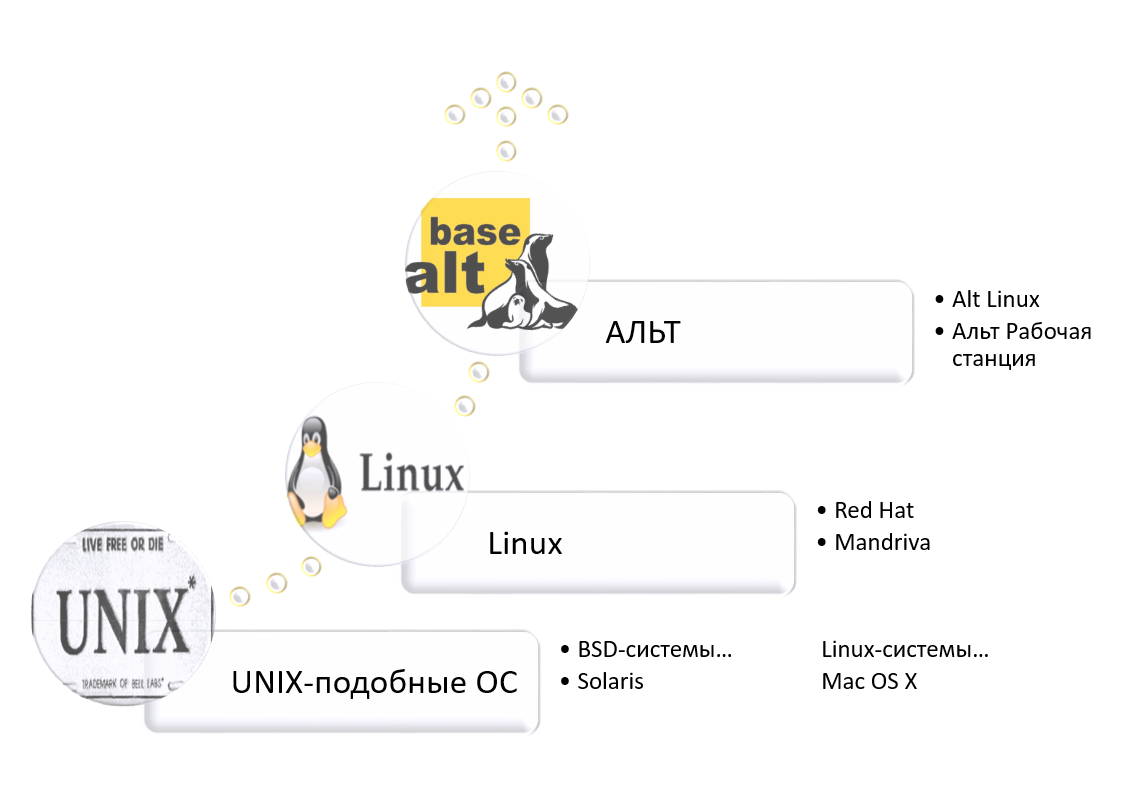


Администрирование ОС Альт часть 1

*конспект по курсу*

МОСКВА 2020

# UNIX-подобные операционные системы



# Особенности UNIX-подобных ОС

1. Переносимость, с платформы на платформу (язык С):
2. Переносимость приложений (стандарты — POSIX, SUS, LSB):
3. Универсальный интерфейс между приложениями (текст):
4. Единый пользовательский интерфейс для интерактивной работы и автоматизации (Shell):
5. Удачная система разграничения привилегий (всё — файл):
6. Многозадачность, многопользовательский доступ:
7. Терминальный и сетевой доступ:

# Интерфейс командной строки

### Стандартный синтаксис POSIX.2

**$ ls -l -rt --color**=always **-I** "\*.mp3" **/tmp /srv**⏎

 Давайте разберем каждый параметр:

1. -l: Этот параметр отображает результат в длинном формате, показывая дополнительную информацию о файлах и директориях.
2. -rt: Флаг -r меняет порядок списка на обратный, а -t сортирует файлы по времени последнего изменения (сначала новые).
3. --color=always: Этот флаг включает цветное форматирование вывода. Он обеспечивает отображение имен файлов разными цветами в зависимости от их типа (например, директории, исполняемые файлы и т. д.).
4. -I "\*.mp3": Флаг -I исключает файлы, соответствующие указанному шаблону (в данном случае файлы с расширением .mp3).

Директории, которые вы перечислили, — это /tmp и /srv. Если вы хотите просмотреть содержимое этих директорий с цветным форматированием, используйте следующие команды:

Для /tmp:

ls -l --color=always /tmp

Для /srv:

ls -l --color=always /srv

# Управление документацией

### Система man (UNIX)

Структура man-страницы

рассмотрим структуру man-страницы:

1. **NAME**: Этот раздел содержит название команды или программы. Например, для команды less, здесь будет указано “less - пейджер для просмотра текстовых файлов”.
2. **SYNOPSIS**: В этом разделе показан синтаксис команды. Он описывает, как правильно использовать команду, какие аргументы и опции можно передавать. Например:
3. less [опции] [файл]
4. **DESCRIPTION**: Здесь предоставляется краткое описание команды. В случае less будет указано, что это пейджер, позволяющий просматривать текстовые файлы постранично.
5. **EXAMPLES**: В этом разделе приводятся примеры использования команды. Например:
6. less myfile.txt
7. **SEE ALSO**: Здесь перечисляются связанные команды или документация. Например, для less это может быть more, другой пейджер.

* Нажмите /, а затем введите ваш шаблон поиска. Шаблоны могут быть регулярными выражениями. Например:
  + /[Oo]ption для поиска слова “option”.
  + /(--)[a-Z] для поиска всех длинных аргументов.
* Для перехода к следующему или предыдущему результату используйте клавиши n и N.
* Секции:
* **Секция 1 (User commands)**: Команды, которые пользователь может выполнять из командной оболочки.
* **Секция 2 (System calls)**: Функции, обертывающие операции, выполняемые ядром операционной системы.
* **Секция 3 (Library calls)**: Все библиотечные функции, исключая обертки системных вызовов (большинство функций из библиотеки libc).
* **Секция 4 (Special files and drivers)**: Описания специальных файлов и драйверов устройств.
* **Секция 5 (File formats and configuration files)**: Различные файловые форматы и конфигурационные файлы.
* **Секция 6 (Games and amusements)**: Игры и развлечения.
* **Секция 7 (Miscellaneous)**: Обзор, соглашения, абстрактные концепции.
* **Секция 8 (System administration commands)**: Команды системного управления (обычно только для администраторов).

### Метасинтаксические конструкции

**setfacl [-bkndRLPvh] [{-m|-x} acl\_spec] \**⏎ **[{-M|-X} acl\_file] file ...**

параметры и их описание:

1. -b: Удаляет все правила ACL для указанного файла.
2. -k: Удаляет правила ACL только для ключа по умолчанию (default ACL).
3. -n: Игнорирует имена пользователей и групп в ACL, оставляя только числовые идентификаторы.
4. -d: Устанавливает правила ACL по умолчанию для указанного файла или директории.
5. -R: Рекурсивно применяет ACL ко всем файлам и поддиректориям.
6. -L: Следует символическим ссылкам при обработке файлов и директорий.
7. -P: Не следует символическим ссылкам при обработке файлов и директорий.
8. -v: Выводит подробную информацию о процессе установки ACL.

Далее параметры, связанные с управлением правилами ACL:

* -m или -x: Добавляет (-m) или удаляет (-x) правило ACL.
* acl\_spec: Определяет правило ACL. Например, user:john:rwx или group:admins:rw-.

И, наконец, указание файлов или директорий, к которым применяются ACL:

* file …: Список файлов или директорий, к которым вы хотите применить ACL.

### Система info (GNU/Linux)

info

* **Переход на ссылку**: Выберите ссылку в тексте и нажмите клавишу Enter, чтобы перейти к связанной информации.
* **Переход по ссылке**: Используйте стрелки вверх/вниз, чтобы перемещаться между разделами документации.
* **Переход обратно**: Нажмите u или Backspace, чтобы вернуться к предыдущему разделу.
* **Поиск**: Нажмите s, чтобы ввести поисковый запрос и найти интересующую вас информацию.
* **Выход**: Для выхода из info нажмите q.
* **Справка**: Нажмите h, чтобы получить справку по доступным командам.

### Встроенная справка приложения

* cut --help
* less (h)
* top (h)
* vim (:help)
* bash (help for)

### Документация из пакета

/usr/share/doc/...

