Джулия Эванс
@b0rk

# права доступа в Unix

drawings.jvns.ca

Есть три операции, которые вы можете производить с файлом:

read write execute

ls -l file.txt показывает ваши полномочия. Вот как интерпретировать выведенные данные:


Права доступа к файлам составляют 12 бит


У файлов:

- r = read (чтение)
- w = write (запись)
- x = execute (выполнение)

У каталогов это соответственно:

- r = показать содержимое
- w = создавать, менять и удалять файлы и папки
- x = запрашивать базовые утилиты и изменять файлы

110 в бинарной системе исчисления равняется 6

| | rw- | r-- | r-- |
|---|---|---|---|
| = | 110 | 100 | 100 |
| = | 6 | 4 | 4 |

chmod 644 file.txt означает изменение прав доступа на rw-, r--, r--.

Всё просто!

Setuid влияет на исполняемые файлы

```
$ls -l /bin/ping
rws r-x r-x root root
```

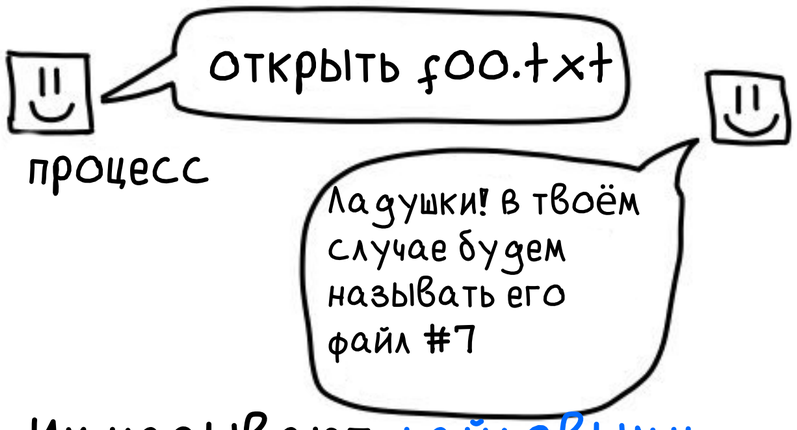
означает, что пинг всегда запускается с правами root

Setgid выполняет три разные, не связанные между собой, действия для исполняемых файлов, каталогов и обычных файлов

Джулия Эванс
@b0rk

# Файловые дескрипторы

Системы Unix используют целые числа, чтобы отслеживать открываемые файлы




Их называют файловыми дескрипторами

lsof (list open files) покажет открытые файлы процесса.

```
$ lsof -p 4242   ← PID, который нас интересует
FD   ИМЯ
 0   /dev/pts/ttY1
 1   /dev/pts/ttY1
 2   pipe: 29174
 3   /home/julia/ofigennaya.txt
 5   /tmp/
```

FD значит файловый дескриптор (file descriptor)

Файловые дескрипторы могут применяться к:

- → файлам на диске
- → программным каналам
- → сокетам (сетевым соединениям)
- → терминалам (например, xterm)
- → устройствам (твоей колонке! /dev/null!)
- → ЕЩЕ МНОГО ЧЕМУ (eventfd, inotify, signalfd, epoll и т.д., и т.д.)


Когда вы открываете или редактируете файл, программный канал или сетевое соединение, вы делаете это, используя файловый дескриптор


Давайте посмотрим под капот. Вот как работает один простой код на Python:

Python:

```
f = open ("file.txt")
f.read lines ()
```

За кадром:


Каждый процесс (нуууу... почти каждый) обладает тремя стандартными файловыми дескрипторами:

| | |
|---|---|
| stdin | → 0 |
| stdout | → 1 |
| stderr | → 2 |

"read from stdin" означает "считать данные из файлового дескриптора 0"

может также являться программным каналом, файлом или терминалом

# Потрясающий каталог: /proc

Джулия Эванс
@b0rk

У каждого процесса на Linux есть PID (Process Identification Number, идентификационный номер процесса), например, 42.

В /proc/42, находится ОЧЕНЬ МНОГО полезной информации о процессе 42

## /proc/PID/cmdline

аргументы командной строки, с которыми был запущен процесс

## /proc/PID/exe

символическая ссылка на бинарник процесса.
МАГИЯ: ссылка сработает, даже если бинарник был удалён!

## /proc/PID/environ

все переменные окружения для процесса

## /proc/PID/status

Программа запущена или приостановлена? Сколько памяти она использует? И много другой полезной информации!

