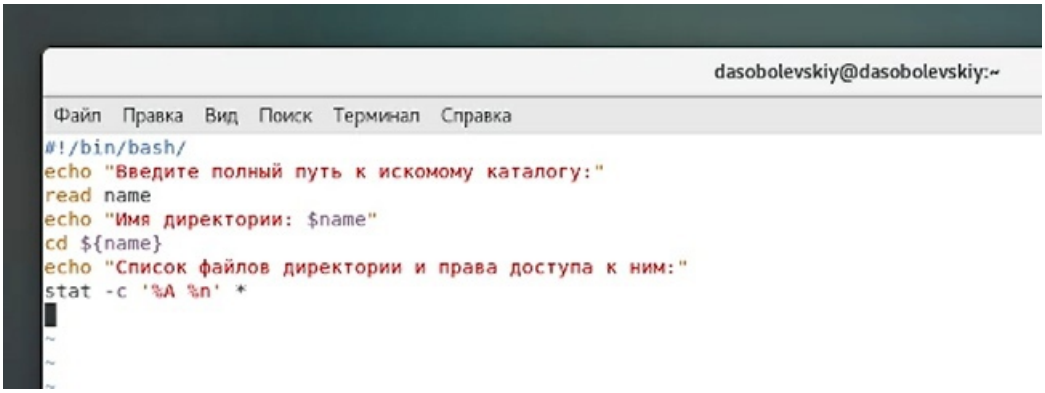
Сперва создадим соответствующий командный файл ls.sh и сразу откроем его в редакторе (рисунок 7).

Рисунок 7:



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 -c 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 8).

Рисунок 8:



Посмотри на резульаты работы нашего командного файла. Рассмотрим директорию, созданную в задании 1 (рисунок 9). Видим, что файл работает исправно.

Рисунок 9:

```
[dasobolevskiy@dasobolevskiy ~]$ bash ls.sh
Введите полный путь к искомому каталогу:
backup
Имя директории: backup
Список файлов директории и права доступа к ним:
-rw-rw-r-- arch.sh
-rw-rw-r-- arch.sh.tar
[dasobolevskiy@dasobolevskiy ~]$
```

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.

Сперва создадим соответствующий командный файл find.sh и сразу откроем его в редакторе (рисунок 10).

Рисунок 10:

```
[dasobolevskiy@dasobolevskiy ~]$ touch find.sh
[dasobolevskiy@dasobolevskiy ~]$ vi find.sh
```

Напишем сам командный файл. Введем обозначения двух переменных: dirt, в которую мы запишем рассматриваемую директорию, и format, в которую запишем искомый формат файла. Им сопутсвуют два вывода echo, сообщающих пользователю о том, что именно необходимо ввести в данный момент. cd ${dirt} - переходим в требуемую директорию. Ищем (команда find) в ней ("." - текущая директория) файлы по именам (-name), в которых встречается нам введенный формат. Конвейером считываем нереализованный вывод и командой wc -l считаем его строки, т.е. - файлы, найденные в данной директории и соответствующие требованиям. (рисунок 11)

Рисунок 11:

dasobolevskiy@dasobolevskiy:~

Файл Правка Вид Поиск Терминал Справка

```
#!/bin/bash/
dirt=""
echo "Введите директорию"
read dirt
format=""
echo "Введите требуемый формат"
read format
cd ${dirt}
echo "Файлов с таким форматом в данной директории: "
find . -name "*.${format}" | wc -l
```

Посмотрим на результат работы написанного файла. Введем с клавиатуры путь к домашней директории, будем искать в ней файлы формата txt, видим, что найдено 14 файлов (рисунок 12).

Рисунок 12:

```
[dasobolevskiy@dasobolevskiy ~]$ bash find.sh
Введите директорию
/home/dasobolevskiy
Введите требуемый формат
txt
Файлов с таким форматом в данной директории:
14
[dasobolevskiy@dasobolevskiy ~]$
```

Вывод:

Изучил основы программирования в оболочке ОС UNIX/Linux. Изучил основы языка bash, научился писать небольшие командные файлы.

Библиография:

[1] Лабораторная работа №11

[2] Архивирование файлов в Linux

[3] Использование массивов в bash

[4] Различные способы составления списка содержимого каталога без использования команды ls

[5] Команда find в Linux

[6] Команда wc в Linux