Отчет по лабораторной работе №9

Операционные системы

Фадин В.В.

Содержание

# 1 Цель работы

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

# 2 Выполнение лабораторной работы

## 2.1 Написать скрипт, который при запуске будет делать резервную копию самого себя.

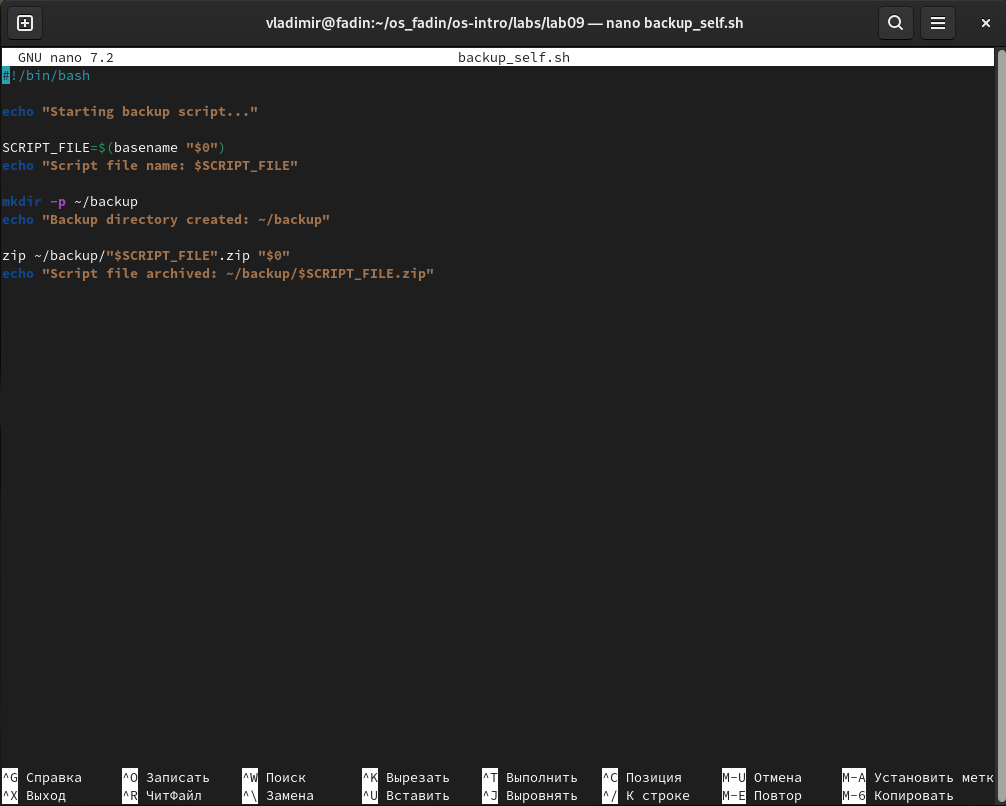


Рис. 1: backup\_self.sh

Тут происходит следующее: 1. SCRIPT\_FILE=$(basename "$0"): эта строка получает текущее имя файла сценария с помощью команды basename, которая удаляет путь к каталогу из имени файла. 2. mkdir -p ~/backup: эта строка создает каталог backup в вашем домашнем каталоге, если он еще не существует. Опция -p указывает mkdir создать каталог с родителями, то есть каталог будет создан, даже если родительский каталог не существует. 3. zip ~/backup/"$SCRIPT\_FILE".zip "$0": эта строка архивирует файл сценария с помощью zip. $0 относится к текущему файлу сценария, а ~/backup/"$SCRIPT\_FILE".zip — это имя выходного файла. Команда zip создаст zip-архив файла сценария и сохранит его в каталоге backup.

Чтобы использовать этот скрипт, сохраним его в файл (например, backup\_self.sh), сделаем файл исполняемым с помощью chmod +x backup\_self.sh, а затем запустим его с ./backup\_self.sh.

