Администрирование Linux

Главная цель: использование терминала, без использования GUI

Осмотреться в системе

```
pwd #текущий рабочий каталог
```

```
whoami #имя пользователя
```

```
groups #покажет группы
```

Основы командной строки

Командную оболочку не редко называют **реплом** (**REPL**,**R**ead-**E**val-**P**rint-**L**oop)

1. Read - shell ждет ввода команды от пользователя
2. Eval - shell исполняет введенную команду
3. Print - shell выводит результат
4. Loop - shell возвращается к 1 пункту

Этот процесс называется **интерактивной сессией**

Чтобы узнать какая оболочка используется для терминала

```
echo $SHELL;
```

Директория, в которой находится терминал называется **рабочей** (**working directory**)

команда `pwd` выводит текущую рабочую директорию

`/` является корневой директорией для UNIX-like систем

Вместо обратных слешей `\` используются прямые `/`

`ls` программа, которая выводит список файлов и директорий в рабочей директории.

`cd` программа для перехода по директориям

Для перемещения могут использоваться как **относительный** так и **абсолютный** путь до папки

Абсолютный путь - это полный путь от корня `/`

Относительный путь - берет свое начало от текущей рабочей директории.

Для перехода на уровень выше используется команда `cd ..`

Символ `~` - абсолютный путь домашней директории

Практически у каждой команды есть какие klb,j опции и параметры. О них можно узнать из справки.

Справку по команде можно вызвать `man <имя команды>`

Интересная статья про командную строку

Команда `stat` позволяет изучить файл (покажет свойства файла)

Команда `cat` позволяет вывести соджержимое файла на экран (удобно с маленькими файлами)

Есть 2 команды `head` и `tail` которые показывают первые/последние 10 строк соответственно. Через параметр `-n` можно указать желаемое количество строк. Обычно эти команды используют для чтения логов.

Иерархия файловой системы Linux

Консольная утилита `grep` служит для поиска по файлу или файлам

Потоки

При старте консольной утилиты, обычно, существуют 3 стандартных потока:

1. STDIN (Standard Input)
2. STDOUT (Standard Output)
3. STDERR (Standard Error)

С помощью символа `>` мы можем перенаправлять потоки (STDOUT), например записать вывод программы в файл

```
cat ~/.bashrc > output.txt
```

При этом `>` всегда **перезаписывает** файл, а `>>` **добавляет** в конец файла.

Кроме того, можно работать с STDIN

```
wc -l < result.txt
```

Просим утилиту `wc` посчитать количество строк в файле `result.txt`

А еще можно объеденить все, и получить вывод результата в файл

```
wc -l < result.txt > output.txt
```

Pipeline (Конвейер)

У одного процесса есть вход, у другого выход, и можно передавать информацию друг другу

Символ `|` называется **пайп**, он указывает `shell` взять STDOUT одного процесса и соединяет его с STDIN другого.

```
#читаем файл source
#грепаем по Dog
#убираем дубли
#сортируем
cat source | grep Dog | uniq | sort
```

Переменная окружения ENV

Команда `env` выводит окружение, в котором работает `shell` именно отсюда, он берет знания о расположении домашней директории и тд.

Переменнуы окружения (Например TERM) это конфигурации системы и программ

Особое место занимает переменная PATH где перечисленны директории, содержащие исполняемые файлы.

История команд

Для поиска ранее введенных команд существует утилита `history` которая выводит список ранее введенных команд.

`CTRL + r` запускает реверсивный поиск по истории, повторное нажатие выберет следующее совпадение, если первое не устроило.

Alias

Посмотреть все алиасы в системе:

```
alias
```

Top

Базовая утилита, для просмотра нагрузки на систему

```
top
    -u <process_name> # вывести для определенного процесса
```

Processes

```
Processes: 551 total, 2 running, 549 sleeping, 2714 threads
# running – процессы, выполняющиеся в данный момент;
# sleeping – “спящие” процессы, процесс находится в состоянии ожидания запроса или в готовности;
# zombie – “зомби”-процесс, дочерний процесс, завершивший свой выполнение, но но ещё присутствующий в списке процессов операционной системы, чтобы дать родительскому процессу считать код завершения.
```

Load Average

```
Load Avg: 1.87, 1.73, 1.55 # показывает данные о средней нагрузке за 1 минуту, 5 минут, 15 минут
```

Для оценки утилизации ресурсов используют данные о CPU:

Для 1 ядра :

< 1 - ресурсов в избытке, процессы практически не ждут ресурсов

1 - 3 - нормальная работа

3 -10 - серьезная нагрузка на ресурсы

10 и выше - система тормозит, нужно смотреть, где не хватает ресурсов

20 и более - доступ к системным ресурсам серьезно затруднен

Соответственно, LA для конкретной машины нужно считать исходя из количества ядер:

Для OSX

```
sysctl -a | grep cpu.family # количество ядер
```

Для Linux

```
lscpu | grep family # количество ядер
```

IOTOP

Базовая утилита для мониторинга нагрузки на I/O систему

Cron

Управление происходит утилитой `crontab`

Вызовом этой утилиты можно сделать 4 вещи:

1. отредактировать `crontab -e`
2. показать текущую таблицу задач `crontab -l`
3. удалить таблицу задач `crontab -r`
4. загрузить таблицу задач из файла `crontab /path/to/file.crontab`

При вызове `crontab -e` будет вызван редактор указанный в стандартной переменной окружения `EDITOR`

