

Отчет по лабораторной работе №9(10)

дисциплина: "Операционные системы"

Студент: Рыскалова Екатерина Андреевна

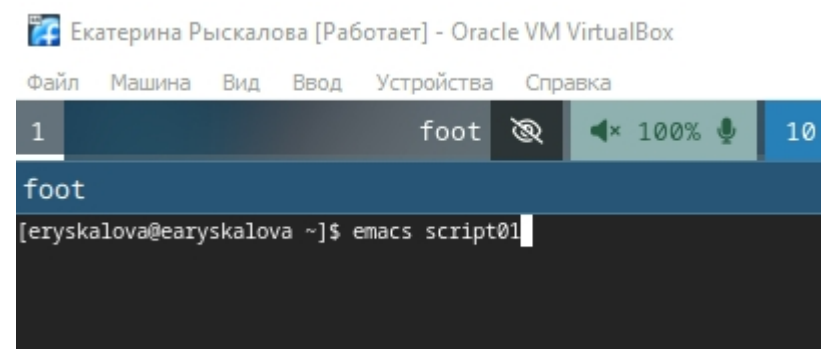
Группа: НПИМбв02-20

Цель работы

Целью данной работы является приобретение практических навыков установки операционной системы на виртуальную машину, настройки минимально необходимых для дальнейшей работы сервисов.

Ход работы

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.

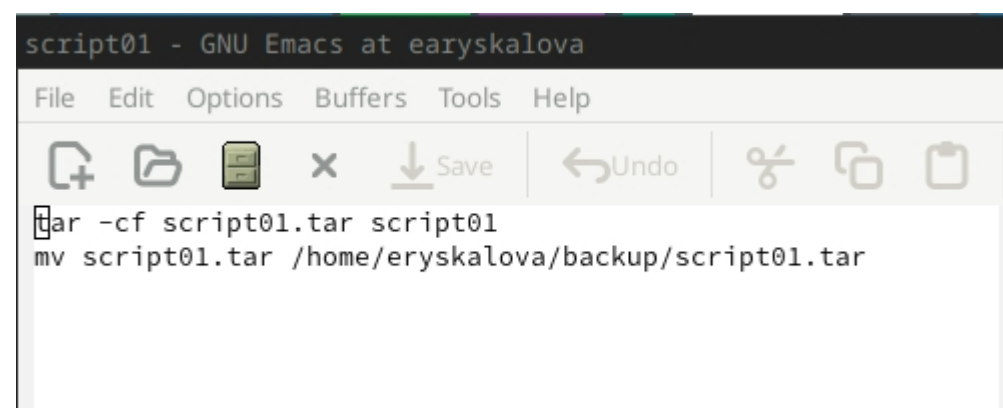


Екатерина Рыскалова [Работает] - Oracle VM VirtualBox

Файл Машина Вид Ввод Устройства Справка

1 foot 100% 10.

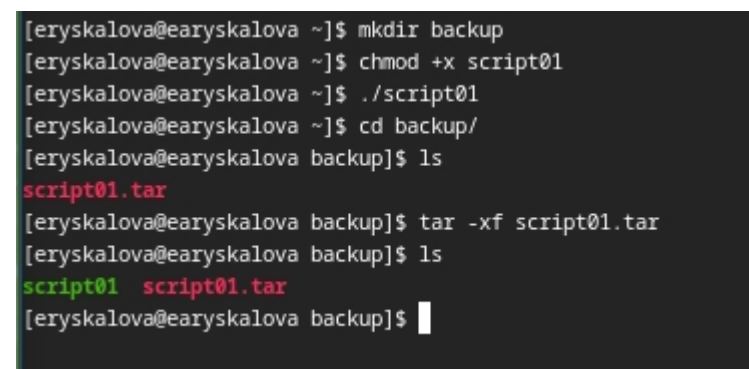
```
foot
[eryskalova@earyskalova ~]$ emacs script01
```



script01 - GNU Emacs at earyskalova

File Edit Options Buffers Tools Help

tar -cf script01.tar script01
mv script01.tar /home/eryskalova/backup/script01.tar



```
[eryskalova@earyskalova ~]$ mkdir backup
[eryskalova@earyskalova ~]$ chmod +x script01
[eryskalova@earyskalova ~]$ ./script01
[eryskalova@earyskalova ~]$ cd backup/
[eryskalova@earyskalova backup]$ ls
script01.tar
[eryskalova@earyskalova backup]$ tar -xf script01.tar
[eryskalova@earyskalova backup]$ ls
script01 script01.tar
[eryskalova@earyskalova backup]$
```

