Linux для спешащих

с примерами на NodeJS и Bash

MtJS # 3

Немного истории

1969 – В Bell Labs (AT&T) разработана ОС UNIX

1983 – Ричард Столлман стартует проект GNU (Свободный UNIX) и создает лицензию GPL.

1987 – Эндрю Танненбаум создал учебную ОС MINIX, ориентированную на 286-й процессор

1991 – Линус Торвальдс начинает разработку Linux (GPL) для 386(486) компьютеров, будучи под впечатлением от MINIX.

1994 – Выходит первая стабильная версия Linux 1.0

Файловые системы

1. В Linux всё файл. Папки, ссылки, устройства, сокеты, процессы...
2. Под "файлом" подразумевается поток данных.
3. Каджому файлу можно задать разрешения на чтение/запись/исполнение.
4. Разрешения устанавливаются для владельца, группы пользователей и остальных пользователей
5. Как правило Linux использует несколько файловых систем одновременно (**procfs, tmpfs, extfs**)

Основные типы файлов

```
ls -l ...
```

Тип файла определяется первым символом

```
-rw-r--r-- index.js                          Обычный файл
drwxr-xr-x /var/powermanagement              Директория
crw------- /dev/null                         Символьное устройство
brw-rw---- /dev/sda                          Блочное устройство ("HDD, ...")
prw-rw---- mypipe                            Именованный канал
lrwxrwxrwx symlink -> /root/original_file    Символическая ссылка
srwxrwxrwx /var/run/mysql.sock               Сокет
```

Файловые дескрипторы

- Процессы работают с файлами посредством файловых дескрипторов **(fd)**.
- Явно **fd** используется в `fs.open, fs.read, fs.write, ...`

```
fs.open("/open/some/file.txt", 'r', (err, fd) => { // Получаем дескриптор файла
  fs.read(fd, ...) // fd например равен 50
});
```

- Также ОС выделяет для каждого процесса файловые дескрипторы fd (0, 1, 2) для стандартных потоков ввода-вывода

Коммуникации с процессами

Основные способы обмена данными между процессами:

- Запись/чтение файлов в т.ч. **стандартные потоки ввода-вывода**
- **Коды возврата** процессов
- **Сигналы** UNIX

Стандартные потоки ввода-вывода

Это потоки данных зарезервированные системой для "общения" процесса с внешним миром.

STD**IN** - стандартный поток *ввода данных*. По умолчанию нацелен на клавиатуру. (**fd 0**)

STD**OUT** - стандартный поток *вывода данных*. По умолчанию нацелен на вывод в терминал. (**fd 1**)

STD**ERR** - стандартный поток *ошибок*. По умолчанию также выводится в терминал. (**fd 2**)

STDIN

Позволяет передать данные *в процесс* с устройства ввода.

```
node
> const fs = require('fs');
```

STDOUT

Позволяет *процессу* передать данные наружу.

```
# echo использует stdout для вывода строки
echo "Привет, Мир!";

# cat считывает файлы и последовательно выводит их в stdout
cat 1.txt 2.txt 3.txt
```

```
// server.js
...
http.createServer(function(req, res) {
  console.log(req.url); # Выводится в STDOUT
}).listen(8080);
```

STDERR

Позволяет отделить ошибки от основного вывода *процесса*.
По умолчанию терминал отображает STDERR.

```
// err.js
console.error('Будет отправлено в STDERR');
console.log('Будет отправлено в STDOUT');
```

```
node err.js
Будет отправлено в STDERR
Будет отправлено в STDOUT

node err.js 2>/dev/null # Перенаправление STDERR (2) в /dev/null
Будет отправлено в STDOUT
```

Перенаправление ввода-вывода

```
# Символ ">" перенаправляет STDOUT или STDERR, перезаписывая целевой файл
echo "Hello, World!" > hello.txt

# Символ ">>" дополняет файл. Теперь в hello.txt приветствие на английском и русском
echo "Привет, мир" >> hello.txt

# Избавляемся от вывода ошибок (fd 2)
find ./public -name "*.bundle.js" -print  -delete 2>/dev/null

# Перенаправление ввода из файла с помощью символа "<"
mysql < dump.sql

# Символ "|" (канал) соединяет STDOUT предыдущей команды с STDIN следующей
curl -s 'https://example.com/libs/some_lib.tar.gz' | gzip -d | tar xvf -
```

Коды возврата процессов (команд)

Код возврата позволяет окружению понять завершился ли процесс успешно, либо произошла ошибка.

