Содержание

[**1.** **Основные команды linux и работа с ними:** 2](#_Toc165657727)

[**2.** **Работа с архивацией файла:** 2](#_Toc165657728)

[**3.** **Задание: создать скрипт на языке Bash, который будет автоматически создавать каталог, копировать файлы, выполнять команды и так далее, используя ваши ранее созданные команды.** 2](#_Toc165657729)

[**4.** **Управление пользователями и группами (создание и удаления пользователя/группы, добавление пользователя в группу, изменение прав доступа):** 2](#_Toc165657730)

[**5.** **Чтение и вывод содержимого файла:** 2](#_Toc165657731)

[**6.** **Создание и запись в файл:** 2](#_Toc165657732)

[**7.** **Запуск команды в терминале:** 2](#_Toc165657733)

[**8.** **Получение информации о системе:** 2](#_Toc165657734)

[**9.** **Управление процессами:** 2](#_Toc165657735)

[**10.** **Задание список студентов:** 2](#_Toc165657736)

1. **Основные команды linux и работа с ними:**
   1. Создание файла:

Создайте файл с именем "hello.txt" и добавьте в него строку "Hello, world!" с использованием команды echo.

* 1. Просмотр содержимого файла:

Выведите содержимое файла "hello.txt" на экран с помощью команды cat.

* 1. Создание каталога:

Создайте новый каталог с именем "my\_directory" с помощью команды mkdir.

* 1. Копирование файла:

Скопируйте файл "hello.txt" в каталог "my\_directory" с помощью команды cp.

* 1. Переименование файла:

Переименуйте файл "hello.txt" в "greetings.txt" с помощью команды mv.

* 1. Удаление файла:

Удалите файл "greetings.txt" с помощью команды rm.

* 1. Поиск файла:

Найдите все файлы с расширением ".txt" в текущем каталоге с помощью команды find.

* 1. Архивация файлов:

Создайте архив "my\_archive.tar.gz", содержащий все файлы из каталога "my\_directory", с помощью команды tar.

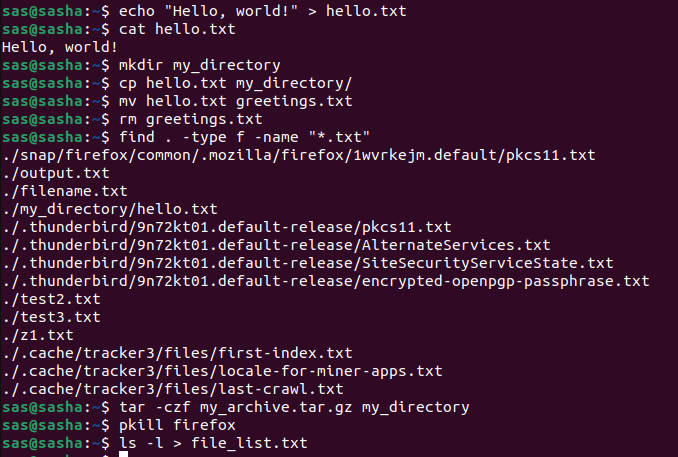
* 1. Управление процессами:

Найдите и завершите все процессы с именем "firefox" с помощью команды pkill.

* 1. Перенаправление вывода:

Запишите вывод команды ls -l в файл "file\_list.txt", используя перенаправление вывода.

Как это выглядит в терминале linux?



1. **Работа с архивацией файла:**

Изображение выглядит как текст, электроника, снимок экрана, программное обеспечение

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

1. **Задание: создать скрипт на языке Bash, который будет автоматически создавать каталог, копировать файлы, выполнять команды и так далее, используя ваши ранее созданные команды.**

Пример скрипта:

#!/bin/bash

# Создание каталога

mkdir my\_backup

# Копирование файла

cp hello.txt my\_backup/

# Переименование файла

mv my\_backup/hello.txt my\_backup/greetings.txt

# Удаление файла

rm my\_backup/greetings.txt

# Поиск файла

echo "Список всех файлов с расширением .txt:"

find . -type f -name "\*.txt"

# Архивация файлов

tar -czf my\_archive.tar.gz my\_backup

# Управление процессами

echo "Завершение всех процессов firefox:"

pkill firefox

# Перенаправление вывода

ls -l > file\_list.txt

# Вывод содержимого созданного файла

echo "Содержимое файла file\_list.txt:"

cat file\_list.txt

Чтобы выполнить этот скрипт нужно:

* Открыть текстовый редактор и скопировать вышеприведенный скрипт.
* Вставить его в файл с расширением .sh, например my\_script.sh.
* Сохранить файл.
* Открыть терминал и перейти в каталог, где находится скрипт.
* Сделать скрипт исполняемым с помощью команды chmod +x my\_script.sh.
* Запустить скрипт с помощью команды ./my\_script.sh.