↳ alt-docs/index.html

↳\*/README.ALT

### Навигация по дереву каталогов

Конечно! Давайте рассмотрим команды и их параметры для навигации по дереву каталогов:

1. ls**(List)**:
   * -l: Выводит список файлов и директорий в длинном формате, включая права доступа, владельца, размер и дату модификации.
   * -h: Опция -h делает размеры файлов более читаемыми (например, “1K” вместо “1024”).
   * -a: Показывает скрытые файлы и директории (начинающиеся с точки).
   * -d: Выводит только информацию о текущем каталоге, а не его содержимом.
   * -t: Сортирует файлы по времени последнего изменения (сначала новые).
   * -S: Сортирует файлы по размеру (сначала большие).
   * -i: Выводит индекс (inode) каждого файла.
2. cd**(Change Directory)**:
   * -: Переходит в предыдущий рабочий каталог.
   * ..: Переходит в родительский каталог.
   * ~: Переходит в домашний каталог текущего пользователя.
   * pwd: Выводит полный путь текущего рабочего каталога.
3. mc**(Midnight Commander)**:
   * mcedit: Открывает текстовый редактор Midnight Commander.

### История команд

history Клавиши:

* ↑/↓
* PgUp/PgDown
* Ctrl-R

### Дополнение имён

Tab

* имена команд/функций/псевдонимов
* имена файлов
* имена переменных
* опции/аргументы

### Перенаправление ввода-вывода

**ls -l *AAA BBB* 2> *CCC* > *DDD***

* ls: Команда для списка файлов и директорий.
* -l: Опция для вывода в длинном формате (с дополнительной информацией).
* AAA BBB: Список файлов или директорий.
* 2> CCC: Перенаправление стандартного потока ошибок (stderr) в файл CCC.
* > DDD: Перенаправление стандартного потока вывода (stdout) в файл DDD.

### ls -l | tr "ol" "01" >> *EEE*

* |: Пайп (перенаправление вывода команды слева в команду справа).
* tr: Команда для замены символов.
* "ol" "01": Замена символов “o” на “0” и “l” на “1”.
* >> EEE: Добавление вывода в файл EEE (дописывает в конец файла).

### tr "ol" "01" < *FFF*

* < FFF: Перенаправление ввода из файла FFF.

### 

### Специальные символы

рассмотрим специальные символы и их значения в командах:

1. **Пробел**: Пробел используется для разделения аргументов в командах. Например, ls -l или echo Hello World.
2. **$**: Символ доллара обозначает переменные в командах. Например, $HOME представляет домашний каталог текущего пользователя.
3. **#**: Знак решетки используется для комментариев в скриптах. Все, что идет после #, игнорируется.
4. **"** и **'**: Кавычки используются для обрамления строк. Одинарные кавычки сохраняют текст “как есть”, а двойные кавычки позволяют интерпретировать переменные и специальные символы.
5. \*\*\*\*: Обратная косая черта используется для экранирования специальных символов. Например, echo \$HOME выведет $HOME.
6. \*\*\*\* или \*\*$()\*\*: Обратные кавычки или синтаксис ()‘используютсядлявыполненияподкомандывнутристроки.Например,‘echo"Сегодня(date)"`.
7. **<**: Символ < используется для перенаправления ввода из файла. Например, cat < file.txt.
8. **>**: Символ > используется для перенаправления вывода в файл. Например, ls > list.txt.
9. **|**: Символ вертикальной черты (пайп) используется для передачи вывода одной команды в другую. Например, ls | grep "file".
10. **&**: Символ амперсанда используется для запуска команды в фоновом режиме. Например, command &.
11. **;**: Символ точки с запятой используется для разделения нескольких команд в одной строке. Например, command1; command2.
12. **\***: Звездочка (звездочка) используется как символ подстановки. Например, ls \*.txt.
13. **?**: Вопросительный знак используется для подстановки одного символа. Например, ls file?.txt.
14. **~**: Тильда обозначает домашний каталог текущего пользователя. Например, cd ~/Documents.
15. **[abc]**, **[^abc]**, **[a-cklx-z]**: Квадратные скобки используются для задания диапазона символов или набора символов. Например, [a-z] означает любую букву от a до z.

### Ввод управляющих последовательностей символов

рассмотрим управляющие последовательности символов и их значения:

1. **Ctrl-c (Control-C)**: Прерывает выполнение текущей команды в терминале.
2. **Ctrl-z (Control-Z)**: Приостанавливает выполнение текущей команды и помещает ее в фоновый режим.
3. **Ctrl-\ (Control-\)**: Отправляет сигнал QUIT, который может завершить выполнение текущей команды.
4. **Ctrl-d (Control-D)**: Обозначает конец ввода (EOF) в терминале.
5. **Ctrl-l (Control-L)**: Очищает экран терминала.
6. **Ctrl-\_ (Control-underscore)**: Отменяет последнее действие (аналог Ctrl-z, но без перевода в фоновый режим).
7. **Ctrl-v (Control-V)**: Вводит следующий символ как буквальный (даже если это специальный символ).
8. **Ctrl-a (Control-A)**: Перемещает курсор в начало строки.
9. **Ctrl-e (Control-E)**: Перемещает курсор в конец строки.
10. **Alt-b (Alt-backspace)**: Удаляет слово перед курсором.
11. **Alt-f (Alt-forward)**: Перемещает курсор вперед на одно слово.
12. **Ctrl-w (Control-W)**: Удаляет слово перед курсором.
13. **Ctrl-y (Control-Y)**: Вставляет удаленный текст (отменяет Ctrl-w или Ctrl-u).
14. **Ctrl-k (Control-K)**: Удаляет текст после курсора до конца строки.
15. **Ctrl-u (Control-U)**: Удаляет текст перед курсором до начала строки.
16. **Ctrl-s (Control-S)**: Останавливает вывод на экран (пауза).
17. **Ctrl-q (Control-Q)**: Возобновляет вывод на экран после Ctrl-s (противоположность паузы).

**reset**: Команда, которая сбрасывает настройки терминала к значениям по умолчанию. **stty**: Команда для настройки параметров терминала (например, режима работы клавиш).

# Обработка текста

### Редактирование текста

vim emacs sed awk

nano pico pluma joe mcedit

ed

*$*EDITOR

*$*VISUAL

### Просмотр текста

cat

less more

view grep cut

*$PAGER*

# Создание скриптов

### Базовые понятия

1. **Скрипт (сценарий)**: Это файл, содержащий последовательность команд, которые выполняются в определенном порядке. Скрипты используются для автоматизации задач.
2. **Командный интерпретатор**: Программа, которая выполняет команды из скрипта. Примеры: bash, sh, Python, Perl.
3. **Текстовый редактор**: Инструмент для создания и редактирования текстовых файлов. Примеры: vim, nano, emacs.
4. **Комментарии**: Специальные строки в скрипте, которые не выполняются, но помогают описать код. Обычно начинаются с символов #.
5. **Шебанг (shebang)**: Строка в начале скрипта, указывающая, какой интерпретатор следует использовать. Например: #!/bin/bash.
6. **Пробельный символ**: Пробелы, табуляции и переводы строки. Они влияют на форматирование и читаемость скрипта.
7. **Разделитель команд**: Обычно точка с запятой (;) или новая строка. Разделяет команды в скрипте.
8. **Файл скрипта**: Текстовый файл, содержащий команды для выполнения.
9. **Запуск на выполнение**: Для запуска скрипта используйте команду ./script.sh (если у файла есть права на выполнение).

# Дерево каталогов Linux

/

├─ home

│  каталоги пользователей

│

├─ root

│ каталог суперпользователя (root)

│

├─ etc

│ Конфигурационные файлы системы

│

├─ bin

│ Бинарные исполняемые файлы (основные команды)

├─ sbin

│ Системные бинарные файлы (команды для администрирования)

├─ usr Системные ресурсы (программы, библиотеки, документация)

│ ├─ bin

│ ├─ sbin

│ ├─ share

│

│

├─ opt

│

│

├─ lib  Виртуальная файловая система, предоставляющая информацию о запущенных процессах и системе.

│

├─ lib64  Виртуальная файловая система, предоставляющая информацию о запущенных процессах и системе.

│

├─ boot

│

│

├─ mnt Точки монтирования для съемных устройств и временных монтирований.

│

├─ media Точки монтирования для съемных устройств и временных монтирований.

