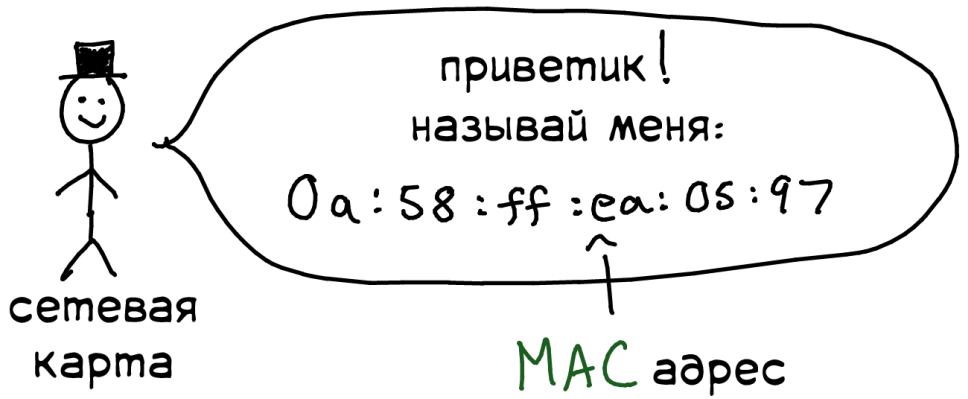
Julia Evans
@b0rk

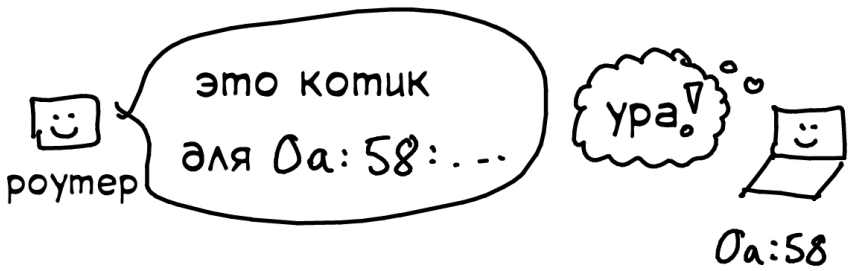
Что такое MAC адрес?

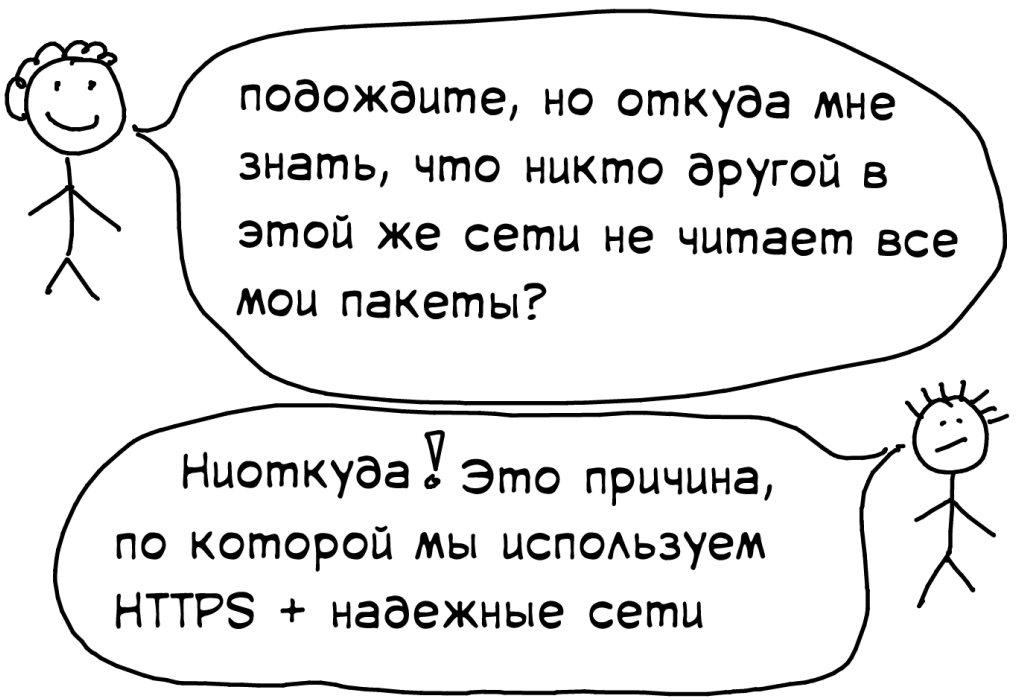
more at: drawings.jvns.ca!

