Linux

инструменты для отладки которые вы ♡

Маленькая волшебная книга для всех, кто пишет (или запускает!!!) программы под Linux

автор:
Джулия Эванс

привет! Это я:

Джулия Эванс

блог: jvns.ca

twitter:@b0rk

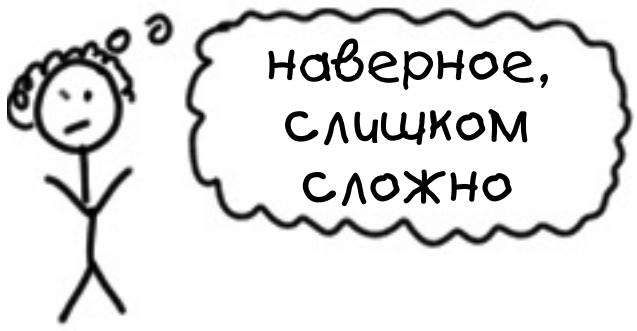
а это журнал, в котором я расскажу

как я стала лучше отлаживать программы

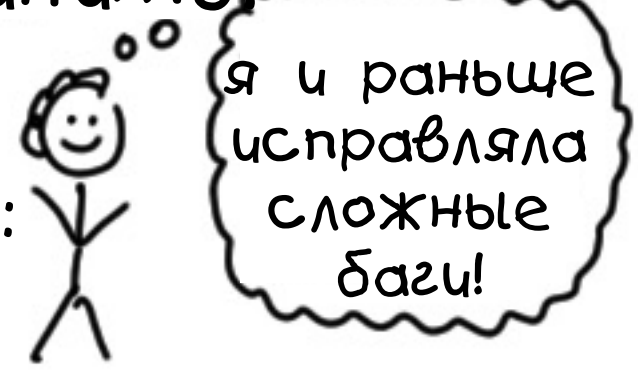
Я стала иначе относиться к 5 вещам:

- Я не забываю о том, что у бага есть логическая причина. Никакой магии.
- Я не сомневаюсь в том, что смогу это починить

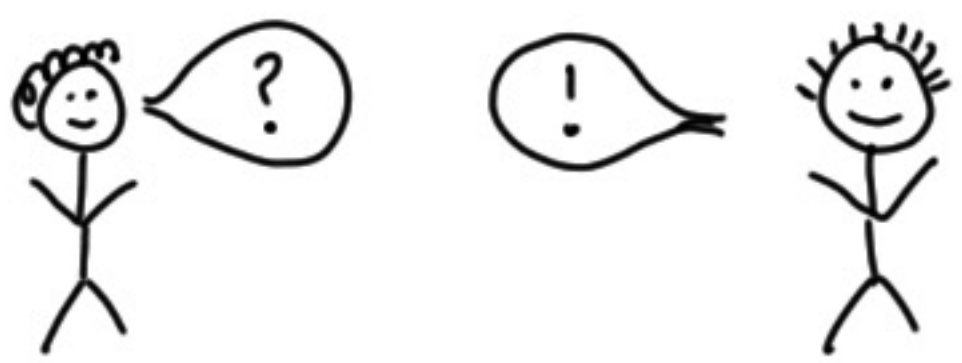
раньше:



сейчас:

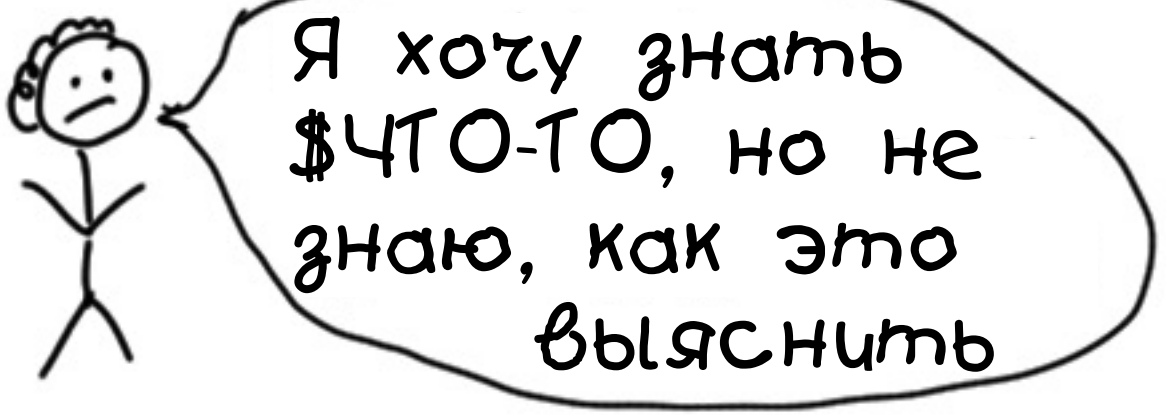


- Советуюсь с другими.



Я хорошо освоила свои инструменты для отладки.

раньше:

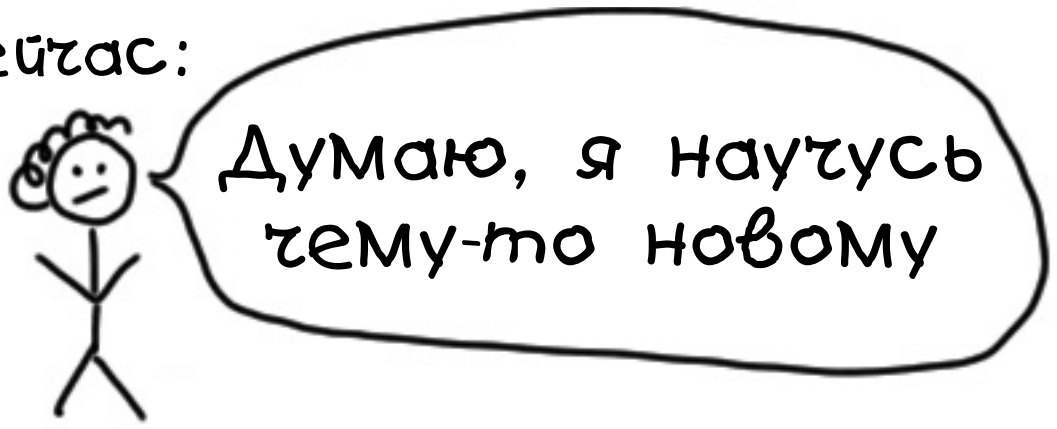


сейчас:

Самое важное: я научилась любить отладку

раньше:

сейчас:



чему научитесь вы

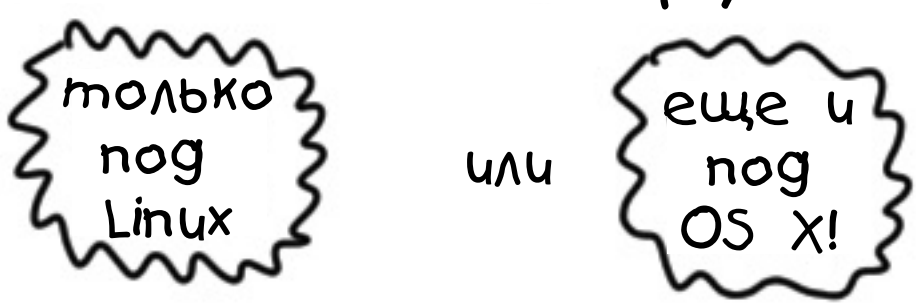
За пару десятков страниц я не смогу придать вам уверенности в себе (Я все равно постараюсь) или научить любить отладку.

Зато я могу показать вам некоторые инструменты для отладки : инструменты, к которым я обращаюсь когда хочу выяснить, что делает моя программа. Надеюсь, в результате вы пополните свой арсенал парой полезных инструментов.

Глава 1: I/O и системные вызовы

Здравствуй, дорогой читатель! Этот журнал содержит три главы про мои любимые инструменты. Я расскажу о пользе каждого и покажу примеры.