│

|

├─ var Изменяемые данные, такие как логи, временные файлы и кэш.

|  |  |  |
| --- | --- | --- |
| │ | ├─ | log |
| │ | ├─ | cache |
| │ | ├─ | spool |
| │ | ├─ | lib |
| │ |  |  |
| │ |  |  |

├─ run

│ Временные файлы, создаваемые во время работы системы.

│

│

├─ tmp

│ Временное пространство, обычно очищается при перезагрузке.

│

│

├─ srv

│ Данные для служб, например, веб-серверов.

│

├─ dev

│ Файлы устройств, представляющие оборудование и драйверы.

│

├─ sys

│

│

│

├─ proc  Виртуальная файловая система, предоставляющая информацию о запущенных процессах и системе.

│ ├─

│ ├─

│ ├─ sys

│

│

│

├─ lost+found Место для восстановления поврежденных файлов.

│

└─ selinux

Политики безопасности SELinux.

# Файловые системы

### Типы файловых системам

1. **Ext2 (Second Extended File System)**: Это файловая система, используемая в Linux. Она предоставляет хорошую производительность, но не поддерживает журналирование, что может привести к потере данных при сбоях.
2. **Ext3 (Third Extended File System)**: Расширение Ext2 с добавлением журналирования. Это обеспечивает более надежное восстановление после сбоев.
3. **Ext4 (Fourth Extended File System)**: Еще более совершенная версия Ext3 с улучшенной производительностью, поддержкой больших файлов и журналированием.
4. **JFS (IBM’s Journaled File System)**: Используется в AIX (Unix-подобной операционной системе). Обеспечивает высокую производительность и надежность.
5. **ReiserFS**:
   1. Разработана для оптимизации обработки множества маленьких файлов.
   2. Имеет высокую производительность при работе с небольшими файлами.
   3. Однако она стала менее популярной из-за проблем с безопасностью и поддержкой.
6. **XFS**:
   1. Используется в Linux.
   2. Обладает высокой производительностью при работе с большими файлами и объемами данных.
7. **Btrfs (B-tree File System)**: Экспериментальная файловая система, разрабатываемая для Linux. Она поддерживает снимки, сжатие, проверку целостности и другие продвинутые функции.
8. **ZFS (Zettabyte File System)**: Создана компанией Sun Microsystems. Она обеспечивает высокую надежность, снимки, сжатие, дедупликацию и другие возможности. Часто используется в серверных окружениях.

### Монтирование файловых систем

**mount -t ext4 -o relatime /dev/sda2 /home**

1. mount: Эта команда используется для монтирования файловых систем в Linux. Она принимает различные параметры и ключи.
2. -t ext4: Этот ключ указывает тип файловой системы. В данном случае, мы монтируем раздел с файловой системой Ext4.
3. -o relatime: Этот ключ устанавливает параметр “relatime” для монтирования. Он обновляет время последнего доступа к файлам только при чтении, что уменьшает нагрузку на файловую систему.
4. /dev/sda2: Это устройство (раздел), которое мы хотим примонтировать.
5. /home: Это точка монтирования (mount point) — каталог, через который обеспечивается доступ к файловой системе.
6. /etc/fstab: Этот файл содержит информацию о монтировании разделов при загрузке системы.
7. 0 0: Это параметры для файла /etc/fstab. Первое число (0) указывает на порядок проверки файловой системы при загрузке, а второе число (0) — на необходимость создания резервной копии.
8. /etc/mtab: Этот файл содержит информацию о текущих монтированиях.

### Управление файловыми системами

краткое описание некоторых команд и ключей для управления файловыми системами в Linux:

1. mount: Команда для монтирования файловой системы. Пример: mount -t ext4 /dev/sda1 /mnt/data.
2. umount: Команда для размонтирования файловой системы. Пример: umount /mnt/data.
3. mke2fs**(или**mkfs.ext2**,**mkfs.ext3**,**mkfs.ext4**)**: Создание файловой системы Ext2, Ext3 или Ext4 на устройстве. Пример: mke2fs -t ext4 /dev/sdb1.
4. e2fsck**(или**fsck.ext2**,**fsck.ext3**,**fsck.ext4**)**: Проверка и восстановление целостности файловой системы Ext2, Ext3 или Ext4. Пример: e2fsck -f /dev/sda2.
5. tune2fs**(или**tunefs.ext2**,**tunefs.ext3**,**tunefs.ext4**)**: Изменение параметров файловой системы. Пример: tune2fs -L "MyDisk" /dev/sdb1.
6. resize2fs: Изменение размера файловой системы Ext2, Ext3 или Ext4. Пример: resize2fs /dev/sda3.
7. df: Отображение информации о свободном месте на диске. Пример: df -h.
8. du: Отображение размера каталогов и файлов. Пример: du -sh /var.
9. truncate: Усечение файла до указанного размера. Пример: truncate -s 1G /my\_large\_file.
10. swapon**и**swapoff: Включение и отключение области подкачки (swap). Пример: swapon /dev/sdb2.
11. dd: Копирование данных между файлами или устройствами. Пример: dd if=/dev/zero of=/swap\_file bs=1024 count=1024.
12. free: Отображение информации о использовании памяти и области подкачки. Пример: free -h.

# Типы файлов

-rwxr-xr-x 1 root root 122760 дек 3 2015 /bin/gzip

1. **Обычные файлы (Regular Files)**: Обычные файлы содержат программы, тексты или данные. Они используются для хранения информации, такой как текст или изображения. Примеры: текстовые файлы, файлы программ, бинарные файлы, изображения (JPG, PNG), архивы (ZIP, RAR) и др.

lrwxrwxrwx 1 root root 4 июл 30 2016 /bin/gunzip -> gzip

**Символические ссылки (Symbolic Link Files)**: Символические ссылки указывают на другие файлы или каталоги. Например, /bin/gunzip является символической ссылкой на /bin/gzip

drwxr-xr-x 2 root root 69632 май 8 15:43 /usr/bin

1. **Файлы каталогов (Directory Files)**: Файлы каталогов предназначены для хранения других обычных файлов, файлов каталогов и специальных файлов, а также связанной с ними информации.

crw-rw-rw- 1 root root 1, 3 май 2 14:26 /dev/null

1. **Символьные файлы (Character Files)**: Символьные файлы представляют символьные устройства, такие как порты и терминалы. Они используются для ввода-вывода символами.

brw-rw---- 1 root disk 8, 0 май 2 14:26 /dev/sda

1. **Блочные файлы (Block Files)**: Блочные файлы представляют блочные устройства, такие как жесткие диски. Они используются для ввода-вывода блоками данных.

prw------- 1 root root 0 май 2 14:26 /dev/initctl

1. **Файлы именованных каналов (Named Pipe Files)**: Именованные каналы обеспечивают межпроцессное взаимодействие. Они используются для передачи данных между процессами.

srw-rw-rw- 1 root root 0 май 7 04:02 /dev/log

1. **Файлы сокетов (Socket Files)**: Сокеты используются для локального обмена данными между процессами.

# Атрибуты файлов

1. **Индексный дескриптор (Inode)**: Это уникальный идентификатор для каждого файла или каталога в файловой системе. Он содержит метаданные о файле, такие как права доступа, владелец, размер и временные метки.
2. **Номер (Inode Number)**: Уникальный числовой идентификатор индексного дескриптора.
3. **Тип файла (File Type)**: Определяет, является ли файл обычным файлом, каталогом, символической ссылкой и т. д.
4. **Размер файла (File Size)**: Размер файла в байтах.
5. **Владелец (Owner)**: Пользователь, который создал файл или имеет права на него.
6. **Группа (Group)**: Группа пользователей, которой принадлежит файл.
7. **Режим доступа (Permissions)**: Права доступа к файлу (чтение, запись, выполнение) для владельца, группы и остальных пользователей.
8. **Количество ссылок (Link Count)**: Количество жестких ссылок на данный файл.
9. **atime (Access Time)**: Время последнего доступа к файлу.
10. **ctime (Change Time)**: Время последнего изменения метаданных файла (например, прав доступа).
11. **mtime (Modification Time)**: Время последнего изменения содержимого файла.
12. **Указатели на блоки данных (Data Block Pointers)**: Указывают на фактические данные файла.
13. **Жёсткие ссылки (Hard Links)**: Жесткие ссылки позволяют создавать несколько имен для одного и того же индексного дескриптора.
14. ln: Команда для создания жестких ссылок.
15. .**и**..: Специальные каталоги в каждом каталоге. . ссылается на текущий каталог, а .. — на родительский каталог.