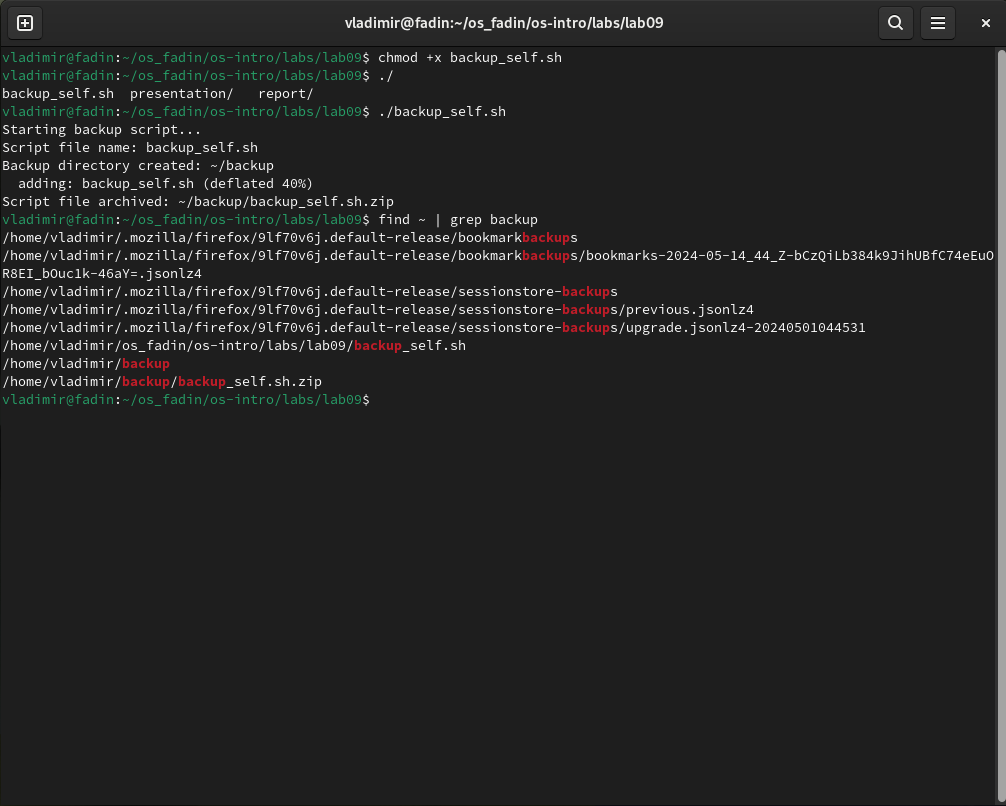


Рис. 2: backup\_self.sh

## 2.2 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.

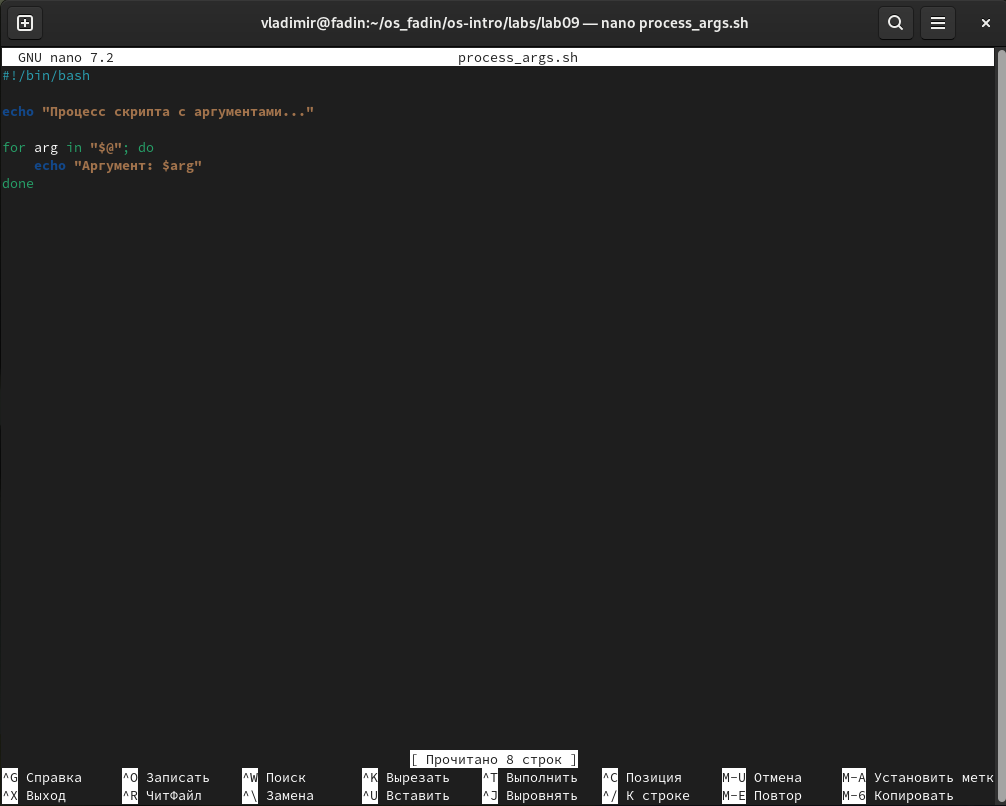


Рис. 3: process\_args.sh

В этом примере $@ — это специальная переменная bash, содержащая все аргументы командной строки. Цикл for проходит по каждому аргументу и выводит его значение с помощью echo.