## /proc/PID/fd

каталог со всеми файлами, которые открывались процессом! Запустите `$ ls -l /proc/42/fd`, чтобы увидеть все файлы для процесса 42.

Эти симлинки тоже заряжены волшебством, поэтому их можно использовать для восстановления удалённых файлов ♥

## /proc/PID/stack

текущий стек ядра. Бывает полезно, если процесс завис в системном вызове

## /proc/PID/maps

карты распределения памяти процесса. Библиотеки общего пользования, динамическая память, анонимные карты и т.д.

и ещё МНОГО всего интересного

загляните в

# man proc

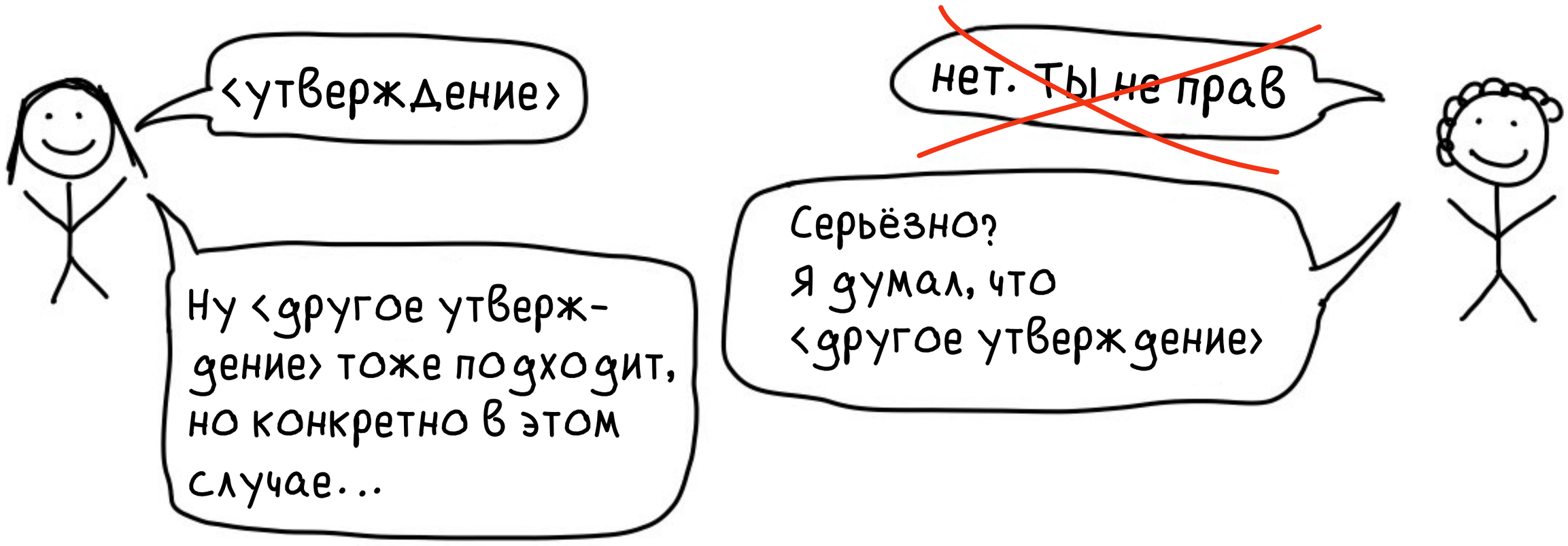
и узнаете больше!

# Как ★эффективно★ общаться, когда ты не согласен

Джулия Эванс
@b0rk


Лайфхак: ♥ прояви любопытство ♥

Когда я с чем-то не согласна, я спрашиваю собеседника о его собственном опыте! Часто оказывается, что он делал что-то такое, что я не пробовала (а бывает и наоборот). В таком случае иногда получается узнать что-то новое



