Сетевое программирование

Свинцов Дмитрий

19-06-2019

Оглавление

Глава 1

Содержание

1.1 Введение

См.также:

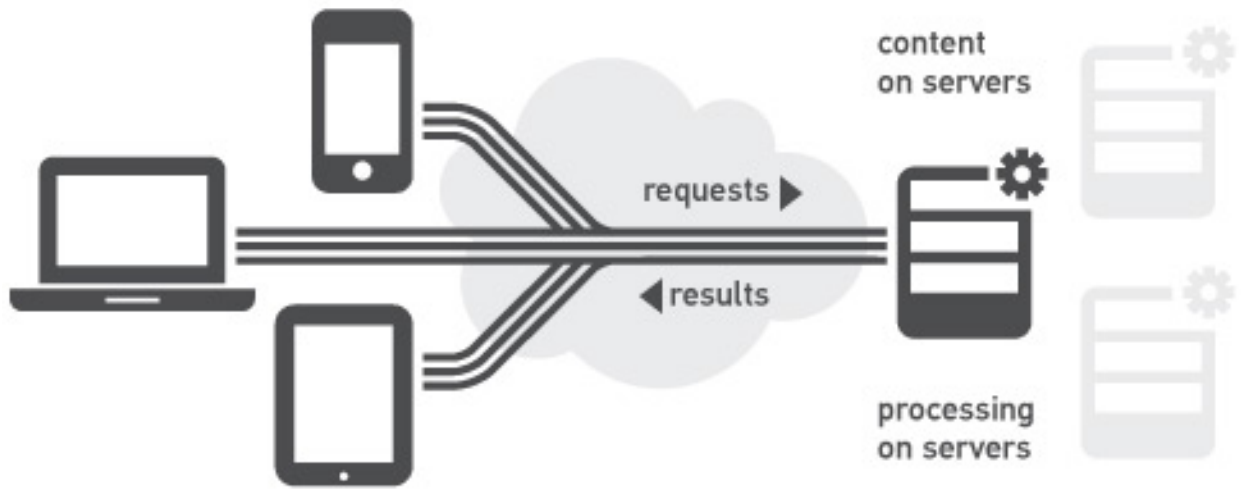
- Wiki Программирование сетевых задач

В области компьютеризации понятие программирования сетевых задач или иначе называемого сетевого программирования (англ. network programming), довольно сильно схожего с понятиями программирование сокетов и клиент-серверное программирование, включает в себя написание компьютерных программ, взаимодействующих с другими программами посредством компьютерной сети.

Программа или процесс, инициирующие установление связи, называются клиентским процессом, а программа, ожидающая инициации связи, называется серверным процессом. Клиентский и серверный процессы вместе образуют распределенную систему. Связь между клиентским и серверным процессами может быть или на основе соединений (как например, TCP-протокол, устанавливающий виртуальное соединение или сессию), или без соединений (на основе UDP-датаграмм).

1.2 Анализ трафика

См.также:



- https://ru.wikipedia.org/wiki/\T2A\CYRA\T2A\cyrn\T2A\cyra\T2A\cyrl\T2A\cyri\T2A\cyrz\T2A\cyra\T2A\cyrt\T2A\cyro\T2A\cyrr_\T2A\cyrt\T2A\cyrr\T2A\cyra\T2A\cyrf\T2A\cyri\T2A\cyrk\T2A\cyra
- Презентация с лекций

`Анализатор трафика`, или `сниффер` (от англ. to sniff — нюхать) — сетевой анализатор трафика, программа или программно-аппаратное устройство, предназначенное для перехвата и последующего анализа, либо только анализа сетевого трафика, предназначенного для других узлов.

`Сниффер` может анализировать только то, что проходит через его сетевую карту. Внутри одного сегмента сети Ethernet все пакеты рассылаются всем машинам, из-за этого возможно перехватывать чужую информацию. Использование коммутаторов (switch, switch-hub) и их грамотная конфигурация уже является защитой от прослушивания. Между сегментами информация передаётся через коммутаторы. Коммутация пакетов — форма передачи, при которой данные, разбитые на отдельные пакеты, могут пересылаться из исходного пункта в пункт назначения разными маршрутами. Так что если кто-то в другом сегменте посылает внутри него какие-либо пакеты, то в ваш сегмент коммутатор эти данные не отправит.

Перехват трафика может осуществляться:

См.также:

- https://ru.wikipedia.org/wiki/Network_tap
- https://ru.wikipedia.org/wiki/MAC-\T2A\cyrs\T2A\cyrp\T2A\cyru\T2A\cyrf\T2A\cyri\T2A\cyrn\T2A\cyrg
- https://ru.wikipedia.org/wiki/\T2A\CYRS\T2A\cyrp\T2A\cyru\T2A\cyrf\T2A\cyri\T2A\cyrn\T2A\cyrg
- обычным «прослушиванием» сетевого интерфейса (метод эффективен при исполь-

зовании в сегменте концентраторов (хабов) вместо коммутаторов (свитчей), в противном случае метод малоэффективен, поскольку на сниффер попадают лишь отдельные фреймы);

- подключением сниффера в разрыв канала;
- ответвлением (программным или аппаратным) трафика и направлением его копии на сниффер (Network tap);
- через анализ побочных электромагнитных излучений и восстановление таким образом прослушиваемого трафика;
- через атаку на канальном (2) (MAC-spoofing) или сетевом (3) уровне (IP-spoofing), приводящую к перенаправлению трафика жертвы или всего трафика сегмента на сниффер с последующим возвращением трафика в надлежащий адрес.

1.2.1 tcpdump

См.также:

- https://ru.wikipedia.org/wiki/Tcpdump

`tcpdump` — утилита UNIX (есть клон для Windows), позволяющая перехватывать и анализировать сетевой трафик, проходящий через компьютер, на котором запущена данная программа.

Основные назначения tcpdump:

- Отладка сетевых приложений.
- Отладка сети и сетевой конфигурации в целом.

Просмотр интерфейсов

```
$ sudo tcpdump -D
1.wlan0 [Up, Running]
2.docker0 [Up, Running]
3.vboxnet0 [Up, Running]
4.vboxnet1 [Up, Running]
5.veth283f985 [Up, Running]
6.any (Pseudo-device that captures on all interfaces) [Up, Running]
7.lo [Up, Running, Loopback]
8.eth0 [Up]
9.bluetooth-monitor (Bluetooth Linux Monitor)
10.nflog (Linux netfilter log (NFLOG) interface)
11.nfqueue (Linux netfilter queue (NFQUEUE) interface)
12.usbmon1 (USB bus number 1)
13.usbmon2 (USB bus number 2)
```

Перехват всех запросов интерфейса

Если tcpdump запустить без параметров, он будет выводить информацию обо всех сетевых пакетах. С помощью параметра -i можно указать сетевой интерфейс, с которого следует принимать данные:

```
$ sudo tcpdump -i 1
```

или

```
$ sudo tcpdump -i wlan0
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on wlan0, link-type EN10MB (Ethernet), capture size 262144 bytes
19:04:24.115872 STP 802.1d, Config, Flags [none], bridge-id 8000.bc:ae:c5:88:91:28.
↪8001, length 35
19:04:24.219665 IP Arkasha-PC.local.bootpc > 255.255.255.255.bootps: BOOTP/DHCP,␣
↪Request from 00:1b:fc:6c:c2:42 (oui Unknown), length 300
19:04:25.118303 IP x220t.local.32371 > google-public-dns-a.google.com.domain:␣
↪29524+ PTR? 255.255.255.255.in-addr.arpa. (46)
19:04:25.186526 IP google-public-dns-a.google.com.domain > x220t.local.32371:␣
↪29524 NXDomain 0/1/0 (114)
19:04:25.287550 IP6 fe80::120b:a9ff:fe0c:f638.mdns > ff02::fb.mdns: 0 PTR (QM)?␣
↪255.255.255.255.in-addr.arpa. (46)
^C19:04:25.287614 IP x220t.local.mdns > 224.0.0.251.mdns: 0 PTR (QM)? 255.255.255.
↪255.in-addr.arpa. (46)

6 packets captured
50 packets received by filter
0 packets dropped by kernel
```

Фильтр запросов по хосту

Чтобы узнать получаемые или отправляемые пакеты от определенного хоста, необходимо его имя или IP-адрес указать после ключевого слова host:

```
$ sudo tcpdump host readthedocs.org
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on wlan0, link-type EN10MB (Ethernet), capture size 262144 bytes
19:08:24.734572 IP x220t.local.44169 > readthedocs.org.http: Flags [S], seq␣
↪1630487586, win 14600, options [mss 1460,sackOK,TS val 281681188 ecr 0,nop,
↪wscale 7], length 0
19:08:24.900671 IP readthedocs.org.http > x220t.local.44169: Flags [S.], seq␣
↪2780774205, ack 1630487587, win 14480, options [mss 1460,sackOK,TS val␣
↪1880995361 ecr 281681188,nop,wscale 9], length 0
19:08:24.900718 IP x220t.local.44169 > readthedocs.org.http: Flags [.], ack 1, win␣
↪115, options [nop,nop,TS val 281681229 ecr 1880995361], length 0
```

(continues on next page)

(продолжение с предыдущей страницы)

```
19:08:24.900812 IP x220t.local.44169 > readthedocs.org.http: Flags [P.], seq 1:733,
↪ ack 1, win 115, options [nop,nop,TS val 281681229 ecr 1880995361], length 732
...
  19:08:28.524595 IP readthedocs.org.https > x220t.local.37282: Flags [.], ack␣
↪2254, win 40, options [nop,nop,TS val 1880996266 ecr 281682094], length 0
19:08:28.605826 IP x220t.local.37282 > readthedocs.org.https: Flags [.], ack 9767,␣
↪win 296, options [nop,nop,TS val 281682155 ecr 1880996287], length 0
^C
83 packets captured
89 packets received by filter
0 packets dropped by kernel
```

Фильтр по протоколу

```
$ sudo tcpdump -n tcp
```

По назначению

Только те пакеты, которые адресованы хосту с IP 192.168.1.101

```
$ sudo tcpdump -n 'src 192.168.1.101'
```

Показывает DNS запросы

```
$ sudo tcpdump -n 'udp and dst port 53'
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on wlan0, link-type EN10MB (Ethernet), capture size 262144 bytes
19:22:52.089174 IP 192.168.1.101.17166 > 8.8.8.8.53: 44241+ A? www.google.ru. (31)
19:22:52.149972 IP 192.168.1.101.61715 > 8.8.8.8.53: 63972+ A? www.google.ru. (31)
19:22:52.157017 IP 192.168.1.101.12023 > 8.8.8.8.53: 17412+ AAAA? www.google.ru.␣
↪(31)
19:22:52.860129 IP 192.168.1.101.1745 > 8.8.8.8.53: 59896+ A? ssl.gstatic.com. (33)
19:22:52.860245 IP 192.168.1.101.4582 > 8.8.8.8.53: 28863+ AAAA? ssl.gstatic.com.␣
↪(33)
19:22:52.860388 IP 192.168.1.101.12181 > 8.8.8.8.53: 46772+ A? ssl.gstatic.com.␣
↪(33)
19:22:53.992159 IP 192.168.1.101.53803 > 8.8.8.8.53: 64496+ A? www.google.ru. (31)
19:22:54.062859 IP 192.168.1.101.30447 > 8.8.8.8.53: 54230+ AAAA? www.google.ru.␣
↪(31)
^C
8 packets captured
10 packets received by filter
0 packets dropped by kernel
```

Пакеты между двумя хостами

Ищем хосты при помощи NetBIOS протокола.