у каждого компьютера есть сетевая карта

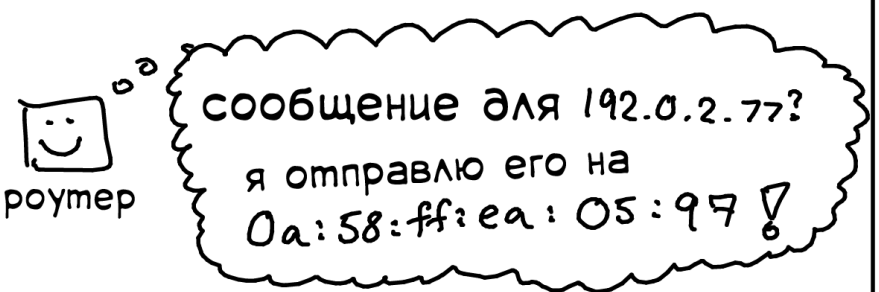


когда вы делаете HTTP запросы через Ethernet/wi-fi каждый пакет отправляется на MAC адрес





в вашем роутере есть таблица, где IP адреса сопоставляются с MAC адресами



(прочитай про ARP (протокол определения адреса))

more at drawings.jvns.ca

✯ Стек ✯

Julia Evans
@b0rk

(в программе на C)

У вашей программы есть

-> локальные переменные

```
int x = 2;
```

-> функция для возврата

```
void parent () {
    do_thing();
}
```

-> аргументы функции

```
make_cat (name, fluffiness)
```

это все живет в части памяти, называемой

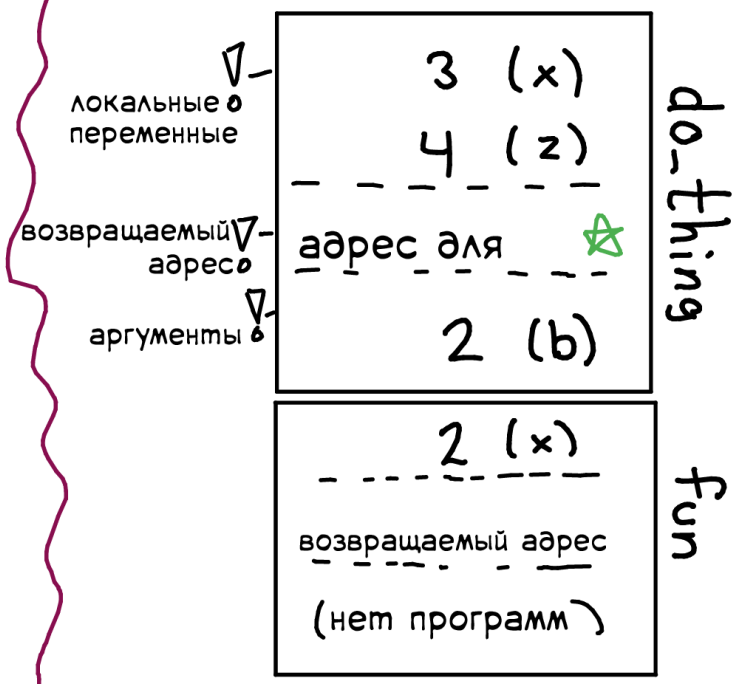
стек

пример программы

```
int fun () {
    int x = 2;
    do_thing(2);
    int y = 4;
}
```

```
void do_thing(b) {
    int x = b+1;
    int z = 4;
    return;
}
```

