***Факультет инфокоммуникационных систем и технологий***

**КАФЕДРА ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И УПРАВЛЯЮЩИХ СИСТЕМ**

**Отчёт по практической работе №8**

**по дисциплине «Операционные системы, среды и оболочки»**

**Выполнил:** *студент гр. ИО-21 Ветров Е.М.*

**Преподаватель:** *доцент каф ИТУС* Исаева Г.Н.

***Цель работы:*** Исследование отличий разработки командных файлов в среде ОС Linux от командных файлов в среде Windows. Создание элементарных скриптов в ОС Linux Ubuntu.

1. Составить командный файл, определяющий: 46 кому принадлежит текущий каталог; дату его создания и состав содержащихся в нём файлов; общее количество файлов в текущем каталоге; последние 10 раз, когда заходил в систему пользователь с текущим «логином».

Для создания командного файла в Linux сценарий нужно написать в файл с расширением ".sh", например, "file\_info.sh". Пример сценария:

```

#!/bin/bash

echo "Owner: $(ls -ld . | awk '{print $3}')"

echo "Creation date: $(ls -ld . | awk '{print $6, $7, $8}')"

echo "Number of files: $(find . -type f | wc -l)"

echo "Last 10 logins:"

last -n 10 $(whoami)

```

Этот скрипт выводит информацию о владельце текущего каталога, дате его создания и количеству файлов в нем. В последней строке скрипт использует команду "last" для вывода информации последних 10 логинов пользователя с текущим именем пользователя.

Чтобы запустить этот скрипт, сначала необходимо дать ему права на выполнение командой `chmod +x file\_info.sh`, а затем запустить его командой `./file\_info.sh`. Результаты будут выведены в консоль.

На рисунке 1 и 2 предоставлено код скрипта и запуска этого скрипта.

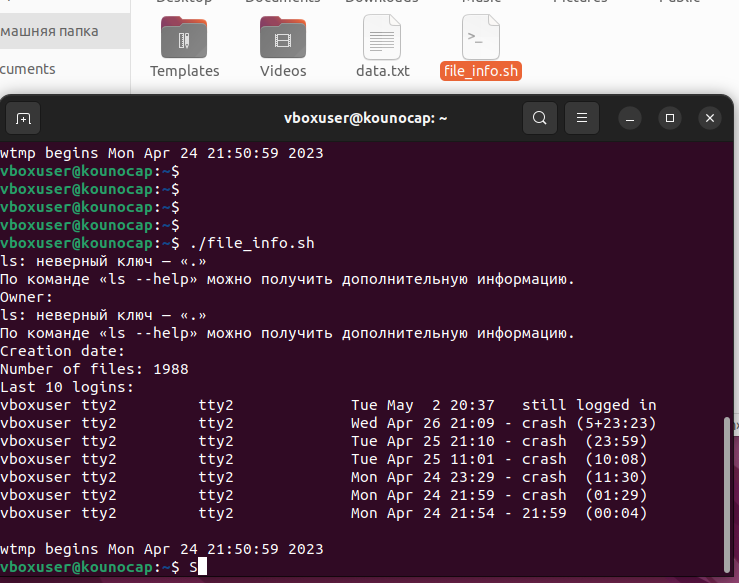


Рисунок 1 Запуск скрипта

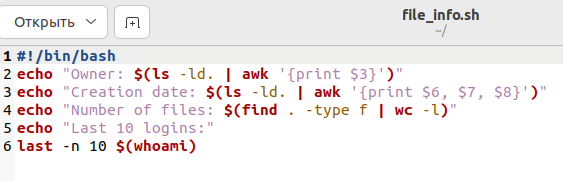


Рисунок 2 Код этого скрипта

2. Написать скрипт: записать ряд Фибоначчи в файл; выдать n-ое число Фибоначчи на консоль, определить сумму n чисел этого ряда. Значение параметра n должно передаваться при вызове скрипта в среде окружения.