# Управление файлами

Давайте рассмотрим некоторые команды в Linux и их параметры:

1. touch: Создает новый файл или обновляет его временную метку. Пример: touch file1 file2.
2. cat: Конкатенирует файлы и выводит их содержимое в стандартный поток вывода (stdout). Пример: cat file1.
3. cp: Копирует файлы. Пример: cp file1 file2.
4. mv: Перемещает файлы или переименовывает их. Пример: mv file1 file2.
5. rm: Удаляет файлы и каталоги. Пример: rm file1.
6. mkdir: Создает новый каталог. Пример: mkdir dir1.
7. rmdir: Удаляет пустой каталог. Пример: rmdir dir1.
8. stat: Выводит информацию о файле, такую как права доступа, размер и временные метки.
9. ln: Создает жесткие ссылки на файлы.
10. mknod: Создает специальные файлы (например, устройства).
11. mkfifo: Создает именованные каналы (FIFO).
12. rsync: Синхронизирует файлы и директории между системами.

### Переменные окружения

1. $PATH: Эта переменная содержит список директорий, в которых система ищет исполняемые файлы (команды). Когда вы вводите команду в терминале, система проверяет каждую директорию в $PATH, чтобы найти соответствующий исполняемый файл.
2. $HOME: Эта переменная указывает на домашний каталог текущего пользователя. Например, /home/username.
3. $USER: Содержит имя текущего пользователя.
4. $PS1: Эта переменная определяет строку приглашения (prompt) в терминале. Она может содержать информацию о текущем пути, имени пользователя и другие детали.
5. $LANG: Устанавливает языковую локаль системы. Определяет, на каком языке будут отображаться системные сообщения и форматирование даты, времени и чисел.
6. $LC\_\*: Эти переменные управляют локализацией (языком и региональными настройками) для конкретных аспектов системы, таких как валюта, числа и даты.
7. $SHELL: Содержит путь к текущему оболочке (shell), которую вы используете (например, /bin/bash или /usr/bin/zsh).

***Пример настройки истории команд***

*HISTFILE=~/.bash\_history HISTSIZE=999 HISTFILESIZE=9999*

*HISTCONTROL=ignoreboth*

*PROMPT\_COMMAND='history -a; history -n'*

### Подстановки Shell

**echo $?**

`seq 1 3`

$(seq 1 3)

$[2+2\*2]

$((2+2\*2))

# Создание скриптов (продолжение)

### Использование переменных

имя

значение $имя присваивание =

автоматически создаваемые переменные:

$?

read

$!

$$

$0

$#

От $1 до $9

$\*

### Условный оператор

if [[*выражение*]]; then *команды;* fi if [[*выражение*]]

then *команды*

else *команды*

fi

[ выражение ] [[ выражение ]]

(( арифметическая\_операция ))

### Команда test

test

-f file

INT1 -eq INT2

|  |  |  |
| --- | --- | --- |
| STRING1 ==  -z STRING | STRING2 |  |
| **Сравнение чисел** |  |
| *-eq* |  | *=* |
| *-ge* |  | *>=* |
| *-ne* |  | *!=* |
| *-gt* |  | *>* |
| *-lt* |  | *<* |
| *-le* |  | *<=* |
| **Проверка файлов**  *-f* |  | |
| *-d* |
| *-r* |
| *-w* |
| *-x* |
| *-O* |
| *-G* |

Оператор **case**

case *переменная* in

*значение 1.1*|*значение 1.2*)

*команды 1*;;

значение 2)

*команды 2*;;

значение 3)

*команды 3;;*

esac

*шаблон \*)*

Цикл **for**

for *переменная* in *список*; do *команда*; done

for *переменная* in *список*

do

*команда*

done

цикл **for со счетчиком**

for ((i=1; i<=N ; i++)) do

*команда*

done

Цикл **while**

counter=0

while [[ $counter -le N ]] do

echo $counter (( counter++ ))

done

# Базовая модель разграничения доступа

brw-rw---- 1 root disk 8, 0 май 2 14:26 /dev/sda

### brw-rw---- … root disk … /dev/sda

* brw-rw----: Это права доступа к файлу. b указывает на блочное устройство. rw означает, что владелец и группа имеют права на чтение и запись, а остальные пользователи не имеют доступа.
* 1 root disk: Это количество жестких ссылок на файл, владелец (root) и группа (disk).
* 8, 0: Это мажорный и минорный номера устройства.
* май 2 14:26: Это дата последнего доступа к файлу.

### Access: (0660/brw-rw )

* Access: (0660/brw-rw): Это права доступа в восьмеричной и символьной нотации.

### Управление доступом

   
chmod: Команда chmod используется для изменения прав доступа к файлам и каталогам. Она позволяет устанавливать, изменять или удалять разрешения на чтение, запись и выполнение. Примеры использования:

* chmod u=rwx,g=rx,o=rx filename: Устанавливает права на чтение, запись и выполнение для владельца (u), на чтение и выполнение для группы (g) и для остальных пользователей (o).
* chmod 755 filename: Устанавливает права rwxr-xr-x для файла.

 chown: Команда chown используется для изменения владельца файла или каталога. Пример: chown user:group filename.

 chgrp: Команда chgrp меняет группу владельца файла или каталога. Пример: chgrp newgroup filename.

 umask: Эта команда устанавливает маску создания файлов и каталогов. Маска определяет, какие права доступа будут автоматически отключены при создании новых файлов. Пример: umask 022.

### Списки управления доступом

**-rw-rw-r--+ … dd dd … file**

 рассмотрим команды getfacl и setfacl, а также их параметры и ключи:

1. getfacl:
   * **Описание**: Команда getfacl используется для просмотра списка контроля доступа (ACL) для файла или каталога.
   * **Синтаксис**: getfacl filename
   * **Пример**: getfacl file
   * **Результат**:
   * # file: file
   * # owner: dd
   * # group: dd
   * user::rw-
   * group::r--
   * group:radio:rw-
   * mask::rw-
   * other::r--
2. setfacl:
   * **Описание**: Команда setfacl используется для установки или изменения списка контроля доступа (ACL) для файла или каталога.
   * **Параметры**:
     + -m: Устанавливает ACL. За ним следует спецификация ACL.
     + -R: Рекурсивно применяет ACL ко всем файлам и подкаталогам.
     + -d: Устанавливает ACL по умолчанию для каталога.
   * **Примеры**:
     + Установка прав доступа для группы radio: setfacl -m g:radio:rw file
     + Рекурсивное применение ACL ко всем файлам и подкаталогам: setfacl -R -m u:user1:rw /path/to/dir
     + Установка ACL по умолчанию для каталога: setfacl -d -m u:user1:rw /path/to/dir

### Расширенные атрибуты файлов

**-rws--x--- … root wheel … /usr/bin/sudo**

* s в правах доступа (например, -rws--x---) означает установку бита SUID (Set User ID). Это позволяет выполнять файл с правами владельца (root), даже если запускающий файл пользователь не является владельцем.

**-rwx--s--x … root shadow … /usr/bin/passwd**

* + s также может быть установлен для группы (SGID) или для всех пользователей (sticky bit) в зависимости от контекста.

**drwxrwxrwt … root root … /tmp**

* drwxrwxrwt для каталога /tmp означает, что установлен бит SUID (sticky bit). Это позволяет пользователям создавать файлы только в своих собственных подкаталогах.

**drwxrws--- … root dept06 … /srv/codename**

drwxrws--- для каталога /srv/codename означает, что установлен бит SGID. Это позволяет новым файлам и каталогам внутри этого каталога наследовать группу владельца.

1. chmod:
   * Команда chmod используется для изменения прав доступа к файлам и каталогам.
   * g+s устанавливает бит SGID для группы. Например: chmod g+s filename.
2. lsattr и chattr:
   * lsattr отображает атрибуты файлов.
   * chattr используется для изменения атрибутов файлов. Например: chattr +i filename устанавливает атрибут “только для чтения”.
   * +i: Устанавливает атрибут “только для чтения” для файла. Это предотвращает его изменение или удаление.