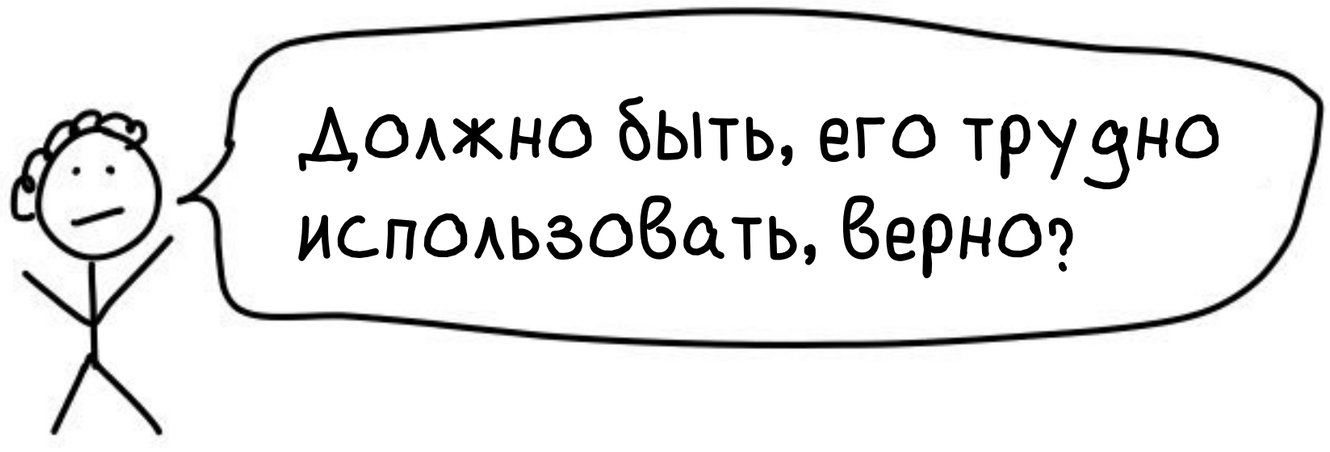

Лайфхак: постарайтесь не говорить, что собеседник неправ. Укажите на то, что неверны его убеждения. НЕТ: «ты не прав» ДА: «это неправильно». Мы все стремимся к одному — к истине! ♥

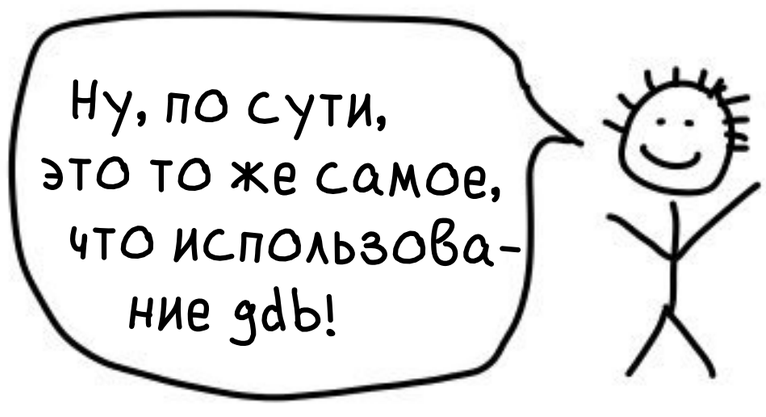
Джулия Эванс
@b0rk

# ★ rr ★

rr-project.org

## RR — отладчик, который позволит тебе вернуться в прошлое!




Вот как его применять:

```
$ rr record /your/application --args
...
Что-то пошло не так!!
Давай посмотрим, что именно...

$ rr replay
GNU gdb (gdb) ...
```

В итоге имеем сессию gdb, содержащую:

- ★ Те же самые результаты системных вызовов
- ★ Те же самые адресные пространства
- ★ прошлые версии многих gdb-команд

reverse-continue reverse-finish
reverse-next reverse-step

Попробуй использовать это вместо gdb!

Джулия Эванс
@b0rk

# СИГНАЛЫ

drawings.jvns.ca

Если ты когда-нибудь запускал kill значит, ты использовал сигналы


Ядро Linux посылает вашему процессу сигналы во многих ситуациях

- Дочерний процесс завершён
- Программный канал закрыт
- Неверная инструкция
- Время на таймере истекло
- Ошибка сегментации

Вы сами можете посылать сигналы с тем же системным вызовом kill или одной из приводимых ниже команд:

| | | |
|---|---|---|
| SIGINT | ctrl-c | разные уровни «смерти» |
| SIGTERM | kill | |
| SIGKILL | kill -9 | |

SIGHUP kill -HUP

Часто интерпретируется как «перезагрузить конфигурацию» (reload config), например, в nginx

Каждый сигнал выполняет одно из стандартных действий

- игнорировать
- уничтожить процесс
- уничтожить процесс и создать файл с содержимым рабочей памяти процесса
- остановить процесс
- восстановить процесс