стек ⇇



есть предел того насколько большим может стать стек

превысьте его и получите

ПЕРЕПОЛНЕНИЕ СТЕКА

Julia Evans
@b0rk

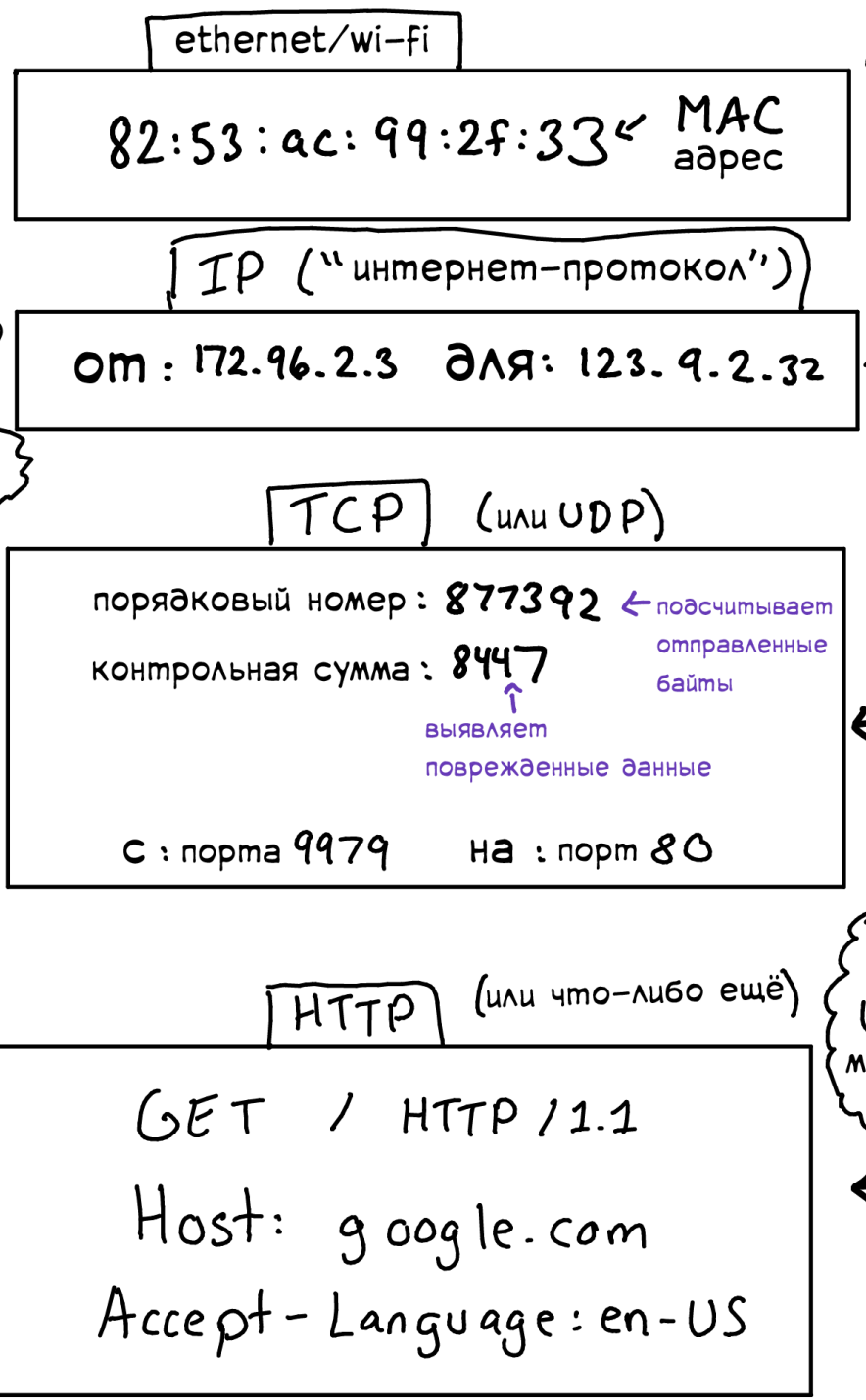
Анатомия пакета

more at drawings.jvns.ca

когда у вас есть страница наподобие Facebook, она попадает на ваш компьютер в виде большого количества маленьких пакетов

давайте посмотрим, как это выглядит!

