Программирование в командном процессоре ОС UNIX. Ветвления и циклы

Лабораторная работа №13

Козомазов Владимир Романович

Содержание

Список иллюстраций

Список таблиц

# 1 Цель работы

**Главная цель:**  
Изучить и практически освоить использование **ветвлений и циклов** в командных процессорах UNIX (Bash, sh, zsh) для создания эффективных и надежных shell-скриптов, автоматизирующих задачи в UNIX-подобных системах.

### 1.0.1 **Конкретные цели работы**

1. **Теоретическое освоение конструкций ветвления и циклов**
2. **Практическое применение знаний**
3. **Оптимизация и отладка скриптов**
4. **Автоматизация реальных задач**
5. **Сравнение и анализ**

Цель работы — не просто изучить синтаксис, а **научиться применять ветвления и циклы для решения реальных задач** в UNIX-системах.

# 2 Задание

### 2.0.1 **Практическое задание**

**по теме:**  
**«Программирование в командном процессоре ОС UNIX. Ветвления и циклы»**

## 2.1 **Цель задания**

Закрепить навыки написания shell-скриптов с использованием:  
✔ **Ветвлений** (if-else, case)  
✔ **Циклов** (for, while, until)  
✔ **Управляющих команд** (break, continue, exit)

## 2.2 **Задачи**

### 2.2.1 **1. Базовые упражнения**

**1.1. Скрипт с ветвлением (if-else)**  
**1.2. Скрипт с case**  
### **2. Работа с циклами**  
**2.1. Скрипт с циклом for**  
**2.2. Скрипт с циклом while**  
**2.3. Скрипт с until**  
### **3. Комбинированные задания**

# 3 Теоретическое введение

## 3.1 **1. Командные процессоры UNIX**

Командные процессоры (shell) — это интерпретаторы команд, обеспечивающие взаимодействие пользователя с операционной системой UNIX/Linux. Наиболее распространенные:  
- **Bash** (Bourne-Again SHell) — стандартный в большинстве дистрибутивов.  
- **sh** (Bourne Shell) — более старый, но обеспечивает совместимость.  
- **zsh**, **ksh** — расширенные версии с дополнительными функциями.

**Особенности shell-скриптов:**  
- Интерпретируются построчно.  
- Не требуют компиляции.  
- Могут включать команды ОС, переменные и управляющие конструкции.

## 3.2 **2. Ветвления в shell-скриптах**

Ветвления позволяют изменять порядок выполнения скрипта в зависимости от условий.

### 3.2.1 **2.1. Конструкция if-elif-else**

if [ условие ]; then   
 # команды при истинности   
elif [ другое\_условие ]; then   
 # альтернативные команды   
else   
 # команды по умолчанию   
fi

**Пример:**

if [ -f "file.txt" ]; then   
 echo "Файл существует."   
else   
 echo "Файл не найден."   
fi

### 3.2.2 **2.2. Конструкция case**

Используется для множественного ветвления:

case $переменная in   
 шаблон1) команды ;;   
 шаблон2) команды ;;   
 \*) команды\_по\_умолчанию ;;   
esac

**Пример:**

case $1 in   
 "start") echo "Запуск службы..." ;;   
 "stop") echo "Остановка..." ;;   
 \*) echo "Неизвестная команда" ;;   
esac

## 3.3 **3. Циклы в shell-скриптах**

Циклы используются для многократного выполнения команд.

### 3.3.1 **3.1. Цикл for**

Перебирает элементы списка:

for переменная in список; do   
 команды   
done

**Пример:**

for file in \*.txt; do   
 echo "Обработка $file"   
done

### 3.3.2 **3.2. Цикл while**

Выполняется, пока условие истинно:

while [ условие ]; do   
 команды   
done

**Пример:**

count=1   
while [ $count -le 5 ]; do   
 echo "Итерация $count"   
 ((count++))   
done

### 3.3.3 **3.3. Цикл until**

Выполняется, пока условие ложно (антипод while):

until [ условие ]; do   
 команды   
done

**Пример:**

until ping -c1 example.com &>/dev/null; do   
 echo "Ожидание ответа от example.com..."   
 sleep 2   
done

## 3.4 **4. Управление выполнением**

* **break** — досрочный выход из цикла.
* **continue** — переход к следующей итерации.
* **exit N** — завершение скрипта с кодом N (0 — успех, 1+ — ошибка).

**Пример:**

for i in {1..10}; do   
 if [ $i -eq 5 ]; then   
 break   
 fi   
 echo $i   
done

## 3.5 **5. Проверка условий**

Условия проверяются с помощью:  
- **[ ]** или **test** — стандартный синтаксис.  
- **[[ ]]** — расширенный (поддерживает &&, ||, регулярные выражения).  
- **(( ))** — для арифметических операций.

**Операторы сравнения:**  
- **Файлы:** -f (существует), -d (директория), -r (доступ на чтение).  
- **Строки:** =, !=, -z (пустая строка).  
- **Числа:** -eq, -ne, -lt, -gt.

**Пример:**

if [[ "$str" == "admin" && $num -gt 10 ]]; then   
 echo "Доступ разрешен."   
fi

## 3.6 **6. Особенности разных shell**

* **Bash**: Поддержка массивов, арифметики (( )), подстановки {1..10}.
* **POSIX sh**: Ограниченный функционал (нет [[ ]], массивов).
* **zsh**: Дополнительные возможности (глобализация, проверка орфографии).

