### ****Тема 1: Основы Bash и работа с терминалом****

### Урок 1: Что такое Bash и зачем он нужен

**Теория:**

**Bash** (Bourne Again SHell) — это одна из самых популярных оболочек командной строки в операционных системах на базе Linux. Это средство взаимодействия пользователя с операционной системой. Через Bash можно управлять файлами, запускать программы, автоматизировать задачи и создавать скрипты.

Основные преимущества Bash:

* Простота в изучении;
* Поддержка сценариев (scripts);
* Возможность автоматизации повседневных задач;
* Универсальность: доступен во всех дистрибутивах Linux и macOS.

Примеры задач, которые удобно автоматизировать с помощью Bash:

* Очистка логов;
* Архивирование и резервное копирование данных;
* Массовое переименование или перемещение файлов;
* Мониторинг работы служб.

**Тест:**

1. Что такое Bash?  
   **Ответ:** оболочка командной строки
2. Назовите одну из основных задач Bash.  
   **Ответ:** автоматизация
3. Какой тип файлов пишется на Bash для автоматизации?  
   **Ответ:** cкрипты

### Урок 2: Основные команды и синтаксис Bash

**Теория:**

Работа в Bash основана на использовании **команд**. Каждая команда состоит из:

* имени команды;
* параметров;
* аргументов.

Примеры базовых команд:

* pwd — показать текущий каталог;
* ls — список файлов и папок;
* cd — переход в другую директорию;
* mkdir — создать папку;
* rm — удалить файл или папку;
* man — открыть справку по команде.

Синтаксис Bash:

* Команды пишутся в одну строку;
* Можно объединять команды с помощью ; или &&;
* Комментарии начинаются с символа #.

Пример:

cd /home/user && ls -l

Эта строка сначала перейдёт в каталог /home/user, а если это успешно — выведет список файлов.

**Тест:**

1. Команда для показа текущей директории:  
   **Ответ:** pwd
2. Чем отделяются комментарии в Bash?  
   **Ответ:** #
3. Какая команда используется для просмотра содержимого папки?  
   **Ответ:** ls

### ****Тема 2: Переменные и операторы****

### Урок 1: Переменные и типы данных

**Теория:**

Переменные в Bash позволяют хранить значения для последующего использования. Переменные **не имеют строгой типизации**, то есть Bash рассматривает все как строки (по умолчанию).

**Создание переменной:**

NAME="Ivan"

AGE=25

Пробелы вокруг = запрещены!

**Доступ к значению переменной:**

echo $NAME

**Типы значений:**

* строки (например, "hello");
* числа (используются в арифметике);
* списки (через массивы).

**Специальные переменные:**

* $0 — имя скрипта;
* $1, $2 ... — позиционные аргументы;
* $# — количество аргументов;
* $? — код завершения предыдущей команды.

**Тест:**

1. Как обратиться к значению переменной AGE?  
   **Ответ:** $age
2. Какой символ используется для получения значения переменной?  
   **Ответ:** $
3. Можно ли использовать пробелы при создании переменной (например x = 5)?  
   **Ответ:** нет

### Урок 2: Арифметические и логические операторы

**Теория:**

Bash поддерживает простые **арифметические операции**:

echo $((5 + 3)) # 8

**Арифметические операторы:**

* + — сложение
* - — вычитание
* \* — умножение
* / — деление
* % — остаток от деления

Пример:

a=10

b=3

sum=$((a + b))

echo $sum

**Логические операторы** часто используются в условных конструкциях:

* -eq — равно
* -ne — не равно
* -lt, -le — меньше / меньше или равно
* -gt, -ge — больше / больше или равно

Пример:

if [ "$a" -gt "$b" ]; then

echo "$a больше $b"

fi

**Тест:**

1. Какой символ используется для арифметики в Bash?  
   **Ответ:** $
2. Напишите оператор, который проверяет, что x больше y.  
   **Ответ:** -gt
3. Что вернёт echo $((10 % 3))?  
   **Ответ:** 1