Вот пример скрипта, который записывает ряд Фибоначчи в файл, а затем находит n-ое число и сумму первых n чисел ряда Фибоначчи, параметр n передается при вызове скрипта:

```

#!/bin/bash

# проверяем, есть ли параметр n

if [ -z "$1" ]

then

echo "Вы не ввели число n"

exit 1

fi

n=$1 # присваиваем значение параметра n

# проверяем, что параметр n - это целое число

if ! [[ $n =~ ^[0-9]+$ ]]

then

echo "n должно быть целым числом"

exit 1

fi

# начальные значения для ряда Фибоначчи

a=0

b=1

sum=1

# записываем ряд Фибоначчи в файл "fibonacci.txt"

echo $a > fibonacci.txt

echo $b >> fibonacci.txt

for ((i=2; i<$n; i++))

do

c=$((a + b)) # вычисляем следующее число ряда

echo $c >> fibonacci.txt # записываем его в файл

a=$b # обновляем значения для следующей итерации

b=$c

sum=$((sum + c)) # вычисляем сумму чисел ряда

done

# выводим n-ое число ряда Фибоначчи и сумму на консоль

echo "n-е число ряда Фибоначчи: $(tail -n 1 fibonacci.txt)"

echo "Сумма первых $n чисел ряда Фибоначчи: $sum"

```

Сохраните этот скрипт в файл с расширением ".sh", например "fibonacci.sh", и запустите его с параметром n, например так:

```

./fibonacci.sh 10

```

Эта команда запустит скрипт, вычислит первые 10 чисел ряда Фибоначчи, запишет эти числа в файл "fibonacci.txt" и выведет n-ое число и сумму первых n чисел ряда на консоль.

На рисунке 3 мы видим вывод числа Фибоначчи.

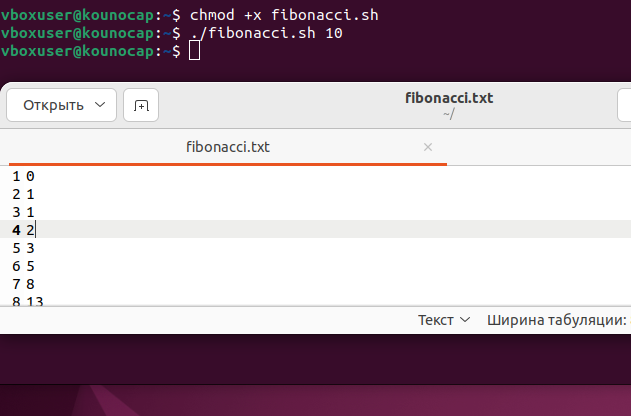


Рисунок 3 Числа Фибоначчи

3. Создать небольшой скрипт, в котором было бы предусмотрено ожидание ввода трёх переменных с консоли, а затем выдача их на экран. Провести эксперимент, задавая недостаточное количество значений переменных или избыточное. Описать, как реагирует командный интерпретатор на такие действия.

Для создания скрипта, который будет ожидать ввода трех переменных с консоли и выводить их на экран, можно использовать следующий код:

#!/bin/bash

# Запрашиваем ввод трех переменных

read -p "Введите первую переменную: " var1

read -p "Введите вторую переменную: " var2

read -p "Введите третью переменную: " var3

# Проверяем, что все переменные были введены

if [ -z "$var1" ] || [ -z "$var2" ] || [ -z "$var3" ]; then

echo "Ошибка: Необходимо ввести значения всех трех переменных."

exit 1

fi

# Выводим значения переменных на экран

echo "Первая переменная: $var1"

echo "Вторая переменная: $var2"

echo "Третья переменная: $var3"

Этот скрипт выведет ошибку, если пользователь не ввел значения всех трех переменных. Кроме того, он завершит выполнение с кодом ошибки 1, что может использоваться в дальнейшей обработке скриптов.

На рисунке 4 и 5 мы можем увидеть скрипт , вывод и ввод чисел.