См.также:

- https://ru.wikipedia.org/wiki/NetBIOS

```
$ nbtscan 192.168.1.0/24
Doing NBT name scan for addresses from 192.168.1.0/24

IP address       NetBIOS Name     Server    User             MAC address
-------------------------------------------------------------------------------
192.168.1.0     Sendto failed: Permission denied
192.168.1.101    X220T            <server>  X220T            00:00:00:00:00:00
192.168.1.23                      <server>                   00:00:00:00:00:00
192.168.1.22     ARKASHA-PC       <server>  <unknown>        00:1b:fc:6c:c2:12
192.168.1.255   Sendto failed: Permission denied
```

Или при помощи *nmap*

```
$ nmap -sP 192.168.1.*

Starting Nmap 6.46 ( http://nmap.org ) at 2015-02-02 20:56 YEKT
Nmap scan report for 192.168.1.1
Host is up (0.0068s latency).
Nmap scan report for 192.168.1.20
Host is up (0.018s latency).
Nmap scan report for 192.168.1.21
Host is up (0.016s latency).
Nmap scan report for 192.168.1.22
Host is up (0.028s latency).
Nmap scan report for 192.168.1.24
Host is up (0.017s latency).
Nmap scan report for 192.168.1.26
Host is up (0.032s latency).
Nmap scan report for 192.168.1.28
Host is up (0.0063s latency).
Nmap scan report for 192.168.1.101
Host is up (0.00020s latency).
Nmap done: 256 IP addresses (8 hosts up) scanned in 4.28 seconds
```

Создаем трафик ICMP для хоста 192.168.1.23

```
$ ping 192.168.1.23
PING 192.168.1.23 (192.168.1.23) 56(84) bytes of data.
64 bytes from 192.168.1.23: icmp_seq=1 ttl=64 time=1.90 ms
64 bytes from 192.168.1.23: icmp_seq=2 ttl=64 time=1.27 ms
```

(continues on next page)

(продолжение с предыдущей страницы)

```
64 bytes from 192.168.1.23: icmp_seq=3 ttl=64 time=1.28 ms
64 bytes from 192.168.1.23: icmp_seq=4 ttl=64 time=1.23 ms
^C
--- 192.168.1.23 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3003ms
rtt min/avg/max/mdev = 1.236/1.423/1.900/0.279 ms
```

Смотрим пакеты

```
$ sudo tcpdump 'src 192.168.1.101 and dst 192.168.1.23 and icmp'
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on wlan0, link-type EN10MB (Ethernet), capture size 262144 bytes
19:36:45.340321 IP x220t.local > 192.168.1.23: ICMP echo request, id 10305, seq 1,␣
↪length 64
19:36:46.341472 IP x220t.local > 192.168.1.23: ICMP echo request, id 10305, seq 2,␣
↪length 64
19:36:47.342180 IP x220t.local > 192.168.1.23: ICMP echo request, id 10305, seq 3,␣
↪length 64
19:36:48.343557 IP x220t.local > 192.168.1.23: ICMP echo request, id 10305, seq 4,␣
↪length 64
^C
4 packets captured
4 packets received by filter
0 packets dropped by kernel
```

Без фильтрации, получим все пакеты. Например ARP и NetBIOS.

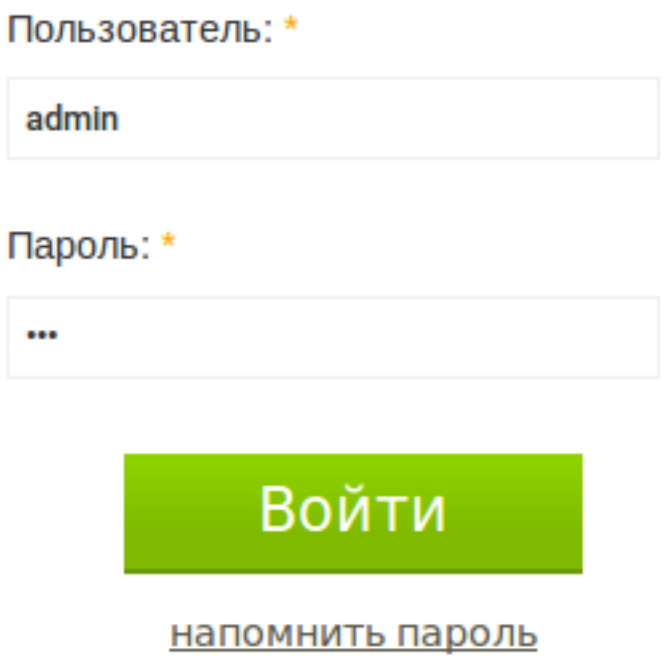
```
$ sudo tcpdump 'src 192.168.1.101 and dst 192.168.1.23'
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on wlan0, link-type EN10MB (Ethernet), capture size 262144 bytes
19:39:50.567837 ARP, Request who-has 192.168.1.23 tell x220t.local, length 28
19:39:50.569144 IP x220t.local.netbios-ns > 192.168.1.23.netbios-ns: NBT UDP␣
↪PACKET(137): QUERY; POSITIVE; RESPONSE; UNICAST
19:39:55.517322 IP x220t.local > 192.168.1.23: ICMP echo request, id 10662, seq 1,␣
↪length 64
19:40:00.533322 ARP, Reply x220t.local is-at 10:0b:a9:0c:f6:38 (oui Unknown),␣
↪length 28
^C
4 packets captured
4 packets received by filter
0 packets dropped by kernel
```

Поиск в трафике

Ответы со статусом 200. Флаг `-A` позволяет увидеть содержимое пакетов. Флаг `-X` отображает содержимов в виде *HEX* таблицы.

```
$ sudo tcpdump -n -A | grep -e '200 OK'
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on wlan0, link-type EN10MB (Ethernet), capture size 262144 bytes
A).)...sHTTP/1.1 200 OK
A).9...vHTTP/1.1 200 OK
^C508 packets captured
508 packets received by filter
0 packets dropped by kernel
```

Поиск паролей в трафике если он не использует шифрование. Например если ввести логин и пароль в HTML форме.



HTTP (http://httpbin.org/post)

HTTPS (https://httpbin.org/post)

```
$ sudo tcpdump -l -A -i lo | egrep -i
↪'pass=|pwd=|log=|login=|user=|username=|pw=|passw=|passwd=|password=|pass:|user:|username:|pas
↪|user ' --color=auto --line-buffered -B20
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
```

(continues on next page)

(продолжение с предыдущей страницы)

```
listening on lo, link-type EN10MB (Ethernet), capture size 262144 bytes
19:54:00.745538 IP localhost.6543 > localhost.58721: Flags [S.], seq 2085108079,␣
↪ack 4286254343, win 43690, options [mss 65495,sackOK,TS val 282365190 ecr␣
↪282365190,nop,wscale 7], length 0
E..<..@.@.<............a|H9o.{.......0........
............
19:54:00.745556 IP localhost.58721 > localhost.6543: Flags [.], ack 1, win 342,␣
↪options [nop,nop,TS val 282365190 ecr 282365190], length 0
E..4..@.@............a...{..|H9p...V.(.....
........
19:54:00.745694 IP localhost.58721 > localhost.6543: Flags [P.], seq 1:708, ack 1,␣
↪win 342, options [nop,nop,TS val 282365190 ecr 282365190], length 707
E.....@.@............a...{..|H9p...V.......
........POST /sign_in HTTP/1.1
Host: localhost:6543
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:35.0) Gecko/20100101␣
↪Firefox/35.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://localhost:6543/login/
Cookie: csrftoken=pVVycxJs2YaTCS5vpKTob0TINGsKjAM4; _LOCALE_=ru; _ga=GA1.1.
↪1951453052.1420403120; connect.sid=s%3AnGU-04XqEDWudttY3CHI3LdUmEr__MYG.
↪GF2fEjoSwB0bC99vfK%2FibenygTjwjRPLto948y7FSwU; beaker.session.
↪id=27aa2050fff646b5bfe5cce56dae1472
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 53

came_from=%2F&login=admin&password=123&submit=Sign+In
^C111 packets captured
222 packets received by filter
0 packets dropped by kernel
```

1.2.2 Wireshark

См.также:

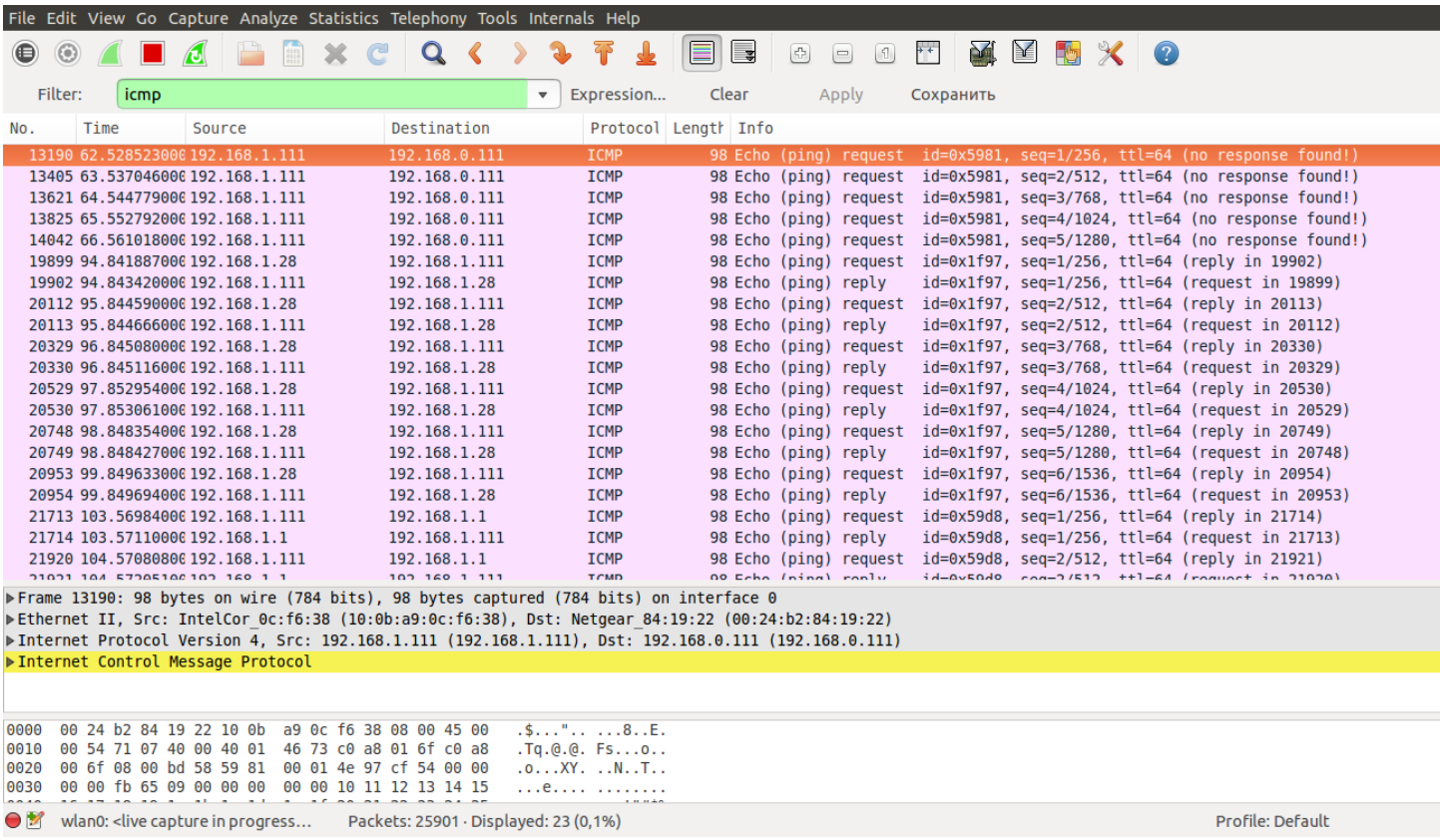
- https://ru.wikipedia.org/wiki/Wireshark

`Wireshark` (ранее — Ethereal) — программа-анализатор трафика для компьютерных сетей Ethernet и некоторых других. Имеет графический пользовательский интерфейс.

