

Программирование в командном процессоре ОС UNIX. Ветвления и ЦИКЛЫ

Лабораторная работа №13

Козомазов Владимир Романович

Содержание

1	Цель работы	5
1.0.1	Конкретные цели работы	5
2	Задание	6
2.0.1	Практическое задание	6
2.1	Цель задания	6
2.2	Задачи	6
2.2.1	1. Базовые упражнения	6
3	Теоретическое введение	8
3.1	1. Командные процессоры UNIX	8
3.2	2. Ветвления в shell-скриптах	8
3.2.1	2.1. Конструкция if-elif-else	9
3.2.2	2.2. Конструкция case	9
3.3	3. Циклы в shell-скриптах	10
3.3.1	3.1. Цикл for	10
3.3.2	3.2. Цикл while	10
3.3.3	3.3. Цикл until	11
3.4	4. Управление выполнением	11
3.5	5. Проверка условий	12
3.6	6. Особенности разных shell	12
4	Выполнение лабораторной работы	14
5	Выводы	17
5.1	Итог	17
	Список литературы	18

Список иллюстраций

4.1	Установка менеджера паролей	14
4.2	Завершение установки менеджера паролей	14
4.3	Просмотр списка ключей	14
4.4	Инициализация хранилища	14
4.5	Создание структуры	15
4.6	Ручное выкладывание изменений	15
4.7	Проверка статуса синхронизаций	15
4.8	Добавление нового пароля	15
4.9	Установка дополнительного программного обеспечения	15
4.10	Установка дополнительных шрифтов	16

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. 4.1).

```
Kozomazov@fedora: ~$ sudo dnf install pass pass-otp
[sudo] пароль для vkozomazov:
Пополнение едб паз.
[sudo] пароль для vkozomazov:
Updating and loading repositories:
Repositories loaded.
Пакет "pass-1.7.4-11.fc41.noarch" уже установлен.
Пакет "pass-otp-1.2.0-15.fc41.noarch" уже установлен.
Nothing to do.
Kozomazov@fedora: ~$
```

Рис. 4.1: Установка менеджера паролей

Завершил установку менеджера паролей командой `sudo dnf install gopass` (рис. 4.2).

```
Kozomazov@fedora: ~$ sudo dnf install gopass
Updating and loading repositories:
Repositories loaded.
Пакет "gopass-1.15.15-2.fc41.x86_64" уже установлен.
Nothing to do.
Kozomazov@fedora: ~$
```

Рис. 4.2: Завершение установки менеджера паролей

Просмотрел список gpg ключей при помощи команды `gpg --list-secret-keys` (рис. 4.3).

```
Kozomazov@fedora: ~$ gpg --list-secret-keys
[keyboard]
-----
sec   Ts44096 2025-03-14 [SC]
      4938B838E2482245D91F62F84FFBD2FEE4272D8C
uid   [ abcomeruo ] Vladimir Kozomazov <voffkoc@gmail.com>
sub   Ts44096 2025-03-14 [E]
Kozomazov@fedora: ~$
```

Рис. 4.3: Просмотр списка ключей

Инициализировал хранилище, написав команду `'pass init` (рис. 4.4).

```
Kozomazov@fedora: ~$ pass init 4938B838E2482245D91F62F84FFBD2FEE4272D8C
Password store initialized for 4938B838E2482245D91F62F84FFBD2FEE4272D8C
```

Рис. 4.4: Инициализация хранилища

Создал структуру с git командой `pass git init` (рис. 4.5).

```
kozomazov@fedora:~$ pass git init
Перинициализирован существующий репозиторий Git в /home/kozomazov/.password-store/.git/
kozomazov@fedora:~$
```

Рис. 4.5: Создание структуры

Синхронизировался с git командами `pass git pull`, `pass git push`
Вручную закоммитил и выложил изменения командами (рис. 4.6).

```
kozomazov@fedora:~$ cd ~/.password-store/
kozomazov@fedora:~/.password-store$ git add .
kozomazov@fedora:~/.password-store$ git commit -am 'edit manually'
[main 6b7c0: 1 commit]
kozomazov@fedora:~/.password-store$ git push
Everything's up-to-date.
kozomazov@fedora:~/.password-store$
```

Рис. 4.6: Ручное выкладывание изменений

Проверил статус синхронизации командой `pass git status` (рис. 4.7).

```
kozomazov@fedora:~/.password-store$ pass git status
Everything's up-to-date.
kozomazov@fedora:~/.password-store$
```

