

1.1 Users & Groups

1. `#learn` `#linux` `#alt` `#bash` `#users` `#groups`

2. Зарегистрировать пользователя test1:

```
sudo useradd -u 1100 -M -N -s /sbin/nologin -d /var/spool/mail/test1 -G users,mail test1
```

- `useradd` : Команда для создания нового пользователя.
- `-u 1100` : Устанавливает UID пользователя в 1100.
- `-M` : Не создавать домашний каталог.
- `-N` : Не создавать группу с именем пользователя.
- `-s /sbin/nologin` : Устанавливает запрет на вход в сеанс.
- `-d /var/spool/mail/test1` : Устанавливает домашний каталог пользователя.
- `-G users,mail` : Добавляет пользователя в группы `users` и `mail`.

2. Создать учетную запись для пользователя test2:

```
sudo useradd test2
```

3. Проверить создание домашнего каталога пользователя test2:

```
ls -l /home/test2
```

- Если домашний каталог создан, вы увидите его в списке, и его владельцем будет пользователь `test2`.

4. Изменить имя пользователя test2 на test3:

```
sudo usermod -l test3 test2
```

- `usermod` : Команда для изменения свойств пользователя.
- `-l test3` : Устанавливает новое имя пользователя.

5. Удалить пользователя test3 и передать права владения его файлами суперпользователю и группе sys:

```
sudo userdel -r test3 sudo chown -R root:sys /home/test3
```

- `userdel -r test3` : Удаляет пользователя `test3` вместе с его домашним каталогом (`-r`).
- `chown -R root:sys /home/test3` : Передает права владения домашним файлам пользователя `test3` суперпользователю и группе `sys`.

6. Найти и изучить другой конфигурационный файл, влияющий на поведение команды `useradd`:

```
man useradd
```

В справке вы найдете упоминание файлов `/etc/login.defs` и

`/etc/default/useradd`, которые могут влиять на поведение команды `useradd`.

Прочтите их содержимое, чтобы найти настройку, отвечающую за минимальный UID для новых пользователей.

7. Зарегистрировать пользователя `test4` с настройками по умолчанию и установить для него пароль:

```
sudo useradd test4 sudo passwd test4
```

8. Изучить содержимое записи пользователя `test4` в файле `/etc/shadow`:

```
sudo grep test4 /etc/shadow
```

9. Установить дату устаревания пароля для пользователя `test4` на 31 декабря текущего года:

```
chage -E 2024-12-31 test4
```

```
sudo chage -E $(date -d '12/31' +%Y-%m-%d) test4
```

- подробнее разберем команду:

1. `sudo`: Это команда, которая предоставляет временные привилегии суперпользователя для выполнения команды. Суперпользователь (или `root`) имеет расширенные права доступа, которые позволяют выполнять действия, требующие специальных разрешений.
2. `chage`: Это команда для изменения атрибутов срока действия учетной записи пользователя в файле `/etc/shadow`. С помощью `chage` можно установить различные параметры, такие как срок действия пароля, срок действия учетной записи и другие.
3. `-E $(date -d '12/31' +%Y-%m-%d)`: Это опция для установки даты истечения срока действия пароля. Здесь мы используем команду `date -d '12/31' +%Y-%m-%d`, чтобы получить дату 31 декабря текущего года в формате `гггг-мм-дд`. Затем эта дата передается в опцию `-E`, чтобы установить дату истечения срока действия пароля.
4. `test4`: Это имя пользователя, для которого мы устанавливаем дату истечения срока действия пароля. Выполняя эту команду, мы устанавливаем для пользователя `test4` дату истечения срока действия пароля на 31 декабря текущего года.
5. Итак, команда `sudo chage -E $(date -d '12/31' +%Y-%m-%d) test4` позволяет установить дату истечения срока действия пароля для пользователя `test4` на 31 декабря текущего года.

```
1. chage -E 2024-12-31 root
```

При наступлении указанного срока учетная запись заданного пользователя

будет автоматически заблокирована.

2. `chage -W 100 root`

Как результат, юзер системы Linux получит предупреждение о смене пароля заранее (за десять дней до истечения срока).

6. Проверить изменения в `/etc/shadow`:

```
sudo grep test4 /etc/shadow
```

1. Удалить пароль пользователя и проверить изменения в `/etc/shadow`:

```
sudo passwd -d test4 sudo grep test4 /etc/shadow
```

12. Заблокировать учетную запись `test4`:

```
sudo usermod -L test4
```

13. Создать группу пользователей `xusers` с `GID 1010`:

```
sudo groupadd -g 1010 xusers
```

14. Зарегистрировать себя в качестве участника группы `xusers`:

```
sudo usermod -aG xusers your_username
```

15. Добавить пользователя `test4` в группу `xusers`:

```
sudo usermod -aG xusers test4
```

16. Найти утилиту для изменения имен и `GID` групп и изменить имя группы на `yusers`:

```
sudo groupmod -n yusers xusers
```

Теперь вы завершили выполнение всех указанных шагов. Убедитесь, что вы внимательно проверили каждый шаг, чтобы избежать возможных ошибок и проблем.

подробнее разберем команду `sudo chage -E $(date -d '12/31' +'%Y-%m-%d') test4`:

- `sudo` : Это команда, которая предоставляет временные привилегии суперпользователя для выполнения команды. Суперпользователь (или `root`) имеет расширенные права доступа, которые позволяют выполнять действия, требующие специальных разрешений.
- `chage` : Это команда для изменения атрибутов срока действия учетной записи пользователя в файле `/etc/shadow`. С помощью `chage` можно установить различные параметры, такие как срок действия пароля, срок действия учетной записи и другие.
- `-E $(date -d '12/31' +'%Y-%m-%d')` : Это опция для установки даты истечения срока действия пароля. Здесь мы используем команду `date -d '12/31' +'%Y-%m-%d'`, чтобы получить дату 31 декабря текущего года в формате `гггг-мм-дд`. Затем эта дата передается в опцию `-E`, чтобы установить дату истечения срока действия пароля.
- `test4` : Это имя пользователя, для которого мы устанавливаем дату истечения срока действия пароля. Выполняя эту команду, мы устанавливаем для пользователя `test4`

дату истечения срока действия пароля на 31 декабря текущего года.