пакеты разделены на несколько секций (или "заголовков")



"физический уровень". постоянно меняется по мере перемещения пакетов между компьютерами

отвечает за доставку ваших пакетов на правильный сервер (как адрес на конверте)

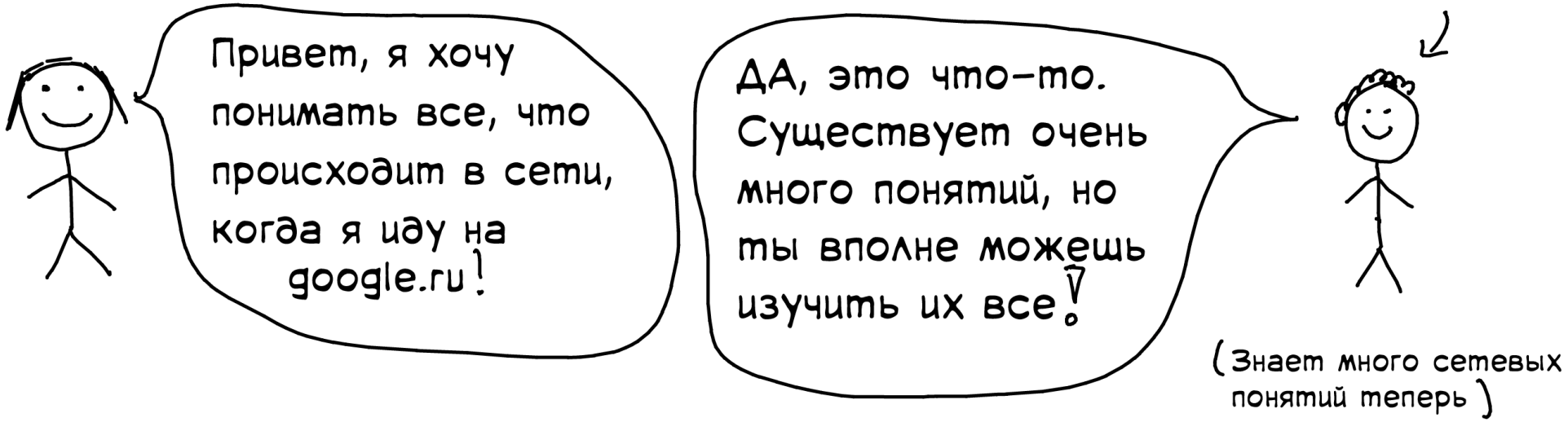
предотвращает повреждение данных и заново отправляет утерянные пакеты

потоковое видео использует UDP. У UDP нет механизмов гарантированной доставки пакетов

фактические данные, которые вы пытаетесь отправить!

Сетевые термины

Julia Evans
@b0rk



протоколы

DNS
SSL/TLS
IP
TCP
UDP
ICMP (ping)
ARP
BGP
ethernet

другие понятия

сокет
пакет
порт
IP адрес
сервер имён
NAT
роутер
TTL
контрольная сумма

+ ещё много чего

здесь много всего для изучения, но вполне возможно понять, как это все взаимодействует, чтобы показать вам котиков ♡

Модель взаимодействия открытых систем OSI для сетей

Julia Evans
@b0rk

На практике редко пригодится, но полезно представлять, что означает «уровень 4»

УРОВНИ

1: электротехнические штуки, провода, частоты, wi-fi

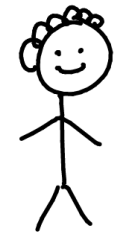
2: Ethernet и другие технологии

3: IP (IP адреса)

4: TCP + UDP (порты)

5+6: Об этих уровнях никто никогда не вспоминает

7: HTTP и все - все - все



Что означает «это прокси 4 уровня»?