Рис. 4.7: Проверка статуса синхронизаций

Добавил новый пароль командой `pass insert [OPTIONAL DIR]/[FILENAME]` (рис. 4.8).

```
kozomazov@fedora:~/.password-store$ pass insert tests/test1
An entry already exists for tests/test1. Overwrite it? (y/N) y
Enter password for tests/test1:
Retype password for tests/test1:
[main a41f640] Add given password for tests/test1 to store.
1 file changed, 0 insertions(+), 0 deletions(-)
kozomazov@fedora:~/.password-store$
```

Рис. 4.8: Добавление нового пароля

Отобразил пароль для указанного имени файла `pass [OPTIONAL DIR]/[FILENAME]`
Установил дополнительное программное обеспечение (рис. 4.9).

```
light \
fuzzel \
swaylock \
kitty \
waybar \
wl-clipboard \
mpv \
grim \
slurp \
[sudo] пароль для kozomazov:
updating and loading repositories:
repositories loaded.
Пакет "dunst-1.12.1-1.fc41.x86_64" уже установлен.
Пакет "fontawesome-fonts-10.2.3-1.fc41.noarch" уже установлен.
Пакет "powerline-fonts-2.8.4-1.fc41.noarch" уже установлен.
Пакет "light-1.2.2-14.fc41.x86_64" уже установлен.
Пакет "fuzzel-1.11.1-2.fc41.x86_64" уже установлен.
Пакет "swaylock-1.8.0-1.fc41.x86_64" уже установлен.
Пакет "kitty-0.19.1-1.fc41.x86_64" уже установлен.
Пакет "waybar-0.11.0-1.fc41.x86_64" уже установлен.
Пакет "swaybg-1.2.1-2.fc41.x86_64" уже установлен.
Пакет "wl-clipboard-2.2.1-3.fc41.x86_64" уже установлен.
Пакет "mpv-0.39.0-1.fc41.x86_64" уже установлен.
Пакет "grim-1.4.1-4.fc41.x86_64" уже установлен.
Пакет "slurp-1.5.0-3.fc41.x86_64" уже установлен.
Nothing to do.
kozomazov@fedora:~$
```

Рис. 4.9: Установка дополнительного программного обеспечения

```

# Install Joseva fonts
usevka-term-ss0-fonts match: Monospace, Liberation Mono Style
usevka-term-ss7-fonts match: Monospace, Monaco Style
usevka-term-ss8-fonts match: Monospace, Pragmata Pro Style
usevka-term-ss9-fonts match: Monospace, Source Code Pro Style
usevka-term-ss10-fonts match: Monospace, Invisio Code K Style
usevka-term-ss11-fonts match: Monospace, X Windows Fixed Style
usevka-term-ss12-fonts match: Monospace, Ubuntu Mono Style
usevka-term-ss13-fonts match: Monospace, Lucida Style
usevka-term-ss14-fonts match: Monospace, JetBrains Mono Style
usevka-term-ss15-fonts match: Monospace, IBM Plex Mono Style
usevka-term-ss16-fonts match: Monospace, PT Mono Style
usevka-term-ss17-fonts match: Monospace, Recursive Mono Style
usevka-term-ss18-fonts match: Monospace, Input Mono Style
# Install Joseva fonts
sudo apt install usevka-alle-fonts usevka-curlly-fonts usevka-zlab-fonts usevka-etotile-fonts usevka-term-fonts
Updating and loading repositories:
Repositories loaded
Paket "isevka-fonts-33.0.1-1.fc41.noarch" уже установлен.
Paket "isevka-alle-fonts-33.0.1-1.fc41.noarch" уже установлен.
Paket "isevka-curlly-fonts-33.0.1-1.fc41.noarch" уже установлен.
Paket "isevka-zlab-fonts-33.0.1-1.fc41.noarch" уже установлен.
Paket "isevka-etotile-fonts-33.0.1-1.fc41.noarch" уже установлен.
Paket "isevka-term-fonts-33.0.1-1.fc41.noarch" уже установлен.

Nothing to do.
```

тарный файл командой `sh -c "$(wget -qO- che`
енный репозиторий при помощи утилит на осн
епозиторий к своей системе
chezmoi на нескольких машинах
ую машину с помощью одной команды
кцию автоматической фиксации и отправлен

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