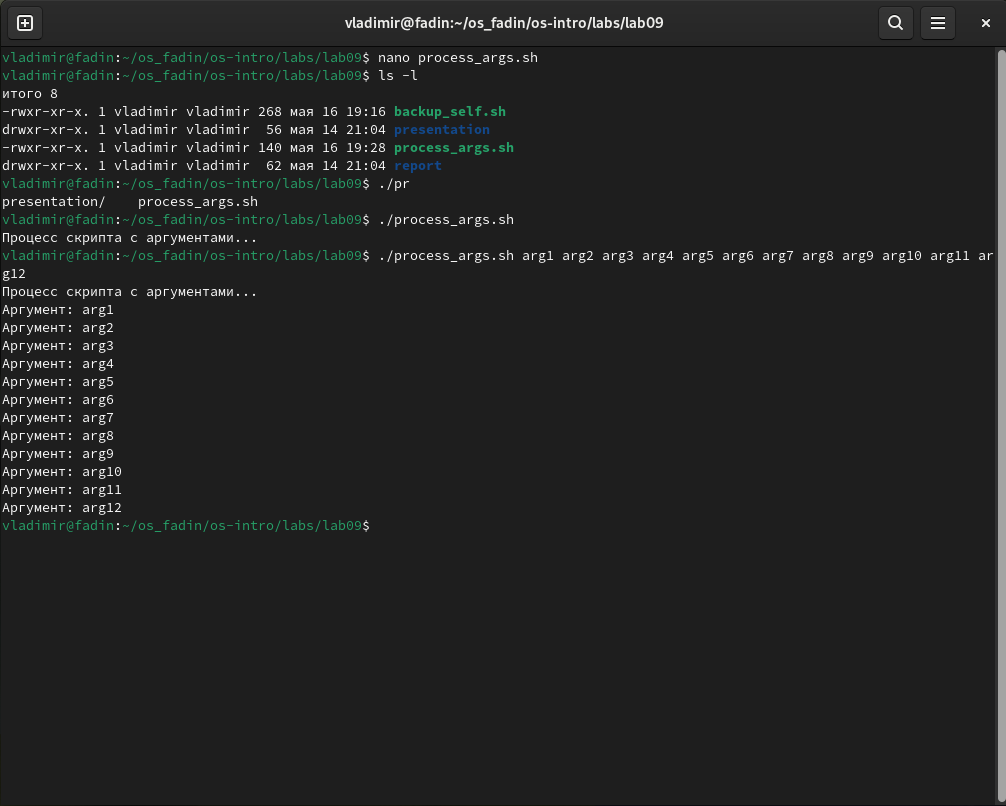


Рис. 4: process\_args.sh

## 2.3 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.

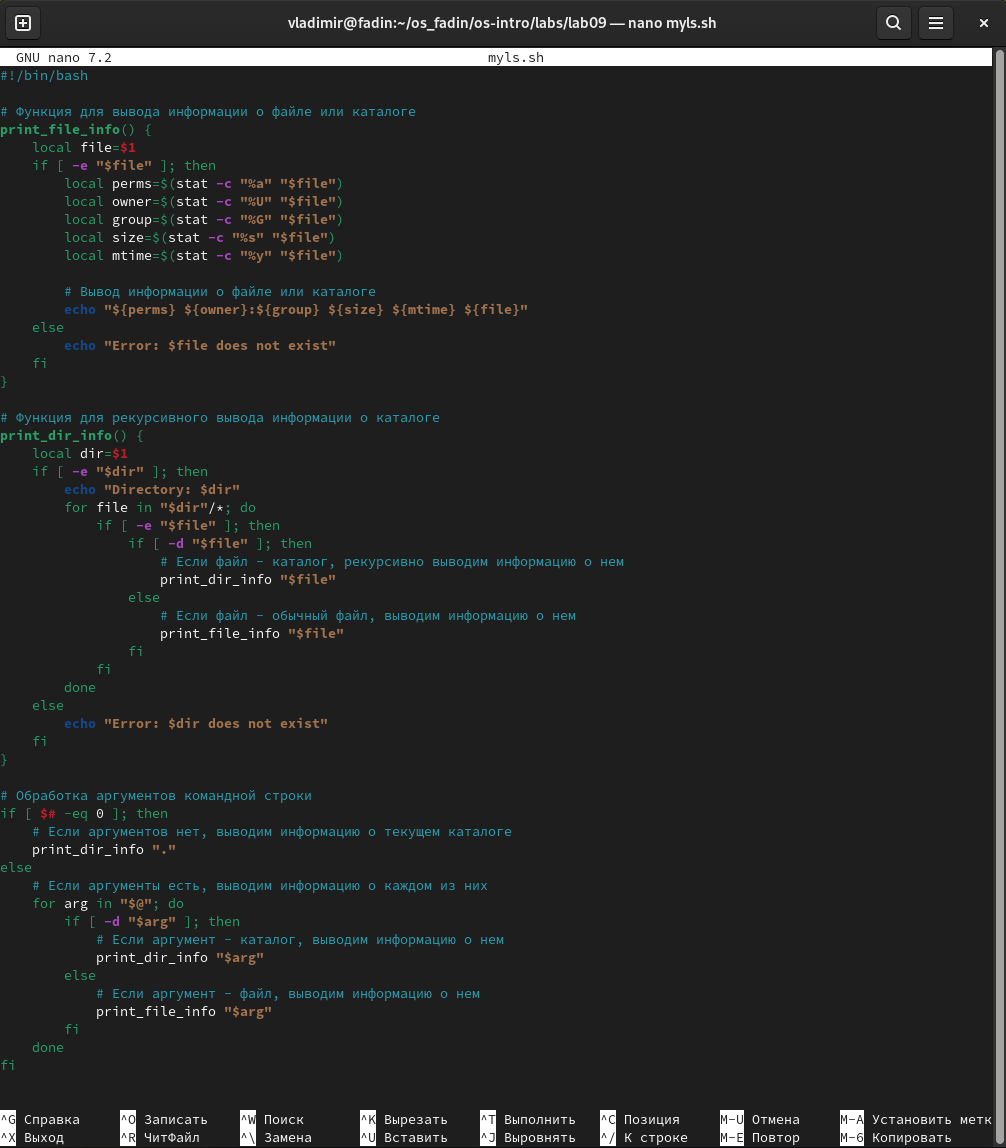


Рис. 5: myls.sh

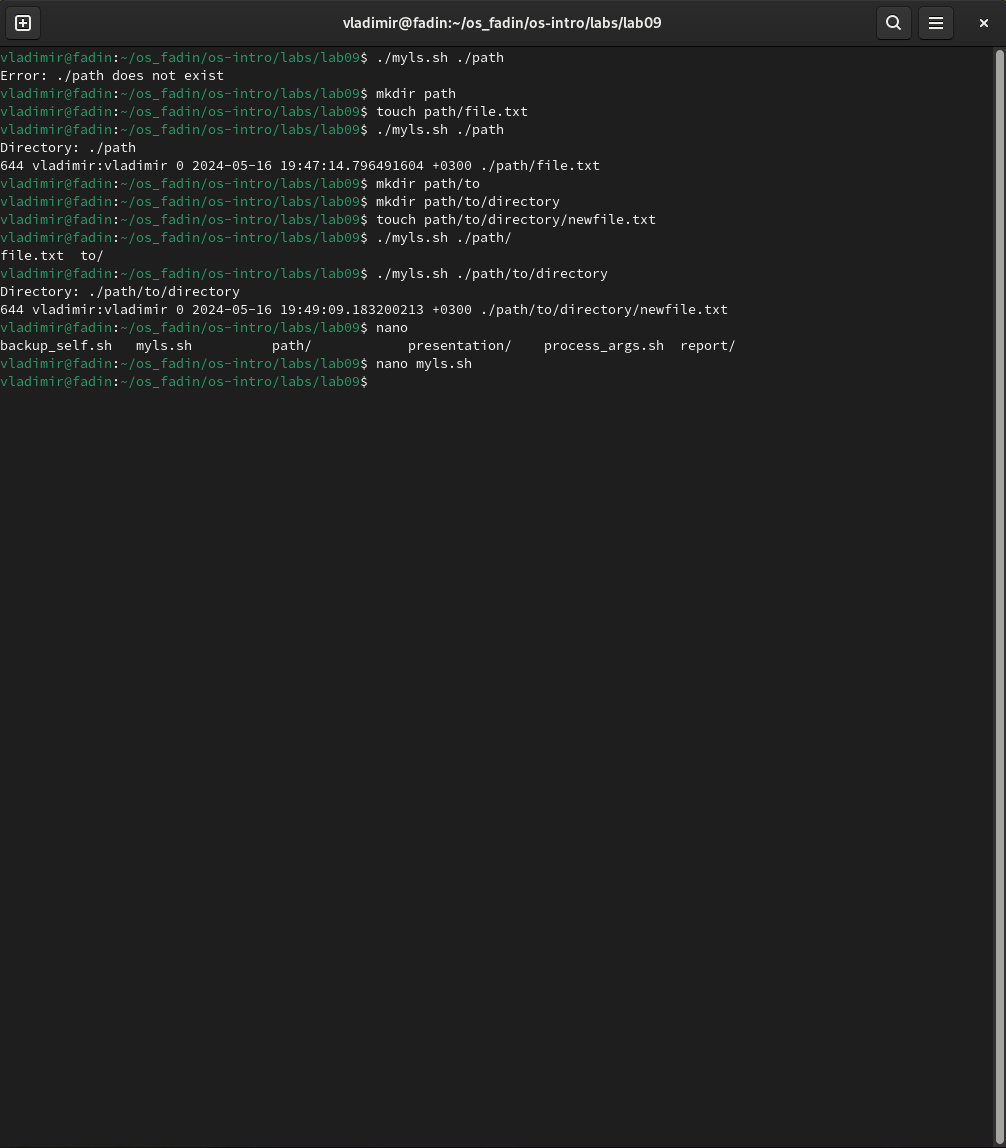


Рис. 6: myls.sh

## 2.4 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

В этом примере мы используем команду find для поиска файлов с расширением в указанной директории. Опция -type f указывает, что мы ищем только файлы, а не каталоги. Опция -name "\*.$file\_extension" указывает, что мы ищем файлы с именем, заканчивающим указанное расширение. Результат поиска приводится в команду wc -l, которая вычисляет количество строк в выводе, что в данном случае равно количеству файлов с расширениями запроса.