если балансировщик помечен «L7» (уровень 7) это обычно означает, что он смотрит на имя хоста: заголовок внутри ваших HTTP пакетов

уровень 3
сетевые инструменты

↑
игнорирует уровень 4 и выше

Я только знаю IP адреса! Я даже не знаю какой порт, не говоря уже о том, что передают пакеты.

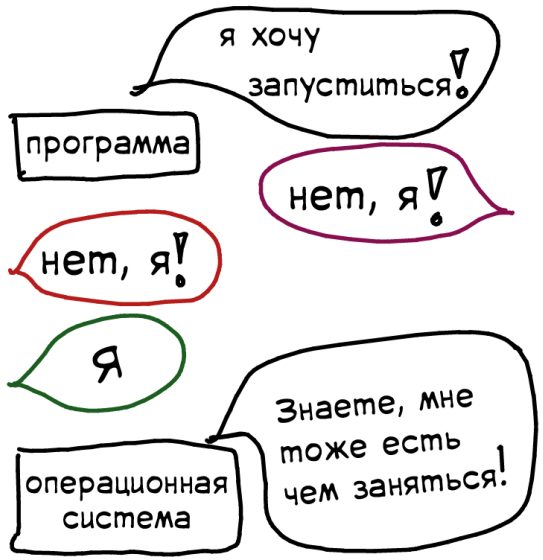
Планирование выполнения задач

drawings.jvns.ca

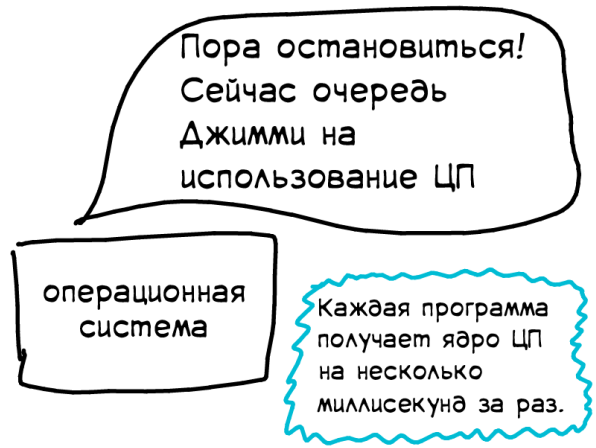
Julia Evans
@b0rk

Однажды...

ваш компьютер сегодня



каждое ядро ЦП может одновременно выполнять только одну программу



шаги для переключения выполняющихся процессов

" переключение контекста "

→ сохранить:

- регистры
- указатели стека
- с каких комана ЦП начать в следующий раз

→ подготовить память для нового процесса

→ загрузить новые регистры и прочее

всё занимает время (2 микросекунды ?).