Все эти инструменты



Главные вопросы, которые могут у вас возникнуть при отладке сбоящей машины:

- она сейчас пишет или считывает с диска? А из сети?

- программы сейчас считывают файлы? Какие файлы?

Начнем с выяснения того,

чем сейчас занимаются наши программы

и какую нагрузку они создают. поехали!

dstat

только под Linux

Я люблю dstat, потому что он очень простой. Каждую секунду dstat показывает текущую нагрузку на диск и сеть.

Однажды у меня был сервер с базой данных, на котором периодически возникали тормоза. Я начала следить за скоростью работы базы, параллельно наблюдая за показателями dstat.

```
$ dstat
отправлено | получено
           0
          3k
          5k
           0
       300 MB
        48 MB
           0
           0
```

в течение этого периода полный порядок

база начала тормозить

снова наладилось

Совет: флаг -t покажет каждую секунду выводит время

Возможно ли, что входящие по сети 300 Мб, это... запрос базы данных?!

Да !

Эта ЗАМЕЧАТЕЛЬНАЯ подсказка помогла нам выявить проблемный запрос.

strace!!

только под Linux

(у меня на смартфоне наклейка strace)

Strace - мой любимый инструмент. Он показывает все системные вызовы, совершаемые программой. Можно получить полное представление о том, что делает приложение. С помощью Strace я люблю отвечать на вопросы типа «какие файлы сейчас открываются».

```
$ strace python my_program.py
... сотни строк ...
open("/home/bork/.config_file") = 3
read(3, "the contents of the file")
... сотни строк ...
connect(5, "172.217.0.163")
sendto(5, "hi!!")
```

файловый дескриптор

чтение файла!

работа по сети!

Strace может в 50 раз снизить производительность вашей программы. Так что не запускайте его на рабочем сервере.

Я не буду подробно описывать здесь этот инструмент, но у меня есть Журнал, полностью посвященный Strace:

jvns.ca/zines

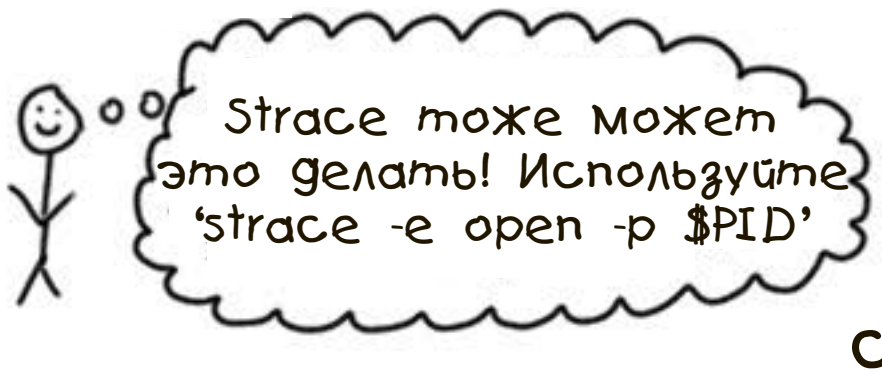
opensnoop !
execsnoop !
eBPF !

еще и под OS X! (типа)

Если выполнить

```
opensnoop -p $PID
```

то будет показывать в реальном времени каждый файл, открываемый программой. Вы можете подумать...



... и будете правы. Но Strace может во много раз снизить скорость вашего приложения.

Opensnoop не снизит.
Execsnoop подскажет, какие программы запускаются.

как его получить

Вам потребуется Ubuntu 16.04+ или ядро версии ~4.4+

Установка:

github.com/iovisor/bcc-tools

На многих серверах сегодня это не будет работать, но держите на заметке! Однажды и ваши серверы будут работать на новом ядре.

как это работает

opensnoop — это скрипт, использующий новое свойство ядра eBPF.