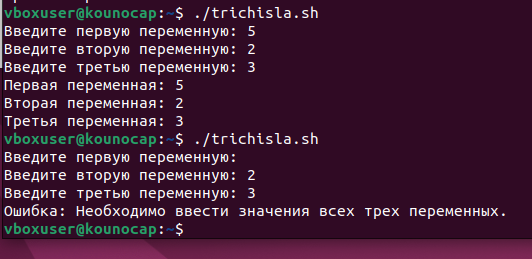


Рисунок 4 Запуск скрипта и ввод чисел и вывод.

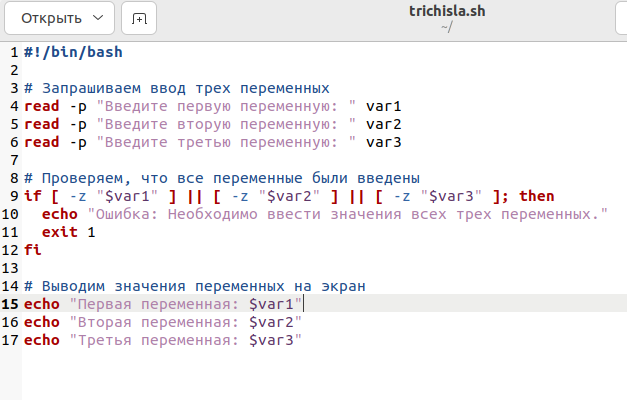


Рисунок 5 Код скрипта

4. Задание по вариантам: разработайте процедуру и выполните ее в среде командной оболочки в ОС Linux. Отладьте полученный сценарий, при необходимости отредактируйте и продемонстрируйте преподавателю.

5) Вводит произвольное число коротких символьных параметров, подсчитывает длину каждого из них и выводит на экран список значений длин и общее число введенных параметров

Код на bash для решения данной задачи:

```bash

#!/bin/bash

count=0 # счетчик количества введенных параметров

lengths=() # массив для хранения длин параметров

read -p "Введите произвольное число коротких символьных параметров: " params

for (( i = 0; i < ${#params}; i++ )); do # перебираем каждый символ ввода

char=${params:$i:1}

if [[ $char != " " ]]; then # проверяем, что символ не является пробелом

(( count++ ))

len=${#char}

lengths+=( $len ) # добавляем длину параметра в массив

echo "Параметр #$count: $char (длина $len)"

fi

done

echo "Общее число введенных параметров: $count"

echo "Длины введенных параметров: ${lengths[@]}" # выводим список длин

```

Теперь по шагам для запуска скрипта в cmd Linux:

1. Открываем терминал (консоль).

2. Создаем новый файл с расширением .sh при помощи команды touch filename.sh (.sh - расширение скрипта, filename - имя нового файла).

3. Открываем редактор nano (или другой) при помощи команды nano filename.sh.

4. Копируем весь код скрипта и вставляем в редактор.

5. Сохраняем скрипт при помощи сочетания клавиш Ctrl+O, затем закрываем редактор при помощи сочетания клавиш Ctrl+X.

6. Делаем скрипт исполняемым при помощи команды chmod +x filename.sh.

7. Запускаем скрипт в терминале при помощи команды ./filename.sh, где filename.sh - имя созданного файла. Скрипт запросит ввод строки с символьными параметрами, после чего выведет на экран длины каждого символа и общее количество введенных параметров.

На рисунке 6 и 7 мы можем видеть сам запуск скрипта и его код.