# Учётные записи

### Пользовательские учётные записи

1. **Суперпользователь (root)**:
   * Суперпользователь (root) имеет абсолютные привилегии на системе. Он может выполнять любые команды, изменять системные настройки, редактировать файлы и устанавливать программное обеспечение без ограничений.
   * В файле /etc/passwd запись для суперпользователя выглядит так:
   * root:x:0:0:System Administrator:/root:/bin/bash
     + root: Имя пользователя.
     + x: Зашифрованный пароль (фактический пароль хранится в файле /etc/shadow).
     + 0: UID (User ID) суперпользователя.
     + 0: GID (Group ID) суперпользователя.
     + System Administrator: Комментарий (информация о пользователе).
     + /root: Домашний каталог суперпользователя.
     + /bin/bash: Путь к оболочке по умолчанию для суперпользователя.
2. **Операторские учётные записи**:
   * Операторские учётные записи могут иметь некоторые привилегии, но не такие широкие, как у суперпользователя.
   * В файле /etc/passwd обычно есть другие пользователи с разными UID и GID.
3. **Служебные учётные записи**:
   * Служебные учётные записи используются для запуска служб, демонов и приложений.
   * Они могут иметь ограниченные привилегии и обычно не имеют доступа к интерактивной оболочке.

Файл /etc/shadow в Linux хранит зашифрованные пароли пользователей и другую связанную с ними информацию. Вот разъяснение:

* /etc/shadow — это текстовый файл, который содержит следующие данные:
  + **Имя пользователя (Username)**: Например, root или bin.
  + **Зашифрованный пароль (Password)**: Вместо фактического пароля в этом файле хранится его зашифрованная версия.
  + **UID (User ID)**: Уникальный идентификатор пользователя.
  + **GID (Group ID)**: Идентификатор группы, к которой принадлежит пользователь.
  + **Другие параметры**: Время последнего изменения пароля, срок действия пароля и другие связанные с безопасностью значения.
* В приведенном примере:
  + root:x:15829:::::: — это запись для пользователя root.
  + bin:\*:15829:::::: — это запись для пользователя bin.

Файл /etc/shadow доступен только для чтения пользователем root, чтобы предотвратить несанкционированный доступ и обеспечить безопасность системы.

1. **Хранилище паролей TCB**:
   * /etc/tcb — это директория, связанная с Trusted Computing Base (TCB), которая обеспечивает безопасность операционной системы.
   * drwx--s--- — это права доступа к директории. Они означают, что только владелец (dd) имеет доступ на чтение и запись, а другие пользователи не имеют доступа.
   * Внутри директории есть файлы auth и shadow.
2. **Групповые учётные записи**:
   * Файл /etc/group содержит информацию о локально настроенных группах на Linux-системе.
   * Пример записи: wheel:x:10:root,admin
     + wheel — имя группы.
     + x — зашифрованный пароль (обычно не используется).
     + 10 — уникальный идентификатор группы (GID).
     + root,admin — пользователи, входящие в эту группу.

/etc/gshadow

*wheel:x::root,admin*

Файл /etc/gshadow в Linux содержит информацию о группах, аналогично файлу /etc/group. Вот разъяснение:

* /etc/gshadow — это файл, который хранит зашифрованные пароли для групп, а также другую связанную с ними информацию.
* Каждая строка в файле /etc/gshadow содержит следующие поля, разделенные двоеточием:
  + **Имя группы (Group Name)**: Например, wheel.
  + **Зашифрованный пароль (Encrypted Password)**: Вместо фактического пароля в этом файле хранится его зашифрованная версия.
  + **Администраторы группы (Group Administrators)**: Пользователи, которые могут добавлять или удалять членов группы с помощью команды gpasswd.
  + **Члены группы (Group Members)**: Пользователи, являющиеся обычными членами группы.

Файл /etc/gshadow доступен только для чтения пользователем root, чтобы обеспечить безопасность системы.

### Управление пользователями

1. **Управление пользователями**:
   * useradd: Создает нового пользователя.
   * userdel: Удаляет пользователя.
   * usermod: Модифицирует параметры существующего пользователя.
   * adduser: Интерактивно добавляет нового пользователя.
   * vipw: Редактирует файл /etc/passwd.
   * passwd: Устанавливает или изменяет пароль пользователя.
   * chage: Управляет параметрами срока действия пароля.
   * chfn: Меняет информацию о пользователе (например, полное имя).
   * chsh: Меняет оболочку пользователя.
2. **Управление группами**:
   * groupadd: Создает новую группу.
   * groupmod: Модифицирует параметры существующей группы.
   * groupdel: Удаляет группу.
   * vigr: Редактирует файл /etc/group.
   * gpasswd: Управляет паролем группы.
3. su**и**sudo:
   * su: Позволяет переключиться на другого пользователя (обычно на root).
   * sudo: Позволяет запускать программы от имени другого пользователя (по умолчанию root).

# Аутентификация и преобразование имён

### Подключаемые модули аутентификации

/etc/pam.d/...

* auth
* account
* passwd
* session
* required
* requisite
* sufficient
* optional
* include/substack

### Коммутатор системы преобразования имён

/etc/nsswitch.conf

**passwd: files**

**shadow: tcb files**

**group: files**

**gshadow: files**

**hosts: files dns**

*getent*

### Архивация файлов

tar gzip bzip2 xz

### Поиск файлов

find

-type f

-size +1M

-mtime -1

-mmin -1

-name '\*.jpg'

-iname '\*.jpg'

-perm -g=s

-ls

-delete

-exec grep text {} +

which locate

# Управление конфигурацией

### Конфигурация (контекст)

### общесистемная (каталог /etc)

### пользовательская (домашний каталог)

### Конфигурация (уровни)

текущая

сохранённая

умолчальная

### Основные форматы конфигурационных файлов UNIX

root:x:0:0:System Administrator:/root:/bin/bash

devpts /dev/pts devpts noexec,gid=tty,mode=620 0 0

Section "Device"

Identifier "Card0"

Driver "nvidia" EndSection

<VirtualHost \*:80>

ServerAdmin [webmaster@example.com](mailto:webmaster@example.com) DocumentRoot ["/var/www/vhosts/www.example.com"](http://www.example.com/) ServerName [www.example.com](http://www.example.com/)

</VirtualHost>

[homes]

|  |  |  |
| --- | --- | --- |
| comment | = | Home Directories |
| browseable | = | no |
| writable | = | yes |

HISTFILESIZE=9999

PS1='[\u@\h \W]\$ '

### Пользовательское окружение

**$ LANG=C ls -l**

echo set

env export

locale

ru\_RU.utf8

### Пользовательский профиль

id

/etc/

* profile, profile.d/...
* bashrc, bashrc.d/...

~/

↳ .profile

↳ .bashrc

↳ .bash\_profile

# Процессы в Linux

### Состояния процесса:

R S T D Z I

< N

s l

+

ps

a x u w *f*

### Атрибуты процесса:

PID PPID EUID RUID EGID RGID GROUPS PGID TPGID TTY NICE

pstree ps -o

pid euser egroup state *command*

### Сигналы:

SIGHUP SIGINT SIGQUIT SIGKILL SIGTERM SIGUSR1 SIGUSR2 SIGCHLD SIGCONT SIGSTOP SIGTSTP

kill killall pkill pgrep top *htop*

### Приоритет процесса

nice renice

### Управление заданиями Shell

bg fg

%1

+

-

jobs

***tar zcf /backup/dir.tgz /srv/dir 2>/dev/null &***

## Фильтры

cat tac head tail grep

-i

-e

-v

-A1

-B2

-Cn

-r

cut tr wc sort uniq comm

# Удалённый доступ

### Диагностика настроек сетевой подсистемы

/etc/net *ifconfig* route

ip

l a r n

netstat

-l

-a

-n

-t

-u

-x

-p ss getent

/etc/resolv.conf

*nslookup, host, dig*

### OpenSSH

ssh

user@host host command

-D

-f

-N

-L

-R

-X

-Y

scp

-p rsync

local/dir host:remote local/dir/ host:remote/dir

~/.ssh

known\_hosts config

*/etc/openssh/ssh\_host\*key*

### Аутентификация по ключам

ssh-keygen

-t

-b

-f

-l

-p

~/.ssh

authorized\_keys

ssh-copy-id ssh-agent

### Полезные утилиты

ping

*iftop dhcpcd*

# Графическая подсистема Linux

## Общая информация

* **X Window System (X, X Window)**
* **X Window - сетевая служба**
* **X-сервер**
* **X-клиенты**
* **X-протокол**