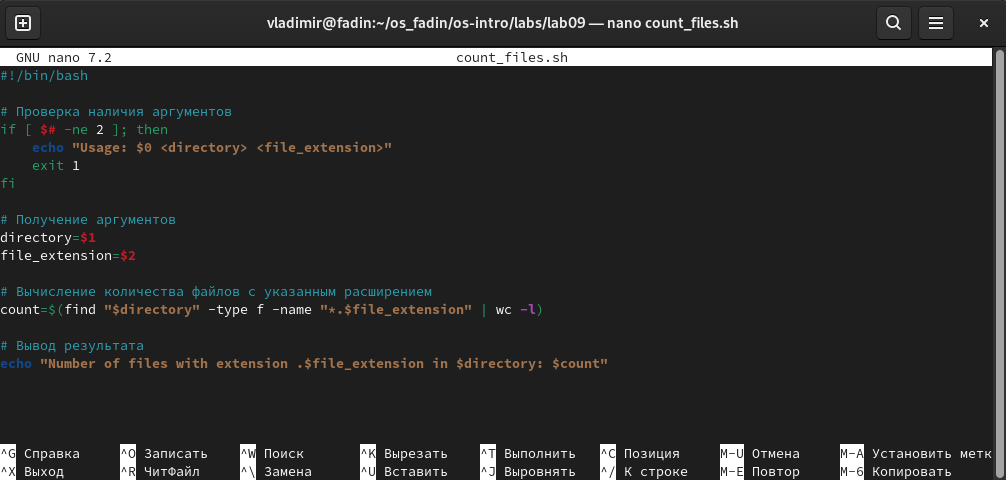


Рис. 7: myls.sh

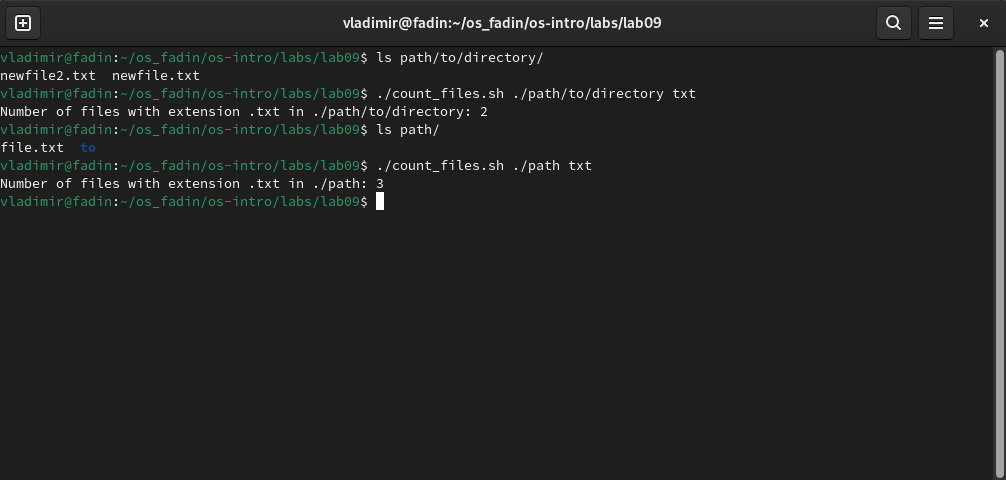


Рис. 8: myls.sh

# 3 Выводы

Изучили основы программирования в оболочке ОС UNIX/Linux. Научились писать небольшие командные файлы.

# 4 Ответы на контрольные вопросы

1. **Что такое командная оболочка? Приведите примеры командных оболочек. Чем они отличаются?** Командная оболочка — это программа, которая позволяет пользователям взаимодействовать с операционной системой путем выполнения команд, сценариев и программ. Примеры командных оболочек: Bash, pdksh, tcsh и zsh. Они различаются синтаксисом, функциональностью и возможностями, такими как завершение команд, история и возможности создания сценариев.
2. **Что такое POSIX?** POSIX (Portable Operating System Interface) — это набор стандартов для операционных систем, включая Unix и Linux, определяющий общий API для взаимодействия с операционной системой. Это обеспечивает совместимость и переносимость программного обеспечения между различными системами.
3. **Как определяются переменные и массивы в Bash?** В Bash переменные определяются с помощью оператора =, а массивы определяются с помощью круглых скобок () и запятых , для разделения элементов.
4. **Какова цель операторов let и read?** Оператор let используется для выполнения арифметических операций, а оператор read используется для чтения ввода от пользователя или файла.
5. **Какие арифметические операции можно выполнять в Bash?** Bash поддерживает базовые арифметические операции, такие как сложение, вычитание, умножение и деление, а также более сложные операции, такие как по модулю и возведение в степень.
6. **Что означает операция (( ))?** Операция (( )) используется для выполнения арифметических операций и вычисления выражений в Bash.