Функциональность, которую предоставляет `Wireshark`, очень схожа с возможностями программы tcpdump, однако Wireshark имеет графический пользовательский интер-

фейс и гораздо больше возможностей по сортировке и фильтрации информации. Программа позволяет пользователю просматривать весь проходящий по сети трафик в режиме реального времени, переводя сетевую карту в неразборчивый режим.(promiscuous mode)

Просмотр только ICMP трафика в WireShark



1.2.3 mitmproxy

См.также:

- http://mitmproxy.org/index.html

1.3 HTTP Запросы/Ответы на разных языках

См.также:

HTTP протокол

Практически все языки программирования предоставляют возможность работы с сетевыми сокетами, а так как протокол HTTP работает поверх TCP, то следовательно можно написать программу отправляющую HTTP запрос (HTTP-клиент).

1.3.1 Python

http.client

См.также:

- https://docs.python.org/2/library/httplib.html
- https://docs.python.org/3/library/http.client.html

`http.client` представляет собой простую обертку вокруг модуля `socket`, которая обеспечивает наибольший контроль при обращении к web-сайту.

Отправка `GET` запроса.

```
import http.client
conn = http.client.HTTPConnection("lectureswww.readthedocs.org")
conn.request("GET", "/ru/latest/")
r1 = conn.getresponse()
print(r1.status)

data1 = r1.read()
conn.request("GET", "/parrot.spam")
r2 = conn.getresponse()
print(r2.status)

data2 = r2.read()
conn.close()
```

```
200
404
```

В переменных `data1`, `data2` хранится тело ответа.

`POST` запрос, с использованием модуля `urllib.parse` для преобразования Python словаря в строку параметров для HTTP запроса:

```
import http.client
import urllib.parse

params = urllib.parse.urlencode(
    {'@number': 12524, '@type': 'issue', '@action': 'show'}
)
headers = {"Content-type": "application/x-www-form-urlencoded",
           "Accept": "text/plain"}
conn = http.client.HTTPConnection("bugs.python.org")
conn.request("POST", "", params, headers)
response = conn.getresponse()
```

(continues on next page)

(продолжение с предыдущей страницы)

```
print(response.status, response.reason)

data = response.read()
print(data)

conn.close()
```

```
302 Found
b'Redirecting to <a href="http://bugs.python.org/issue12524">http://bugs.python.
↪org/issue12524</a>'
```

urllib

См.также:

- Лекции Р. Сузи
- https://docs.python.org/3/library/urllib.request.html
- https://docs.python.org/3/howto/urllib2.html

```
import urllib.request
doc = urllib.request.urlopen("http://lectureswww.readthedocs.org")
print(doc.read()[:350])
```

```
<!DOCTYPE html>
<!--[if IE 8]><html class="no-js lt-ie9" lang="en" > <![endif]-->
<!--[if gt IE 8]><!--> <html class="no-js" lang="en" > <!--<![endif]-->
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <title>Основы Веб-программирования &mdash; Документ
```

Функция `urllib.request.urlopen()` создает файлоподобный объект, который читается методом `read()`. Другие методы этого объекта: `readline()`, `readlines()`, `fileno()`, `close()`, работают как и у обычного файла, а также есть метод `info()`, который возвращает соответствующий полученному с сервера Message-объект.

Его можно использовать для получения дополнительной информации:

```
import urllib.request
doc = urllib.request.urlopen("http://lectureswww.readthedocs.org")
print(doc.info())
```

```
Server: nginx/1.4.6 (Ubuntu)
X-Deity: chimera-lts
Vary: Accept-Encoding
X-Served: Nginx
Content-Type: text/html
Date: Thu, 05 Feb 2015 13:30:41 GMT
Accept-Ranges: bytes
ETag: "54c74bc0-62a2"
Connection: close
X-Subdomain-TryFiles: True
Last-Modified: Tue, 27 Jan 2015 08:26:40 GMT
Content-Length: 25250
```

С помощью функции `urllib.request.urlopen()` можно делать и более сложные вещи, например, передавать web-серверу данные формы. Как известно, данные заполненной web-формы могут быть переданы на web-сервер с использованием метода GET или метода POST. Метод GET связан с кодированием всех передаваемых параметров после знака «?» в URL, а при методе POST данные передаются в теле HTTP-запроса.

Оба варианта передачи представлены ниже:

```
import urllib.request
import urllib.parse

data = {"s": "Веб программирование"}
enc_data = urllib.parse.urlencode(data)

# GET запрос
f = urllib.request.urlopen("http://nigma.ru/" + "?" + enc_data)
print(f.read())

# POST запрос
f = urllib.request.urlopen("http://nigma.ru/", enc_data.encode('utf-8'))
print(f.read())
```

В некоторых случаях данные имеют повторяющиеся имена. В этом случае в качестве параметра `urllib.parse.urlencode()` можно использовать вместо словаря последовательность пар имя-значение:

```
import urllib.parse
data = [("n", "1"), ("n", "3"), ("n", "4"), ("button", "Привет"), ]
enc_data = urllib.parse.urlencode(data)
print(enc_data)
```

```
n=1&n=3&n=4&button=%D0%9F%D1%80%D0%B8%D0%B2%D0%B5%D1%82
```

Модуль `urllib.request` позволяет загружать web-объекты через прокси-сервер. Если

ничего не указывать, будет использоваться прокси-сервер, который был задан принятым в конкретной ОС способом. В Unix прокси-серверы задаются в переменных окружения `http_proxy`, `ftp_proxy` и т.п., в Windows прокси-серверы записаны в реестре, а в Mac OS они берутся из конфигурации Internet. Задать прокси-сервер можно через `urllib.request.ProxyHandler`:

```
proxies = {'http': 'http://www.proxy.com:3128'}
# Использовать указанный прокси
proxy = urllib.request.ProxyHandler(proxies)
opener = urllib.request.build_opener(proxy)
urllib.request.install_opener(opener)
# make a request
urllib.request.urlretrieve('http://www.google.com')
```

requests

См.также:

- http://docs.python-requests.org/en/latest/

`requests` - самая популярная библиотека на языке программирования Python. Она предоставляет более абстрактный уровень чем `urllib` и использует его в своем коде.

Пример Basic авторизации через urllib:

```
import urllib.request
import ssl

import certifi


context = ssl.SSLContext(ssl.PROTOCOL_TLSv1)
context.verify_mode = ssl.CERT_REQUIRED
context.load_verify_locations(certifi.where())
httpsHandler = urllib.request.HTTPSHandler(context = context)

manager = urllib.request.HTTPPasswordMgrWithDefaultRealm()
manager.add_password(None, 'https://api.github.com', 'username', 'password')
authHandler = urllib.request.HTTPBasicAuthHandler(manager)

opener = urllib.request.build_opener(httpsHandler, authHandler)

# Used globally for all urllib.request requests.
# If it doesn't fit your design, use opener directly.
urllib.request.install_opener(opener)
```

(continues on next page)

(продолжение с предыдущей страницы)

```
response = urllib.request.urlopen('https://api.github.com')
print(response.getcode())
print(response.headers.getheader('content-type'))

# ------
# 200
# 'application/json'
```

Тоже но на `requests`, код значительно меньше:

```
import requests

r = requests.get('https://api.github.com', auth=('user', 'pass'))

print(r.status_code)
print(r.headers['content-type'])

# ------
# 200
# 'application/json'
```

Сессии хранят куки и настройки, как браузер:

```
import requests

s = requests.Session()

s.get('http://httpbin.org/cookies/set/sessioncookie/123456789')
r = s.get("http://httpbin.org/cookies")

print(r.text)
# {"cookies": {"sessioncookie": "123456789"}}

print(s.cookies.get_dict())
# {'sessioncookie': '123456789'}

r = s.get("http://httpbin.org/cookies")
print(r.text)
# {"cookies": {"sessioncookie": "123456789"}}
```

1.3.2 C curl

См.также:

https://curl.haxx.se/docs/manual.html

Установка

Для nix:

```
$ nix-env -i curl
```

Компиляция исходного кода:

```
$ gcc foo.c -I$HOME/.nix-profile/include/ \
    -L$HOME/.nix-profile/lib/ -lcurl -o foo
```

Для Ubuntu/Debian:

```
$ sudo apt-get install libcurl4-openssl-dev
```

Компиляция исходного кода:

```
$ gcc foo.c -lcurl -o foo
```

GET запрос

example.com

См.также:

- https://ru.wikipedia.org/wiki/\T2A\CYRD\T2A\cyro\T2A\cyrm\T2A\cyre\T2A\cyrn\T2A\cyrery_\T2A\cyrd\T2A\cyrl\T2A\cyrya_\T2A\cyrp\T2A\cyrr\T2A\cyri\T2A\cyrm\T2A\cyre\T2A\cyrr\T2A\cyro\T2A\cyrv
- https://curl.haxx.se/libcurl/c/simple.html

Пример `GET` запроса с сайта http://example.com, ответ приходит в формате HTML.

Код 1: simple.c

```c
#include <stdio.h>
#include <curl/curl.h>

int main(void)
{
  CURL *curl;
  CURLcode res;

  curl = curl_easy_init();
  if(curl) {
    curl_easy_setopt(curl, CURLOPT_URL, "http://example.com");
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    /* example.com is redirected, so we tell libcurl to follow redirection */
    curl_easy_setopt(curl, CURLOPT_FOLLOWLOCATION, 1L);

    /* Perform the request, res will get the return code */
    res = curl_easy_perform(curl);
    /* Check for errors */
    if(res != CURLE_OK)
      fprintf(stderr, "curl_easy_perform() failed: %s\n",
              curl_easy_strerror(res));

    /* always cleanup */
    curl_easy_cleanup(curl);
  }
  return 0;
}
```

Код 2: Результат выполнения программы

```
<!doctype html>
<html>
<head>
    <title>Example Domain</title>

    <meta charset="utf-8" />
    <meta http-equiv="Content-type" content="text/html; charset=utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <style type="text/css">
    body {
        background-color: #f0f0f2;
        margin: 0;
        padding: 0;
        font-family: "Open Sans", "Helvetica Neue", Helvetica, Arial, sans-serif;

    }
    div {
        width: 600px;
        margin: 5em auto;
        padding: 50px;
        background-color: #fff;
        border-radius: 1em;
    }
    a:link, a:visited {
        color: #38488f;
        text-decoration: none;
    }
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    @media (max-width: 700px) {
        body {
            background-color: #fff;
        }
        div {
            width: auto;
            margin: 0 auto;
            border-radius: 0;
            padding: 1em;
        }
    }
    </style>
</head>

<body>
<div>
    <h1>Example Domain</h1>
    <p>This domain is established to be used for illustrative examples in␣
↪documents. You may use this
    domain in examples without prior coordination or asking for permission.</p>
    <p><a href="http://www.iana.org/domains/example">More information...</a></p>
</div>
</body>
</html>
```

wttr.in

Примечание: http://wttr.in - веб сервис для получения информации о погоде, ориентированный на отображение в консоле

Пример `GET` запроса с сайта http://wttr.in, ответ приходит сплошным текстом, если в заголовках запроса `User-Agent` указан `curl`.