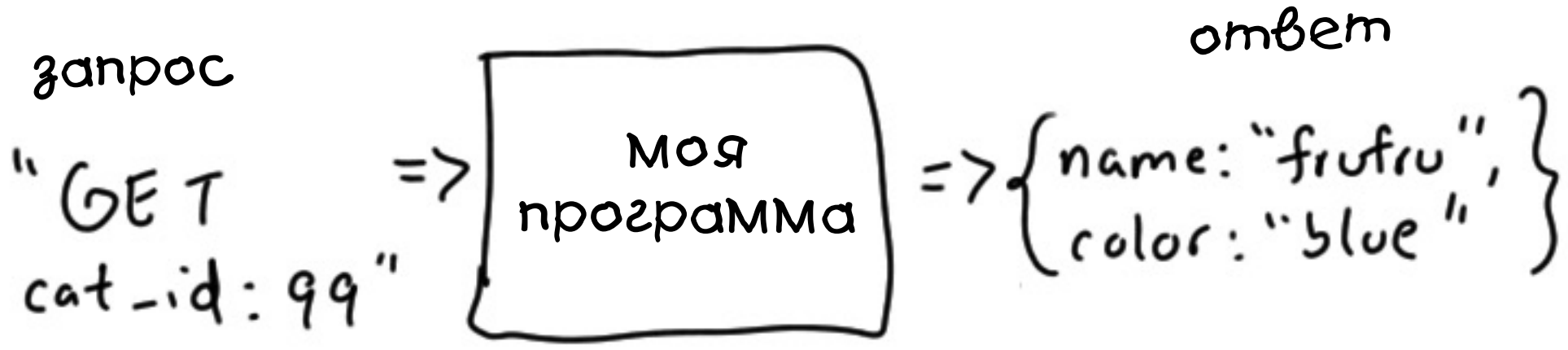
eBPF работает быстро!

Еще есть opensnoop под OS X и BSD!
Он работает на базе DTrace.
eBPF очень мощная вещь, подробности читайте в блоге Бренданa Грегга. (www.brendangregg.com/blog/)

Глава 2 Работа с сетью

В этом журнале я уделила много внимания инструментам для работы с сетью, и хочу объяснить, почему.

Многие программы, с которыми я работаю, взаимодействуют по протоколу HTTP.



Все языки программирования используют единые сетевые протоколы! Так что вне зависимости от того, на каком языке вы пишете, сеть — очень удобное место для получения ответов на вопросы:

- где была ошибка — в запросе или в отклике?
- Мой сервер вообще работает?
- Моя программа тормозит. Кто виноват?

приступим!

netcat

сетевые запросы своими руками

еще и под OS X!

HTTP-запросы, по сути, очень просты — это просто текст!

Чтобы в этом убедиться, давайте сделаем запрос своими руками! Сначала создадим файл:

request.txt

```
GET / HTTP/1.1
Host: ask.metafilter.com
User-Agent: zine
(2 новые строки! важно!!)
```

Далее:

nc означает netcat

```
$ cat request.txt | nc metafilter.com 80
```

В ответ вы получите кучу HTML-кода!

Также можно использовать netcat для быстрой отправки огромных файлов по локальной сети.

шаг 1 (на целевой машине)

```
$ hostname -I
  192.168.2.132 ...

$nc -l 9931 > bigfile
```

(прослушивание порта!)

шаг 2 (на исходной машине)

```
cat bigfile |
nc 192.168.2.132 9931
```

(отправка данных)

☆ netstat ☆

еще и под OS X!

Каждый сетевой запрос отправляется на какой-то порт. Чтобы запрос был получен, порт должен «прослушиваться» программой («сервером»). И можно очень легко выяснить, какие программы какие порты слушают. Это можно сделать с помощью инструмента

☆ «тунец, пожалуйста!» ☆
(«tuna, please!»)

также известного как

`sudo netstat -tunapl`

Вот что он покажет:

```
протокол  локальный адрес      PID / имя программы
tcp       0.0.0.0:5353         2993 / python
                  ↑
                порт
```

Так вот! Я люблю netstat за то, что он говорит мне, какие порты используются процессами.

А под OS X используйте `lsof -i -P`

ngrep

еще и под OS X!

Гренайте свою сеть!