7. **Какие стандартные имена переменных вы знаете?** Стандартные имена переменных в Bash включают, среди прочего, SHELL, PATH, HOME и USER.
8. **Что такое метасимволы?** Метасимволы — это специальные символы в Bash, имеющие определенное значение, например \*, ? и [, которые используются для сопоставления с образцом и подстановки под шаблон.
9. **Как избежать метасимволов?** Метасимволы можно экранировать с помощью обратной косой черты \ или заключая их в кавычки.
10. **Как создавать и запускать командные файлы?** Командные файлы, также известные как сценарии, можно создавать с помощью текстового редактора и запускать с использованием нотации ./, за которой следует имя сценария.
11. **Как определяются функции в Bash?** Функции в Bash определяются с помощью ключевого слова function, за которым следует имя функции и аргументы в круглых скобках.
12. **Как определить, является ли файл каталогом или обычным файлом?** Вы можете использовать команду test или оператор [, чтобы определить, является ли файл каталогом или обычным файлом, используя параметры -d и -f соответственно.
13. **Какова цель команд set, typeset и unset?** Команда set используется для установки параметров оболочки, typeset используется для объявления переменных, а unset используется для удаления переменных или функций.
14. **Как параметры передаются в командные файлы?** Параметры можно передавать в командные файлы с использованием синтаксиса $1, $2 и т. д., который представляет первый, второй и т. д. аргумент командной строки.
15. **Что такое специальные переменные в Bash и каково их назначение?** К специальным переменным в Bash относятся, среди прочего, SHELL, PATH, HOME и USER, которые используются для хранения информации об оболочке и среде пользователя.