- Итак, команда `sudo chage -E $(date -d '12/31' +%Y-%m-%d) test4` позволяет установить дату истечения срока действия пароля для пользователя `test4` на 31 декабря текущего года.

1.3 Окружение

#learn #linux #alt #bash

- псевдотаблица (без шапки) с таб все поля заполняем
- параметр - значение
- ini
- Конфигурация - скрипт

1. Пользовательское окружение

- Профиль
- Переменные

2. Переменные окружения

- echo - вывод потока вывода))
- env
- set
- export
- locale

3. Профиль

- /etc/
- ~/

1. Настройка sudo для групп wheel и admin:

Для этого вам нужно отредактировать файл **sudoers**, используя команду *visudo*, которая безопасно редактирует файл конфигурации sudo.

```
sudo visudo
```

Добавьте следующие строки в файл **sudoers**:

Разрешить членам группы wheel выполнять любую команду

```
%wheel ALL=(ALL) ALL
```

Разрешить членам группы admin выполнять только *passwd* от имени root

```
%admin ALL=(ALL) /usr/bin/passwd
```

Сохраните и закройте файл **sudoers**. Теперь члены группы wheel могут выполнять любую команду от имени root, в то время как члены группы admin могут выполнять только команду *passwd* от имени **root**.

Проверка конфигурации *sudo*:

Вы можете выполнить команду *sudo -l*, чтобы увидеть список команд, которые пользователь может выполнить с помощью *sudo*. Если вы входите в группу wheel или admin, вы сможете выполнить соответствующие команды.

2. Установка значения *umask* для пользователей с оболочкой Bash по умолчанию:

Чтобы установить значение **umask** для пользователей с оболочкой Bash по умолчанию, вы можете добавить команду *umask* в глобальный файл конфигурации Bash.

Откройте файл */etc/profile* для редактирования:

```
sudo nano /etc/profile
```

Измените *umask* :

umask 027

Сохраните и закройте файл. Это устанавливает значение *umask* равным 027 для всех пользователей с оболочкой Bash по умолчанию.

Вы можете проверить текущее значение *umask*, запустив команду *umask* в вашем терминале. Она должна отобразить установленное значение

3. Установка псевдонима 'l' для 'ls -LI' в собственном профиле оболочки:

Откройте файл вашего собственного профиля оболочки (например, *.bashrc* или *.bash_profile*):

```
nano ~/.bashrc
```

Добавьте псевдоним в файл:

alias l='ls -LI'

Сохраните и закройте файл. Теперь, когда вы вводите 'l' в терминале, будет выполнена команда 'ls -LI'.

4. Установка псевдонима 'l' для 'ls -LI' для всех новых пользователей с оболочкой Bash по умолчанию:

Это можно сделать, добавив псевдоним в шаблонный профиль оболочки Bash, который используется при создании новых учетных записей пользователей.

Откройте файл */etc/skel/.bashrc* для редактирования:

```
sudo nano /etc/skel/.bashrc
```

Добавьте псевдоним в файл:

```
alias l='ls -LI'
```

Сохраните и закройте файл. Теперь, когда создается новая учетная запись пользователя с оболочкой Bash по умолчанию, у нее будет установлен псевдоним 'l'.

5. Добавление каталога в переменную окружения PATH:

Самое удобное место для добавления каталога в переменную окружения PATH для обычного пользователя - это его собственный файл профиля оболочки.

Откройте ваш файл профиля оболочки (например, .bashrc или .bash_profile):

```
nano ~/.bashrc
```

Добавьте каталог в переменную PATH:

```
export PATH=$PATH:/путь/к/каталогу/bin
```

Замените /путь/к/каталогу/bin на реальный путь к каталогу bin в вашем домашнем каталоге.

Сохраните и закройте файл.

6. Изменение вторичного приглашения командной строки:

Вы можете изменить вторичное приглашение командной строки (PS2), изменив переменную окружения PS2 в вашем файле профиля оболочки.

Откройте ваш файл профиля оболочки (например, .bashrc или .bash_profile):

```
nano ~/.bashrc
```

Добавьте или измените переменную PS2, чтобы установить желаемое вторичное приглашение:

```
PS2='-> '
```

```
export PS2
```

Сохраните и закройте файл. Теперь, когда вы запускаете новую сессию Bash, вторичное приглашение будет установлено на '->'.

1.3.2 Day Процессы

#learn

#linux

#alt

#bash

1. Запустите в фоновом режиме два задания: `sleep 200` и `sleep 2000`, выведите информацию о состоянии заданий.

Команда:

```
sleep 200 &  
sleep 2000 &  
jobs
```

- Описание:
 - `sleep 200 &`: Запускает процесс "спящего" состояния на 200 секунд в фоновом режиме.
 - `sleep 2000 &`: Запускает процесс "спящего" состояния на 2000 секунд в фоновом режиме.
 - `jobs`: Выводит информацию о заданиях, включая их состояние и идентификаторы.

2. Снимите с выполнения 2-е задание, выведите информацию о заданиях.

Команда:

```
kill %2  
jobs  
jobs
```

- Описание:
 - `kill %2`: Отправляет сигнал завершения процессу с идентификатором 2 (второму запущенному заданию).
 - `jobs`: Выводит информацию о заданиях, чтобы убедиться, что второе задание завершено.
3. Как изменяется общая картина жизненного цикла процесса в системе по сравнению с описанным, если процесс запускается в фоновом режиме.
 - При запуске процесса в фоновом режиме, он продолжает выполняться в фоне, не блокируя терминал. Пользователь может продолжать работу с командной

строкой, в то время как процесс выполняется. Пользователь может отправлять сигналы процессу или прервать его выполнение.