все это хорошо, но вы же не хотите постоянно переключаться между процессами

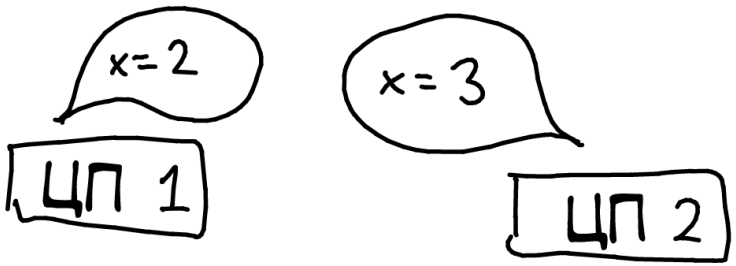
вы не используете ЦП пока ждете

drawings.jvns.ca

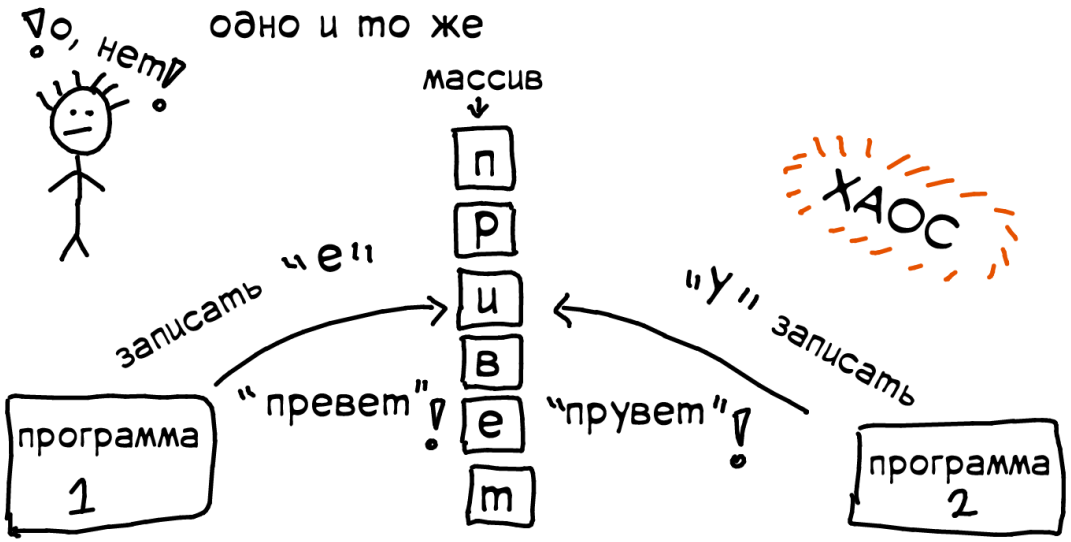
мьютексы

Julia Evans
@b0rk

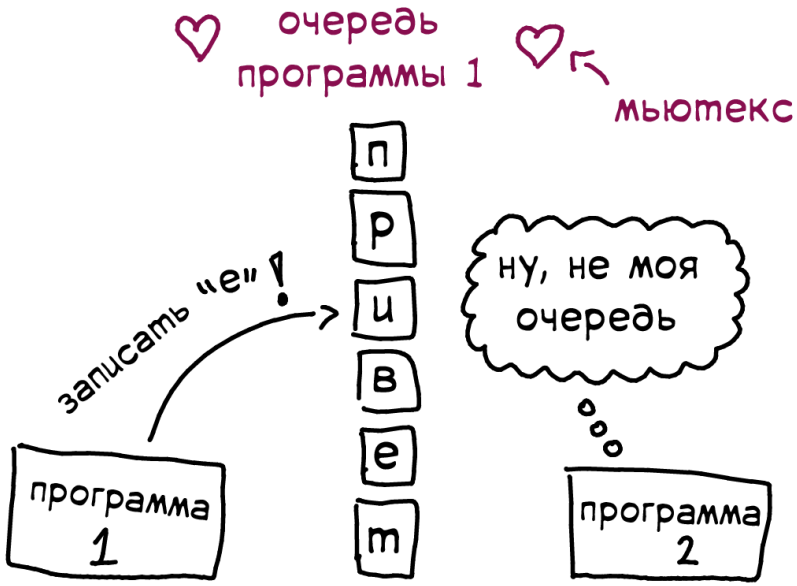
иногда вы одновременно запускаете на выполнение код на 2-х ЦП (центральных процессорах)



иногда 2 потока хотят изменить одно и то же



мьютекс отслеживает, используется ли память программой в данный момент



когда вы закончили, сообщаете мьютексу

есть еще много всего, но место закончилось

Ассемблер

Julia Evans
@b0rk

Мы наслышаны, что компьютер "думает в двоичном коде". Что же это значит??

память вашего компьютера (ОЗУ)

0100100100001010010010

0100 — что-то из этого-картинка котика

что-то из этого-программы

программы являются двоичными

0100101100101011006

Что же ЭТО значит?