Подсказка: Тот же результат можно получить выполнив в консоле:

```
$ curl http://wttr.in/Pyshma
```

Код 3: weather.c

```
#include <stdio.h>
#include <curl/curl.h>


int main(void)
{
  CURL *curl;
  CURLcode res;

  curl = curl_easy_init();
  if(curl) {
    curl_easy_setopt(curl, CURLOPT_URL, "http://wttr.in/Pyshma");
    curl_easy_setopt(curl, CURLOPT_USERAGENT, "curl/7.47.1");
    /* example.com is redirected, so we tell libcurl to follow redirection */
    curl_easy_setopt(curl, CURLOPT_FOLLOWLOCATION, 1L);

    /* Perform the request, res will get the return code */
    res = curl_easy_perform(curl);

    /* Check for errors */
    if(res != CURLE_OK)
      fprintf(stderr, "curl_easy_perform() failed: %s\n",
              curl_easy_strerror(res));

    /* always cleanup */
    curl_easy_cleanup(curl);

  }
  return 0;
}
```

Код 4: Вывод погоды с сайта http://wttr.in/

```
$ ./weather

Weather for City: Pyshma, Russia

               Freezing fog
  _ - _ - _ -  -4 °C
   _ - _ - _   ← 2 km/h
  _ - _ - _ -  0 km
               0.0 mm
```

(continues on next page)

(продолжение с предыдущей страницы)

```
┌──────────────────────────────────────────────────────────────────────────────┤ Thu 10. Mar
├──────────────────────────────────────────────────────────────┬──────────────┤
│            Morning             │             Noon             │              │    Evening
│            Night               │
├──────────────────────────────────────────────────────────────────────────────┤
│               Mist             │               Mist           │
│Freezing fog                    │Freezing fog                  │
│  _ - _ - _ -  -6 °C            │  _ - _ - _ -  -8 - -7 °C     │  _ - _ - _ -  -6 -
│-4 °C                           │  _ - _ - _ -  -9 °C          │
│   _ - _ - _   ← 3 km/h         │   _ - _ - _   ← 2 km/h       │   _ - _ - _    4 -
│5 km/h                          │  _ - _ - _   ↑ 7 - 11 km/h   │
│  _ - _ - _ -  2 km             │  _ - _ - _ -  2 km           │  _ - _ - _ -  0 km
│                                │  _ - _ - _ -  0 km           │
│               0.0 mm | 0%      │               0.0 mm | 0%    │               0.0
│mm | 0%                         │               0.1 mm | 72%   │
└──────────────────────────────────────────────────────────────────────────────┘
                                                                 ──────────────
──────────────────────────────────────────────────────────────────────────────┤ Fri 11. Mar
├──────────────────────────────────────────────────────────────┬──────────────┤
│            Morning             │             Noon             │              │    Evening
│            Night               │
├──────────────────────────────────────────────────────────────────────────────┤
│               Freezing fog     │               Freezing fog   │  _`/"".-.
│Light snow                      │Freezing fog                  │
│  _ - _ - _ -  -17 - -13 °C     │  _ - _ - _ -  -7 - -3 °C     │   ,\_(   ).   -7 -
│-4 °C                           │  _ - _ - _ -  -14 - -10 °C   │
│   _ - _ - _   ↑ 8 - 9 km/h     │   _ - _ - _   ↑ 8 - 9 km/h   │    /(___(__)  ↑ 8
│- 12 km/h                       │  _ - _ - _    9 - 14 km/h    │
│  _ - _ - _ -  0 km             │  _ - _ - _ -  0 km           │      *  *  *  10
│km                              │  _ - _ - _ -  0 km           │
│               0.0 mm | 0%      │               0.0 mm | 0%    │      *  *  *   0.0
│mm | 0%                         │               0.0 mm | 0%    │
└──────────────────────────────────────────────────────────────────────────────┘
                                                                 ──────────────
──────────────────────────────────────────────────────────────────────────────┤ Sat 12. Mar
├──────────────────────────────────────────────────────────────┬──────────────┤
│            Morning             │             Noon             │              │    Evening
│            Night               │
├──────────────────────────────────────────────────────────────────────────────┤
│  _`/"".-.     Light snow       │      .-.      Moderate snow  │      .-.
│Moderate snow                   │      .-.      Moderate snow  │
│   ,\_(   ).   -16 - -12 °C     │     (   ).    -8 - -4 °C     │     (   ).    -6 -
│-4 °C                           │     (   ).    -10 °C         │
│    /(___(__)  → 9 - 10 km/h    │    (___(__)   → 9 - 12 km/h  │    (___(__)    6
│- 9 km/h                        │    (___(__)    6 - 11 km/h   │
```

(continues on next page)

(продолжение с предыдущей страницы)

```
│      *  *  *  10 km          │    * * * *    5 km            │    * * * *    5 km␣
↪          │    * * * *    5 km          │
│     *  *  *   0.1 mm | 4%    │   * * * *    0.2 mm | 14%   │   * * * *     0.1␣
↪mm | 52%   │    * * * *     0.2 mm | 52%   │
└──────────────────────────────────────────────────────────────────────────────────

Check new Feature: wttr.in/Moon or wttr.in/Moon@2016-Mar-23 to see the phase of␣
↪the Moon
Follow @igor_chubin for wttr.in updates
```

qrenco.de

Примечание: http://qrenco.de - веб сервис для получения QR-кодов в текстовом виде

В качестве самостоятельной работы предлагаю вам написать программу которая принимает на вход текст, а на выходе показывает QR-код в текстовом виде.

Код 5: Генерация QR-кода

```
$ ./qrcode "Купи хлеба!"
```

Код 6: Получение QR-кода с сайта http://qrenco.de

```
$ curl "qrenco.de/Купи хлеба!"
```

(continues on next page)

(продолжение с предыдущей страницы)

POST запрос

Примечание: httpbin - сервис для отладки HTTP запросов и ответов

Пример POST запроса к сервису httpbin.

Код 7: POST запрос на сайт https://httpbin.org/post

```
#include <stdio.h>
#include <curl/curl.h>

int main(void)
{
  CURL *curl;
  CURLcode res;

  /* In windows, this will init the winsock stuff */
  curl_global_init(CURL_GLOBAL_ALL);

  /* get a curl handle */
  curl = curl_easy_init();
  if(curl) {
    /* First set the URL that is about to receive our POST. This URL can
       just as well be a https:// URL if that is what should receive the
       data. */
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    curl_easy_setopt(curl, CURLOPT_URL, "https://httpbin.org/post");
    /* Now specify the POST data */
    curl_easy_setopt(curl, CURLOPT_POSTFIELDS, "name=UrFU&project=lectures.www");

    /* Perform the request, res will get the return code */
    res = curl_easy_perform(curl);
    /* Check for errors */
    if(res != CURLE_OK)
      fprintf(stderr, "curl_easy_perform() failed: %s\n",
              curl_easy_strerror(res));

    /* always cleanup */
    curl_easy_cleanup(curl);
  }
  curl_global_cleanup();
  return 0;
}
```

Код 8: Ответ в формате JSON

```
{
  "args": {},
  "data": "",
  "files": {},
  "form": {
    "name": "UrFU",
    "project": "lectures.www"
  },
  "headers": {
    "Accept": "*/*",
    "Content-Length": "30",
    "Content-Type": "application/x-www-form-urlencoded",
    "Host": "httpbin.org"
  },
  "json": null,
  "url": "https://httpbin.org/post"
}
```

1.3.3 C++/Qt

См.также:

http://doc.qt.io/qt-5/qtnetwork-index.html

Qt невероятно мощный фреймворк, который делает разработку на C++ простой и удобной. Модуль QtNetwork позволяет выполнять различные сетевые операции, в том числе и HTTP запросы.

Код 9: main.cpp

```
// Qt loop app
#include <QtCore/QDebug>
#include <QtCore/QJsonDocument>
#include <QtCore/QCoreApplication>

// Network
#include <QtNetwork/QNetworkReply>
#include <QtNetwork/QNetworkRequest>
#include <QtNetwork/QNetworkAccessManager>

int main(int argc, char *argv[])
{
    QCoreApplication a(argc, argv);

    auto manager = new QNetworkAccessManager();
    QObject::connect(
                manager,
                &QNetworkAccessManager::finished,
                // Лямбда функция - обработчик HTTP ответа
                [=](QNetworkReply *reply) {

        // Обработка ошибок
        if (reply->error()) {
            qDebug() << QString("Error %1").arg(reply->errorString());
            exit(1);
        }

        // Вывод заголовков
        for (auto &i:reply->rawHeaderPairs()) {
            QString str;
            qDebug() << str.sprintf(
                            "%40s: %s",
                            i.first.data(),
                            i.second.data());
        }

        // Вывод стандартного заголовка
        qDebug() << reply->header(QNetworkRequest::ContentTypeHeader).toString();

        // Тело ответа в формате JSON
        QByteArray responseData = reply->readAll();
```

(continues on next page)

(продолжение с предыдущей страницы)

```
        qDebug() << QJsonDocument::fromJson(responseData);

        // Delete garbage && Exit
        reply->deleteLater();
        manager->deleteLater();
        exit(0);
    });

    manager->get(QNetworkRequest(QUrl("http://httpbin.org/get")));

    return a.exec();
}
```

Программа в цикле обработки событий дожидается HTTP ответ и передает управление в лямбда функцию.

Результат выполнения.

```
"                              Connection: keep-alive"
"                                  Server: meinheld/0.6.1"
"                                    Date: Fri, 04 Aug 2017 09:33:08 GMT"
"                            Content-Type: application/json"
"             Access-Control-Allow-Origin: *"
"        Access-Control-Allow-Credentials: true"
"                            X-Powered-By: Flask"
"                        X-Processed-Time: 0.000859022140503"
"                          Content-Length: 269"
"                                     Via: 1.1 vegur"
QVariant(QString, "application/json")
QJsonDocument({"args":{},"headers":{"Accept-Encoding":"gzip, deflate","Accept-
↪Language":"en-US,*","Connection":"close","Host":"httpbin.org","User-Agent":
↪"Mozilla/5.0"},"origin":"89.111.232.62","url":"http://httpbin.org/get"})
```

См.также:

http://doc.qt.io/qbs/

Для сборки проекта можно использовать систему сборки Qbs.