## сервер

$ ls -l `which X`

lrwxrwxrwx 1 root root 4 ноя 16 01:21 /usr/bin/X -> Xorg

* + **Xorg**
  + **Xephyr/Xnest**

$ pgrep -a X

1583 X :0 -nolisten tcp vt1 . . .

* + **-nolisten tcp**
* **vtX**
* **:0**

## Запуск вложенного X-сервера

$ Xephyr -resizeable :1 &

$ xterm & xcalc & xeyes &

$ DISPLAY=:1 xcalc

$ export DISPLAY=:1

$ xterm & xeyes &

## Оконные менеджеры (Window Managers)

* **Оконные менеджеры**
* **twm / olwm / mwm**

$ LANG=C twm &

$ xterm & xcalc & xeyes &

$ pkill twm

* **icewm**

$ icewm-session &

$ pkill icewm-session

## Менеджеры входа (дисплейные менеджеры)

* **GDM**
* **Sddm**
* **XDM**
* **LightDM**

## Настольные пользовательские окружения (графические среды)

* **GNOME**
* **KDE**
* **MATE**
* **XFCE**
* **LXDE**
* **. . .**

## Настроечные файлы X-сервера и его запуск

$ cat /etc/X11/xorg.conf

$ ls /etc/X11/xorg.conf.d/

* **xinit**
* **Display Manager**

## Подключение к X-серверу по сети (MATE, LightDM)

* Переключение X-сервера в сетевой режим

$ ps xf | grep -A 1 lightdm

$ mcedit /etc/lightdm/lightdm.conf

...

xserver-allow-tcp=true

...

$ telinit 3

$ telini 5

$ ss -atnp | grep X

## Разграничение доступа на X-сервере

* **xhost [+|-[NAME]]**

$ xhost +192.168.50.103

192.168.50.103 being added to access control list

$ xhost

access control enabled, only authorized clients can connect INET:192.168.50.103

$ DISPLAY=192.168.50.101:0 xeyes

$ ps

## Аутентификация средствами xauth

* **MIT-MAGIC-COOKIE**

## Управление X-сеансами средствами SSH

$ ssh -Y user@host [user@host]$ echo $DISPLAY localhost:10

[user@host]$ xeyes &

* **ssh -X**
* **ssh -Y**

*дополнительная тема*

# Использование потокового редактора SED

### Синтаксис командной строки

$ sed OPTIONS 'SCRIPT' [INPUTFILE] [> OUTFILE]

SCRIPT – команды+регулярные выражения+модификаторы INPUTFILE, OUTFILE – текстовые файлы

или

$ COMMAND | sed OPTIONS 'SCRIPT'

COMMAND — команда или скрипт shell (bash) с выводом в стандартный вывод.

### Ключи (опции) sed

-n

-i

-e

-f

-l

-r

-s

-u

-z

--follow-symlinks

--help

--posix

--sandbox

--version

### Часто используемые (основные) команды sed

s q d p n

### дополнительные команды sed

=

a c D

h H

g G

i l N

p w x

### Модификаторы замены

/(число), например /3

/g

/e

/i (/I)

/p

/w

### Преобразование регистра

\L

\l

\u

\U

\E

# Лабораторная работа 1

1. Результат выполнения лабораторных работ — это **три** сценария на языке командного интерпретатора.:

**первый** (основной) выполняет задание;

**второй** (тестовый) проверяет правильность работы основного;

**третий** (обнуляющий) возвращает систему в исходное состояние так, чтобы можно было запустить процесс выполнения сценариев сначала.

1. Задача **основного** сценария первой лабораторной работы: создать файл **my\_file**, записать в него с помощью соответствующих команд:

*системную дату,*

*текущий рабочий каталог, имя пользователя,*

*название и версию ядра операционной системы.*

1. **Тестовый** сценарий должен проверить наличие файла **my\_file** и вывести его содержимое.
2. **Третий** сценарий удаляет файл **my\_file** или сообщает об его отсутствии.
3. Для начала работы запустите текстовый редактор **nano**. Наберите шебанг и первые комментарии с описанием сценария:

**#!/bin/sh**

**# сценарий записи системной информации в файл # лабораторная работа № 1**

Дальше следуют команды создания файла **my\_file** и записи данных в него:

**touch my\_file**

**date > my\_file pwd >> my\_file who >> my\_file**

**uname -sr >> my\_file**

1. Сохраните файл, нажав **^+X** (Ctrl+X).

Имя файла первого скрипта первой работы должно быть **base1.sh**

1. Для того, чтобы можно было запустить скрипт, необходимо сделать файл исполняемым (наберите в командной строке, после выхода из редактора): **chmod +x ./base1.sh**
2. Попытайтесь выполнить сценарий:

**./base1.sh**

1. Отобразите на экране содержимое файла **my\_file**, выполнив:

**cat my\_file**

1. Вернитесь к редактированию файла **base1.sh**, набрав:

**nano base1.sh**

1. Обратите внимание на цветное оформление сценария. Добавьте комментарии в строках создания и записи файла:

**touch my\_file # создание файла**

**date > my\_file # запись даты и времени pwd >> my\_file # запись текущего каталога**

**who >> my\_file # запись имени пользователя uname >> my\_file # запись названия ОС**

1. Сохраните изменения и закройте редактор.
2. Теперь создайте **тестовый** скрипт командой:

**nano test1.sh**

Так же начните с шебанга и описания скрипта:

**#! /bin/sh**

**# сценарий проверки правильности выполнения # Лабораторной работы № 1**

**cat my\_file # вывод на экран содержимого файла my\_file echo $? # вывод кода завершения последней операции**

1. Сохраните файл и закройте редактор.
2. Настройте права для выполнения и этого сценария:

**chmod +x ./test1.sh**

1. Выполните сценарий:

**./test1.sh**

1. Убедитесь, что код завершения (последнее число) равен **0**.

Это говорит об **отсутствии ошибок** при выполнении команды.

1. Скопируйте файл **test1.sh** в новый файл **reset1.sh**. Затем вызовите его на редактирование:

**cp test1.sh reset1.sh nano reset1.sh**

1. Измените комментарии и содержание редактируемого файла:

**#! /bin/sh**

# сценарий очистки результатов выполнения # Лабораторной работы № 1

**# удаление файла my\_file**

**rm my\_file && echo 'рабочий файл удалён'**

**echo $?** # вывод кода завершения последней операции

1. Сохраните файл и завершите редактирование. Как в предыдущих случаях сделайте файл исполняемым и запустите его:

**chmod +x ./reset1.sh**

**./reset1.sh**

1. Можно проверить все три файла ещё одним последовательным запуском, а затем сверить с образцом содержание самих скриптов:

|  |  |
| --- | --- |
| *base1.sh* | #!/bin/sh  # сценарий записи системной информации в файл # Лабораторная работ № 1  **touch my\_file** # создание файла  **date > my\_file** # запись даты и времени  **pwd >> my\_file** # запись текущего каталога в файл  **who >> my\_file** # запись имени в файл  **uname >> my\_file** # запись названия ОС в файл |
| *test1.sh* | #! /bin/sh  # сценарий проверки правильности выполнения # Лабораторной работы № 1  **cat my\_file** # вывод на экран содержимого файла my\_file  **echo $?** # вывод кода завершения последней операции |
| *reset1.sh* | #! /bin/sh  # сценарий очистки результатов выполнения # Лабораторной работы № 1  **rm my\_file && echo 'рабочий файл удалён'** # удаление файла my\_file  **echo $?** # вывод на экран кода завершения последней операции |

**Задание 1.2**

В качестве самостоятельного задания разработайте три аналогичных сценария, при этом в файл **my\_hist\_file** должно быть записано 10 последних команд текущего сеанса работы.

# Лабораторная работа 2

1. Результат выполнения лабораторных работ — это **три** сценария на языке командного интерпретатора.:

**первый** (основной) выполняет задание;

**второй** (тестовый) проверяет правильность работы основного;

**третий** (обнуляющий) возвращает систему в исходное состояние так, чтобы можно было запустить процесс выполнения сценариев сначала.

