Титульный лист

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Лабораторная работа 11

По дисциплине "Операционные системы"

Выполнил:

Студент группы НПВбм-01-19

Студенческий билет №: 1032193844

Саидов Ахият Магомадович

Руководитель: Валиева Татьяна Рефатовна

Москва 2023

Цель работы

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

Начало работы

1. Напишем 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.
   - Создадим файл *script1.sh*, который будет в дальнейшем скрипт.

     ```
     [AMSaidov@amsaidov ~]$ mkdir lab11
     [AMSaidov@amsaidov ~]$ cd lab11
     [AMSaidov@amsaidov lab11]$ touch script1.sh
     ```

     Рисунок1
   - Перейдем в наш файл *script1.sh* при помощи редактора `vi`.

     ```
     [AMSaidov@amsaidov lab11]$ vi script1.sh
     ```

     Рисунок2
   - Запишем код скрипта такой последовательностью:
     - создадим каталог, в который будет копироваться скрипт `mkdir`;
     - скопируем скрипт в этот каталог `cp`;
     - архивируем скрипт `gzip`.

     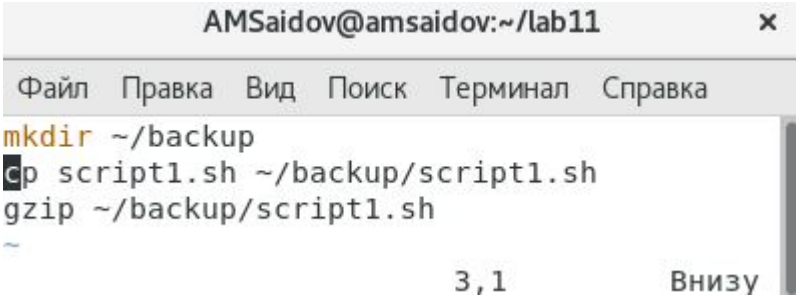

     Рисунок3
   - Разрешим управление для владельца, запустим скрипт и проверим на работоспособность.

     ```
     [AMSaidov@amsaidov lab11]$ chmod +x script1.sh
     [AMSaidov@amsaidov lab11]$ ./script1.sh
     [AMSaidov@amsaidov lab11]$ ls ~/backup/
     script1.sh.gz
     ```

     Рисунок4
2. Напишем пример командного файла, обрабатывающего любое произвольное число аргументов командной строки, в том числе превышающее десять. Например, скрипт может последовательно распечатывать значения всех переданных аргументов.
   - Создадим файл *script2.sh*, который будет в дальнейшем скрипт.

     ```
     [AMSaidov@amsaidov ~]$ cd lab11
     [AMSaidov@amsaidov lab11]$ touch script2.sh
     ```

     Рисунок5

- Перейдем в наш файл *script2.sh* при помощи редактора `vi`.

```
[AMSaidov@amsaidov lab11]$ vi script2.sh
```

Рисунок6

- Запишем код скрипта такой последовательностью:
  - создадим цикл для всех переданных аргументов `for i`;
  - выведем аргумент `do echo $1`;
  - удалим первый аргумент, смещая оставшееся `shift`;
  - закончим цикл `done`.

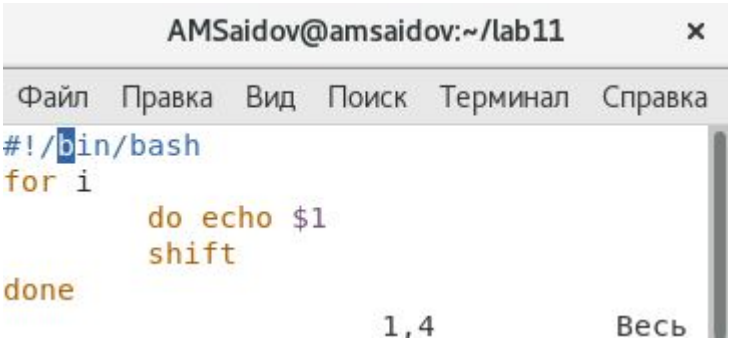

Рисунок7

- Разрешим управление для владельца, запустим скрипт и проверим на работоспособность.

```
[AMSaidov@amsaidov lab11]$ chmod +x script2.sh
[AMSaidov@amsaidov lab11]$ ./script2.sh 11
11
[AMSaidov@amsaidov lab11]$ ./script2.sh 11 12 13 14 24124 151613461345
11
12
13
14
24124
151613461345
```