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

1. **Управление пользователями и группами (создание и удаления пользователя/группы, добавление пользователя в группу, изменение прав доступа):**

Изображение выглядит как текст, снимок экрана, меню, Шрифт

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, меню, Шрифт

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

1. **Чтение и вывод содержимого файла:**

Программа, которая открывает файл и выводит его содержимое на экран.

Листинг программы:

#include <stdio.h>

#include <stdlib.h>

int main() {

FILE \*file;

char ch;

// Открытие файла для чтения

file = fopen("z1.txt", "r");

// Проверка на успешное открытие файла

if (file == NULL) {

printf("Ошибка открытия файла.\n");

return 1;

}

// Чтение и вывод содержимого файла

printf("Содержимое файла z1.txt:\n");

while ((ch = fgetc(file)) != EOF) {

putchar(ch);

}

// Закрытие файла

fclose(file);

return 0;

}

Работа в линукс:

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

1. **Создание и запись в файл:**

Программа, которая создает новый файл и записывает в него некоторую информацию.

Листинг программы:

#include <stdio.h>

int main() {

FILE \*file;

char text[] = "Привет я студентка 32ИС!\n";

// Создание нового файла для записи

file = fopen("output.txt", "w");

// Проверка на успешное создание файла

if (file == NULL) {

printf("Ошибка создания файла.\n");

return 1;

}

// Запись текста в файл

fprintf(file, "%s", text);

// Закрытие файла

fclose(file);

printf("Текст успешно записан в файл 'output.txt'.\n");

return 0;

}

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

1. **Запуск команды в терминале:**

Программа, которая выполняет команду в терминале и выводит результат.

Листинг программы:

#include <stdio.h>

#include <stdlib.h>

int main() {

char command[100];

// Ввод команды от пользователя

printf("Введите команду для выполнения: ");

fgets(command, sizeof(command), stdin);

// Выполнение команды и вывод результата

printf("Результат выполнения команды:\n");

system(command);

return 0;

}

Работа в линукс:

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

1. **Получение информации о системе:**

Программа, которая выводит информацию о системе (например, версию Ubuntu, количество ядер процессора).

Листинг программы:

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/utsname.h>

int main() {

struct utsname unameData;

int num\_cores;

// Получение информации о системе

uname(&unameData);

num\_cores = sysconf(\_SC\_NPROCESSORS\_ONLN);

// Вывод информации о системе

printf("Информация о системе:\n");

printf(" - Операционная система: %s %s\n", unameData.sysname, unameData.release);

printf(" - Версия Ubuntu: %s\n", unameData.version);

printf(" - Количество ядер процессора: %d\n", num\_cores);

return 0;

}

Работа с линукс:

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

1. **Управление процессами:**

Программа, которая создает новый процесс, ожидает его завершения и выводит его код возврата.

Листинг программы:

#include <stdio.h>

#include <stdlib.h>

#include <sys/wait.h>

#include <unistd.h>

int main() {

pid\_t pid;

int status;

// Создание нового процесса

pid = fork();

if (pid < 0) {

// Ошибка при создании процесса

perror("Ошибка при создании процесса");

exit(1);

} else if (pid == 0) {

// Это дочерний процесс

printf("Дочерний процесс с PID: %d\n", getpid());

// Имитация работы дочернего процесса

sleep(3);

exit(42); // Код возврата дочернего процесса

} else {

// Это родительский процесс

printf("Родительский процесс с PID: %d\n", getpid());

// Ожидание завершения дочернего процесса

waitpid(pid, &status, 0);

if (WIFEXITED(status)) {

printf("Дочерний процесс завершился с кодом: %d\n", WEXITSTATUS(status));

}

}

return 0;

}

Работа с линукс:

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

1. **Задание список студентов:**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define MAX\_STUDENTS 100

typedef struct {

char name[50];

int age;

float grade;

} Student;

void addStudent(Student students[], int \*count) {

if (\*count < MAX\_STUDENTS) {

printf("Введите имя студента: ");

scanf("%s", students[\*count].name);

printf("Введите возраст студента: ");

scanf("%d", &students[\*count].age);

printf("Введите средний балл студента: ");

scanf("%f", &students[\*count].grade);

(\*count)++;

} else {

printf("Нет места для добавления нового студента.\n");

}

}

void displayStudents(Student students[], int count) {

printf("Список студентов:\n");

for (int i = 0; i < count; i++) {

printf("Имя: %s, Возраст: %d, Средний балл: %.2f\n", students[i].name, students[i].age, students[i].grade);

}

}

int compareStudents(const void \*a, const void \*b) {

return strcmp(((Student \*)a)->name, ((Student \*)b)->name);