1. Задача **основного** сценария **второй** лабораторной работы:

создать в **/tmp** в новом файле **my\_sys\_2M.img** размером **2M** файловую систему **ext2**; провести монтирование новой ФС в **/mnt/new\_disk**;

исследовать возможность заполнения ФС в двух вариантах: по **количеству** файлов;

по **объёму** ФС (дополнительное самостоятельное задание);

1. Тестовый сценарий должен проверить наличие ФС, показать её параметры, количество созданных файлов.
2. Для возврата к начальным условиям необходимо очистить и отмонтировать файловую систему, а затем удалить файл **my\_sys\_2M.img** .
3. Для начала работы запустите текстовый редактор **nano**:

**nano base2.sh**

1. Наберите шебанг и первые комментарии с описанием сценария:

**#!/bin/sh**

# сценарий создания и монтирования ФС # лабораторная работа № 2

Дальше следуют команды подготовки файла **my\_sys\_2M.img** и создания на нем ФС:

**truncate -s 2M /tmp/my\_sys\_2M.img** # файл заданного размера

**mkfs.ext2 /tmp/my\_sys\_2M.img** # созд. ФС ext2 **mkdir /mnt/new\_disk** # созд.точку монтирования **mount /tmp/my\_sys\_2M.img /mnt/new\_disk** # монтируем ФС **df -H /mnt/new\_disk** # выводим сведения о ФС

Теперь приступаем к заполнению ФС:

**mkdir /mnt/new\_disk/new\_dir** # создаем каталог в новой ФС

# в новом каталоге: 1000 файлов,прячем сообщения об ошибках

**cd /mnt/new\_disk/new\_dir**

**i=1; while touch file\_$i; do echo file\_$i; i=$[i+1];done**

(Предложите альтернативный вариант цикла, например с использованием **for**.)

1. Сохраните файл и закройте редактор. Настройте права на исполнение и запустите сценарий с правами суперпользователя. После выполнения не забудьте завершить сеанс работы суперпользователя.

**chmod +x ./base2.sh su -**

**./base2.sh**

1. Начните работу с тестовым сценарием:

**nano test2.sh**

1. Наберите шебанг и первые комментарии с описанием сценария:

**#! /bin/sh**

# сценарий исследования ФС # лабораторная работа № 2

1. Тестовый сценарий должен вывести информацию о файловой системе и посчитать количество файлов в созданном каталоге:

**df -i /mnt/new\_disk** # выводим сведения о количестве I-нодов

**cd /mnt/new\_disk/new\_dir**

**echo "количество созданных файлов:"**

**ls -l | wc -l** # считаем количество файлов

1. Сохраните файл и закройте редактор. Настройте права на исполнение и запустите сценарий.

**chmod +x ./test2.sh**

**./test2.sh**

1. Начните работу с обнуляющим сценарием:

**nano reset2.sh**

1. Наберите шебанг и первые комментарии с описанием сценария:

**#!/bin/sh**

# сценарий удаления ФС

# лабораторная работа № 2

1. Для возврата в исходное состояние нужно удалить созданные на новой ФС файлы и каталоги, размонтировать её, удалить файл самой системы и каталог в /mnt. Часть этих операций потребует прав суперпользователя:

**cd /mnt/new\_disk/** # переходим в ФС

**rm -rf** # удаляем файлы

**umount /mnt/new\_disk** # размонтируем ФС

**rm /tmp/my\_sys\_2M.img** # удаляем файл my\_sys\_2M.img

1. Сохраните файл и закройте редактор. Настройте права на исполнение и запустите сценарий и запустите сценарий с правами суперпользователя. После выполнения не забудьте завершить сеанс работы суперпользователя. .

**chmod +x ./reset2.sh su -**

**./reset2.sh**

**Задание 2.2**

В качестве самостоятельного задания разработайте три аналогичных сценария, но исследуйте переполнение файловой системы по объёму.

# Лабораторная работа 3

* 1. Как обычно, результат выполнения лабораторной работы — это **три** сценария на языке командного интерпретатора.:

**первый** (основной) выполняет задание;

**второй** (тестовый) проверяет правильность работы основного; **третий** (обнуляющий) возвращает систему в исходное состояние так, чтобы можно было запустить процесс выполнения сценариев сначала.

* 1. Задача **основного** сценария третьей лабораторной работы:

завести две локальные **групповые** учётные записи: **group1** и **group2**;

создать три локальные **пользовательские** учётные записи: **user1**, **user2**, **user3**; включить пользователей в группы следующим образом:

**первый** пользователь входит только в **первую** группу, **второй** пользователь — в **первую** и **вторую** группы, **третий** — только во **вторую**;

создать в пределах ветки дерева каталогов /srv каталоги **dir1** и **dir2**; разграничить права следующим образом:

подкаталог **dir1** доступен на запись только группе **group1**, **dir2** — только группе **group2**.

* 1. Тестовый сценарий должен проверить соответствие прав на файлы, создаваемые в каталогах dir1 и dir2.
  2. Для возврата к начальным условиям необходимо удалить пользователей и группы, а затем удалить созданные каталоги и файлы.
  3. Для начала работы запустите текстовый редактор **nano**:

**nano base3.sh**

* 1. Наберите шебанг и первые комментарии с описанием сценария:

**#!/bin/sh**

# сценарий создания групп пользователей # лабораторная работа № 3

Дальше следуют команды создания групп и пользователей:

**groupadd group1** # создаем группы

**groupadd group2**

**adduser user1** # новые пользователи

**adduser user2 adduser user3**

**gpasswd -a user1 group1** # добавляем пользователей в группы

**gpasswd -a user2 group1 gpasswd -a user2 group2 gpasswd -a user3 group2**

**for i in `seq 1 3` do** # инфо о пользователях и группах

**echo "пользователь user$i" id user$i**

**done**

**mkdir /srv/dir1 /srv/dir2** # создаем каталоги **chgrp group1 dir1** # "отдаем" каталоги соотв. группам **chgrp group2 dir2**

**chmod 2775 /srv/dir1 /srv/dir2** # права на запись в каталоги

**ls -la /srv**

* 1. Сохраните файл и закройте редактор. Настройте права на исполнение и запустите сценарий и запустите сценарий с правами суперпользователя. После выполнения не забудьте завершить сеанс работы суперпользователя.
  2. Разработайте тестовый сценарий (обратите внимание на различные варианты использования команды echo):

nano test3.sh

#!**/bin/sh**

# сценарий проверки прав пользователей # лабораторная работа № 3

**su —l user2 -c** ' **cd /srv/dir1 touch file\_u2d1**

**whoami && echo "создал(а) файл в каталоге" && pwd cd /srv/dir2**

**touch file\_u2d2**

**echo "`whoami` создал(а) файл в каталоге `pwd`" '**

# user1 - доступ к файлам в dir1

**su —l user1 -c ' cd /srv/dir1/**

**whoami && echo "доступ к файлу в каталоге" && pwd mv /srv/dir1/file\_u2d1 /srv/dir1/file\_u1d1**

**echo $?**

# user1 - доступ к файлам в dir2

**cd /srv/dir2/**

**echo "`whoami` создал(а) файл в каталоге `pwd`" mv /srv/dir1/file\_u2d2 /srv/dir1/file\_u1d2**

**echo $? '**

* 1. Аналогично — для пользователя user3.

Затем сохраните файл и закройте редактор. Настройте права на исполнение и запустите сценарий.

* 1. Разработайте обнуляющий сценарий:

nano reset3.sh

#!**/bin/sh**

# сценарий удаления файлов, групп и пользователей # лабораторная работа № 3

# удаление каталогов и файлов

**rm -rf /srv/dir1 /srv/dir2**

# удаление групп и пользователей

**userdel —r user1 userdel —r user2 userdel —r user3**

**groupdel group1 groupdel group2**

* 1. Сохраните файл и закройте редактор. Настройте права на исполнение и запустите сценарий и запустите сценарий с правами суперпользователя. После выполнения не забудьте завершить сеанс работы суперпользователя.

**Задание 3.2**

В качестве самостоятельного задания разработайте три аналогичных сценария, при этом имена пользователей и названия групп должны задаваться как параметры сценария. Должна проводиться проверка количества передаваемых параметров.

# Лабораторная работа 4

1. В этой лабораторной работы нужно создать **четыре** сценария на языке командного интерпретатора.:

**стартовый** (подготовительный) задает рабочую среду;

**первый** (основной) выполняет задание;

**второй** (тестовый) проверяет правильность работы основного; **третий** (обнуляющий) возвращает систему в исходное состояние так, чтобы можно было запустить процесс выполнения сценариев сначала.

1. Задача **стартового** сценария четвертой лабораторной работы: запустить в фоновом режиме 5 процессов;

3 из которых являются **целевыми**, а 2 — **дополнительными**.

Для примеров фоновых процессов можно выбрать ping ya.ru и ping yandex.ru.