Твоя программа может использовать специальные обработчики практически для любого сигнала


исключения:

SIGSTOP и SIGKILL не могут быть проигнорированы

на него кастанули SIGKILL →

Обработка сигналов может стать непростой задачей, потому что в ЛЮБОЙ момент может появиться новый сигнал

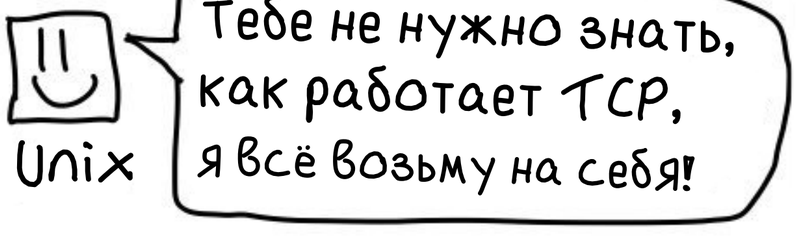
Джулия Эванс
@b0rk

# сокеты

drawings.jvns.ca

## Сетевые протоколы зачастую сложны


Unix системы наделены программным интерфейсом приложения, который называется «Socket API». Он упрощает создание сетевых соединений (на Windows тоже!!!)



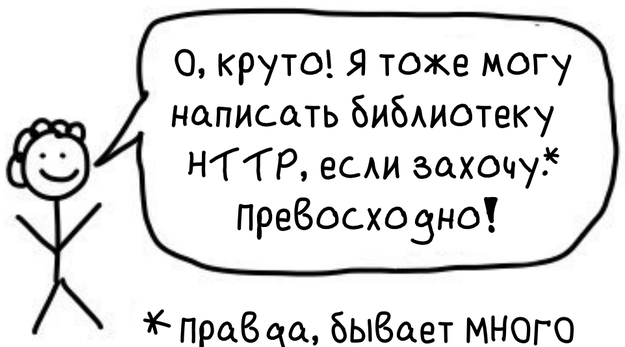

Вот как выглядит процесс загрузки фотографии с котиками при помощи Socket API:

1. Создайте сокет
   `fd=socket(AF_INET, SOCK_STREAM ...`
2. Подключитесь к IP/порту
   `connect(fd, 12.13.14.15:80)`
3. Сделайте запрос
   `write(fd, "GET /cat.png HTTP/1.1 ...`
4. Получите ответ
   `cat_picture=read(fd...`

Если посмотреть под капот любой библиотеки HTTP, можно заметить, что везде используются сокеты.

```
$ curl awesome.com
Python: requests.get("yay.us")
```

сокеты




* правда, бывает МНОГО пограничных случаев!

## AF_INET? Что это?

AF_INET, в общем-то, это и есть «интернет-сокет»: он позволяет устанавливать соединение с другими компьютерами по Интернету по IP'шнику.

Его главной альтернативой является AF_UNIX («сокет домена Unix»), служащий для установления соединения между программами на одном компьютере.

## Три вида Интернет сокетов (AF_INET):

**SOCK_STREAM = TCP**
это использует curl

**SOCK_DGRAM = UDP**
это использует dig (DNS)

**SOCK_RAW** = просто дайте мне отправить IP-пакеты, и я выполню свой собственный протокол
это использует ping

# Сокеты домена Unix

Джулия Эванс
@b0rk

Сокеты домена Unix — это файлы.

```
$ file mysock.sock
 socket
```

Права доступа к файлу определяют, кто может отправлять данные этому сокету.

Они позволяют двум программам на одном компьютере взаимодействовать друг с другом.

Например, сокеты домена Unix используются Docker'ом!


Существует два вида сокетов домена Unix:

**stream** — как TCP! Позволяет отправлять непрерывный поток байтов

**datagram** — как UDP! Отправляет дискретные фрагменты данных.

## Преимущество 1

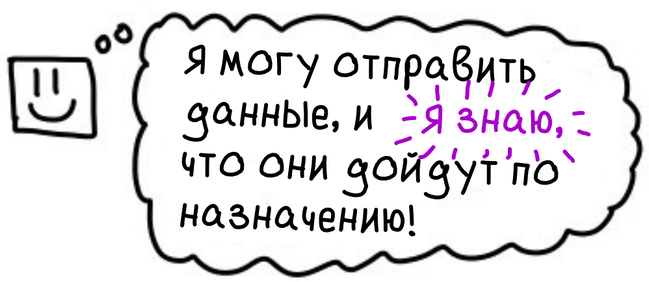
Позволяет тебе работать с правами доступа к файлу, чтобы ограничить доступ к HTTP/базам данных!

```
chmod 600 secret.sock
```

Поэтому Docker использует сокеты домена Unix 🔒


## Преимущество 2

UDP сокеты не всегда надёжны (даже на одном и том же компьютере).
Датаграмный сокет домена Unix всегда надёжен и не подвергается повторному упорядочиванию!




