Отчёт по лабораторной работе №12

Операционные системы

Пономарева Татьяна Александровна

Содержание

# 1 Цель работы

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

# 2 Теоретическое введение

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; – С-оболочка (или csh) — надстройка над оболочкой Борна, использующая С-подобный 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/Linux-подобных операционных систем и переносимости прикладных программ на уровне исходного кода. POSIX-совместимые оболочки разработаны на базе оболочки Корна.

# 3 Выполнение лабораторной работы

Даю права на выполнение командой chmod +x lab12\_(номер задания).sh

## 3.1 Задание 1

1. Скрипт, который делает резервную копию самого себя в архиве

#!/bin/bash  
  
# Полный путь к скрипту  
script\_path=$(readlink -f "$0")  
script\_name=$(basename "$script\_path")  
  
# Создание директории backup в домашнем каталоге, если её нет  
backup\_dir="$HOME/backup"  
mkdir -p "$backup\_dir"  
  
# Формирование имени архива с датой и временем  
timestamp=$(date +"%Y%m%d\_%H%M%S")  
backup\_name="$script\_name-$timestamp.tar.gz"  
  
# Архивирование скрипта  
tar -czf "$backup\_dir/$backup\_name" "$script\_path"  
  
# Вывод сообщения  
echo "Резервная копия создана: $backup\_dir/$backup\_name"

Описание:

Создание директории backup — создаёт каталог для хранения резервных копий.

Формирование имени архива — создаёт имя файла с уникальным временем для предотвращения перезаписи.

Копирование и архивация — копирует скрипт и архивирует его в формате tar.gz.

Вывод сообщения — информирует пользователя о завершении операции.

Исполнение (рис. 1).



Рис. 1: Исполнение 1

## 3.2 Задание 2

1. Скрипт, который обрабатывает любое число аргументов и выводит их

#!/bin/bash  
  
# Скрипт выводит все переданные аргументы, независимо от их количества  
  
echo "Всего передано аргументов: $#"  
  
index=1  
for arg in "$@"; do  
 echo "Аргумент $index: $arg"  
 ((index++))  
done

Описание:

Вывод количества аргументов — отображает количество переданных аргументов.

Перебор и вывод аргументов — поочередно выводит каждый аргумент с порядковым номером.

Исполнение (рис. 2).

Исполнение 2

Рис. 2: Исполнение 2

## 3.3 Задание 3

1. Скрипт — аналог команды ls (без использования ls и dir)

#!/bin/bash  
  
#Аналог ls: выводит права доступа, размер и имя каждого файла в каталоге  
  
# Устанавливаем директорию (по умолчанию текущая)  
directory="${1:-.}"  
  
# Проверяем, существует ли каталог  
if [ ! -d "$directory" ]; then  
 echo "Ошибка: Каталог не существует: $directory"  
 exit 1  
fi  
  
echo "Содержимое каталога: $directory"  
echo  
  
# Перебираем файлы и каталоги в указанной директории  
for file in "$directory"/\*; do  
 if [ -e "$file" ]; then  
 perms=$(stat -c "%A" "$file") # Права доступа  
 size=$(stat -c "%s" "$file") # Размер  
 name=$(basename "$file") # Имя файла  
 echo "$perms $size байт $name"  
 fi  
done

Описание:

Установка директории — задаёт директорию для работы (по умолчанию текущая).

Проверка существования директории — проверяет, существует ли указанная директория.

Перебор файлов — перебирает все файлы в директории, выводит права доступа, размер и имя каждого файла.

Исполнение (рис. 3).



Рис. 3: Исполнение 3

## 3.4 Задание 4

1. Скрипт, который считает файлы с заданным расширением в указанной директории

#!/bin/bash  
  
# Проверка корректности аргументов  
if [ $# -ne 2 ]; then  
 echo "Использование: $0 <расширение\_файла> <путь\_к\_директории>"  
 echo "Пример: $0 .txt /home/user/documents"  
 exit 1  
fi  
  
extension="$1"  
directory="$2"  
  
# Проверка существования директории  
if [ ! -d "$directory" ]; then  
 echo "Ошибка: Директория не найдена: $directory"  
 exit 1  
fi  
  
# Подсчёт количества файлов с заданным расширением  
count=$(find "$directory" -type f -name "\*$extension" | wc -l)  
  
# Вывод результата  
echo "Файлов с расширением $extension в каталоге $directory: $count"

Описание:

Проверка аргументов — проверяет количество переданных аргументов и выводит ошибку, если их не два.

Задание переменных — сохраняет расширение файла и путь к директории.

Проверка существования директории — проверяет, существует ли указанная директория.

Подсчёт файлов — с помощью find ищет файлы с нужным расширением и считает их количество.