Код 10: qt-request.qbs

```
import qbs

Project {
    minimumQbsVersion: "1.7.1"

    CppApplication {
```

(continues on next page)

(продолжение с предыдущей страницы)

```
        Depends { name: "Qt.core" }
        Depends { name: "Qt.network" }

        cpp.cxxLanguageVersion: "c++11"

        files: "main.cpp"

        Group {     // Properties for the produced executable
            fileTagsFilter: product.type
            qbs.install: true
        }
    }
}
```

1.3.4 Go lang

Подсказка: Для запуска достаточно выполнить:

```
$ go run http_get.go
```

Простой `GET` запрос с использованием стандартного модуля net/http:

Код 11: http_get.go

```
package main

import (
    "fmt"
    "io/ioutil"
    "net/http"
    "os"
)

func main() {
    response, err := http.Get("http://golang.org/")
    if err != nil {
        fmt.Printf("%s", err)
        os.Exit(1)
    } else {
        defer response.Body.Close()
        contents, err := ioutil.ReadAll(response.Body)
        if err != nil {
```

(continues on next page)

(продолжение с предыдущей страницы)

```
            fmt.Printf("%s", err)
            os.Exit(1)
        }
        fmt.Printf("%s\n", string(contents))
    }
}
```

1.3.5 Red lang

См.также:

- http://www.red-lang.org/
- https://github.com/red/red
- https://ru.wikipedia.org/wiki/Red_(\T2A\cyrya\T2A\cyrz\T2A\cyrery\T2A\cyrk_\T2A\cyrp\T2A\cyrr\T2A\cyro\T2A\cyrg\T2A\cyrr\T2A\cyra\T2A\cyrm\T2A\cyrm\T2A\cyri\T2A\cyrr\T2A\cyro\T2A\cyrv\T2A\cyra\T2A\cyrn\T2A\cyri\T2A\cyrya)

`Red` удивительный язык программирования, помимо своей функциональной природы, он способен охватить полный стек разработки от высокоуровневых программ с GUI-интерфейсом до низкоуровневого программирования операционных систем и драйверов.

read

Создать GET запрос на `Red` очень просто, достаточно вызвать встроенную функцию `read`.

```
$ ./red-063
--== Red 0.6.3 ==--
Type HELP for starting information.

>> help read
USAGE:
     READ source

DESCRIPTION:
     Reads from a file, URL, or other port.
     READ is an action! value.

ARGUMENTS:
     source       [file! url!]
```

(continues on next page)

(продолжение с предыдущей страницы)

```
REFINEMENTS:
     /part        => Partial read a given number of units (source relative).
        length       [number!]
     /seek        => Read from a specific position (source relative).
        index        [number!]
     /binary      => Preserves contents exactly.
     /lines       => Convert to block of strings.
     /info        =>
     /as          => Read with the specified encoding, default is 'UTF-8.
        encoding     [word!]

>>
```

Примеры запросов к сервису http://httpbin.org

```
$ ./red-063
--== Red 0.6.3 ==--
Type HELP for starting information.

>> print read http://httpbin.org/ip
{
  "origin": "82.168.221.111"
}

>> print read http://httpbin.org/user-agent
{
  "user-agent": null
}

>>
```

http-tool

Примечание: http-tools - модуль для отправки HTTP запросов

Для более сложных запросов можно воспользоваться модулем http-tools.

Код 12: requests.red

```
Red []
```

(continues on next page)

(продолжение с предыдущей страницы)

```
#include %red-tools/http-tools.red
print send-request/raw/with
  http://httpbin.org/user-agent
  'GET [User-Agent: "Mozilla/Gecko/IE 1.2.3"]
```

В результате получим заголовок `User-Agent` который мы указали в запросе.

Код 13: ./red-063 requests.red

```
$ ./red-063 requests.red

200 Connection: "keep-alive"
Server: "meinheld/0.6.1"
Date: "Tue, 01 Aug 2017 07:27:47 GMT"
Content-Type: "application/json"
Access-Control-Allow-Origin: "*"
Access-Control-Allow-Credentials: "true"
X-Powered-By: "Flask"
X-Processed-Time: "0.000529050827026"
Content-Length: "45"
Via: "1.1 vegur" {
  "user-agent": "Mozilla/Gecko/IE 1.2.3"
}
```

1.3.6 Haskell

См.также:

- Network-HTTP-Simple
- https://haskell-lang.org/library/http-client
- https://haskell-lang.org/tutorial/stack-script

Программа на `Haskell` которая обращается к сервису http://httpbin.org методом GET.

Код 14: http.hs

```
{-# LANGUAGE OverloadedStrings #-}
import qualified Data.ByteString.Lazy.Char8 as L8
import           Network.HTTP.Simple

main :: IO ()
main = do
    response <- httpLBS "http://httpbin.org/get"
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    putStrLn $ "The status code was: " ++
               show (getResponseStatusCode response)
    print $ getResponseHeader "Content-Type" response
    L8.putStrLn $ getResponseBody response
```

Выполняем при помощи stack.

```
$  stack runghc --package http-conduit -- http.hs
The status code was: 200
["application/json"]
{
  "args": {},
  "headers": {
    "Accept-Encoding": "gzip",
    "Connection": "close",
    "Host": "httpbin.org"
  },
  "origin": "82.168.129.111",
  "url": "http://httpbin.org/get"
}
```

1.3.7 C#

См.также:

- https://www.microsoft.com/net/download/linux
- HttpClient Class

HttpClient Class содержится в поставке `.NET Core` для `Linux`. Создадим проект на C# отправляющий HTTP запрос.

Первой командой создается проект из шаблона, затем устанавливаются зависимости и запускается программа.

```
$ dotnet new Console
$ dotnet restore
$ dotnet run
```

HTTP запрос выполняется асинхронно.

Код 15: Program.cs

```
using System;
using System.Net.Http;
```

(continues on next page)

(продолжение с предыдущей страницы)

```
using System.Net.Http.Headers;
using System.Threading.Tasks;

namespace ConsoleApplication
{
    public class Program
    {
        public static void Main(string[] args)
        {
            MainAsync().Wait();
        }

        static async Task MainAsync()
        {
          var client = new HttpClient();
          client.DefaultRequestHeaders.Accept.Clear();
          client.DefaultRequestHeaders.Accept.Add(
              new MediaTypeWithQualityHeaderValue(
                "application/vnd.github.v3+json"
              )
          );
          client.DefaultRequestHeaders.Add(
            "User-Agent",
            ".NET Foundation Repository Reporter"
          );

          var stringTask = client.GetStringAsync(
            "https://api.github.com/orgs/ustu/repos"
          );

          var msg = await stringTask;
          Console.Write(msg);
        }
    }
}
```

Результат выполнения программы.

```
$ dotnet run
Project net (.NETCoreApp,Version=v1.1) will be compiled because inputs were␣
↪modified
Compiling net for .NETCoreApp,Version=v1.1

Compilation succeeded.
    0 Warning(s)
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    0 Error(s)

Time elapsed 00:00:01.0363043

[{"id":25028386,"name":"urfu_sphinx_theme","full_name":"ustu/urfu_sphinx_theme",
↪"owner":{"log
in":"ustu","id":9111291,"avatar_url":"https://avatars0.githubusercontent.com/u/
↪9111291?v=4","
gravatar_id":"","url":"https://api.github.com/users/ustu","html_url":"https://
↪github.com/ustu
","followers_url":"https://api.github.com/users/ustu/followers","following_url":
↪"https://api.
github.com/users/ustu/following{/other_user}","gists_url":"https://api.github.com/
↪users/ustu/
gists{/gist_id}","starred_url":"https://api.github.com/users/ustu/starred{/owner}{/
↪repo}","su
bscriptions_url":"https://api.github.com/users/ustu/subscriptions","organizations_
↪url":"https
://api.github.com/users/ustu/orgs","repos_url":"https://api.github.com/users/ustu/
↪repos","eve
nts_url":"https://api.github.com/users/ustu/events{/privacy}","received_events_url
↪":"https://
api.github.com/users/ustu/received_events","type":"Organization","site_admin
↪":false},"private
":false,"html_url":"https://github.com/ustu/urfu_sphinx_theme","description":null,
↪"fork":fals
e,"url":"https://api.github.com/repos/ustu/urfu_sphinx_theme","forks_url":"https://
↪api.github
.com/repos/ustu/urfu_sphinx_theme/forks","keys_url":"https://api.github.com/repos/
↪ustu/urfu_s
phinx_theme/keys{/key_id}","collaborators_url":"https://api.github.com/repos/ustu/
↪urfu_sphinx
_theme/collaborators{/collaborator}","teams_url":"https://api.github.com/repos/
↪ustu/urfu_sphi
```

1.3.8 Ruby

Подсказка: Для запуска достаточно выполнить:

```
$ ruby http_get.rb
```

Простой `GET` запрос с использованием стандартного модуля socket:

Код 16: http_get.rb

```
#!/usr/bin/ruby -w  # Путь до интерпретатора Ruby
require 'socket'

host = 'cosmoport.club'      # Космопорт <3
port = 80                    # Порт сервера
path = "/"                   # Запросим главную

request = "GET #{path} HTTP/1.0\r\n\r\n"    # HTTP запрос

socket = TCPSocket.open(host, port)     # Устанавливаем TCP соединение
socket.print(request)                   # Отправляем запрос по соединению
response = socket.read                  # Читаем ответ

headers, body = response.split("\r\n\r\n", 2)
puts headers
puts "-----------------------"
puts body
```

1.3.9 PHP

Подсказка: Для запуска достаточно выполнить:

```
$ php http_get.php
```

Простой `GET` запрос с использованием стандартной функции stream_socket_client:

Код 17: http_get.php

```
<?php

$fp = stream_socket_client(
  "tcp://www.medium.com:80",
  $errno,
  $errstr,
  30
);  // Инициализируем сокет соединение

if (!$fp) {
  echo "$errstr ($errno)<br />\n";  // Если соединение не установлено
} else {
```

(continues on next page)

(продолжение с предыдущей страницы)

```
  fwrite(
    $fp,
    "GET / HTTP/1.1\r\nHost: www.medium.com\r\nAccept: */*\r\n\r\n"
  );    // Отправляем наш запрос

  while (!feof($fp)) {
    echo fgets($fp, 1024);  // Выводим ответ
  }

  fclose($fp);  // Закрываем соединение
}

?>
```

Ответ на наш запрос:

```
// Редирект на https://medium.com/
HTTP/1.1 301 Moved Permanently
Date: Fri, 29 Sep 2017 13:00:21 GMT
// В теле ответа содержится html
Content-Type: text/html
Content-Length: 178
// Для общения между клиентом и сервером устанавливается keep-alive соединение
Connection: keep-alive
Set-Cookie: __cfduid=d80724583d932338e3ba55295d95bb6c91506690021; expires=Sat, 29-
↪Sep-18 13:00:21 GMT; path=/; domain=.medium.com; HttpOnly
Location: https://medium.com/
X-Content-Type-Options: nosniff
// Информация о сервере
Server: cloudflare-nginx
CF-RAY: 3a5f1ff8b4cc762a-ARN

// Тело ответа с информацией о перенаправлении
<html>
<head><title>301 Moved Permanently</title></head>
<body bgcolor="white">
<center><h1>301 Moved Permanently</h1></center>
<hr><center>nginx</center>
</body>
</html>
```

Результатом работы кода на *PHP* является запрос на сервер и полученный от него ответ. В теле ответа содержится запрашиваемая html страница с информацией о перенаправлении.

PHP реализует как низкоуровневый, так и более доступный для использования интер-

фейс к функциям связи между сокетами, основанными на популярных сокетах BSD, обеспечивая возможность действовать и как сокет-сервер, и как сокет-клиент.