4. Выполните команду `exec ls`. Изучите ее поведение.

Команда:

```
exec ls
```

- Описание:
 - `exec` : Заменяет текущий процесс новым процессом, указанным в аргументах командной строки.
 - `ls` : Команда для вывода содержимого текущего каталога.
 - Команда `exec ls` заменит текущий процесс процессом `ls`, поэтому после выполнения этой команды вместо оболочки Bash будет запущена утилита `ls`, и после её завершения пользователь снова вернется в оболочку Bash.

5.Получите информацию о процессах в обычном и подробном форматах.

Команда:

```
ps  
ps aux
```

- Описание:
 - `ps` : Выводит список текущих процессов в обычном формате.
 - `ps aux` : Выводит список всех процессов в подробном формате, включая процессы всех пользователей.

6. Выведите список всех процессов, запущенных всеми пользователями, вошедшими в сеанс, в системе.

Команда:

```
ps -e
```

- Описание:
 - `ps -e` : Выводит список всех процессов, запущенных на системе.

7. Проследите в течение минуты поведение процессов, использующих наибольшее процессорное время, используя наиболее подходящую для этого утилиту.

Команда:

```
top
```

- Описание:
 - Утилита `top` показывает текущие процессы, использующие наибольшее процессорное время, а также другую информацию о загрузке системы. Обновляется периодически и позволяет отслеживать изменения в реальном времени.

8.Получите иерархический список процессов с помощью команды `ps`.

Команда:

```
ps axjf
```

- Описание:
 - `ps axjf` : Выводит иерархический список процессов в виде дерева.

9.Запустите порожденную оболочку Bash. Исследуйте, посылая родительской оболочке Bash сигналы TERM, INT, QUIT и HUP, что при этом происходит.

Команда:

```
bash
```

- Описание:
 - Запускает новую оболочку Bash. При посылке сигналов родительской оболочке:
 - SIGTERM (15): Обычно приводит к завершению процесса.
 - SIGINT (2): Прерывает процесс (аналогично нажатию Ctrl+C).
 - SIGQUIT (3): Приводит к корректному завершению процесса с созданием core dump.
 - SIGHUP (1): Обычно используется для перезапуска процесса.

10. От имени обычного пользователя пошлите сигнал KILL любому процессу, запущенному от имени другого пользователя. Что произойдет?

Команда:

```
kill -9 <PID>
```

- Описание:

- `kill -9 <PID>`: Отправляет сигнал KILL (9) процессу с указанным идентификатором.

Процесс будет насильственно завершен, даже если он будет запущен от имени другого пользователя. Это происходит потому, что сигнал KILL не может быть перехвачен или обработан процессом, и он немедленно завершает процесс, игнорируя все другие обстоятельства.

11. Запустите в фоновом режиме команду `sleep 1000`. Проверьте, на какие сигналы из следующих: TERM, INT, QUIT и HUP, реагирует эта команда.

Команда:

```
sleep 1000 &
```

- Описание:

Команда `sleep 1000 &` запускает процесс "спящего" состояния на 1000 секунд в фоновом режиме.

Реакция на различные сигналы:

SIGTERM (15): Процесс будет завершен.

SIGINT (2): Процесс будет прерван.

SIGQUIT (3): Процесс будет завершен, создав core dump.

SIGHUP (1): Поведение зависит от настройки терминала. Обычно процесс будет прерван, если он связан с терминалом.

1.4 Day Поиск и Сортировка

#learn #linux #alt #bash

1. **Вопрос:** Получите список групп пользователей в системе, отсортированный по GID в обратном числовом порядке.

Ответ: Для получения списка групп пользователей, отсортированного по GID в обратном числовом порядке, можно воспользоваться командой

```
cat /etc/group | sort -n -r -k3
```

Команда `cat /etc/group | sort -n -r -k3` используется для вывода содержимого файла `/etc/group`, а затем сортировки этого вывода по третьему столбцу в обратном порядке.

Вот подробное разъяснение аргументов и переменных в этой команде:

1. ``cat``: Это утилита Unix/Linux, используемая для вывода содержимого файлов. В данном случае, ``cat`` используется для вывода содержимого файла ``/etc/group``.
2. ``/etc/group``: Это путь к файлу, содержащему информацию о группах пользователей в системе Unix/Linux. Этот файл содержит строки, каждая из которых представляет информацию о конкретной группе.
3. ``|`` (вертикальная черта): Это оператор конвейера (pipeline), который направляет вывод одной команды вводу другой команды. В данной команде он направляет вывод ``cat /etc/group`` на ввод команды ``sort``.
4. ``sort``: Эта команда используется для сортировки строк текстового ввода. В данном контексте она будет сортировать вывод, полученный от ``cat /etc/group``.
5. ``-n``: Этот аргумент ``sort`` указывает на нечисловую сортировку. Он обеспечивает сортировку строк текста в порядке, который соответствует их числовому значению.
6. ``-r``: Этот аргумент ``sort`` указывает на обратный порядок сортировки. То есть строки будут отсортированы в обратном порядке, начиная с наибольшего значения.
7. ``-k3``: Этот аргумент ``sort`` указывает на то, что сортировка должна производиться по третьему столбцу. В файле ``/etc/group`` третий столбец содержит идентификаторы группы (GID), и мы сортируем строки по этому столбцу.

2. **Вопрос:** С помощью утилит `find`, `head` и `sort` получите список из десяти файлов в домашнем каталоге, занимающих наибольшее дисковое пространство.

Ответ: Для этого можно использовать комбинацию утилит `find`, `du`, `sort` и `head`.

Команда будет выглядеть примерно так:

```
find ~ -type f -exec du -h {} + | sort -rh | head -n 10
```

Эта команда найдет все файлы в домашнем каталоге (`~`), затем посчитает их размеры с помощью `du`, сортирует результаты в обратном порядке по размеру и выводит первые 10 файлов с наибольшим размером.