Вывод результата — информирует пользователя о количестве найденных файлов.

Исполнение (рис. 4).

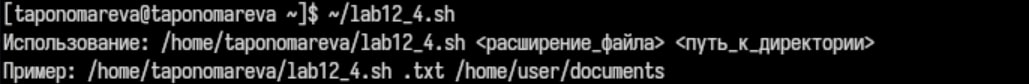


Рис. 4: Исполнение 4

# 4 Ответы на контрольные вопросы

1. Что такое командная оболочка? Примеры и различия Командная оболочка (shell) — это программа, позволяющая пользователю взаимодействовать с операционной системой с помощью командной строки. Она интерпретирует команды пользователя и запускает соответствующие программы.

Примеры оболочек:

* sh (Bourne Shell) — базовая оболочка UNIX.
* bash (Bourne Again Shell) — улучшенная версия sh, популярная в Linux.
* csh (C Shell) — использует C-подобный синтаксис.
* ksh (Korn Shell) — сочетает черты sh и csh.
* zsh — расширенная оболочка с поддержкой автодополнения и расширений.

Различия:

* Синтаксис и доступные функции (например, история команд, работа с массивами, автодополнение)
* Поддержка расширенных возможностей и скриптов

1. Что такое POSIX? POSIX (Portable Operating System Interface) — это набор стандартов, разработанных IEEE, обеспечивающих совместимость между UNIX-подобными операционными системами на уровне исходного кода.

POSIX описывает:

* Поведение оболочек.
* Стандартные команды.
* Системные вызовы и API.

1. Как определяются переменные и массивы в bash?

Переменные:

name="Alice"  
echo $name

Массивы:

* array=(one two three)
* echo ${array[0]} # one
* echo ${array[@]} # one two three

1. Назначение операторов let и read

let — выполняет арифметические операции: let a=5+3

read — считывает ввод пользователя: read name echo “Привет, $name”

1. Арифметические операции в bash

В bash доступны:

Сложение: +

Вычитание: -

Умножение: \*

Деление: /

Остаток от деления: %

Инкремент/декремент: ++, –

Сравнения: ==, !=, <, >, <=, >=

1. Что означает операция (( ))?

(( выражение )) — конструкция для выполнения арифметических вычислений.

Пример: ((a=3+5)) echo $a # 8

Также может использоваться как условие: if ((a > 5)); then echo “Больше 5”; fi

1. Стандартные имена переменных bash

$HOME — домашняя директория пользователя.

$USER — имя пользователя.

$PWD — текущая директория.

$PATH — пути поиска исполняемых файлов.

$SHELL — путь к текущей оболочке.

$HOSTNAME — имя хоста.

1. Что такое метасимволы?

Метасимволы — специальные символы, которые имеют особое значение в оболочке:

* — любое количество любых символов.

? — любой один символ.

[] — любой символ из указанного набора.

$, &, |, >, <, ;, (), {} и другие — служебные символы.

1. Как экранировать метасимволы?

Используй обратный слеш  или заключи в кавычки:

echo $HOME # выведет $HOME, а не значение переменной echo “*file*” # кавычки предотвратят расширение по шаблону

1. Как создавать и запускать командные файлы?

Создадим файл с расширением .sh, например: nano myscript.sh

Напишем команды в bash внутри.

Сделаем файл исполняемым: chmod +x myscript.sh

Запустим: ./myscript.sh

1. Как определяются функции в bash?

my\_function() { echo “Это функция” }

Вызов: my\_function 12. Как узнать, является ли файл каталогом или обычным файлом?

if [ -d “путь” ]; then echo “Это каталог” elif [ -f “путь” ]; then echo “Это файл” fi

1. Назначение команд set, typeset, unset

set — устанавливает параметры оболочки или выводит переменные.

typeset — (в некоторых оболочках) управляет атрибутами переменных.

unset — удаляет переменные или функции:

unset myvar

1. Как передаются параметры в командные файлы?

Аргументы передаются через специальные переменные:

$1, $2, … — позиционные параметры.

$@ — все аргументы по отдельности.

$# — количество аргументов.

Пример: + echo “Первый аргумент: $1” + echo “Всего аргументов: $#”

1. Специальные переменные bash и их назначение

* Переменная Назначение
* $0 Имя скрипта
* $1..$9 Позиционные аргументы
* $# Количество аргументов
* $@ Все аргументы как отдельные слова
* $\* Все аргументы как одна строка
* $? Код возврата последней команды
* $$ PID текущего процесса
* $! PID последнего фонового процесса

# 5 Выводы

Были изучены основы программирования в оболочке ОС UNIX/Linux. Были получены знания о написании небольших командных файлов.

# Список литературы

1. [Курс на ТУИС](https://esystem.rudn.ru/course/view.php?id=113)