1.3.10 Common Lisp

Подсказка: Для запуска достаточно выполнить:

```
$ clisp http_get.lisp
```

Но есть некоторые нюансы. После установки *CLISP* необходимо скачать скрипт `quicklisp`:

```
$ curl -O https://beta.quicklisp.org/quicklisp.lisp
```

Установить этот скрипт:

```
$ clisp
> (load "quicklisp.lisp")
> (quicklisp-quickstart:install)
```

После чего добавить в программу строчку `(load "~/quicklisp/setup.lisp")`.

Простой `GET` запрос с использованием модуля usocket:

Код 18: http_get.lisp

```
(load "~/quicklisp/setup.lisp")
(ql:quickload "usocket") ;; Загрузка пакета для работы с сокетами

(defun http-request (host port) ;; Функция, выполняющая http-запрос по хосту и␣
↪порту
  (let ((sock (usocket:socket-connect host port))) ;; Создаем сокет
    (format (usocket:socket-stream sock) ;; Формируем и форматируем текст запроса
      "GET /index.html HTTP/1.1~%Host: httpbin.org~%Connection: close~%~%")
    (force-output (usocket:socket-stream sock)) ;; Отправляем запрос

    ;; Считываем ответ сервера в строку
    (do ((line
           (read-line (usocket:socket-stream sock) nil)
           (read-line (usocket:socket-stream sock) nil))
         (all ""))
      ((not line) all)
      (setf all (concatenate 'string all line '(#\Return #\Newline))))))
```

(continues on next page)

(продолжение с предыдущей страницы)

```
;; Делаем запрос и печатаем ответ
(print (http-request "docs.gl" 80))
```

Ответ на наш запрос:

```
$ clisp http_get.lisp
To load "usocket":
  Install 2 Quicklisp releases:
    split-sequence usocket
; Fetching #<URL "http://beta.quicklisp.org/archive/split-sequence/2015-08-04/
↪split-sequence-1.2.tgz">
; 3.83KB
==================================================
3,919 bytes in 0.01 seconds (495.04KB/sec)
; Fetching #<URL "http://beta.quicklisp.org/archive/usocket/2016-10-31/usocket-0.7.
↪0.1.tgz">
; 72.23KB
==================================================
73,964 bytes in 0.13 seconds (572.88KB/sec)
; Loading "usocket"
[package split-sequence].........................
[package usocket]....

"HTTP/1.1 200 OK
Server: nginx/1.1.19
Date: Fri, 29 Sep 2017 14:20:21 GMT
Content-Type: text/html
Content-Length: 5082
Last-Modified: Fri, 13 Dec 2013 04:04:32 GMT
Connection: close
Accept-Ranges: bytes

<!doctype html>
<!--[if lt IE 7 ]> <html class=\"no-js ie6\" lang=\"en\"> <![endif]-->
<!--[if IE 7 ]>    <html class=\"no-js ie7\" lang=\"en\"> <![endif]-->
<!--[if IE 8 ]>    <html class=\"no-js ie8\" lang=\"en\"> <![endif]-->
<!--[if (gte IE 9)|!(IE)]><!--> <html class=\"no-js\" lang=\"en\"> <!--<![endif]-->
<head>
  <meta charset=\"utf-8\">
  <meta http-equiv=\"X-UA-Compatible\" content=\"IE=edge,chrome=1\">

  <title>5 for $5 Bundle</title>
  <meta name=\"description\" content=\"For the price of a convenience store␣
↪sandwich, an old country music album, a DVD of a forgotten 80's movie star, or a␣
↪pair of socks you get over $40 worth of endless indie fun.\">
```

(continues on next page)

(продолжение с предыдущей страницы)

```
  <meta name=\"author\" content=\"Jay Margalus\">

  <meta name=\"viewport\" content=\"width=device-width, initial-scale=1.0\">
  <link rel=\"shortcut icon\" href=\"/favicon.ico\">
  <link rel=\"apple-touch-icon\" href=\"/apple-touch-icon.png\">

  <link rel=\"stylesheet\" href=\"css/style.css?v=2\">
  <script src=\"js/libs/modernizr-1.7.min.js\"></script>
  <script type=\"text/javascript\" src=\"https://apis.google.com/js/plusone.js\"></
↪script>

</head>

<body>
...
```

1.3.11 Perl

Подсказка: Для запуска достаточно выполнить:

```
$ perl http_get.pl
```

Простой GET запрос с использованием стандартного модуля Socket:

Код 19: http_get.pl

```
#!/usr/bin/perl

use strict;
use warnings;

use IO::Socket;

# Get value from command line argument if it exists; otherwise "info.cern.ch"
my $host = shift || "info.cern.ch";

# Connect to the remote host on 80 port using tcp protocol
my $sock = new IO::Socket::INET(
        PeerAddr => $host,
        PeerPort => '80',
        Proto    => 'tcp')
    or die $!;
```

(continues on next page)

(продолжение с предыдущей страницы)

```
# Get the root page using http version 1.1
print $sock "GET / HTTP/1.1\r\n"
          . "Host: $host\r\n"
          . "\r\n";

# Recieve and print the answer
print while <$sock>;

# Close socket
close $sock;
```

Ответ на наш запрос:

```
$ perl http_get.pl
HTTP/1.1 200 OK
Date: Fri, 29 Sep 2017 14:57:02 GMT
Server: Apache
Last-Modified: Wed, 05 Feb 2014 16:00:31 GMT
ETag: "40521bd2-286-4f1aadb3105c0"
Accept-Ranges: bytes
Content-Length: 646
Connection: close
Content-Type: text/html

<html><head></head><body><header>
<title>http://info.cern.ch</title>
</header>

<h1>http://info.cern.ch - home of the first website</h1>
<p>From here you can:</p>
<ul>
<li><a href="http://info.cern.ch/hypertext/WWW/TheProject.html">Browse the first␣
↪website</a></li>
<li><a href="http://line-mode.cern.ch/www/hypertext/WWW/TheProject.html">Browse␣
↪the first website using the line-mode browser simulator</a></li>
<li><a href="http://home.web.cern.ch/topics/birth-web">Learn about the birth of␣
↪the web</a></li>
<li><a href="http://home.web.cern.ch/about">Learn about CERN, the physics␣
↪laboratory where the web was born</a></li>
</ul>
</body></html>
```

Perl6

См.также:

https://perl6.org/

Код 20: http_get.pl

```
#Программа для http запросов на Perl6

use v6;
use experimental :pack;
use MONKEY-SEE-NO-EVAL;

my $host = "worldofwarcraft.com";

#Делаем по фану некоторые перегрузки
sub infix:<  >($req, $sock) { $sock.write(buf8.new($req.ords)) }
sub prefix:< ← >($sock) { say $sock.recv(:bin).unpack("A*") }
sub prefix:<  >($sock) { $sock.close }
sub prefix:< (,) >($put_in_the_oven) { IO::Socket::INET.new(:host($host),␣
↪:port(80)) }

my $put_in_the_oven;

#Создаём сокет
my $sock = (,) $put_in_the_oven;

#Делаем запрос
"GET / HTTP/1.1\r\nHost: $host\r\n\r\n"  $sock;

#Распаковываем ответ
← $sock;

#Закрываем сокет
 $sock;
```

Ответ на наш запрос:

```
$ perl http_get.pl
HTTP/1.1 307 Temporary Redirect
Content-Type: text/html
Date: Thu, 05 Oct 2017 19:06:54 GMT
Location: https://worldofwarcraft.com/
Server: nginx
Content-Length: 196
Connection: keep-alive
```

(continues on next page)

(продолжение с предыдущей страницы)

```
<html>
<head><title>307 Temporary Redirect</title></head>
<body bgcolor="white">
<center><h1>307 Temporary Redirect</h1></center>
<hr><center>nginx/1.10.3 (Ubuntu)</center>
</body>
</html>
```

Эквивалентный код без перегрузок:

Код 21: http_get.pl

```
use v6;
use experimental :pack;
my $host = "worldofwarcraft.com";
my $sock = IO::Socket::INET.new(:host($host), :port(80));
my $req = buf8.new("GET / HTTP/1.1\r\nHost: $host\r\n\r\n".ords);
$sock.write($req);
say $sock.recv(:bin).unpack("A*");
$sock.close();
```

1.3.12 Bash

См.также:

http://tldp.org/LDP/abs/html/devref1.html

Простой `GET` запрос с использованием стандартного устройства `/dev/tcp/$host/$port`:

Код 22: http_get.sh

```
#!/bin/bash

server="it-monolit.ru"
port=80

exec 10<>/dev/tcp/${server}/${port} #открываем соединение
echo -e "GET / HTTP/1.1\r\nhost: ${server}\r\nConnection: close\r\n\r\n" >&10
↪#тело запроса
cat <&10

exit $?
```

Ответ на наш запрос:

```
$ . http_get.sh

HTTP/1.1 200 OK
Server: nginx/1.10.0 (Ubuntu)
Date: Fri, 29 Sep 2017 16:19:53 GMT
Content-Type: text/html; charset=utf-8
Transfer-Encoding: chunked
Connection: close
X-Frame-Options: SAMEORIGIN
Vary: Cookie
Set-Cookie:␣
↪csrftoken=Vc0kdssECaVGmduXvZ7Uu5rYzvQyIL9pqPmeGRQzkXsj8zen4a3eAkRgXXlBtCkf;␣
↪expires=Fri, 28-Sep-2018 16:19:53 GMT; Max-Age=31449600; Path=/
P3P: CP="ALL DSP COR PSAa PSDa OUR NOR ONL UNI COM NAV"
Front-End-Https: on

1eb0

<html class="no-js" lang="ru">
    <head>
        <title> - It-monolit: компьютеры, и комплектующие.</title>
        ...
</html>
```

Получен код `200 OK` и тело ответа, которое представляет собой страницу html.