Команда `find ~ -type f -exec du -h {} + | sort -rh | head -n 10` используется для поиска всех файлов в домашней директории пользователя, затем подсчета их размера и вывода 10 наиболее крупных файлов.

Вот подробное разъяснение аргументов, переменных и знаков в этой команде:

1. ``find``: Это утилита Unix/Linux, используемая для поиска файлов и каталогов в файловой системе. В данном случае, она будет искать файлы в указанной директории.
2. ``~``: Это символ, обозначающий домашнюю директорию текущего пользователя. В данной команде ``~`` указывает ``find`` начать поиск файлов с домашней директории пользователя.
3. ``-type f``: Этот аргумент ``find`` указывает на то, что нужно искать только обычные файлы (не каталоги, не ссылки и т.д.).
4. ``-exec``: Этот аргумент ``find`` используется для выполнения команды для каждого найденного файла.
5. ``du -h {} +``: Это команда ``du`` с аргументами ``-h`` для вывода размера файлов в удобном для чтения формате (в гигабайтах, мегабайтах и т.д.). ``{}`` является заполнителем, который будет заменен именем найденного файла. ``+`` указывает на то, что ``du`` должен обрабатывать несколько файлов за один вызов.
6. ``|``: Это оператор конвейера (pipeline), который направляет вывод одной команды на ввод другой команды. В данном случае, вывод команды ``du -h`` будет направлен на ввод команды ``sort``.
7. ``sort -rh``: Это команда ``sort`` с аргументами ``-rh``. ``-r`` указывает на обратный порядок сортировки (от большего к меньшему), а ``-h`` указывает на сортировку в "гуманном" формате, который учитывает размеры файлов в удобном для чтения формате.
8. ``head -n 10``: Это команда ``head`` с аргументом ``-n 10``, которая выводит только первые 10 строк из ввода. В данном случае, она выводит 10 наиболее крупных файлов, которые были отсортированы командой ``sort``.

3. **Вопрос:** Определите, используя файл `/etc/passwd`, содержащий данные об учетных записях пользователей, сколько пользователей зарегистрировано в системе.

Ответ: Для подсчета количества пользователей можно воспользоваться командой `cat /etc/passwd | wc -l`

Эта команда считает количество строк в файле `/etc/passwd`, каждая строка соответствует одной учетной записи пользователя.

Вот подробное разъяснение аргументов, переменных и знаков в этой команде:

1. ``cat``: Это утилита Unix/Linux, используемая для вывода содержимого файлов. В данном случае, ``cat`` используется для вывода содержимого файла ``/etc/passwd``.
2. ``/etc/passwd``: Это путь к файлу, который содержит информацию о пользователях системы, такую как их имена, идентификаторы (UID), домашние каталоги и т.д.
3. ``|``: Это оператор конвейера (pipeline), который направляет вывод одной команды на ввод другой команды. В данном случае, он направляет вывод ``cat /etc/passwd`` на ввод команды ``wc``.
4. ``wc``: Это утилита Unix/Linux, используемая для подсчета количества строк, слов и символов в текстовом вводе.
5. ``-l``: Этот аргумент ``wc`` указывает на то, что нужно подсчитывать только количество строк во вводе.

4. **Вопрос:** Определите для скольких пользователей оболочкой по умолчанию является `bash`.

Ответ: Для этого можно воспользоваться командой

```
grep -c '/bash$' /etc/passwd
```

Эта команда найдет строки, в которых оболочка по умолчанию оканчивается на `"/bash"` и подсчитает их количество.

Команда `grep -c '/bash$' /etc/passwd` используется для поиска строк в файле `/etc/passwd`, которые содержат `/bash` в конце строки, и подсчета количества найденных совпадений.

Вот подробное разъяснение аргументов, переменных и знаков в этой команде:

1. ``grep``: Это утилита Unix/Linux, используемая для поиска строк в текстовых файлах по заданному шаблону. В данном случае, ``grep`` будет искать строки, соответствующие шаблону ``/bash$``.
2. ``-c``: Этот аргумент ``grep`` указывает на то, что нужно вывести только количество строк, содержащих совпадения с заданным шаблоном, а не сами совпадения.

3. ``'/bash$'``: Это регулярное выражение, которое задает шаблон поиска. ``'/bash$`` означает строку, которая оканчивается на ``'/bash``.

4. ``/etc/passwd``: Это путь к файлу, который будет просмотрен утилитой ``grep``. Файл ``/etc/passwd`` содержит информацию о пользователях системы.

5. Вопрос: Сколько имеется пользователей, UID которых больше 100?

Ответ: Для этого можно воспользоваться командой

```
awk -F: '$3 > 100 {count++} END {print count}' /etc/passwd
```

Эта команда использует `awk` для проверки каждой строки файла `/etc/passwd` и подсчитывает количество пользователей, у которых UID (третье поле) больше 100.

Команда `awk -F: '$3 > 100 {count++} END {print count}' /etc/passwd` используется для обработки содержимого файла `/etc/passwd` с использованием утилиты AWK. Она считает количество пользователей с идентификаторами UID (user ID), превышающими 100.

Вот подробное разъяснение аргументов, переменных и знаков в этой команде:

1. ``awk``: AWK – это язык программирования и утилита для обработки и анализа текстовых файлов. Он используется для выполнения действий на основе шаблонов и правил, заданных в программе AWK.

2. ``-F:``: Этот аргумент устанавливает разделитель полей для AWK. В данном случае, разделителем является символ ":". Таким образом, AWK будет разделять строки файла на поля, используя ":" в качестве разделителя.

3. ``'$3 > 100 {count++}'``: Это основная часть программы AWK, которая выполняется для каждой строки в файле. Она проверяет значение третьего поля (UID) в строке. Если значение UID больше 100, то увеличивает счетчик ``count`` на единицу.

4. ``END {print count}``: Это часть программы AWK, которая выполняется после обработки всех строк файла. Она выводит значение переменной ``count``, которая содержит количество пользователей с UID больше 100.