Это мой любимый шпионский сетевой инструмент для начинающих! Запустите:

```
sudo ngrep -d any metafilter
```

Зайдите на http://metafilter.com в своем браузере. Вы увидите, как ngrep выводит соответствующие сетевые пакеты! Мы шпионы :)

Недавно я внесла изменения в клиентскую часть приложения, чтобы она отправляла {"специальный_id": ...} в каждом запросе.

Для проверки ее работы я запустила

```
sudo ngrep специальный_id
```

Выяснилось, что все в порядке :)

tcpdump

еще и под OS X!

Самый сложный сетевой инструмент, описанный в этом журнале, и я не сразу его оценила. С помощью tcpdump я сохраняю трафик, чтобы проанализировать его позднее!

```
sudo tcpdump port 8997
  -w service.pcap
```

! офигенная вещь

"port 8997" это маленькая программа на языке «Berkeley Packet Filter» (BPF) BPF-фильтры могут компилироваться и работают очень быстро !

! "pcap file" ("packet capture") это стандарт хранения сетевого трафика. Его понимают любые инструменты ♥

В каких ситуациях я использую tcpdump:

— Когда я отправляю запрос на машину и хочу знать, получила ли она его
(`tcpdump port 80` покажет все пакеты на порт 80)

— Когда некоторые сетевые подключения оказываются слишком медленными, и я хочу знать, кто виноват — клиент или сервер (понадобится еще и wireshark!)

— Когда нужно просто отобразить и проанализировать пакеты (tcpdump -A)

wireshark

еще и под OS X!

Wireshark — это крутая утилита для сетевого анализа с графическим интерфейсом. Маленькое упражнение для ее изучения! Запустите:

```
sudo tcpdump port 80 -w http.pcap
```

пока выполняется, откройте в браузере metafilter.com (или jvns.ca!). Затем нажмите Ctrl+C для остановки tcpdump.

Теперь у нас есть pcap!

```
wireshark http.pcap
```

Изучите интерфейс Wireshark!

попробуйте ответить на вопросы:

1. Какие HTTP-заголовки отправил ваш браузер на metafilter.com?

   (подсказка: ищите `фрейм, содержащий GET`)

2. Сколько времени занял самый долгий запрос?

   (подсказка: Statistics -> Conversations)

3. Сколькими пакетами вы обменялись сервером metafilter.com?

   (подсказка: `ip.dst == 54.186.13.33`) — IP из 'ping metafilter.com'

только под Linux

Глава 3: CPU + perf

Ваши программы тратят много времени CPU! Миллиарды циклов. Что они ДЕЛАЮТ?!

В этой главе с помощью perf мы ответим на этот вопрос.

Perf есть только под Linux, это очень полезный инструмент, незаслуженно обделенный вниманием.

(в этом журнале я хочу осветить инструменты, которые считаю недооцененными ♡♡♡)

У меня не хватило места в журнале, чтобы рассмотреть эти инструменты, но я хочу хотя бы упомянуть о них :

- valgrind (отладчик использования памяти)
- фантастические инструменты из Java-экосистемы, которых, вероятно не хватает в вашем языке (jstack, VisualVM, Mission Control, YourKit)
- ftrace (для решения проблем с производительностью ядра Linux)
- eBPF

♡ perf ♡

perf нельзя назвать простым или элегантным. Это какой-то странный «швейцарский нож», умеющий делать много полезных вещей.

Во-первых, это профилировщик

попробуйте выполнить:

```
$ sudo perf record python
```

(через 2 секунды нажмите Ctrl+C)

сохраняет файл perf.data

Каждые несколько миллисекунд будет сохраняться лог работы процесса python. посмотрим на результат:

```
$ sudo perf report
```

Оказалось, что у меня 5% времени тратится на функцию PyDict_GetItem. Круто!

только функции на языке C

работает везде ♡

perf можно установить практически на любые Linux-машины. при этом набор его возможностей будет частично зависеть от версии ядра.