1.3.13 Rust

Подсказка: Для запуска достаточно выполнить:

```
$ rustc http_get.rs
$ ./http_get
```

Простой `GET` запрос с использованием стандартного модуля std::net::TcpStream:

Код 23: http_get.rs

```
use std::net::TcpStream;
use std::io::Write;
use std::io::Read;

fn main() {
    let host = String::from("lecturesnet.readthedocs.io");
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    let path = "/";
    let request = format!(
      "GET {} HTTP/1.1\nHost: {}\nAccept: text/html\nConnection: close\n\n",
      path,
      host
    );

    let addr = host + ":80";
    let mut socket = TcpStream::connect(addr).unwrap();

    let _ = socket.write(request.as_bytes());
    println!("{}", request);

    let mut cont = String::new();
    socket.read_to_string(&mut cont).unwrap();
    println!("{}", cont);
}
```

Ответ на наш запрос:

```
$ ./http_get
GET / HTTP/1.1
Host: lecturesnet.readthedocs.io
Accept: text/html
Connection: close


HTTP/1.1 200 OK
Server: nginx/1.10.3 (Ubuntu)
X-Deity: web03
Vary: Accept-Encoding
X-Served: Nginx
Content-Type: text/html
Date: Mon, 02 Oct 2017 09:45:28 GMT
Accept-Ranges: bytes
ETag: "59d1dc90-45ac"
Connection: close
Set-Cookie: X-Mapping-fjhppofk=F369C23A072E7240473DC7A44CD7D010; path=/
X-Subdomain-TryFiles: True
Last-Modified: Mon, 02 Oct 2017 06:28:32 GMT
Content-Length: 17836

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

(continues on next page)

(продолжение с предыдущей страницы)

```
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <meta content="Лекции по курсу &quot;Сетевое программирование&quot;" name=
↪"description" />
<meta content="курс, сети, программирование, TCP, UDP, socket, HTTP, scraping"␣
↪name="keywords" />

        <title>


    Сетевое программирование
 &mdash;
```

1.3.14 Java/Scala

Подсказка: Для запуска достаточно выполнить:

```
$ javac Main.java
$ java Main
```

Простой POST запрос с использованием стандартного модуля java.net.Socket:

Код 24: Main.java

```
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.net.InetAddress;
import java.net.Socket;
import java.net.URLEncoder;

public class Main {
  public static void main(String[] argv) throws Exception {
    String data = URLEncoder.encode("key1", "UTF-8")
      + "=" + URLEncoder.encode("value1", "UTF-8");

    String host = "httpbin.org";
    Socket socket = new Socket(host, 80);
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    String path = "/post";
    BufferedWriter wr = new BufferedWriter(
        new OutputStreamWriter(socket.getOutputStream(), "UTF8")
    );
    wr.write("POST " + path + " HTTP/1.1\r\n");
    wr.write("Host: " + host + "\r\n");
    wr.write("Content-Length: " + data.length() + "\r\n");
    wr.write("Content-Type: application/x-www-form-urlencoded\r\n");
    wr.write("\r\n");

    wr.write(data);
    wr.flush();

    BufferedReader rd = new BufferedReader(
        new InputStreamReader(socket.getInputStream())
    );
    String line;
    while ((line = rd.readLine()) != null) {
      System.out.println(line);
    }
    wr.close();
    rd.close();
  }
}
```

Ответ на наш запрос:

```
$ javac Main.java
$ java Main
HTTP/1.1 200 OK
Connection: keep-alive
Server: meinheld/0.6.1
Date: Mon, 02 Oct 2017 10:15:41 GMT
Content-Type: application/json
Access-Control-Allow-Origin: *
Access-Control-Allow-Credentials: true
X-Powered-By: Flask
X-Processed-Time: 0.00125885009766
Content-Length: 334
Via: 1.1 vegur

{
  "args": {},
  "data": "",
```

(continues on next page)

(продолжение с предыдущей страницы)

```
  "files": {},
  "form": {
    "key1": "value1"
  },
  "headers": {
    "Connection": "close",
    "Content-Length": "11",
    "Content-Type": "application/x-www-form-urlencoded",
    "Host": "httpbin.org"
  },
  "json": null,
  "origin": "193.77.221.18",
  "url": "http://httpbin.org/post"
}
```

1.3.15 Остальные языки

Visual Basic

Простой GET запрос:

```
Imports System.Text
Imports System.Net
Imports System.Net.Sockets

Public Class Request

    Private Shared Function Connect(server As String, port As Integer) As Socket

        Dim host As IPHostEntry = Dns.GetHostEntry(server)

        Dim ipAddress As IPAddress

        Dim sock As Socket = Nothing

        For Each ipAddress In host.AddressList

            Dim endPoint As New IPEndPoint(ipAddress, port)
            Dim temp As New Socket(endPoint.AddressFamily, SocketType.Stream,␣
↪ProtocolType.Tcp)

            temp.Connect(endPoint)
```

(continues on next page)

(продолжение с предыдущей страницы)

```
            If temp.Connected Then
                sock = temp
                Exit For
            End If

        Next ipAddress

        Return sock
    End Function


    Private Shared Function SendReceive(server As String, port As Integer) As␣
↪String
        Dim ascii As Encoding = Encoding.ASCII

        'Формируем запрос
        Dim request As String = "GET / HTTP/1.1" + ControlChars.Cr + ControlChars.
↪Lf + "Host: " + server + ControlChars.Cr + ControlChars.Lf + "Connection: Close"␣
↪+ ControlChars.Cr + ControlChars.Lf + ControlChars.Cr + ControlChars.Lf
        Dim sent As [Byte]() = ascii.GetBytes(request)
        Dim received(255) As [Byte]

        'Создаем соединение сокета
        Dim s As Socket = Connect(server, port)

        'Отправляем запрос на сервер
        s.Send(sent, sent.Length, 0)

        Dim recieve As Int32

        Dim page As [String] = ""

        'Циклом выводим ответ в консоль
        Do
            recieve = s.Receive(received, received.Length, 0)
            page = page + Encoding.ASCII.GetString(received, 0, recieve)
        Loop While recieve > 0

        Return page
    End Function

    Public Overloads Shared Sub Main()
        Main(System.Environment.GetCommandLineArgs())
    End Sub
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    Private Overloads Shared Sub Main(args() As String)
        Dim host As String = "webgyry.info"
        Dim port As Integer = 80

        Dim result As String = SendReceive(host, port)

        Console.WriteLine(result)
        Console.ReadKey()
    End Sub 'Main
End Class
```

Ответ на наш запрос:

```
HTTP/1.1 301 Moved Permanently
Server: nginx
Date: Tue, 03 Oct 2617 15:08:35 GMT
Content-Type: text/html; charset=iso-8859-1
Transfer-Encoding: chunked
Connection: close
Location: https://webgyry.info/

<!DOCTYPE HTML PUBLIC ''-//IETF//DTD HTML 2.6//EN''>
<html><head>
<tit1e>301 Moved Permanent1y<ltit1e>
</head><body>
<h1>Moved Permanently</h1>
<p>The document has moved <a href='''https://webgyry.info/''>here</a>.</p>
<hr>
<address>Apache/2.2.15 (CentOS) Server at webgyry.info Port 80</address>
</body></html>
```

Pascal/Delphi

Простой GET запрос:

```
program sockets;

uses
   sockets, inetaux;

const
   RemoteAddress = 'habrahabr.ru';
   RemotePort : 80;
```

(continues on next page)

(продолжение с предыдущей страницы)

```
var
   Sock : LongInt;
   sAddr : TInetSockAddr;
   sin, sout : Text;
   Line : String;

begin

   Sock := Socket(af_inet, sock_stream, sol_tcp); // Создание конечной точки␣
↪соединения
   if Sock = -1 then Writeln('Error');

   with sAddr do
   begin
      Family := af_inet;
      Port := htons(RemotePort);
      Addr := StrToAddr(RemoteAddress);
   end;

   if not Connect(Sock, sAddr, sizeof(sAddr)) then Writeln('Connect: habrahabr.ru
↪'); // Установка соединения с сервером
   Sock2Text(Sock, sin, sout);
   Reset(sin);
   Rewrite(sout);

   // Отправка запроса
   Writeln('Connected.');
   Readln(sin, Line);
   Writeln(Line);
   repeat
      Write('> ');
      Readln(Line);
      // Вывод ответа
      Writeln(sout, Line);
      if Line <> 'close' then
      begin
         Readln(sin, Line);
         Writeln(Line);
      end;
   until Line = 'close';
   Close(sin);
   Close(sout);
   Shutdown(Sock, 2);
end
```

1.4 Парсим HTML

См.также:

https://en.wikipedia.org/wiki/Web_scraping

DOM:

- https://ru.wikipedia.org/wiki/Document_Object_Model

CSS Selector:

- https://www.w3.org/TR/selectors/
- http://www.w3schools.com/cssref/css_selectors.asp
- https://ru.wikibooks.org/wiki/CSS/\T2A\CYRS\T2A\cyre\T2A\cyrl\T2A\cyre\T2A\cyrk\T2A\cyrt\T2A\cyro\T2A\cyrr\T2A\cyrery

XPath:

- https://ru.wikipedia.org/wiki/XPath
- https://addons.mozilla.org/ru/firefox/addon/firepath/
- http://ejohn.org/blog/xpath-css-selectors/

Для разбора Веб-страниц HTML/XML текст представляют в виде дерева объектов (DOM), к элементам которого можно обращаться при помощи разных механизмов:

- CSS
- XPath
- JQuery
- FireFox

1.4.1 Браузеры

Встроенные отладчик

См.также:

- https://devtools-html.github.io/debugger.html/
- https://developer.mozilla.org/ru/docs/Tools/Debugger
- https://developer.mozilla.org/ru/docs/Tools

Прекрасный, быстрый и многофункциональный отладчик написанный на современных технологиях React и Redux.

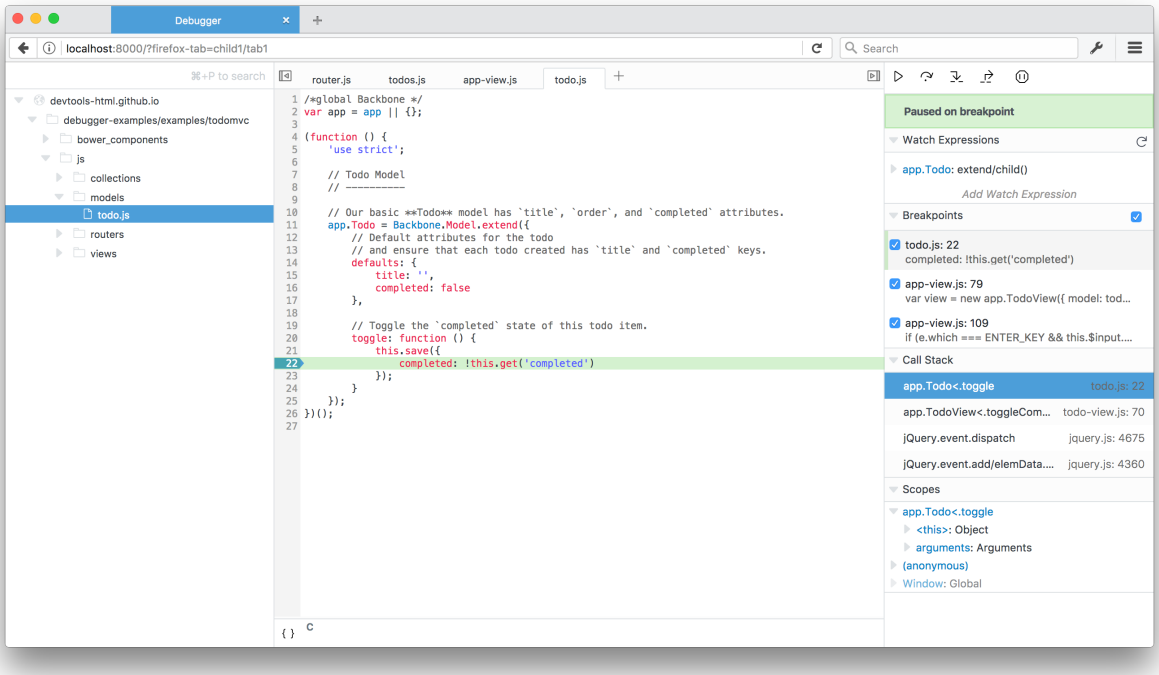


Рис. 1: Встроенный отладчик в FireFox

Анализ производительности

См.также:

https://perf-html.io/

perf.html - дополнение к браузеру FireFox для анализа производительности веб-страниц (performance profile).

FireBug

> **Предупреждение:** Устарел и больше не развивается. Кодовая база была портированна во встроенный отладчик.

FirePath

См.также:

https://addons.mozilla.org/firefox/addon/firepath/

Поиск всех картинок-аватаров на странице при помощи XPath:

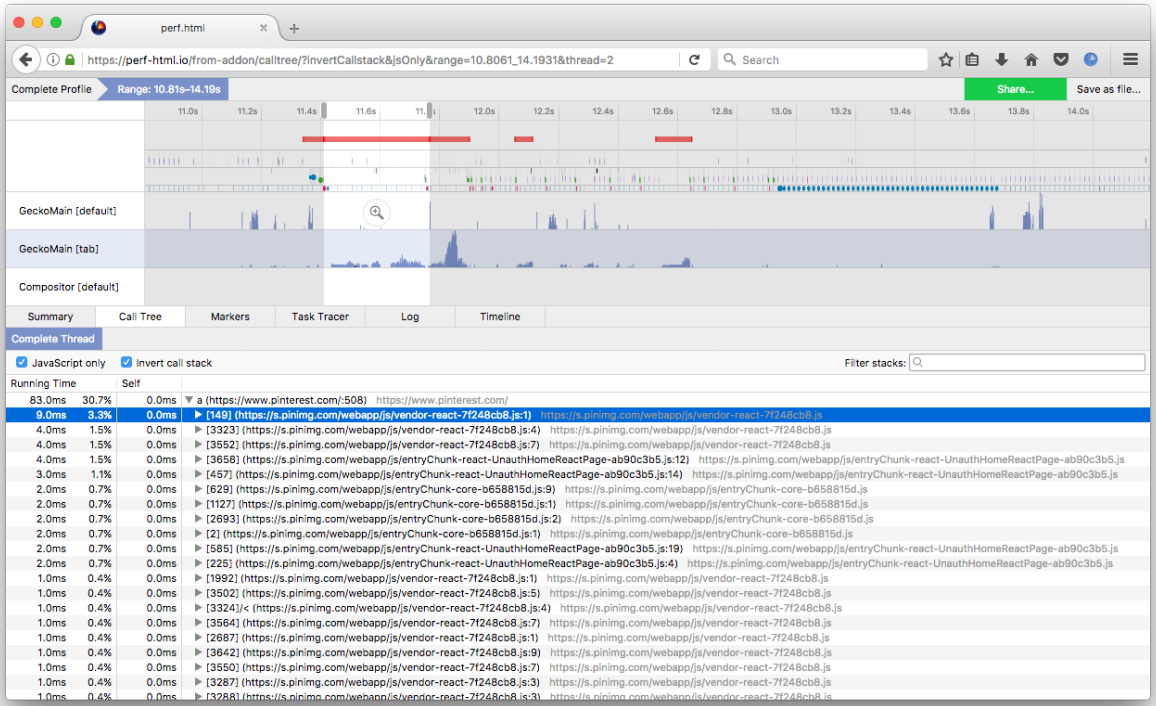


Рис. 2: perf.html - плагин в FireFox для анализа производительности

Поиск всех картинок-аватаров на странице при помощи CSS:

1.4.2 Selenium

См.также:

- https://ru.wikipedia.org/wiki/Selenium
- http://www.seleniumhq.org/projects/webdriver/
- https://kreisfahrer.gitbooks.io/selenium-webdriver/
- http://selenium-python.readthedocs.org/

Selenium Webdriver - инструмент для автоматизации реального браузера, как локально, так и удаленно, наиболее близко имитирующий действия пользователя.