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.

```
[earyskalova@earyskalova ~]$ emacs script02
```

```
script02 - GNU Emacs at earyskalova
File Edit Options Buffers Tools Help
[Icons] Save Undo [Icon]
count=1
for param in "$@"
do
echo "$count: $param"
count=$((count +1))
done
```

```
[root@earyskalova ~]# chmod +x script02
[root@earyskalova ~]# ./script02 1 2 3
1: 1
2: 2
3: 3
[root@earyskalova ~]#
```

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.

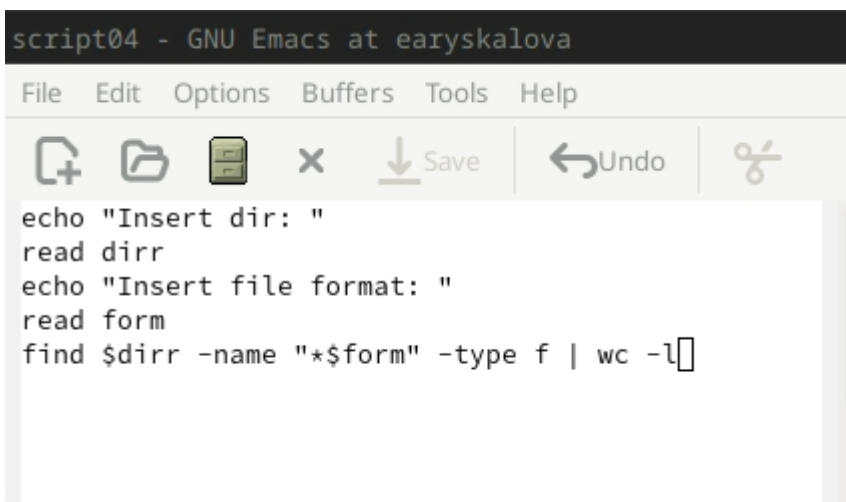
```
foot
[root@earyskalova ~]# emacs script03
```

```
script03 - GNU Emacs at earyskalova
File Edit Options Buffers Tools Help
[Icons] Save Undo [Icon]
if test -w $A
then echo writeable
elif test -r $A
then echo readable
else echo unwriteable and unreadable
fi
done
```

```
[eryskalova@earyskalova ~]$ chmod +x script03
[eryskalova@earyskalova ~]$ ./script03
a1: is a file and writeable
conf.txt: is a file and writeable
file.txt: is a file and writeable
may: is a file and writeable
monthly: is dir
reports: is dir
script02: is a file and writeable
script03: is a file and writeable
Видео: is dir
Документы: is dir
Загрузки: is dir
Изображения: is dir
Музыка: is dir
Общедоступные: is dir
./script03: строка 2: test: Рабочий: ожидается бинарный оператор
Рабочий стол: is a file and ./script03: строка 5: test: Рабочий: ожидается бинарный оператор
./script03: строка 7: test: Рабочий: ожидается бинарный оператор
upwriteable and unreadable
Шаблоны: is dir
[eryskalova@earyskalova ~]$
```

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.

```
[eryskalova@earyskalova ~]$ emacs script04
```



```
[eryskalova@earyskalova ~]$ chmod +x script04
[eryskalova@earyskalova ~]$ ./script04
Insert dir:
/home/eryskalova/
Insert file format:
.txt
3
[eryskalova@earyskalova ~]$
```

Контрольные вопросы

1. Объясните понятие командной оболочки. Приведите примеры командных оболочек. Чем они отличаются?