1. **Основной** сценарий должен прекратить выполнение только **целевых** процессов.
2. **Тестовый** сценарий должен сообщать сколько целевых и дополнительных процессов выполняется в данный момент.
3. Для возврата к начальным условиям необходимо прекратить выполнение всех запущенных процессов.
4. В этой лабораторной работе первым разработайте **тестовый** сценарий. Запустите текстовый редактор:

**nano test4.sh**

1. Наберите шебанг и первые комментарии с описанием сценария:

**#!/bin/sh**

# сценарий мониторинга запущенных процессов # лабораторная работа № 4

**echo "ping по адресу ya.ru"** #количество основных процессов

**pgrep -f ya.ru| wc -l**

**echo "ping по адресу yandex.ru"** # количество доп. процессов

**pgrep -f yandex.ru| wc -l**

1. Сохраните файл и закройте редактор. Настройте права на исполнение и запустите сценарий. При первом запуске в результате должны получиться нули.
2. Разработайте стартовый сценарий, :

nano start4.sh

#!**/bin/sh**

# сценарий запуска фоновых процессов # лабораторная работа № 4

# запускаем 3 целевых процесса

**for i in `seq 1 3`; do ping ya.ru >/dev/null &; done**

# запускаем 2 дополнительных процесса

**for i in 1 2; do ping yandex.ru >/dev/null &; done**

1. Сохраните файл и закройте редактор. Настройте права на исполнение и запустите сценарий.
2. После этого снова запустите test4.sh, проверьте количество целевых и дополнительных процессов.
3. Основной сценарий должен «убить» процессы, связанные с ya.ru, и «пощадить» все остальные:

nano base4.sh

#!**/bin/sh**

# сценарий целевого прекращения процессов # лабораторная работа № 4

# выбираем и прекращаем процессы

**for p in `pgrep -f ya.ru` do**

**kill $p done**

1. Сохраните файл и закройте редактор. Настройте права на исполнение и запустите сценарий. После чего запустите test4.sh и проверьте количество целевых и дополнительных процессов.
2. Очищающий сценарий прекращает все процессы, запущенные в ходе работы:

**nano reset4.sh**

#!**/bin/sh**

# сценарий очистки для

# лабораторная работа № 4

# прекращаем все процессы из Лаб.работы 4

**for p in `pgrep -f ya.ru` do**

**kill $p done**

**for p in `pgrep -f yandex.ru` do**

**kill $p done**

1. Сохраните файл и закройте редактор. Настройте права на исполнение и запустите сценарий. После чего запустите test4.sh и проверьте количество целевых и дополнительных процессов.

**Задание 4.2**

В качестве самостоятельного задания разработайте аналогичные сценарии, изменив их так, чтобы адреса сайтов (целевого и дополнительного) задавались как параметры при запуске сценария. Должна проводиться проверка количества передаваемых параметров.

**Оглавление**

[UNIX-подобные операционные системы 2](#_bookmark0)

[Особенности UNIX-подобных ОС 2](#_bookmark1)

[Интерфейс командной строки 3](#_bookmark2)

[Стандартный синтаксис POSIX.2 3](#_bookmark3)

[Управление документацией 4](#_bookmark4)

[Система man (UNIX) 4](#_bookmark5)

[Метасинтаксические конструкции 5](#_bookmark6)

[Система info (GNU/Linux) 6](#_bookmark7)

[Встроенная справка приложения 6](#_bookmark8)

[Документация из пакета 6](#_bookmark9)

[Навигация по дереву каталогов 7](#_bookmark10)

[История команд 7](#_bookmark11)

[Дополнение имён 8](#_bookmark12)

[Перенаправление ввода-вывода 8](#_bookmark13)

[Специальные символы 9](#_bookmark14)

[Ввод управляющих последовательностей символов 10](#_bookmark15)

[Обработка текста 11](#_bookmark16)

[Редактирование текста 11](#_bookmark17)

[Просмотр текста 11](#_bookmark18)

[Создание скриптов 12](#_bookmark19)

[Базовые понятия 12](#_bookmark20)

[Дерево каталогов Linux 13](#_bookmark21)

[Файловые системы 15](#_bookmark22)

[Типы файловых системам 15](#_bookmark23)

[Монтирование файловых систем 16](#_bookmark24)

[Управление файловыми системами 17](#_bookmark25)

[Типы файлов 18](#_bookmark26)

[Атрибуты файлов 19](#_bookmark27)

[Индексный дескриптор 19](#_bookmark28)

[Жёсткие ссылки 19](#_bookmark29)

[Управление файлами 20](#_bookmark30)

[mc 20](#_bookmark31)

[Переменные окружения 21](#_bookmark32)

[Пример настройки истории команд 21](#_bookmark33)

[Подстановки Shell 21](#_bookmark34)

[Создание скриптов (продолжение) 22](#_bookmark35)

[Использование переменных 22](#_bookmark36)

[Условный оператор 22](#_bookmark37)

[Команда test 23](#_bookmark38)

[Сравнение чисел 23](#_bookmark39)

[Проверка файлов 23](#_bookmark40)

[Оператор case 24](#_bookmark41)

[цикл for со счетчиком 24](#_bookmark42)

[Цикл while 24](#_bookmark43)

[Базовая модель разграничения доступа 25](#_bookmark44)

[Управление доступом 25](#_bookmark45)

[Списки управления доступом 26](#_bookmark46)

[Расширенные атрибуты файлов 27](#_bookmark47)

[Дополнительные атрибуты файлов 27](#_bookmark48)

[Учётные записи 28](#_bookmark49)

[Пользовательские учётные записи 28](#_bookmark50)

[Хранилище паролей TCB 29](#_bookmark51)

[Групповые учётные записи 29](#_bookmark52)

[Управление пользователями 30](#_bookmark53)

[Управление группами 30](#_bookmark54)

[Аутентификация и преобразование имён 31](#_bookmark55)

[Подключаемые модули аутентификации 31](#_bookmark56)

[Коммутатор системы преобразования имён 31](#_bookmark57)

[Архивация файлов 32](#_bookmark58)

[Поиск файлов 32](#_bookmark59)

[Управление конфигурацией 33](#_bookmark60)

[Конфигурация (контекст) 33](#_bookmark61)

[общесистемная (каталог /etc) 33](#_bookmark62)

[пользовательская (домашний каталог) 33](#_bookmark63)

[Конфигурация (уровни) 34](#_bookmark64)

[Основные форматы конфигурационных файлов UNIX 35](#_bookmark65)

[Пользовательское окружение 36](#_bookmark66)

[Пользовательский профиль 36](#_bookmark67)

[Процессы в Linux 37](#_bookmark68)

[Состояния процесса: 37](#_bookmark69)

[Атрибуты процесса: 38](#_bookmark70)

[Сигналы 39](#_bookmark71)

[Приоритет процесса 40](#_bookmark72)

[Управление заданиями Shell 40](#_bookmark73)

[Фильтры 41](#_bookmark74)

[Удалённый доступ 42](#_bookmark75)

[Диагностика настроек сетевой подсистемы 42](#_bookmark76)

[OpenSSH 43](#_bookmark77)

[Аутентификация по ключам 44](#_bookmark78)

[Полезные утилиты 44](#_bookmark79)

[Графическая подсистема Linux 45](#_bookmark80)

[Общая информация 45](#_bookmark81)

[X-сервер 45](#_bookmark82)

[Запуск вложенного X-сервера 46](#_bookmark83)

[Оконные менеджеры (Window Managers) 46](#_bookmark84)

[Менеджеры входа (дисплейные менеджеры) 46](#_bookmark85)

[Настольные пользовательские окружения (графические среды) 47](#_bookmark86)

[Настроечные файлы X-сервера и его запуск 47](#_bookmark87)

[Подключение к X-серверу по сети (MATE, LightDM) 47](#_bookmark88)

[Разграничение доступа на X-сервере 48](#_bookmark89)

[Аутентификация средствами xauth 48](#_bookmark90)

[Управление X-сеансами средствами SSH 48](#_bookmark91)

[дополнительная тема Использование потокового редактора SED 49](#_bookmark92)

[Синтаксис командной строки 49](#_bookmark93)

[Ключи (опции) sed 49](#_bookmark94)

[Часто используемые (основные) команды sed 49](#_bookmark95)

[дополнительные команды sed 50](#_bookmark96)

[Модификаторы замены 50](#_bookmark97)

[Преобразование регистра 50](#_bookmark98)

[Лабораторная работа 1 51](#_bookmark99)

[Лабораторная работа 2 54](#_bookmark100)

[Лабораторная работа 3 57](#_bookmark101)

[Лабораторная работа 4 60](#_bookmark102)