## Преимущество 3

Вы можете отправить файловый дескриптор через сокет домена Unix. Удобно, когда нужно обработать сомнительные входные файлы!

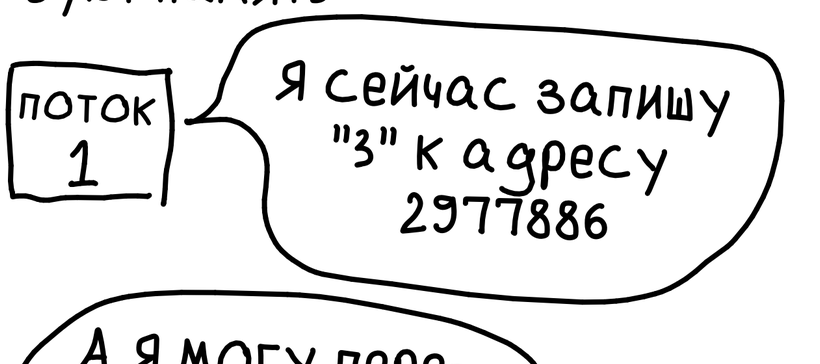
# Что такое программные потоки

Джулия Эванс
@b0rk
drawings.jvns.ca

Каждый процесс имеет множество потоков

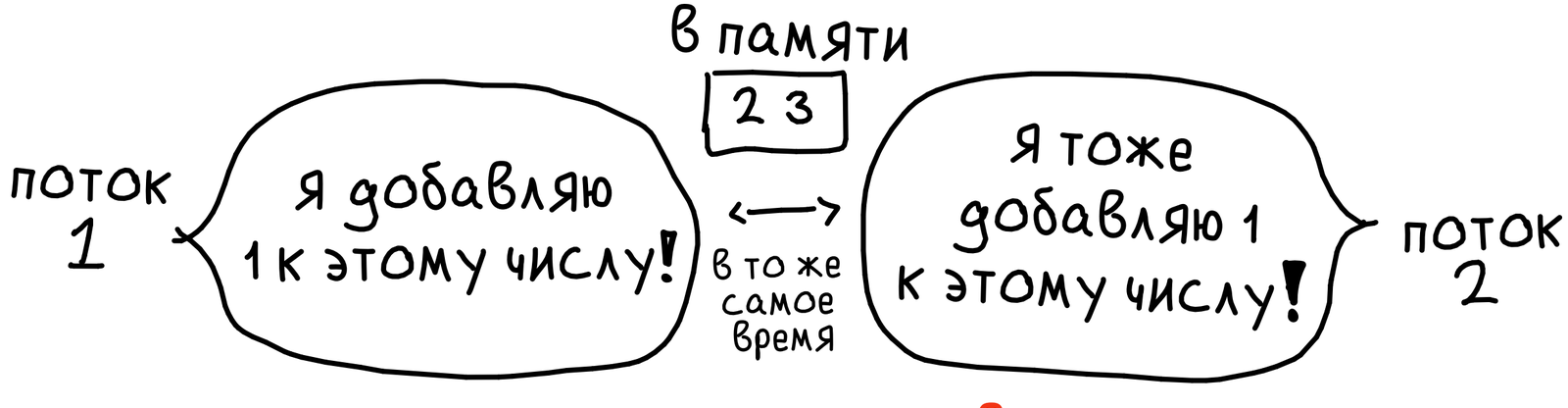
| process id | thread id |
|---|---|
| 1888 | 1888 |
| 1888 | 1892 |
| 1888 | 1893 |
| 1888 | 2007 |

Потоки совместно используют память




Но они могут запускать разный код


Совместное использование памяти может вызвать проблему «состояния гонки»




РЕЗУЛЬТАТ: 24 → Неверно
Должно быть 25

Если у тебя процессор с 8 ядрами, ты можешь запускать код для 8 разных потоков одновременно