}

void searchUnsorted(Student students[], int count, char \*name) {

int found = 0;

for (int i = 0; i < count; i++) {

if (strcmp(students[i].name, name) == 0) {

printf("Студент найден:\nИмя: %s, Возраст: %d, Средний балл: %.2f\n", students[i].name, students[i].age, students[i].grade);

found = 1;

break;

}

}

if (!found) {

printf("Студент с именем %s не найден.\n", name);

}

}

void searchSorted(Student students[], int count, char \*name) {

Student \*result = bsearch(name, students, count, sizeof(Student), compareStudents);

if (result != NULL) {

printf("Студент найден:\nИмя: %s, Возраст: %d, Средний балл: %.2f\n", result->name, result->age, result->grade);

} else {

printf("Студент с именем %s не найден.\n", name);

}

}

void selectionSort(Student students[], int count) {

for (int i = 0; i < count - 1; i++) {

int minIndex = i;

for (int j = i + 1; j < count; j++) {

if (students[j].grade < students[minIndex].grade) {

minIndex = j;

}

}

if (minIndex != i) {

Student temp = students[i];

students[i] = students[minIndex];

students[minIndex] = temp;

}

}

}

void bubbleSort(Student students[], int count) {

for (int i = 0; i < count - 1; i++) {

for (int j = 0; j < count - i - 1; j++) {

if (students[j].grade > students[j + 1].grade) {

Student temp = students[j];

students[j] = students[j + 1];

students[j + 1] = temp;

}

}

}

}

void insertionSort(Student students[], int count) {

for (int i = 1; i < count; i++) {

Student key = students[i];

int j = i - 1;

while (j >= 0 && students[j].grade > key.grade) {

students[j + 1] = students[j];

j = j - 1;

}

students[j + 1] = key;

}

}

void writeToFile(Student students[], int count, char \*filename) {

FILE \*file = fopen(filename, "w");

if (file != NULL) {

for (int i = 0; i < count; i++) {

fprintf(file, "%s %d %.2f\n", students[i].name, students[i].age, students[i].grade);

}

fclose(file);

printf("Данные успешно записаны в файл %s.\n", filename);

} else {

printf("Ошибка при открытии файла %s.\n", filename);

}

}

int main() {

Student students[MAX\_STUDENTS];

int count = 0;

int choice;

char name[50];

do {

printf("\nМеню:\n");

printf("1. Добавить студента\n");

printf("2. Вывести список студентов\n");

printf("3. Поиск студента по имени в неупорядоченном списке\n");

printf("4. Поиск студента по имени в упорядоченном списке\n");

printf("5. Сортировка списка студентов: \n");

printf("6. Записать данные студентов в файл\n");

printf("0. Выйти\n");

printf("Введите ваш выбор: ");

scanf("%d", &choice);

switch (choice) {

case 1:

addStudent(students, &count);

break;

case 2:

displayStudents(students, count);

break;

case 3:

printf("Введите имя студента для поиска: ");

scanf("%s", name);

searchUnsorted(students, count, name);

break;

case 4:

printf("Введите имя студента для поиска: ");

scanf("%s", name);

qsort(students, count, sizeof(Student), compareStudents);

searchSorted(students, count, name);

break;

case 5:

printf("Выберите метод сортировки:\n");

printf("1. Сортировка выбором\n");

printf("2. Сортировка обменом\n");

printf("3. Сортировка вставками\n");

printf("Введите ваш выбор: ");

scanf("%d", &choice);

switch (choice) {

case 1:

selectionSort(students, count);

break;

case 2:

bubbleSort(students, count);

break;

case 3:

insertionSort(students, count);

break;

default:

printf("Неверный выбор.\n");

}

printf("Список студентов после сортировки:\n");

displayStudents(students, count);

break;

case 6:

writeToFile(students, count, "students.txt");

break;

case 0:

printf("До свидания!\n");

break;

default:

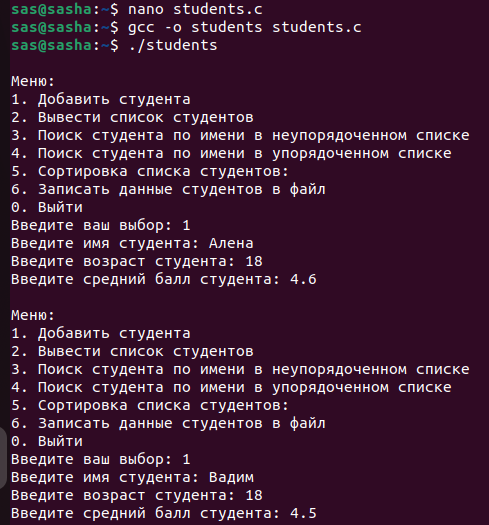
printf("Неверный выбор.\n");

}

} while (choice != 0);

return 0;

}



Изображение выглядит как текст, снимок экрана, меню

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, меню

Автоматически созданное описание