- 0 – успешное завершение (true)
- 1-255 – завершение с кодом ошибки (false)

```
# Схематичное изображение файла .git/hooks/pre-commit
./node_modules/pre-commit/hook
RESULT=$?  # Код возврата предыдущей команды
[ $RESULT -ne 0 ] && exit 1
exit 0

# Также коды возврата позволяют объединять команды в цепочки
test -f ./tmp/pids/server.pid && ( cat ./tmp/pids/server.pid | xargs kill ) || /usr/bin/true
```

Сигналы UNIX

Пользователь или другой процесс могут послать **сигнал** текущему процессу.
В UNIX сигналы посылаются с помощью команды `kill`

```
// Если запустить приложение и закрыть терминал, процесс получит сигнал SIGHUP и завершится
process.on('SIGHUP', () => {
  console.log('Переопределили поведение по умолчанию и не даем завершить процесс');
});
// делаем долгую задачу ...
```

```
// ... worker.js
// В PM2 мастер-процесс при завершении посылает SIGINT чтобы воркеры смогли безопасно завершить работу
process.on('SIGINT', function() {
  db.close();
});
```

Некоторые типы сигналов

ID	Сигнал	Действие по умолчанию	Описание
1	SIGHUP	Завершение	Закрытие терминала
2	SIGINT	Завершение	Прерывание с терминала (CTRL+C)
9	SIGKILL	Завершение	Безусловное завершение процесса
15	SIGTERM	Завершение	Завершение процесса
18	SIGCHILD	Нет	Дочерний процесс завершен или остановлен

Пользователи и группы

Пользователь – объект, имеющий определенные **права**, на работу с файлами и процессами.

Группа – сущность, позволяющая выдать **права** нескольким пользователям.

```
ls -l /home
drwxr-xr-x 11 admin        admin  352   1 янв  1970 admin
drwxr-xr-x 62 bachinskiyv  admin  1984 23 апр 10:22 bachinskiyv
drwxr-xr-x 34 someuser     users  416   5 апр 12:35 someuser

# Больше данных о пользователях можно получить с помощью cat /etc/passwd
nobody:*:-2:-2:Unprivileged User:/var/empty:/usr/bin/false
root:*:0:0:System Administrator:/var/root:/bin/sh
```

Суперпользователи

root (UserID 0) – пользователь с абсолютными привилегиями.

Выполнять команды от суперпользователя можно с помощью команд

- su
- sudo

sudo более мощная, позволяет сделать тонкую настройку разрешений через `visudo`

Соединение с удаленным сервером

Самый распространенный способ – с помощью команды `SSH`

```
# Secure SHell
ssh root@192.168.1.1 # Подключаемся к серверу 192.168.1.1 как root-пользователь
# При первом подключении ssh сохраняет адрес и публичный ключ сервера
# на вашей машине в ~/.ssh/known_hosts
# При последущих подключениях информация будет проверяться
```

Два самых распространненых типа аутентификации:

- По логину и паролю
- По публичному ключу

Аутентификация по публичному ключу

Основные принципы:

- Используется ваша пары ключей: Приватный -> Публичный
- Приватный `~/.ssh/id_rsa` известен только вам
- Публичный `~/.ssh/id_rsa.pub` нужно скопировать на сервер в файл `~/.ssh/authorized_keys`
- При подключении к серверу, сервер шифрует случайную последовательность вашим публичным ключом и расшифровать ее сможете только вы, *приватным ключом*
- Отправляете данную последовательность серверу

Secure CoPy – программа для копирования фалов по сети.

Поддерживает копирование с локального сервера на удаленный, с удаленного на локальный и с удаленного на удаленный

```
scp [...options] source_file_or_dir target_file_or_dir
```

```
# Подключаемся к hostname по порту 1337, под пользователем user и копируем локальный файл
# в файл remote_file.txt в home-директории на сервере
scp -P1337 ~/local/file.txt user@hostname.com:remote_file.txt

# Рекурсивно (-r) копируем файлы папки backup01 на backup.host
# под текущим пользователем в директорию /var/backups/backup01
scp -r backup01 backup.host:/var/backups
```

Справочные команды

`man` – команда Unix, предназначенная для вывода справки по программам

```
man 5 passwd # Выведем информацию о формате passwd из 5-го раздела man
man passwd # Информация о команде passwd расположена на первой странице
```

`info` – гипертекстовый вариант документации. Позволяет ходить по ссылкам в документе и смотреть разделы и смежные документы.

Поиск команд

`which` – отображает полный путь к команде

```
which yarn
/usr/bin/yarn
```

`apropos` – ищет команды подходящие по ключевому слову

```
apropos disk
dmc(1)                   - controls the Disk Mount Conditioner
du(1)                    - display disk usage statistics
fdisk(8)                 - DOS partition maintenance program
```

Источники

История Linux

Unix: Wikipedia

Все является файлом

Аутентификация по ключу

Что записано в файле known_hosts

Брайан Уорд, Внутренее устройство Linux, 2018