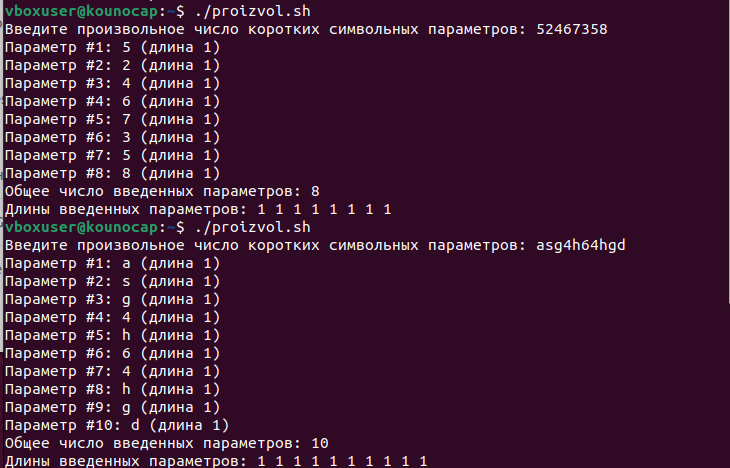


Рисунок 6 Запуск скрипта

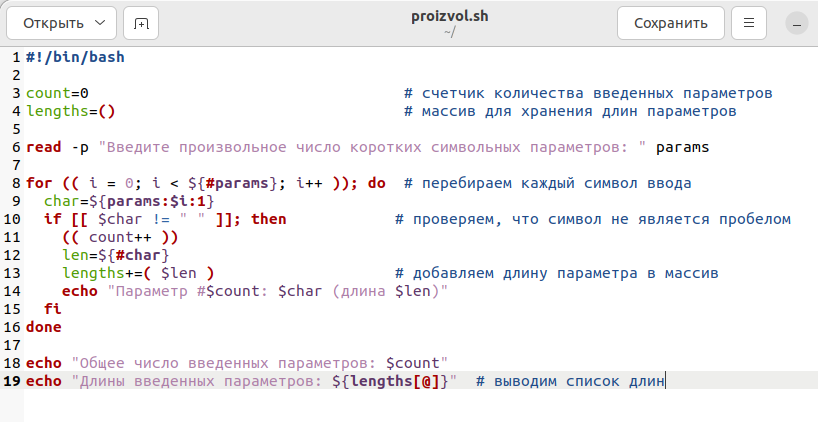


Рисунок 7 Код скрипта

**Вывод:** Я исследовал и нашел отличие разработки командных файлов в среде ОС Linux от командных файлов в среде Windows. Создал элементарные скрипты в ОС Linux Ubuntu.

**Контрольные вопросы**

1. Командные файлы используются для автоматизации задач, которые нужно повторять регулярно или выполнять быстро. Например, создание резервных копий, настройка сетевых соединений, обновление системы, настройка различных настроек и многих других задач, где нужно выполнять множество команд в определенной последовательности.

2. В Linux-подобных ОСых командные файлы написаны на языке shell (bash, sh, zsh и др.), который предоставляет расширенную функциональность для работы с операционной системой. Внешнее изображение командных файлов в Linux-подобных ОС может быть выполнено в соответствии с требованиями, определенными в файловой системе, и содержит расширение .sh или другое соответствующее расширение командного файла. В Windows-подобных ОС командные файлы обычно написаны на языке Batch, имеют расширение .bat или .cmd, и используют команды Windows для работы с операционной системой.

3. Достоинствами программирования в средах окружения являются удобный интерфейс для написания кода, автодополнение, проверка синтаксиса и другие инструменты, упрощающие и ускоряющие процесс написания и отладки кода. Недостатками могут быть медленная работа при больших объемах кода, потеря контроля над процессом выполнения, необходимость в дополнительных инструментах для автоматизации и сборки кода.

4. В командном файле может быть неограниченное количество параметров, но доступны только первые 9 параметров для обработки внутри файла как %1, %2, %3 и так далее. Остальные параметры можно получить, используя переменную %\*, которая содержит все параметры в строке.

5. Для расширения количества формальных параметров для исполняемых файлов в Linux-подобных ОС можно использовать переменную $@, которая содержит все аргументы командной строки, которые были переданы в скрипт, кроме имени скрипта. Например:

```bash

#!/bin/bash

echo "Название скрипта: $0"

echo "Все переданные параметры: $@"

echo "Первый параметр: $1"

echo "Второй параметр: $2"

```

В этом примере используется специальная переменная $0, которая содержит имя скрипта. В результате в терминале будут выведены значения переданных параметров командной строки, а также первого и второго параметра, если они были переданы. При необходимости можно добавить больше формальных параметров в скрипт, используя переменную $3, $4 и так далее.