это ИНСТРУКЦИИ

у каждого центрального процессора (ЦП) есть набор инструкций (x86 или ARM обычно)

некоторые из инструкций

jmp mov addl xor push inc

инструкции – это числа

инструкция inc ("приращение")

на x86 будет 1000000 или 0x40

транслятор переводит „читаемый для человека" код ассемблера в двоичный

код ассемблера	трансляция	двоичный код
mov $1, %rax		...01001001
mov $1, %rdi		0....
xor %rdi, %rdi		

регистр

Регистр инструкций указывает на адрес в памяти

ЦП → IR (инструкция регистр) 0x5884

Я посмотрю в ОЗУ и выполню код, который там найду!

Пространство пользователя против пространства ядра

Julia Evans
@b0rk

drawings.jvns.ca

ядро Linux содержит **миллионы** строк кода

- читает и записывает файлы
- решает какие программы получают в пользование ЦП
- принимает ввод с клавиатуры

когда работает код ядра Linux, это называется **пространство ядра**

когда работает ваша программа–это **пространство пользователя**

время для **переключения контекста** на пространство ядра

ваша программа переключается туда – обратно

```
str= "my string"
x= x+2
file.write(str)   ← ☆ переключаемся на пространство ядра ☆

y= x+4
str= str*y        ← ☆ и мы вернулись на пространство пользователя! ☆
```

расчет времени вашего процесса

```
$ time find /home

0.15 пользователь   0.73 система
```

↑ время, использованное в вашем процессе

↑ время, использованное ядром для обработки вашего процесса

Julia Evans
@b0rk

Таблица страниц (в 32 битах памяти)

кажый процесс имеет свое пространство памяти

0x aeff3 000

по этому адресу записано «кот»

для меня-там записано «собака»

процесс 1

процесс 2

каждый адрес соотвествует «реальному» адресу в физической памяти (ОЗУ)

процесс 1 0x28ea4000

0x aeff3000 процесс 3 → недействительный!

процесс 2 0x 3942f000

В памяти расположена таблица страниц, которая хранит все соответствия для процессов

0x12345 000 -> 0xae925...
0x 23f 49000 -> 0x 12345...

соответствия – обычно блоки по 4kB (4kB – нормальный размер страницы)

каждое* обращение к памяти использует таблицу страниц

ЦП

мне нужен доступ к 0x ae 923 456

таблица страниц сообщает реальный адрес это 0x 99234456

* ну, почти

когда вы переключаете процессы...

kernel

теперь, используй эту таблицу страниц

хорошо, спасибо!

ЦП

некоторые страницы не соответствуют физическим адресам ОЗУ

процесс

я собираюсь получить доступ к 0x 0004 0000

НЕТ! недопустимый адрес!

ЦП

= ошибка сегментации =

Julia Evans
@b0rk

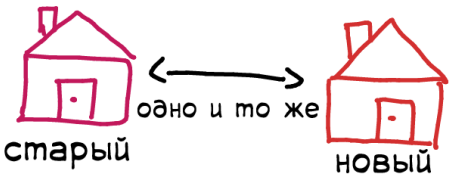
Копирование при записи

drawings.jvns.ca

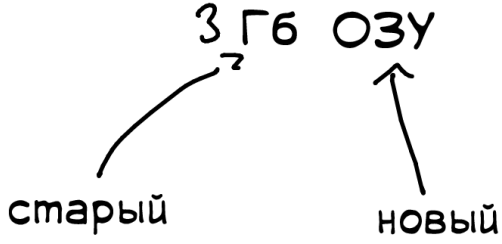
каждый раз, когда вы запускаете дочерний процесс, Linux

выполняет функцию клонирования `fork()`

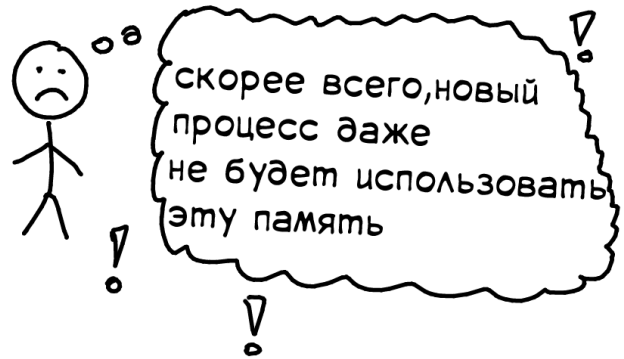
она копирует родительский процесс.