Рисунок8

3. Напишем 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.
   - Создадим файл *script3.sh*, который будет в дальнейшем скрипт.

```
[AMSaidov@amsaidov ~]$ cd lab11
[AMSaidov@amsaidov lab11]$ touch script3.sh
```

Рисунок9

   - Перейдем в наш файл *script3.sh* при помощи редактора `vi`.

```
[AMSaidov@amsaidov lab11]$ vi script3.sh
```

Рисунок10

   - Запишем код скрипта такой последовательностью:
     - запросим путь каталога `read`;
     - сохраним аргумент `change=`;
     - перейдем в каталог `cd`;
     - выведем поочередно файлы с определенными правами пользователя `find`.

```
#!/bin/bash
echo "Введите путь каталога"
read catalog
change=$catalog
cd $change
echo "Выведены файлы с правами пользователя по категориям"
echo "R"
find $change -maxdepth 1 -perm /u=r
echo "W"
find $change -maxdepth 1 -perm /u=w
echo "X"
find $change -maxdepth 1 -perm /u=x
```

Рисунок11

- Разрешим управление для владельца, запустим скрипт и проверим на работоспособность.

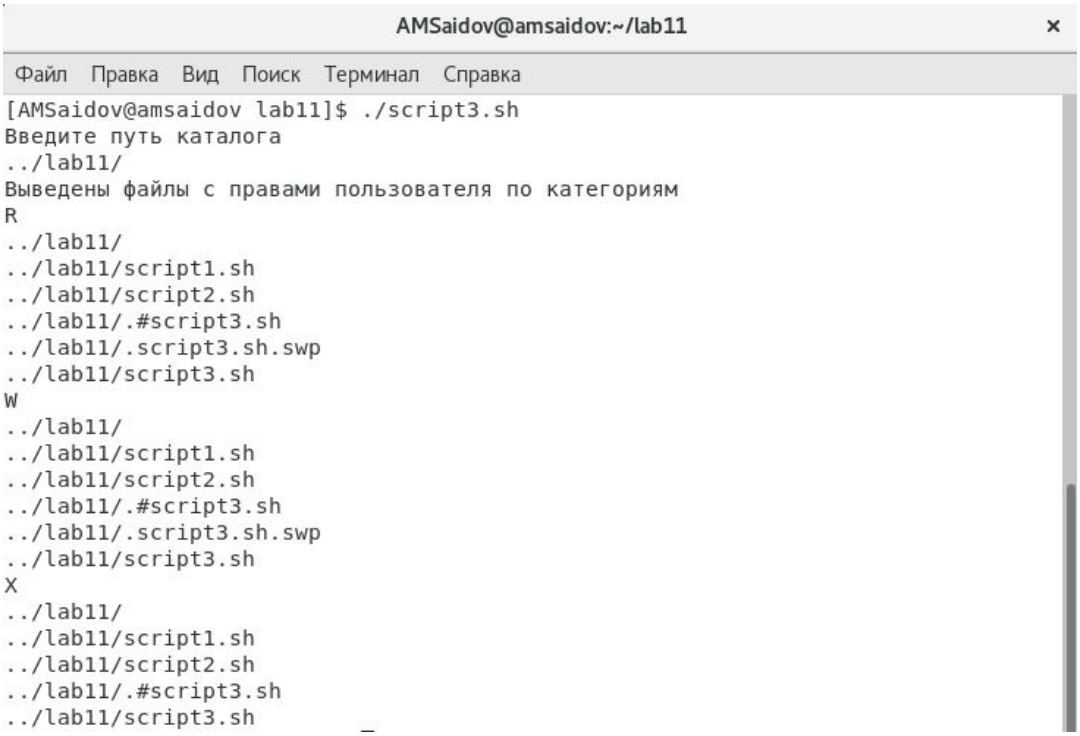

Рисунок12

Вывод

мы изучили основы программирования в оболочке ОС UNIX/Linux. Научились писать небольшие командные файлы.