5. ``/etc/passwd``: Это путь к файлу ``/etc/passwd``, который содержит информацию о пользователях системы, включая их UID в третьем поле каждой строки.

6. Вопрос: Выведите пронумерованный список файлов в текущем каталоге.

Ответ: Для этого можно воспользоваться командой

```
ls -l | cat -n
```

Эта команда выводит список файлов в текущем каталоге с номерами строк, что позволяет пронумеровать файлы.

Команда `ls -1 | cat -n` используется для вывода списка файлов и каталогов в текущей директории, пронумерованных по порядку.

Вот подробное разъяснение аргументов, переменных и знаков в этой команде:

1. `ls` : Это утилита Unix/Linux, используемая для вывода списка файлов и каталогов в указанной директории.
2. `-1` : Этот аргумент `ls` указывает на то, что нужно выводить каждый элемент списка в отдельной строке. Это обеспечивает простой формат вывода, где каждый файл или каталог находится на отдельной строке.
3. `|` : Это оператор конвейера (pipeline), который направляет вывод одной команды на ввод другой команды. В данном случае, он направляет вывод `ls -1` на ввод команды `cat -n`.
4. `cat` : Это утилита Unix/Linux, используемая для конкатенации и вывода содержимого файлов. Здесь она используется без аргументов, поэтому просто принимает ввод из предыдущей команды.
5. `-n` : Этот аргумент `cat` используется для нумерации строк во входных данных. Он добавляет номер каждой строки перед ее содержимым.

7. **Вопрос:** Получите столбец из имен пользователей, находящийся в данный момент в сеансе.

Ответ: Для этого можно воспользоваться командой `w | awk '{print $1}'`. Эта команда использует `w` для вывода информации о текущих сеансах, а затем `awk` для извлечения первого столбца, содержащего имена пользователей.

Команда `w | awk '{print $1}'` используется для вывода имен пользователей, которые в данный момент вошли в систему.

Вот подробное разъяснение аргументов, переменных и знаков в этой команде:

1. `w` : Это утилита Unix/Linux, используемая для отображения информации о текущих пользователях системы, а также о том, как они используют ресурсы системы (например, время входа в систему, запущенные процессы и т. д.).
2. `|` : Это оператор конвейера (pipeline), который направляет вывод одной команды на ввод другой команды. В данном случае, он направляет вывод команды `w` на ввод команды `awk`.
3. `awk` : AWK - это язык программирования и утилита для обработки и анализа текстовых файлов. Он используется для выполнения различных операций с данными в текстовых файлах или выводе других команд.
4. `'{print $1}'` : Это основная часть программы AWK, которая выполняется для каждой

строки ввода. В данном случае, она выводит первое поле (разделенное пробелом) каждой строки. В контексте вывода команды `w`, первое поле содержит имена пользователей, которые вошли в систему.

8. **Вопрос:** Выведите список тех файлов текущего каталога, для которых можно найти в этом каталоге файлы, первые три символа в именах которых совпадают не менее, чем у двух файлов. Приведите два примера решения, используя `awk` и `sed`.

Ответ:

- Пример с использованием `awk`:

```
ls | awk '{print substr($0, 1, 3)}' | sort | uniq -d | while read -r prefix;
do ls | grep "^$prefix"; done
```

В этом примере команда `ls` выводит список файлов, затем `awk` извлекает первые три символа из каждого имени файла, `sort` сортирует их, `uniq -d` оставляет только повторяющиеся строки (т.е. префиксы, которые совпадают у нескольких файлов), а затем `while read -r prefix; do ls | grep "^$prefix"; done` выводит файлы, имена которых начинаются с этих префиксов.

1. `ls`: Эта команда используется для вывода содержимого текущего каталога. Она перечисляет все файлы и каталоги, находящиеся в текущем рабочем каталоге.
2. `|`: Этот символ (вертикальная черта) используется для передачи вывода одной команды в качестве ввода другой команде. В данном случае он передает вывод команды `ls` на вход команды `awk`.
3. `awk '{print substr($0, 1, 3)}'`: Это команда `awk`, которая обрабатывает вывод команды `ls`. Она использует функцию `substr` для извлечения первых трех символов из каждой строки вывода команды `ls`. Флаг `{print}` указывает `awk` на то, что нужно вывести результат на экран.
4. `|`: Второй символ вертикальной черты используется для передачи вывода `awk` на вход команды `sort`.
5. `sort`: Эта команда сортирует строки в лексикографическом порядке по умолчанию. В данном случае она сортирует список первых трех символов файлов по алфавиту.
6. `|`: Третий символ вертикальной черты используется для передачи вывода `sort` на вход команды `uniq`.
7. `uniq -d`: Эта команда `uniq` удаляет дублирующиеся строки из ввода, оставляя только уникальные. Флаг `-d` указывает ей выводить только повторяющиеся строки.
8. `|`: Четвертый символ вертикальной черты используется для передачи вывода `uniq` на вход команды `while`.

9. `while read -r prefix; do ls | grep "^$prefix"; done`: Это цикл `while`, который читает каждую строку из вывода `uniq` и присваивает ее переменной `prefix`. Затем он выполняет команду `ls | grep "^$prefix"`, которая выводит только те файлы, имена которых начинаются с префикса, полученного из предыдущих шагов. `grep "^$prefix"` ищет строки, начинающиеся с содержимого переменной `prefix` и выводит их на экран.

- Пример с использованием `sed`:

```
ls | sed 's/^\(.\\{3\\}\).*\/\1/' | sort | uniq -d | while read -r prefix; do ls  
| grep "^$prefix"; done
```

Этот пример похож на предыдущий, но вместо `awk` используется `sed` для извлечения первых трех символов из каждого имени файла. Остальные шаги такие же, как и в предыдущем примере.