Если вы пишете на Python/Ruby/Java/Node, то наверняка сейчас встревожены. «Я хочу знать, какая функция Ruby, а не функция C, сейчас выполняется!». Я вас понимаю — читайте дальше.

perf для всех

Когда-то у меня был сервер, целиком нагружавший процессор. В течение одной минуты я выяснила, что он выполнял поиск по регулярным выражениям на Ruby. Как мне это удалось?

```
$ sudo perf top
```

process	PID	%	function
ruby	1957	77	match-at

Perf top помогает не всегда. Далеко не всегда. Но зато он прост в использовании, и очень здорово, когда он все-таки помогает.

внутренняя функция поиска регулярных выражений на Ruby

...для Java- и Node-разработчиков!

помните, я говорила, что perf понимает только С-функции? Это не совсем так. Node.js и JVM (Java, Scala, Clojure...) тоже научили perf распознавать свои функции.

node

Используйте опцию командной строки --perf-basic-prof

Java

Найдите на Github 'perf-map-agent' и следуйте инструкциям

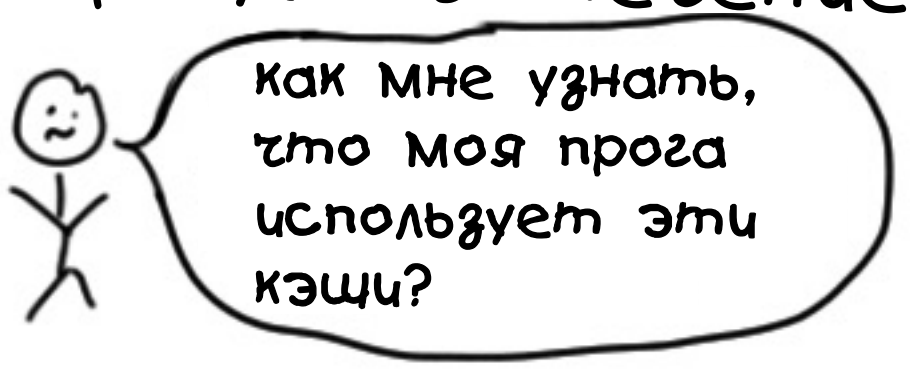
Следите за процессором!

Ваш процессор оснащен маленьким кэшем (L1 кэш), обращение к которому занимает всего 0,5 наносекунды!

в 200 раз быстрее, чем к оперативной памяти!

* совет: гуглите «Latency numbers every programmer should know»!

Имеет смысл использовать кэш проца в том случае, если вам нужно выполнять операцию в течение микросекунд!



как использовать

```
perf stat -e L1-dcache-load-misses ls
```

Будет запущен `ls` и выведен отчет.