Командная оболочка — это пользовательский интерфейс для доступа к сервисам операционной системы. В зависимости от её типа, она может принимать команды в текстовом формате (текстовая оболочка) или предоставлять графический интерфейс (графическая оболочка). Примеры текстовых командных оболочек: **bash**, **zsh**, **fish**. Они отличаются по синтаксису, функциональности, настройкам по умолчанию и другим специфическим возможностям, например, улучшенная автодополнение в **zsh**, поддержка скриптов на основе **bash** и уникальные функции для работы с текстом и данными в **fish**.

2. Что такое POSIX?

POSIX (Portable Operating System Interface) — это набор стандартов, разработанных для обеспечения совместимости между UNIX-системами. Стандарты описывают интерфейсы программирования приложений (API), команды оболочки и утилиты, которые должны быть доступны в операционной системе.

3. Как определяются переменные и массивы в языке программирования bash?

В **bash**, переменные определяются без указания типа, простым присваиванием значения, например: `variable_name="value"`. Массивы определяются при помощи круглых скобок, значения разделяются пробелом, например: `array_name=(value1 value2 value3)`.

4. Каково назначение операторов let и read?

`let` используется для выполнения арифметических операций, например: `let "a=5+4"`. `read` используется для считывания строки из стандартного ввода (обычно это клавиатура) и присваивания её значение переменной, например: `read var_name`.

5. Какие арифметические операции можно применять в языке программирования bash?

В **bash** можно использовать: сложение (+), вычитание (-), умножение (*), деление (/), остаток от деления (%), возведение в степень (**).

6. Что означает операция (()):

Операция `(())` используется для выполнения арифметических операций в **bash**. Внутри `(())` переменные используются без знака доллара, а результаты арифметических вычислений можно прямо использовать в условиях, например: `if ((a > b)); then`.

7. Какие стандартные имена переменных Вам известны?

Некоторые стандартные переменные включают: `HOME` (домашний каталог текущего пользователя), `PWD` (текущий рабочий каталог), `USER` (имя текущего пользователя), `PATH` (список каталогов, в которых система ищет команды).

8. Что такое метасимволы?

Метасимволы — это символы, которые имеют особое значение для оболочки. Например, `*`, `?`, `[]`, `|`, `&`, `;` и другие. Они используются в паттернах (шаблонах) поиска, разделении команд и т.д.

9. Как экранировать метасимволы? Метасимволы можно экранировать при помощи обратного слэша \ перед метасимволом для его литерального использования, например: * означает символ *, а не операцию поиска всех файлов.

10. Как создавать и запускать командные файлы?

Командные файлы (скрипты) создаются в любом текстовом редакторе с присвоением файла расширения `.sh`. Запускаются они командой `bash script_name.sh` или после установки права на выполнение с помощью `chmod +x script_name.sh` — запуском `./script_name.sh`.

11. Как определяются функции в языке программирования bash?

Функции в **bash** определяются по следующему шаблону:

```
function_name() {  
    command1  
    command2  
}
```

вызываются простым указанием их имени: `function_name`.

12. Каким образом можно выяснить, является файл каталогом или обычным файлом?

В **bash** для проверки используется условный оператор `if` с ключами `-d` (для каталога) и `-f` (для файла), например: `if [-d "$directory_name"]; then.`

13. Каково назначение команд `set`, `typeset` и `unset`?

`set` используется для установки и показа значений позиционных параметров и опций оболочки. `typeset` в некоторых оболочках (например, `ksh`, `zsh`) используется для объявления типов переменных. `unset` используется для сброса значений переменных или функций.

14. Как передаются параметры в командные файлы?

Параметры передаются в командные файлы через пространство после имени файла при его вызове `script_name.sh param1 param2`. В скрипте к ним можно обращаться через `$1`, `$2` и т. д., где `$1` это первый параметр, `$2` второй и так далее.

15. Назовите специальные переменные языка **bash** и их назначение.

- `$0` - имя скрипта
- `$1 ... $9` - позиционные параметры (аргументы командной строки)
- `$#` - количество позиционных параметров
- `$?` - статус выхода последней выполненной команды
- `$$` - PID (идентификатор процесса) текущего скрипта
- `$_` - PID последней выполненной в фоне команды
- `$@`, `$*` - все позиционные параметры как список