1. `ls`: Это команда для вывода списка файлов и каталогов в текущем рабочем каталоге.
2. `sed 's/^\(.\\{3\\}\).*\/\1/'`: Это использование `sed` (поточного редактора) для выполнения поиска и замены в строках. В данном случае:
 - `s/`: Это опция команды `sed`, которая указывает на начало операции замены.
 - `^\(.\\{3\\}\).*`: Это регулярное выражение, которое ищет первые три символа в строке.
 - `^`: Символ начала строки.
 - `\(.\\{3\\}\)`: Это группа захвата, которая находит любой символ (`.`) повторяющийся три раза (`\{3\}`).
 - `.*`: Любое количество любых символов после первых трех символов.
 - `\1`: Это обратная ссылка на первую группу захвата, то есть на первые три символа. Это то, что будет выведено в результате.
3. `sort`: Эта команда сортирует вывод в лексикографическом порядке.
4. `uniq -d`: Эта команда выводит только повторяющиеся строки из входного потока. Опция `-d` указывает, что нужно вывести только повторяющиеся строки.
5. `while read -r prefix; do`: Это начало цикла `while`, который читает каждую строку из входного потока и присваивает ее переменной `prefix`.
6. `ls | grep "^$prefix"`: Эта команда использует `grep` для поиска файлов, имена которых начинаются с содержимого переменной `prefix`. Знак `^` в регулярном выражении указывает, что строка должна начинаться с указанного текста (`$prefix`).
7. `done`: Это завершение блока цикла `while`.

1.5 Day Scripts

[#learn](#) [#linux](#) [#alt](#) [#bash](#)

Вот решение задания 3.1 с подробными комментариями:

Основной сценарий (base3.sh):

```
#!/bin/sh
```

Сценарий создания групп пользователей

Лабораторная работа № 3

Создание групп

```
groupadd group1
groupadd group2
```

Создание пользователей

```
adduser user1
adduser user2
adduser user3
```

Добавление пользователей в группы

```
gpasswd -a user1 group1
gpasswd -a user2 group1
gpasswd -a user2 group2
gpasswd -a user3 group2
```

Вывод информации о пользователях и группах

```
for i in `seq 1 3`; do
    echo "Пользователь user$i"
    id user$i
done
```

Создание каталогов

```
mkdir /srv/dir1 /srv/dir2
```

Установка прав доступа к каталогам

```
chgrp group1 /srv/dir1
chgrp group2 /srv/dir2
chmod 2775 /srv/dir1 /srv/dir2
```

Проверка прав на каталоги

```
ls -la /srv
```

Тестовый сценарий (test3.sh):

```
#!/bin/sh
```

Сценарий проверки прав пользователей

Лабораторная работа № 3

Пользователь user2 создает файл в dir1 и dir2

 Bash

```
su -l user2 -c
cd /srv/dir1
touch file_u2d1
echo "(whoami)создал(а)файлвкаталоге$(pwd) cd /srv/dir2 touch file_u2d2 echo (whoami)
создал(а) файл в каталоге $(pwd)"
```

Пользователь user1 перемещает файлы из dir1 и dir2

```
su -l user1 -c '
cd /srv/dir1/
mv /srv/dir1/file_u2d1 /srv/dir1/file_u1d1
cd /srv/dir2/
mv /srv/dir2/file_u2d2 /srv/dir2/file_u1d2
```

Обнуляющий сценарий (reset3.sh):

```
#!/bin/sh
```

Сценарий удаления файлов, групп и пользователей

Лабораторная работа № 3

Удаление каталогов и файлов

```
rm -rf /srv/dir1 /srv/dir2
```

Удаление пользователей

```
userdel -r user1
```

```
userdel -r user2
```

```
userdel -r user3
```

Удаление групп

```
groupdel group1
```

```
groupdel group2
```

Запуск сценариев:

Для каждого сценария установите права на выполнение:

```
chmod +x base3.sh test3.sh reset3.sh
```

Запустите основной сценарий с правами суперпользователя:

```
sudo ./base3.sh
```

Запустите тестовый сценарий:

```
./test3.sh
```

Запустите обнуляющий сценарий с правами суперпользователя после завершения тестирования:

```
sudo ./reset3.sh
```

Дополнительное задание (3.2):

Вам нужно разработать три аналогичных сценария, но с именами пользователей и названиями групп в качестве параметров. Добавьте проверку количества передаваемых параметров.

User

Сделай на основе 3.1 дополнительное задание (3.2)

ChatGPT

Хорошо, давайте создадим три сценария на основе основного сценария из задания 3.1, но с возможностью передачи имен пользователей и названий групп в качестве параметров. Также мы добавим проверку количества передаваемых параметров.

Сценарий 3.2.1 (create_users_groups.sh):

```
#!/bin/sh
```

Сценарий создания пользователей и групп

Лабораторная работа № 3.2

Проверка количества переданных параметров

```
if [ "$#" -ne 6 ]; then  
echo "Использование: $0 "  
exit 1  
fi
```

```
group1=$1  
group2=$2  
user1=$3  
user2=$4  
user3=$5
```

Создание групп

```
groupadd $group1  
groupadd $group2
```

Создание пользователей

```
adduser $user1  
adduser $user2  
adduser $user3
```

Добавление пользователей в группы

```
gpasswd -a $user1 $group1
gpasswd -a $user2 $group1
gpasswd -a $user2 $group2
gpasswd -a $user3 $group2
```

Вывод информации о пользователях и группах

```
for user in $user1 $user2 $user3; do
echo "Пользователь $user"
id $user
done
```

Разберем данный сценарий подробно:

Объявление интерпретатора:

```
#!/bin/sh
```

Эта строка указывает на то, что интерпретатором для выполнения сценария будет обычный оболочка командной строки (shell).

Комментарий с описанием сценария:

Сценарий создания пользователей и групп

Лабораторная работа № 3.2

Этот блок содержит комментарии, которые описывают назначение и цель данного сценария.

Проверка количества переданных параметров:

```
if [ "$#" -ne 6 ]; then
```

```
echo "Использование: $0 <group1> <group2> <user1> <user2> <user3>"
exit 1
```

```
fi
```

Этот блок проверяет, что количество переданных параметров соответствует ожидаемому (в данном случае, 6 параметров). Если количество параметров не совпадает, выводится сообщение об использовании скрипта и происходит выход из сценария с кодом ошибки 1.

