

Программирование в командном процессоре ОС UNIX. Расширенное программирование

Лабораторная работа №14

Козомазов Владимир Романович

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	10
5	Выводы	13
	Список литературы	14

Список иллюстраций

4.1	Установка менеджера паролей	10
4.2	Завершение установки менеджера паролей	10
4.3	Просмотр списка ключей	10
4.4	Инициализация хранилища	10
4.5	Создание структуры	11
4.6	Ручное выкладывание изменений	11
4.7	Проверка статуса синхронизаций	11
4.8	Добавление нового пароля	11
4.9	Установка дополнительного программного обеспечения	11
4.10	Установка дополнительных шрифтов	12

Список таблиц

1 Цель работы

Главная цель данной работы — **углублённое освоение методов и приёмов программирования в командных процессорах UNIX** (Bash, sh, ksh и др.) для создания эффективных, надёжных и безопасных shell-скриптов, способных решать сложные задачи системного администрирования, автоматизации и обработки данных.

2 Задание

1. Написать командный файл, реализующий упрощённый механизм семафоров. Командный файл должен в течение некоторого времени t_1 дожидаться освобождения ресурса, выдавая об этом сообщение, а дождавшись его освобождения, использовать его в течение некоторого времени $t_2 < t_1$, также выдавая информацию о том, что ресурс используется соответствующим командным файлом (процессом)
2. Реализовать команду `man` с помощью командного файла. Изучите содержимое каталога `/usr/share/man/man1`.
3. Используя встроенную переменную `$RANDOM`, напишите командный файл, генерирующий случайную последовательность букв латинского алфавита.

3 Теоретическое введение

3.0.0.1 1. Понятие командного процессора в UNIX

Командный процессор (shell) — это интерпретатор команд, обеспечивающий взаимодействие пользователя с операционной системой. В UNIX-подобных системах распространены: - **Bash** (Bourne-Again SHell) — наиболее популярный, с расширенными возможностями - **sh** (Bourne Shell) — стандартный, но с ограниченным функционалом - **zsh/ksh** — альтернативные оболочки с дополнительными функциями

3.0.0.2 2. Особенности shell-программирования

1. Интерпретируемый характер:

- Скрипты выполняются построчно без предварительной компиляции
- Требуют указания интерпретатора в первой строке (`#!/bin/bash`)

2. Работа с процессами:

- Каждая команда запускает новый процесс
- Механизмы управления: `&`, `jobs`, `fg`, `bg`, `kill`

3. Перенаправление ввода-вывода:

- Основные операторы: `>`, `<`, `>>`, `|`, `2>`
- Особые файлы: `/dev/null`, `/dev/zero`

3.0.0.3 3. Уровни программирования в shell

Уровень	Характеристики	Примеры
Базовый	Простые линейные скрипты	Автоматизация рутинных команд
Продвинутый	Использование условий, циклов, функций	Системный мониторинг, обработка данных
Профессиональный	Интеграция с другими языками, сложная логика	DevOps-инструменты, CI/CD системы

3.0.0.4 4. Ключевые технологии расширенного программирования

1. Регулярные выражения:

- Реализации в `grep`, `sed`, `awk`
- Пример: `grep -E "^[A-Za-z0-9_]+@[a-z]+\.[a-z]{2,4}$" emails.txt`

2. Работа с текстом:

- **awk** — язык обработки строк с возможностью программирования

```
awk '{print $1, $3}' access.log | sort | uniq -c
```

3. Параллельное выполнение:

- Механизмы: `&`, `wait`, `xargs -P`, `GNU parallel`

```
find . -type f -print0 | xargs -0 -P 4 md5sum
```

3.0.0.5 5. Безопасность shell-скриптов

1. Типичные уязвимости:

- Инъекции через непроверенные параметры
- небезопасное использование временных файлов

- Утечка чувствительных данных

2. Методы защиты:

- Валидация ввода: `[["$var" =~ ^[0-9]+$]]`
- Экранирование: `printf "%q" "$unsafe_input"`
- Безопасные временные файлы: `mktemp`

3.0.0.6 6. Современные тенденции

1. Интеграция с DevOps:

- Использование в CI/CD-пайплайнах
- Взаимодействие с Docker/Kubernetes

2. Альтернативные подходы:

- Замена сложных скриптов на Python/Perl
- Использование специализированных языков (e.g., Ansible YAML)

3.0.0.7 7. Исторический контекст

- 1971: Первая версия shell в UNIX (Thompson Shell)
- 1977: Bourne Shell (sh) — основа современного scripting
- 1989: Bash — расширенная версия с возможностями csh/ksh

Теоретическая значимость:

Изучение расширенного программирования в shell позволяет: - Понимать архитектуру UNIX-систем на глубинном уровне - Эффективно решать задачи системного администрирования - Создавать переносимые и надежные решения

Практическая ценность:

Приобретенные навыки применяются в: - Автоматизации серверных задач - Обработке больших объемов данных - Разработке инструментов DevOps

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 `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 -am 'edit manually'
[main 0b7c0: 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.mnarch Monospace, Liberation Mono Style  
usevka-term-s87-fonts.mnarch Monospace, Monaco Style  
usevka-term-ss98-fonts.mnarch Monospace, Pragmata Pro Style  
usevka-term-us89-fonts.mnarch Monospace, Source Code Pro Style  
usevka-term-u816-fonts.mnarch Monospace, Envy Code K Style  
usevka-term-ts11-fonts.mnarch Monospace, X Windows Fixed Style  
usevka-term-tsl2-fonts.mnarch Monospace, Ubuntu Mono Style  
usevka-term-vl13-fonts.mnarch Monospace, Lucida Style  
usevka-term-ts14-fonts.mnarch Monospace, JetBrains Mono Style  
usevka-term-ts15-fonts.mnarch Monospace, IBM Plex Mono Style  
usevka-term-us16-fonts.mnarch Monospace, PT Mono Style  
usevka-term-ts17-fonts.mnarch Monospace, Recursive Mono Style  
usevka-term-ts18-fonts.mnarch Monospace, Input Mono Style
```

```
$ sudo apt install usevka-aile-fonts usevka-curly-fonts usevka-zlab-fonts usevka-etotile-fonts usevka-term-fonts
```

```
Updating and loading repositories:
```

```
Repositories loaded:  
Flower "isevka-konts-33.0.1-l.fc4i.noarch" уже установлен.  
Flower "isevka-aile-fonts-33.0.1-l.fc4i.noarch" уже установлен.  
Flower "isevka-curly-fonts-33.0.1-l.fc4i.noarch" уже установлен.  
Flower "isevka-zlab-fonts-33.0.1-l.fc4i.noarch" уже установлен.  
Flower "isevka-etotile-fonts-33.0.1-l.fc4i.noarch" уже установлен.  
Flower "isevka-term-fonts-33.0.1-l.fc4i.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

В ходе выполнения лабораторной работы №14 были получены не только теоретические знания, но и **практические навыки**, позволяющие эффективно работать в UNIX-средах, сокращая время на рутинные операции за счёт автоматизации. Владение расширенным shell-программированием делает специалиста более универсальным и востребованным в IT-индустрии.

Список литературы