как работает

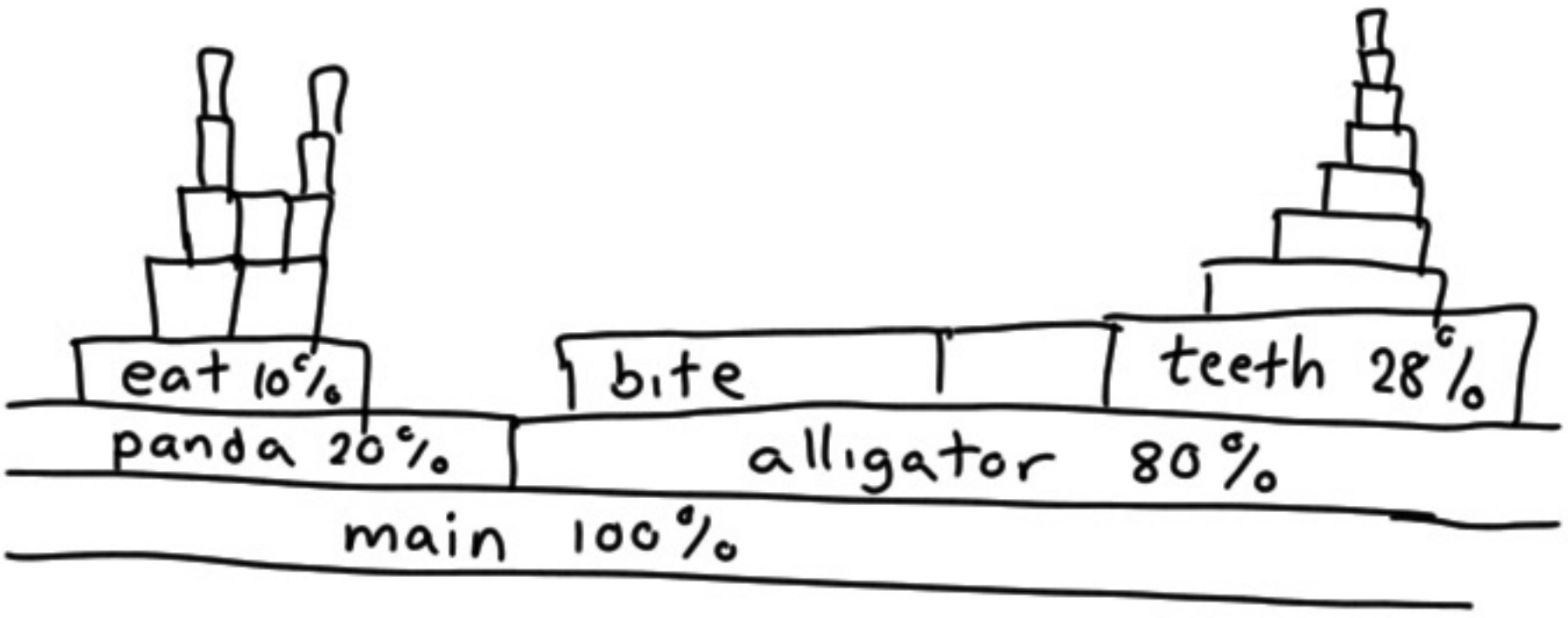
Ваш процессор может хранить все метрики о своей деятельности. `perf stat` просто подсчитывает их и забирает результат.

Не пренебрегайте возможностями «железа»! Если вы изучите его работу и возможности, это может сильно облегчить вам жизнь. ★

flamegraphs

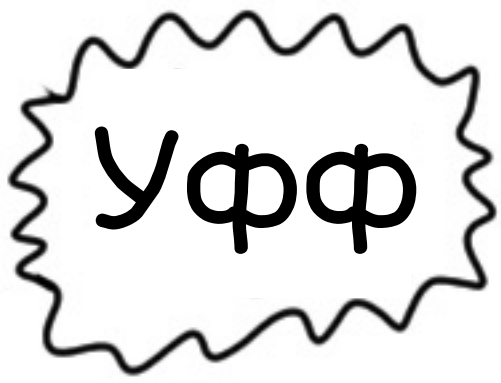
«Огненные графики» - прекрасный способ визуализации производительности проца. Они снискали популярность благодаря инструменту flamegraph.pl Брендана Грегга.

Выглядит «огненный график» примерно так:



Графики строятся по результатам трассировок стека приложения (обычно счет идет на тысячи). График, показанный выше, означает, что 80% стека начинается с "main" и 10% начинается с "main" ... alligator ... panda eat ...

Строить графики можно по записям perf (гуглите «Brendan Gregg flamegraph»). Но это могут делать и многие другие инструменты.



Надеюсь, вы узнали что-то новое для себя.

Спасибо за чтение ♡

Спасибо моему партнеру Камалю за бесконечные рецензии, спасибо Монике Динкулеску (@notwaldorf) за обложку журнала и многое другое.

На моем сайте (jvns.ca) есть еще много всего, как и на brendangregg.com.

Экспериментируйте, повсюду применяйте описанные инструменты. Смотрите, где они могут вам помочь, а где нет. Нужно много практиковаться в работе с этими инструментами, чтобы они действительно начали помогать в отладке реальных проблем. Я изучала их два года, но еще есть чему учиться.

Это действительно увлекательно

понравилось?

Можете распечатать еще!

бесплатно ♡

≡ http://jvns.ca/zines ≡

Перевела Команда FirstVDS.ru