Контрольные вопросы

1. Объясните понятие командной оболочки. Приведите примеры командных оболочек. Чем они отличаются?
   Ответ:
   a) sh — стандартная командная оболочка UNIX/Linux, содержащая базовый, полный набор функций
   b) csh — использующая С-подобный синтаксис команд с возможностью сохранения истории выполнения команд
   c) ksh — напоминает оболочку С, но операторы управления программой совместимы с операторами оболочки Борна

d) bash — сокращение от Bourne Again Shell (опять оболочка Борна), в основе своей совмещает свойства оболочек C и Корна

2. Что такое POSIX?
   Ответ: POSIX (Portable Operating System Interface for Computer Environments) — набор стандартов описания интерфейсов взаимодействия операционной системы и прикладных программ.

3. Как определяются переменные и массивы в языке программирования bash?
   Ответ: Переменные вызываются $var, где var=чему-то, указанному пользователем, неважно что бы то не было, название файла, каталога или еще чего.
   Для массивов используется команда set -A

4. Каково назначение операторов let и read?
   Ответ: let — вычисляет далее заданное математическое значение
   read — позволяет читать значения переменных со стандартного ввода

5. Какие арифметические операции можно применять в языке программирования bash?
   Ответ: Прибавление, умножение, вычисление, деление), сравнение значений, экспонирование и др.

6. Что означает операция (( ))?
   Ответ: Это обозначение используется для облегчения программирования для условий bash

7. Какие стандартные имена переменных Вам известны?
   Ответ: Нам известны HOME, PATH, BASH, ENV, PWD, UID, OLDPWD, PPID, GROUPS, OSTYPE, PS1 - PS4, LANG, HOSTFILE, MAIL, TERM, LOGNAME, USERNAME, IFS и др.

8. Что такое метасимволы?
   Ответ: Метасимволы это специальные знаки, которые могут использоваться для сокращения пути, поиска объекта по расширению, перед переменными, например «$» или «*» .

9. Как экранировать метасимволы?
   Ответ: Добавить перед метасимволом метасимвол «\»

10. Как создавать и запускать командные файлы?
    Ответ: При помощи команды chmod. Надо дать права на запуск chmod +x название файла, затем запустить bash ./название файла
    Например у нас файл lab
    Пишем:
    chmod +x lab
    ./lab

11. Как определяются функции в языке программирования bash?
    Ответ: Объединяя несколько команд с помощью function

12 Каким образом можно выяснить, является файл каталогом или обычным файлом?
Ответ: Можно задать команду на проверку диретория ли это test -d директория

13 Каково назначение команд set, typeset и unset?
Ответ:
Set — используется для создания массивов
Unset — используется для изъятия переменной
Typeset — используется для присваивания каких-либо функций

14. Как передаются параметры в командные файлы?
    Ответ: Добавлением аршументов после команды запуска bash скрипта

15. Назовите специальные переменные языка bash и их назначение.

    Ответ:

```
-   $* – отображается вся командная строка или параметры оболочки;
-   $? – код завершения последней выполненной команды;
-   $$ – уникальный идентификатор процесса, в рамках которого выполняется
командный процессор;
-   $! – номер процесса, в рамках которого выполняется последняя вызванная на
выполнение в командном режиме команда;
-   $- – значение флагов командного процессора;
-   ${#*} – возвращает целое число – количество слов, которые были результатом
$*;
-   ${#name} – возвращает целое значение длины строки в переменной name;
-   ${name[n]} – обращение к n-му элементу массива;
-   ${name[*]} – перечисляет все элементы массива, разделённые пробелом;
-   ${name[@]} – то же самое, но позволяет учитывать символы пробелы в самих
переменных;
-   ${name:-value}–еслизначениепеременнойnameнеопределено,тоонобудет заменено на
указанное value;
-   ${name:value} – проверяется факт существования переменной;
-   ${name=value} – если name не определено, то ему присваивается значение value;
-   ${name?value} – останавливает выполнение, если имя переменной не определено,
и выводит value как сообщение об ошибке;
-   ${name+value} – это выражение работает противоположно ${name-value}. Если
переменная определена, то подставляется value;
-   ${name#pattern} – представляет значение переменной name с удалённым самым
коротким левым образцом (pattern);
-   ${#name[*]}и${#name[@]}–этивыражениявозвращаютколичествоэлементов в массиве
name.
```