### ****Тема 3: Управляющие конструкции****

### Урок 1: Условия: if, else, elif

**Теория:**

Условные конструкции позволяют выполнять различные действия в зависимости от условий. Основной блок:

if [ условие ]; then

команды

elif [ другое\_условие ]; then

другие\_команды

else

альтернативные\_команды

fi

**Примеры условий:**

if [ "$AGE" -ge 18 ]; then

echo "Доступ разрешен"

else

echo "Доступ запрещен"

fi

**Обратите внимание:**

* Квадратные скобки [ ] обязательны (или можно использовать [[ ]]);
* Не забывайте ставить then после if;
* Блок завершается fi.

**Часто используемые проверки:**

* -z "$VAR" — переменная пуста;
* -n "$VAR" — переменная не пуста;
* -f file.txt — файл существует;
* -d папка — каталог существует.

**Тест:**

1. Как заканчивается блок if в Bash?  
   **Ответ:** fi
2. Как проверить, существует ли файл log.txt?  
   **Ответ:** [ -f log.txt ]
3. Какой оператор означает "иначе если"?  
   **Ответ:** elif

### Урок 2: Циклы: for, while, until

**Теория:**

Циклы позволяют повторять действия. В Bash есть три типа циклов: for, while, until.

**Цикл for:**

for i in 1 2 3; do

echo "Число: $i"

done

**Цикл while:**

i=1

while [ $i -le 3 ]; do

echo $i

((i++))

done

**Цикл until (обратный while):**

i=1

until [ $i -gt 3 ]; do

echo $i

((i++))

done

Цикл for часто применяется при работе с файлами:

for file in \*.txt; do

echo "Обрабатываю $file"

done

**Тест:**

1. Какой цикл используется, чтобы повторять до наступления условия?  
   **Ответ:** until
2. Какая команда инкрементирует значение переменной i на 1?  
   **Ответ:** ((i++))
3. Что выведет for i in 1 2 3; do echo $i; done?  
   **Ответ:** 1 2 3

### ****Тема 4: Функции и скрипты****

### Урок 1: Создание Bash-скриптов

**Теория:**

Скрипт — это файл, содержащий последовательность команд Bash. Такой файл можно запускать как программу.

**Шаги создания скрипта:**

1. Создайте файл:

nano my\_script.sh

1. В начале файла укажите **шебанг**:

#!/bin/bash

1. Напишите команды:

#!/bin/bash

echo "Привет, мир!"

1. Сделайте файл исполняемым:

chmod +x my\_script.sh

1. Запустите скрипт:

./my\_script.sh

**Шебанг (#!)** указывает, какой интерпретатор использовать для выполнения команд. Обычно это /bin/bash.

**Советы:**

* Расширение .sh необязательно, но удобно для распознавания.
* Комментарии начинаются с #.

**Тест:**

1. Какой шебанг используется в Bash-скриптах?  
   **Ответ:** #!/bin/bash
2. Команда для запуска скрипта script.sh:  
   **Ответ:** ./script.sh
3. Команда для выдачи прав на запуск скрипта:  
   **Ответ:** chmod +x script.sh

### Урок 2: Функции в Bash

**Теория:**

Функции в Bash позволяют группировать команды и повторно их использовать.

**Объявление функции:**

greet() {

echo "Привет, $1!"

}

**Вызов функции:**

greet "Иван"

**Особенности:**

* $1, $2, ... — позиционные аргументы;
* return может возвращать код (0–255), но не значение;
* echo используется для вывода результата.

**Пример:**

sum() {

result=$(( $1 + $2 ))

echo $result

}

res=$(sum 2 3)

echo "Сумма: $res"

**Тест:**

1. Как объявить функцию с именем hello?  
   **Ответ:** hello()
2. Как получить первый аргумент внутри функции?  
   **Ответ:** $1
3. Как вернуть результат из функции?  
   **Ответ:** echo