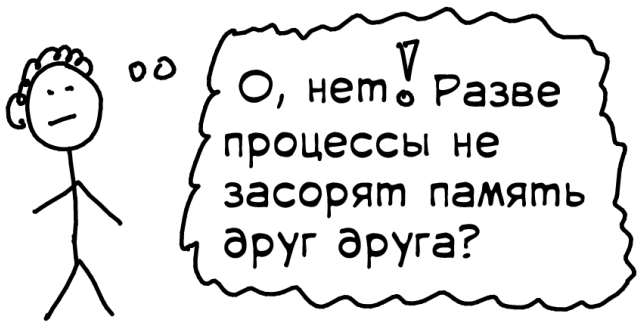
клонированный процесс использует ту же самую память



копировать всю память при каждом fork'е – это медленно, и уж точно напрасная трата места

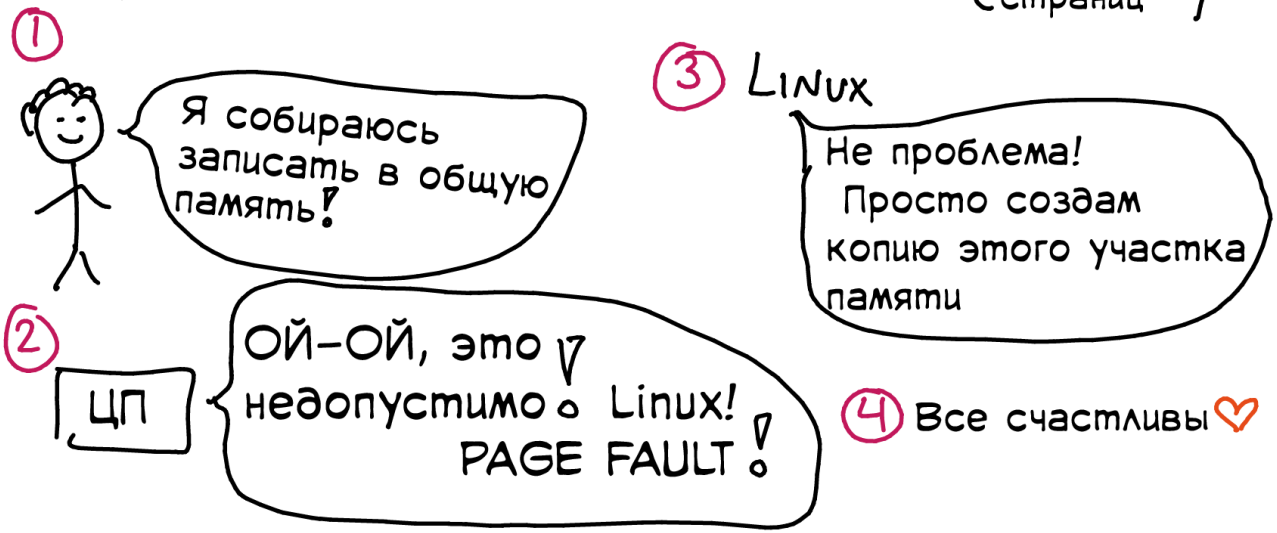


Поэтому Linux позволяет им совместно использовать ОЗУ вместо создания копии



Как мы сделаем так, чтобы это работало?!

Linux помечает всю память для обоих процессов как «только для чтения» (в таблице страниц)



Перевела Команда FirstVDS.ru