# 4 Выполнение лабораторной работы

Установил менеджер паролей pass с помощью команды sudo dnf install pass pass-otp (рис. 1).



Рис. 1: Установка менеджера паролей

Завершил установку менеджера паролей командой sudo dnf install gopass (рис. 2).



Рис. 2: Завершение установки менеджера паролей

Просмотрел список gpg ключей при помощи команды gpg --list-secret-keys (рис. 3).



Рис. 3: Просмотр списка ключей

Иницилизировал хранилище, написав команду `pass init (рис. 4).

Рис. 4: Иницилизация хранилища

Рис. 4: Иницилизация хранилища

Создал структуру с git командой pass git init (рис. 5).

Рис. 5: Создание структуры

Рис. 5: Создание структуры

Синхронизировался с git командами pass git pull, pass git push

Вручную закоммитил и выложил изменения командами (рис. 6).



Рис. 6: Ручное выкладывание изменений

Проверил статус синхронизации командой pass git status (рис. 7).

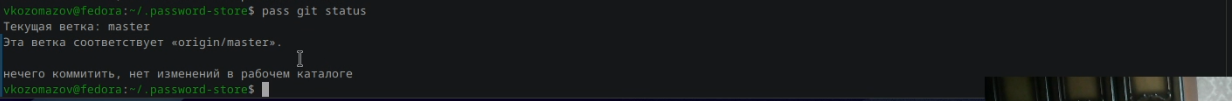


Рис. 7: Проверка статуса синхронизаций

Добавил новый пароль командой pass insert [OPTIONAL DIR]/[FILENAME] (рис. 8).

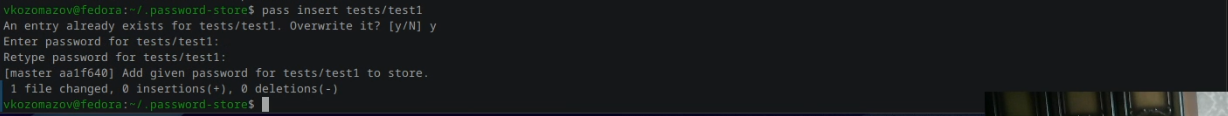


Рис. 8: Добавление нового пароля

Отобразил пароль для указанного имени файла pass [OPTIONAL DIR]/[FILENAME]

Установил дополнительное программное обеспечение (рис. 9).

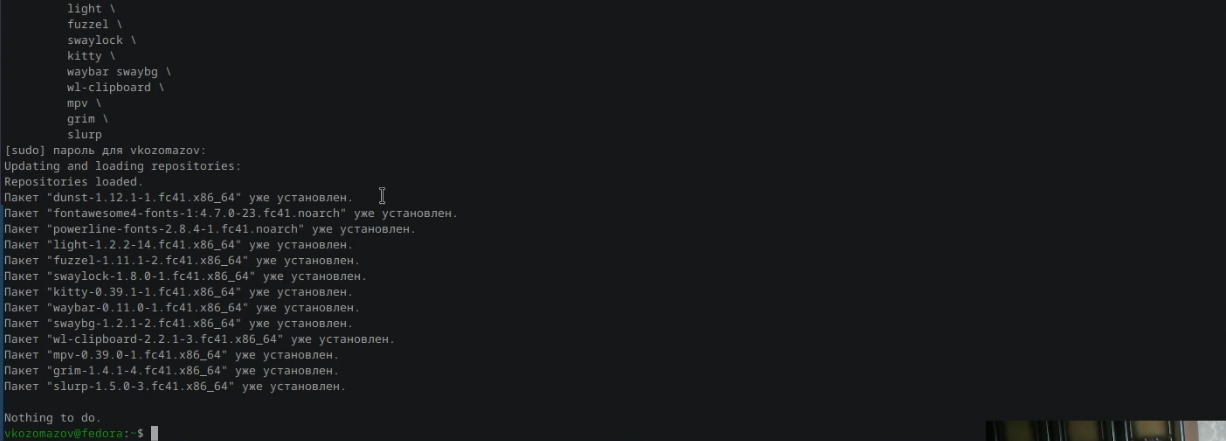


Рис. 9: Установка дополнительного программного обеспечения

Установил дополнительно шрифты командами sudo dnf copr enable peterwu/iosevka, sudo dnf search iosevka, sudo dnf install iosevka-fonts iosevka-aile-fonts iosevka-curly-fonts iosevka-slab-fonts iosevka-etoile-fonts iosevka-term-fonts (рис. 10).



Рис. 10: Установка дополнительных шрифтов

Установил бинарный файл командой sh -c "$(wget -qO- chezmoi.io/get)"

Создал собственный репозиторий при помощи утилит на основе шаблона

Подключил репозиторий к своей системе

Использовал chezmoi на нескольких машинах

Настроил новую машину с помощью одной команды

Включил функцию автоматической фиксации и отправлении изменений в репозиторий

# 5 Выводы

## 5.1 **Итог**

Ветвления и циклы — **фундамент shell-программирования**, который открывает возможности для:  
✅ **Создания сложных скриптов** с минимальными усилиями.  
✅ **Эффективного администрирования** UNIX-систем.  
✅ **Построения автоматизированных рабочих процессов**.

**Освоение этих концепций — важный шаг** к профессиональной работе с командной строкой Linux/UNIX.

# Список литературы