Присвоение значений параметров:

```
group1=$1
group2=$2
user1=$3
user2=$4
user3=$5
```

Эти строки присваивают значения переданных аргументов переменным для дальнейшего использования в сценарии.

Создание групп и пользователей:

```
groupadd $group1
groupadd $group2
adduser $user1
adduser $user2
adduser $user3
```

Здесь используются команды `groupadd` для создания групп и `adduser` для создания пользователей на основе переданных аргументов.

Добавление пользователей в группы:

```
gpasswd -a $user1 $group1
gpasswd -a $user2 $group1
gpasswd -a $user2 $group2
gpasswd -a $user3 $group2
```

Команды `gpasswd -a` добавляют пользователей в указанные группы.

Вывод информации о пользователях и группах:

```
for user in $user1 $user2 $user3; do
    echo "Пользователь $user"
    id $user
done
```

Этот блок использует цикл `for`, чтобы пройти по каждому из пользователей и вывести информацию о них с помощью команды `id`.

Это подробный разбор сценария создания пользователей и групп.

Сценарий 3.2.2 (`test_permissions.sh`):

```
#!/bin/sh
```

Сценарий проверки прав доступа к файлам

Лабораторная работа № 3.2

Проверка количества переданных параметров

```
if [ "$#" -ne 2 ]; then
echo "Использование: $0 "
exit 1
fi

user1=$1
user2=$2
```

Пользователь user2 создает файлы

```
su -l user2 -c 'cd /srv/dir1; touch file_u2d1; echo $(whoami) > $(pwd)/file_u2d1; cd /srv/dir2; touch file_u2d2; echo $(whoami) > $(pwd)/file_u2d2'
```

Пользователь user1 перемещает файлы

```
su -l $user1 -c '
cd /srv/dir1/
mv /srv/dir1/file_u2d1 /srv/dir1/file_u1d1
cd /srv/dir2/
mv /srv/dir2/file_u2d2 /srv/dir2/file_u1d2
'
```

Сценарий 3.2.3 (cleanup.sh):

```
#!/bin/sh
```

Сценарий очистки после выполнения

Лабораторная работа № 3.2

Удаление каталогов и файлов

```
rm -rf /srv/dir1 /srv/dir2
```

Удаление пользователей

```
for user in $user1 $user2 $user3; do
userdel -r $user
done
```

done

Удаление групп

```
groupdel $group1
```

```
groupdel $group2
```

Запуск сценариев:

Для каждого сценария установите права на выполнение:

```
chmod +x create_users_groups.sh test_permissions.sh cleanup.sh
```

Запустите сценарий создания пользователей и групп с передачей необходимых параметров:

```
./create_users_groups.sh group1 group2 user1 user2 user3
```

Запустите тестовый сценарий с передачей имен пользователей:

```
./test_permissions.sh user1 user2
```

После окончания тестирования запустите сценарий очистки:

```
./cleanup.sh
```

Это должно обеспечить создание пользователей, назначение их в группы, проверку прав доступа к файлам и очистку после завершения.

4.1 Disk RAID

#linux #bash #raid #disk

Для выполнения этих задач вам потребуется виртуальная машина, настроенная в среде Oracle VM VirtualBox, и установленная операционная система Linux. В качестве примера, я буду использовать CentOS, но вы можете выбрать любой другой дистрибутив Linux по вашему усмотрению.

Прежде чем начать, убедитесь, что у вас есть доступ к дискам или виртуальным дискам в вашей виртуальной машине, которые вы можете использовать для создания разделов и RAID-массивов.

Давайте начнем.

Шаг 1: Разбиение диска

Для разбиения диска на соответствующие разделы выполните следующие шаги:

```
# Запустите утилиту разбиения диска
sudo fdisk /dev/sda

# Создайте первый раздел размером 5 ГБ
# Введите последовательно следующие команды:
n      (для создания нового раздела)
p      (для создания первичного раздела)
<Enter> (для выбора номера раздела по умолчанию)
<Enter> (для выбора начального сектора по умолчанию)
+5G    (для указания размера раздела в 5 ГБ)
n      (для создания следующего раздела)
p      (для создания первичного раздела)
<Enter> (для выбора номера раздела по умолчанию)
<Enter> (для выбора начального сектора по умолчанию)
+5G    (для указания размера раздела в 5 ГБ)
n      (для создания третьего раздела)
p      (для создания первичного раздела)
<Enter> (для выбора номера раздела по умолчанию)
<Enter> (для выбора начального сектора по умолчанию)
<Enter> (для указания оставшейся части диска)

# Сохраните изменения и выйдите
w
```

Шаг 2: Создание VG и LV

Теперь мы создадим VG (Volume Group) на двух разделах по 5 ГБ и создадим на нем LV (Logical Volume) размером 8 ГБ.

```
# Создание VG
sudo vgcreate myvg /dev/sda1 /dev/sda2

# Создание LV на 8 ГБ
sudo lvcreate -L 8G -n mylv myvg
```

Шаг 3: Расширение LV

Расширим LV до 10 ГБ:

```
# Расширение LV до 10 ГБ
sudo lvextend -L 10G /dev/myvg/mylv

# Обновление размера файловой системы
sudo resize2fs /dev/myvg/mylv
```

Шаг 4: Создание RAID5

Создадим RAID5 на LV размером 10 ГБ и одним из оставшихся разделов на 10 ГБ.

```
# Создание RAID5
sudo mdadm --create --verbose /dev/md0 --level=5 --raid-devices=2
/dev/myvg/mylv /dev/sda3
```

Шаг 5: Создание ФС и монтирование

Создадим файловую систему ext4 и смонтируем RAID в /mnt/raid:

```
# Создание файловой системы ext4
sudo mkfs.ext4 /dev/md0

# Создание точки монтирования
sudo mkdir /mnt/raid

# Монтирование RAID
sudo mount /dev/md0 /mnt/raid
```