Сами задачи описаны в следующем формате:

```
# строки-комментарии игнорируются
#
# задача, выполняемая ежеменутно
* * * * * /path/to/exec -a -b -c
# задача, выполняемая на 10 минуте каждого часа
10 * * * * /path/to/exec -a -b -c
# задача, выполняемая на 10 минуте второго часа каждого для и использующая перенаправление стандартного потока вывода
10 2 * * * /path/to/exec -a -b -c > /tmp/cron-job-output.log
```

Первые 5 (пять) полей значений :

Минуты [1 ... 60]

Часы [0 ... 23]

Дни месяца [1 ... 31]

Месяцы [1 ... 12]

Дни недели [0 ... 6] - где 0 - воскресение

Значения можно перечислять через запятую:

```
# задача, выполняемая в первую и десятую минуту каждого часа
1,10 * * * * /path/to/exec -a
```

Или через дефис:

```
# задача, выполняемая в каждую из первыз десяти минут каждого часа
0-9 * * * * /path/to/exec -a
```

Или через косую черту:

```
# задача, выполняемая раз в 10 минут
*/10 * * * * /path/to/exec -a -b
```

Доступы к crontab

Регламентируются двумя файлами cron.allow и cron.deny в которых перечисляются, соответственно, пользователи с доступом и без доступа

Логи крона

Логи пишутся в /var/log/syslog иногда может лежать в /var/log/cron

Модификация файловой системы

`touch` - обычно используют побочный эффект этой утилиты - создание файла

`rm` - удаление

`mv` - перемещает файл (сайд эффект - переименование файла, этим эффектом очень часто пользуются)

`cp` - копирование файла

`mkdir` - создание директории

Пробнее о каждой команде в `man` соответствующей команды

Система прав

```
ls -l # выведет список файлов с правами
-rw-rw-r--@  1 web  staff  93498809 Oct 17  1997 CRITTER.DAT
# формат данных следующий - | --- | --- | --- ,
# где
# - - тип файла
#       - - обычный файл
#       d - каталог
# далее идут блоки
# --- - для владельца файла
# --- - для группы
# --- - для остальных пользователей
```

Значения

Для владельца, группы и остальных пользователей

Возможны следующие варианты:

`r--` - только чтение

`-w-` - права на запись в файл

`--x` -права на исполнение

двоичная	восьмеричная	символьная	права на файл	права на каталог
000	0	`---`	нет	нет
001	1	`--x`	выполнение	чтение файлов и их свойств
010	2	`-w-`	запись	нет
011	3	`-wx`	запись и выполнение	всё, кроме чтения списка файлов
100	4	`r--`	чтение	чтение имён файлов
101	5	`r-x`	чтение и выполнение	доступ на чтение
110	6	`rw-`	чтение и запись	чтение имён файлов
111	7	`rwx`	все права	все права

Выдача и изменение прав

```
chmod 664 <filename> # владелец имеет rw, все остальные только r
```

Утилита `chmod` имеет следующие возможности:

1. указание пользователей:
   u - владелец

g - группы

o - остальные

a - все

2. оператор
   `+` - добавить права
   `-` - забрать права
   `=` - установить права
3. Параметры:
   `-R` - рекурсивно
4. Пример:

```
chmod u=rwx,g=rx,o=rx filename # установить права 755
```

CRON

Посмотреть текущее расписание в системе:

```
crontab -l
```

Как редактировать:

```
crontab -e
```

Запись выглядит следующим образом

```
 минута час день_месяца месяц день_недели <команда>
```

Значение	Диапазон	Дополнительно
минуты	0-59	
часы	0-23	
дни месяца	1-31	
месяцы	1-12	можно задавать и в 3-х буквенном варианте
дни недели	0-6	можно задавать и в 3-х буквенном варианте (0=воскресенье)

Символ '*' подразумевает - любое значение.

Минимальное время 1-а минута. Это связано с тем что cron каждую минуту просматривает список заданий, и ищет которые нужно выполнить.

Дни недели и месяца в трех буквенном варианте:
sun mon tue wed thu fri sat
jan feb mar apr may jun jul aug sep oct nov dec

Дополнительные переменные cron

Переменная	Описание	Эквивалент
@reboot	Запуск при загрузке	
@yearly	Раз в год	0 0 1 1 *
@annually	Тоже что и @yearly	
@monthly	Раз в месяц	0 0 1 * *
@weekly	Раз в неделю	0 0 * * 0
@daily	Раз в день	0 0 * * *
@midnight	В полночь (00:00)	
@hourly	Каждый час	0 * * * *

Модификаторы (могут быть применены для минут часов дней_месяцев месяцев дней__недели)

Символ	Значение
*	Любое (каждый минимальный шаг - минимальный шаг 1 (минута))
,	Перечисление (1,2,3,4,5) 1,2,3,4,5 (минуту)
-	Диапазон (4 - 7) каждый раз с 4 до 7
/	Каждый n шаг (*/4) - каждый четвертый

Онлайн редактор

https://crontab.guru

Утилита grep

```
git status -s | grep -i "CORE-380" # фильтрует переданную информацию по заданному шаблону "Regexp"
             -i # регистронезависимо
             -o # вернуть только результат совпадения
```

Опции: **Синтаксис**:

Утилита awk

Утилита sort

Утилита uniq

Утилита netstat

Утилита column

Утилита sed

Синтаксис:

```
sed опции -e команды файл
```

Основные опции: -n, --quiet - не выводить содержимое буффера после редактирования -e - команды, которые надо выполнить для редактирования

Отличия в Mac OS