УРОК 17. ЗАВЕРШЕНИЕ РАБОТЫ СО СКРИПТАМИ

ПРАКТИЧЕСКАЯ РАБОТА СО СКРИПТАМИ

Задача 1

Создать скрипт testscript.sh который выполняет следующее:

- В каталоге /home создаст 5 каталогов с именами Dir1 … Dir5
- В каждый из созданных каталогов создаст 5 файлов с интервалом 5 секунд с именами File1.txt … File5.txt
- В каждый файл запишет текущие время в формате H-M-S
- По окончанию создания каждого каталога со списком файлов выводит на экран список файлов.
- Создаст сжатый tar архив в каталоге /tmp/Arh с именем Arh- «ТЕКУЩАЯ ДАТА» (Формат даты d-m-y) в архив упакует все созданные выше каталоги.
- Создаст файл ArhList.txt со списком содержимого архива.
- Разархивировать получившийся архив в новый путь /opt/newfolder/ сохранить структуру каталогов.

В скрипте необходимо предусмотреть возможность изменить пути для создания и распаковки архива т.е используем переменные для этих путей

Решение 1

```
#!/bin/bash
#
ARHPATH=/tmp/Arh
EXTPATH=/opt/newfolder
DATE=`date '+%d-%m-%y'`
echo $DATE
for i in {1..5}
do
mkdir -p Dir$i
for j in {1..5}
do
date +'%H-%M-%S' > Dir$i/File-$j.txt
sleep 5
done
ls Dir$i
```


```
done
tar -czvf $ARHPATH/Arh-$DATE.tar.gz Dir* >> ArhList.txt
sleep 5
mkdir -p $EXTPATH
tar -xzf $ARHPATH/Arh-$DATE.tar.gz -C $EXTPATH
```

Задача 2

Написать скрипт, который позволяет пинговать указанный адрес и запрашивает количество запросов для проверки. Как результат - возвращает среднее значение ответа в ms.

Команда ping в операционных системах UNIX и Linux используется для проверки доступности и времени отклика (задержки) удаленного узла (обычно компьютера или сетевого устройства) в сети. Она отправляет запросы ICMP ECHO_REQUEST на указанный узел и ждет ответа.

Команда ping отправляет пакеты на узел с каким-то адресом и ожидает ответа. Если узел доступен и настроен на ответ на запросы ICMP, команда ping будет выводить информацию о каждом отправленном пакете, времени его отправки и времени получения ответа. Также будет отображаться общее время, количество отправленных и полученных пакетов, а также статистика о потерянных пакетах.

Решение 2

```
#!/bin/bash

read -p "Введите адрес для пинга: " address
read -p "Введите количество запросов для проверки: " count
ping_result=$(ping -c "$count" "$address")
  avg_ping=$(echo "$ping_result" | awk -F'/' '/^rtt/ { print $5 }')
   echo "Среднее время пинга до $address составляет: $avg_ping мс"
```

Важно - при использовании ping на mac - ответ будет немного другим и вместо rtt мы получим round-trip.

Задача 3

Написать скрипт по проверке целостности файлов в указанной директории. Скрипт вычисляет MD5-хэш каждого файла в этой директории и сохраняет результаты в текстовом файле, который содержит имена файлов и соответствующие им MD5-хэши.

 **MD5-хэш (Message Digest Algorithm 5) - это криптографическая хэш-функция, которая принимает на вход сообщение произвольной длины и выдает на выходе хэш-значение фиксированной длины, обычно 128 бит (или 32 шестнадцатеричных символа).**

MD5-хэш используется для проверки целостности данных. Если MD5-хэш файла до передачи и после получения совпадают, это означает, что файл не был изменен в процессе передачи.

Решение 3

```
#!/bin/bash
read -p "Enter path for control: "  pathcheck
read -p "enter pat for result file: " pathresult
mkdir -p  $pathresult
        for var in  $pathcheck/*
          do
                echo  "file name - " $var
                cd $pathresult
                md5sum $pathcheck/$var >> $pathcheck-md5sum.txt
          done
```

Таким образом скрипт запрашивает у пользователя путь к директории, содержимое которой требуется контролировать, запрашивает у пользователя путь для сохранения результата контроля.
Затем создает директорию для сохранения результата контроля (если она не существует).

Для каждого файла в указанной директории выполняет следующие действия:

- Выводит на экран имя файла.
- Вычисляет MD5-хэш файла с использованием команды md5sum.
- Сохраняет результат (имя файла и соответствующий MD5-хэш) в файле

$pathcheck-md5sum.txt в указанной директории для сохранения результата контроля.

Задача 4

Написать скрипт, который:

- Запрашивает у пользователя путь, где будет создан новый скрипт.
- Запрашивает у пользователя имя для нового скрипта.
- Создает директорию для скрипта (если она не существует).
- Проверяет, существует ли уже файл с указанным именем в указанной директории.
- Если файл с таким именем уже существует, выводит сообщение об ошибке.
- Если файл с указанным именем не существует, создает новый скрипт с указанным именем в указанной директории.
- Добавляет шаблонный заголовок в созданный скрипт.
- Назначает права выполнения для скрипта.
- Выводит сообщение о завершении работы.

Решение 4

```
#!/bin/bash

read -p "Enter path for script: " SCRIPTPATH
read -p "Enter name for your script: " NAME
mkdir -p $SCRIPTPATH
ls  $SCRIPTPATH/$NAME 2>/dev/null
if [ $? -eq 0 ]
  then
        echo "BAD NAME " $NAME "file exists"
  else
echo "Script will be created using path " $SCRIPTPATH/$NAME
echo -e "#!/bin/bash\n#\n#Write the code here\n#\n" > $SCRIPTPATH/$NAME
chmod +x $SCRIPTPATH/$NAME
echo "DONE"
fi
```

Задача 5

Написать скрипт, который:

- Запрашивает у пользователя директорию, в которой нужно произвести изменения (или использует текущую директорию, если пользователь не вводит никаких данных).
- Запрашивает текущее и новое расширения файлов.
- Меняет расширение всех файлов с указанным текущим расширением в указанной директории на новое.

Решение 5

```
#!/bin/bash

# Запрос директории у пользователя
read -p "Введите директорию для работы (или нажмите Enter для использования текущей): " directory

# Если директория не указана, используем текущую
if [ -z "$directory" ]; then
    directory="."
fi

# Проверка, существует ли указанная директория
if [ ! -d "$directory" ]; then
    echo "Директория '$directory' не существует или недоступна."
    exit 1
fi

# Запрос текущего расширения и нового расширения
read -p "Введите текущее расширение файлов: " current_extension
read -p "Введите новое расширение файлов: " new_extension

# Переходим в указанную директорию
cd "$directory" || exit

# Меняем расширение файлов
for file in *.$current_extension; do
    # Проверка, существует ли файл с текущим расширением
    if [ -e "$file" ]; then
```


```
        # Получаем имя файла без расширения
        filename=$(basename "$file" ".$current_extension")
        # Меняем расширение файла
        mv "$file" "$filename.$new_extension"
        echo "Изменено расширение файла: $file → $filename.$new_extension"
    fi
done

echo "Готово."
```