

Программирование в командном процессоре ОС UNIX. Командные файлы

Лабораторная работа №12

Козомазов Владимир Романович

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	11
5	Выводы	14
	Список литературы	15

Список иллюстраций

4.1	Установка менеджера паролей	11
4.2	Завершение установки менеджера паролей	11
4.3	Просмотр списка ключей	11
4.4	Инициализация хранилища	11
4.5	Создание структуры	12
4.6	Ручное выкладывание изменений	12
4.7	Проверка статуса синхронизаций	12
4.8	Добавление нового пароля	12
4.9	Установка дополнительного программного обеспечения	12
4.10	Установка дополнительных шрифтов	13

Список таблиц

1 Цель работы

Цель работы “Программирование в командном процессоре ОС UNIX. Командные файлы” заключается в освоении навыков создания и выполнения командных файлов (shell-скриптов) в UNIX-подобных операционных системах.

2 Задание

1. Написать скрипт, который при запуске будет делать резервную копию самого себя в другую директорию backup в моём домашнем каталоге.
2. Написать пример командного файла, обрабатывающего любое произвольное число аргументов командной строки, в том числе превышающее десять.
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.
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.

3 Теоретическое введение

3.0.0.1 1. Командный процессор (shell) в UNIX

Командный процессор (интерпретатор команд, *shell*) — это программа, обеспечивающая взаимодействие пользователя с операционной системой UNIX. Он принимает команды, выполняет их и возвращает результат.

Популярные shell-процессоры:

- **Bash (Bourne-Again SHell)** — стандартный в большинстве Linux-дистрибутивов.
- **sh (Bourne Shell)** — более старый, но совместимый с Bash.
- **zsh, ksh, csh** — альтернативные оболочки с дополнительными возможностями.

3.0.0.2 2. Командные файлы (shell-скрипты)

Командный файл — это текстовый файл, содержащий последовательность команд для выполнения в shell.

Особенности shell-скриптов:

- Исполняются интерпретатором (не требуют компиляции).
- Могут принимать аргументы командной строки.
- Поддерживают переменные, условия, циклы, функции.
- Могут вызывать другие программы и скрипты.

3.0.0.3 3. Основные элементы shell-программирования

3.0.0.3.1 3.1. Структура скрипта

```
#!/bin/bash  # Шелл (указывает интерпретатор)  
# Комментарии начинаются с #
```

```
echo "Hello, World!"  # Простая команда
```

3.0.0.3.2 3.2. Переменные

- Объявление: VAR=value (без пробелов!).
- Использование: \$VAR или \${VAR}.

```
name="User"  
echo "Hello, $name!"  # Hello, User!
```

3.0.0.3.3 3.3. Параметры командной строки

- \$0 — имя скрипта.
- \$1, \$2, ... — аргументы.
- \$# — количество аргументов.
- \$* или \$@ — все аргументы.

3.0.0.3.4 3.4. Управляющие конструкции

- **Условия:**

```
if [ "$1" -eq 10 ]; then  
    echo "Аргумент равен 10"  
else
```



```
    echo "Аргумент не равен 10"
fi
```

- **Циклы:**

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

3.0.0.3.5 3.5. Перенаправление ввода/вывода

- > — вывод в файл (перезапись).
- >> — вывод в файл (дополнение).
- < — ввод из файла.
- | — конвейер (передача вывода одной команды на вход другой).

3.0.0.3.6 3.6. Коды возврата и обработка ошибок

- \$? — код завершения последней команды (0 — успех, иначе ошибка).
- set -e — завершить скрипт при ошибке.
- trap — перехват сигналов.

3.0.0.4 4. Пример скрипта

```
#!/bin/bash
# Скрипт для поиска файлов и вывода информации

if [ $# -eq 0 ]; then
```

```
    echo "Использование: $0 <каталог>"
    exit 1
fi
```

```
dir=$1
echo "Файлы в каталоге $dir:"
find "$dir" -type f -exec ls -l {} \;
```

3.0.0.5 5. Практическое применение

- Автоматизация рутинных задач (резервное копирование, логирование).
- Обработка текстовых данных (логи, CSV).
- Управление системными процессами.

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 vkozmazov:
Пополнение едб паз.
[sudo] пароль для vkozmazov:
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 `grp` ключей при помощи команды `grp --list-secret-keys` (рис. 4.3).

```
Kozomazov@fedora: ~$ grp --list-secret-keys
[keyboard]
-----
sec  Ts4d096 2025-03-14 [SC]
    4938B838E2482245D91F62F84FFBD2FEE4272D8C
uid  [ abcomeruo ] Vladimir Kozomazov <voffkoc@gmail.com>
sub  Ts4d096 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
Перициализирован 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 -m 'edit manually'
[main 8b7c0: 1 file changed, 0 insertions(+), 0 deletions(-)]
komazov@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
Находясь ветка: master
Ветка соответствует «origin/master».
ничего коммитить, нет изменений в рабочем каталоге
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 swaybg \
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 "fontawesome4-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

```
usevka-term-ss80-fonts.monoarch Monospace, Liberation Mono Style  
usevka-term-s87-fonts.monoarch Monospace, Monaco Style  
usevka-term-is98-fonts.monoarch Monospace, Pragmata Pro Style  
usevka-term-us90-fonts.monoarch Monospace, Source Code Pro Style  
usevka-term-u816-fonts.monoarch Monospace, Envy Code K Style  
usevka-term-ts11-fonts.monoarch Monospace, X Windows Fixed Style  
usevka-term-tsl2-fonts.monoarch Monospace, Ubuntu Mono Style  
usevka-term-lt13-fonts.monoarch Monospace, Lucida Style  
usevka-term-ts14-fonts.monoarch Monospace, JetBrains Mono Style  
usevka-term-ts15-fonts.monoarch Monospace, IBM Plex Mono Style  
usevka-term-us16-fonts.monoarch Monospace, PT Mono Style  
usevka-term-ts17-fonts.monoarch Monospace, Recursive Mono Style  
usevka-term-ts18-fonts.monoarch Monospace, Input Mono Style
```

`$ sudo dnf install usevka-aile-fonts usevka-curly-fonts usevka-zlab-fonts usevka-etotille-fonts usevka-term-fonts`

Updating and loading repositories:

Repositories loaded.

```
Famer "isevka-fonts-33.0.1-1.fc41.noarch" уже установлен.  
Famer "isevka-aile-fonts-33.0.1-1.fc41.noarch" уже установлен.  
Famer "isevka-curly-fonts-33.0.1-1.fc41.noarch" уже установлен.  
Famer "isevka-zlab-fonts-33.0.1-1.fc41.noarch" уже установлен.  
Famer "isevka-etotille-fonts-33.0.1-1.fc41.noarch" уже установлен.  
Famer "isevka-term-fonts-33.0.1-1.fc41.noarch" уже установлен.
```

Nothing to do.

5 Выводы

В ходе выполнения работы были изучены ключевые аспекты создания и использования командных файлов (shell-скриптов) в UNIX-подобных системах.

Список литературы