Шаг 6: Симуляция отказа диска и восстановление

Для симуляции отказа одного из дисков на RAID и его восстановления, выполните следующие шаги:

```
# Отключение одного из дисков
sudo mdadm --manage /dev/md0 --fail /dev/myvg/mylv

# Удаление отказавшего диска из RAID
sudo mdadm --manage /dev/md0 --remove /dev/myvg/mylv

# Проверка статуса RAID
cat /proc/mdstat

# Вставка нового диска (симулируем замену)
sudo mdadm --manage /dev/md0 --add /dev/sda2

# Проверка статуса RAID после восстановления
cat /proc/mdstat
```

Эти команды создадут и настроят нужные разделы, VG, LV, RAID5, файловую систему и точку монтирования. После выполнения этих шагов, вы сможете продолжить работу с вашим RAID-массивом.

GRUB 2.1

#learn

#linux

#alt

#bash

#grub

#systemctl

#journalctl

#service

- Задание выполняется в контексте безопасности суперпользователя
 1. Посмотрите текущий уровень работы операционной системы инструментарием SystemV

```
runlevel
```
 2. Посмотрите цель инициализации системы по-умолчанию

```
systemctl get-default
```
 3. Выполните перезагрузку системы
 4. В момент работы загрузчика GRUB перейдите в режим редактирования опций загрузки
e
 5. Допишите в строке загрузки ядра - начинается с `linux` указание загрузиться в 3ий уровень выполнения, дописав в конце строки номер уровня

 В конце строки `linux`

3


1. Выполните загрузку
F10
2. По окончании загрузки войдите в систему и перейдите в контекст безопасности суперпользователя
3. Посмотрите текущий уровень работы операционной системы инструментарием SystemV

```
runlevel
```

2.2. Загрузка с замененным процессом `init`

- Задание выполняется в контексте безопасности суперпользователя
 1. С использованием команды *which* определите путь к исполняемому файлу командного интерпретатора `bash`. Запомните/запишите его.
 2. Выполните перезагрузку системы

3. В момент работы загрузчика GRUB перейдите в режим редактирования опций загрузки
4. Допишите в строчке загрузке ядра ОС необходимость использовать командный интерпретатор вместо процесса `init`

 Пишем в конце строки `linux` например `/bin/bash`
`init=<путь к bash>`

5. Выполните загрузку операционной системы.
6. Можно изменить `ro` на `rw` и сбросить **пароль суперпользователя**

2.3. Создание описания для сервиса

1. Перезагрузите систему в режиме загрузки по-умолчанию
2. Задание выполняется в контексте безопасности суперпользователя
3. Запросите с `github.com` получение дополнительных файлов для проведения лабораторных занятий

```
git clone https://github.com/hse-labs/linux-lf.git
```
4. При необходимости выполните установку пакета `git-core` средствами `apt-get`

```
apt-get install git-core
```
5. Ознакомьтесь с содержимым файла `fake.service` (Unit-файл сервиса) в полученном каталоге

```
cat linux-lf/fake.service
```
6. Ознакомьтесь с текущим списком описаний сервисов для SystemD

```
ls /etc/systemd/system
```

```
ls /lib/systemd/system
```
7. Скопируйте файл `fake.service` с описанием сервиса в каталог `/etc/systemd/system`

```
cp linux-lf/fake.service /etc/systemd/system/
```
8. Убедитесь, что описание сервиса скопировано

```
ls /etc/systemd/system/fake.service
```

2.4. Запуск-статус-остановка сервиса

1. Задание выполняется в контексте безопасности суперпользователя
 - Для выполнения дальнейших действий используйте утилиту `systemctl`
2. Выполните запуск созданного сервиса `fake.service`
3. Посмотрите статус работы сервиса
4. Перезапустите созданный сервис
5. Остановите сервис

6. Настройте сервис на автоматический запуск

2.5. Управление сервисами

1. Задание выполняется в контексте безопасности суперпользователя
 - Для выполнения дальнейших действий (п.2,4,6,7,8,10) используйте утилиту *systemctl*
2. Уберите сервис **CUPS (cups.service)** из автозагрузки.
3. Перезагрузитесь в третий уровень загрузки.
`telinit 3`
4. Проверьте, запущен ли **CUPS**.
5. Залогиньтесь пользователем `sysadmin` и перейдите в пятый уровень загрузки.
`telinit 5`
6. Проверьте, запущен ли **CUPS**.
7. Запустите **CUPS**.
8. Верните **CUPS** в автозагрузку.
9. Перезагрузитесь в пятый уровень загрузки
`telinit 5`
10. Проверьте, запущен ли **CUPS**.

2.6. Журнал событий journald

1. Задание выполняется в контексте безопасности суперпользователя
2. Ознакомьтесь с настройками системы журнализации `journald`
`cat /etc/systemd/journald.conf`
3. Обратите внимание на значение параметров
 - `Storage`
 - `ForwardToSyslog`
4. Ознакомьтесь с расположением файлов журналов `journald`
 - в каталоге на диске `/var/log/journal` (если есть)
 - в каталоге в RAM-диске `/run/log/journal` (если есть)

2.7. Запросы средствами `journalctl`

1. Задание выполняется в контексте безопасности суперпользователя
2. Ознакомьтесь со встроенной справкой утилиты `journalctl`

```
man journalctl
journalctl --help
```


3. Выполните вывод событий приоритета emerg

```
journalctl -p emerg
```

4. Выполните вывод событий ядра

```
journalctl -k
```

5. Выполните вывод событий службы sshd

```
journalctl $(which sshd)
```

6. Выполните вывод событий пользователя sysadmin

```
journalctl _UID=$(id -u sysadmin)
```

7. Выполните вывод событий с момента загрузки системы

```
journalctl -b
```

8. Выполните вывод последних сообщений с возможной расшифровкой

```
journalctl -xe
```