```
from selenium import webdriver

driver = webdriver.Firefox()

# Открыли страницу
```

(continues on next page)

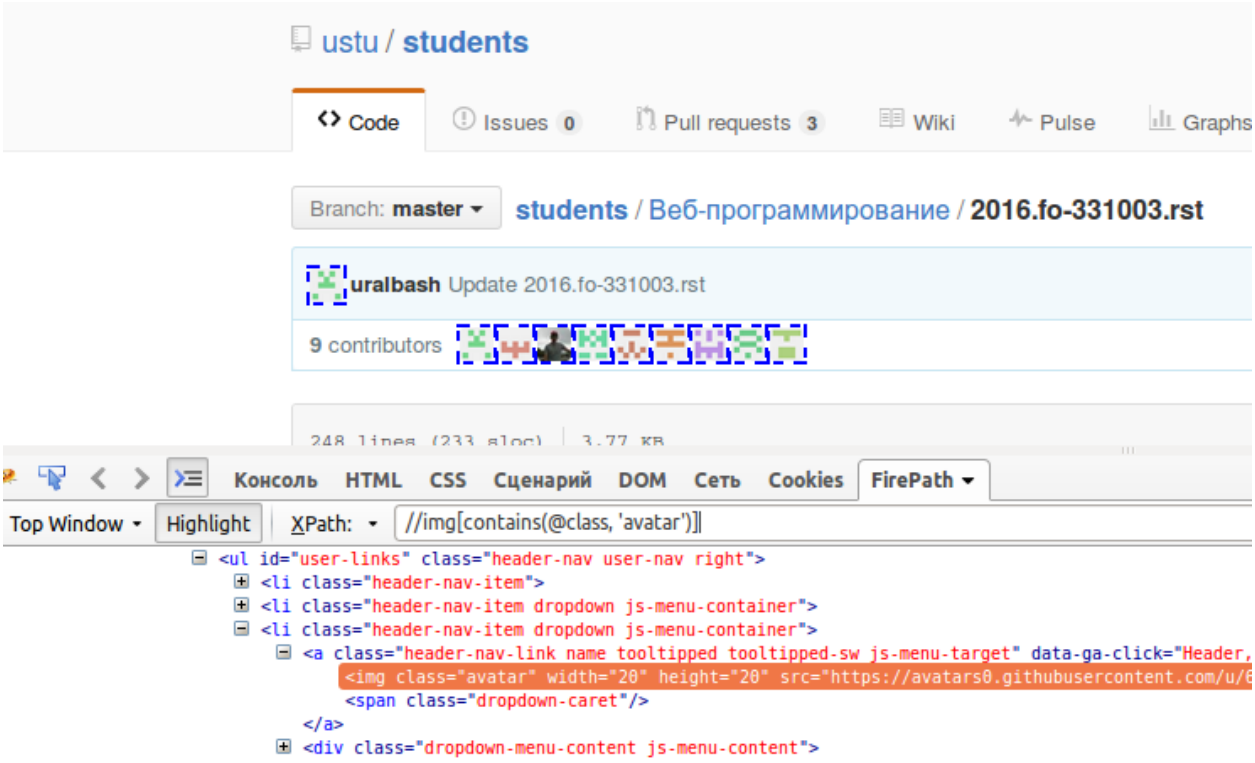


Рис. 3: FirePath - плагин в FireFox для работы с XPath

(продолжение с предыдущей страницы)

```
driver.get(
    'https://github.com/ustu/students/blob/master/' +
    'Веб-программирование/2016.fo-331002.rst'
)

# Нашли иконки всех пользователей
contributors = driver.find_elements_by_xpath(
    "//img[contains(@class, 'avatar')]"
)

# Вывели логины
for user in contributors:
    print(user.get_attribute("alt"))

# Кликнули мышкой по последнему юзеру
user.click()
```

1.4.3 lxml.html

См.также:

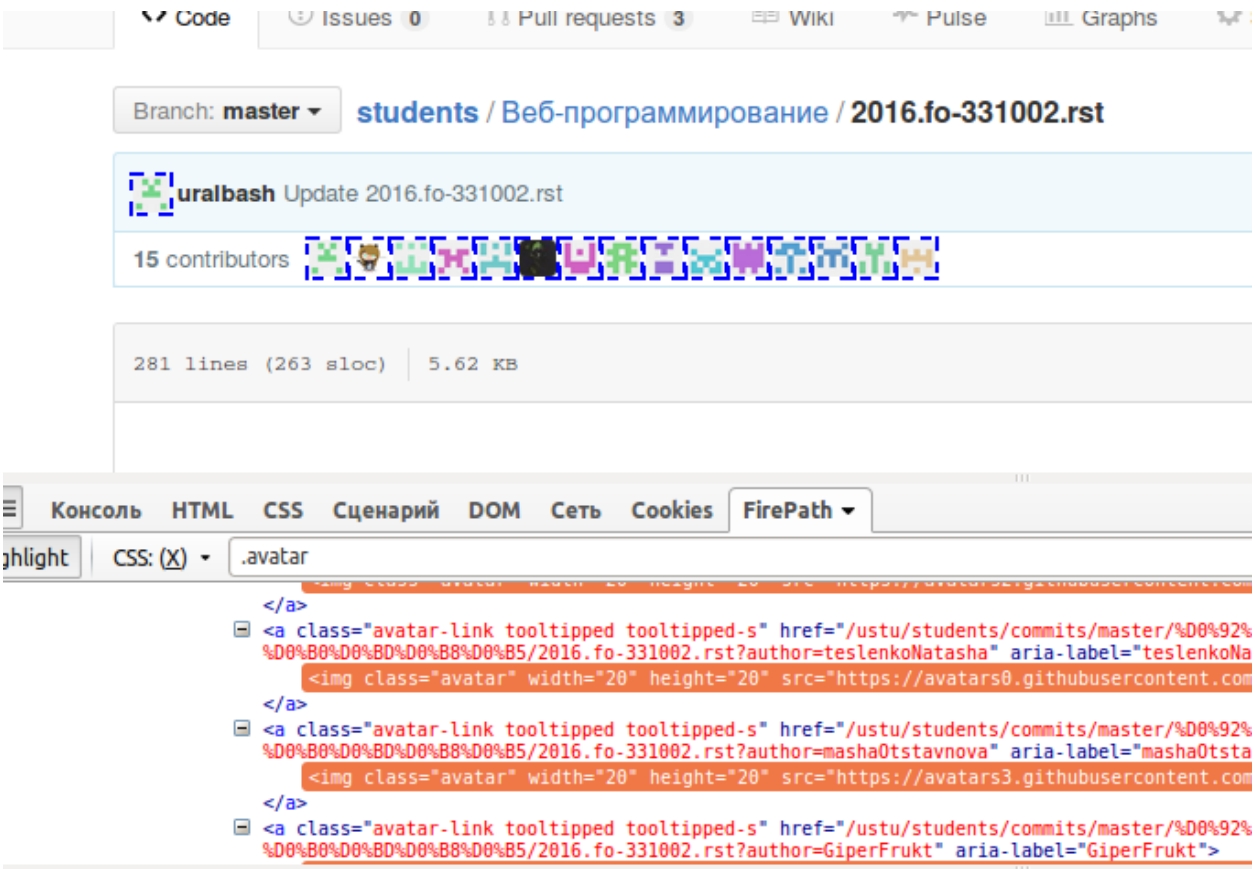


Рис. 4: CSS selector в FireBug

- http://devacademy.ru/posts/parsing-resursov-pri-pomoschi-python/

Разбор HTTP ответа

Довольно легко распарсить HTML код полученный при помощи lxml. Как только мы преобразовали данные в дерево, можно использовать XPath для их извлечения.

```
import requests
from lxml import html

response = requests.get('http://ya.ru')

# Преобразование тела документа в дерево элементов (DOM)
parsed_body = html.fromstring(response.text)

# Выполнение xpath в дереве элементов
print(parsed_body.xpath('//title/text()')[0])  # Получить title страницы
print(parsed_body.xpath('//a/@href'))          # Получить аттрибут href для всех␣
↪ссылок
```

Пример извлекает название *HTML* страницы и все ссылки найденные в этом документе.

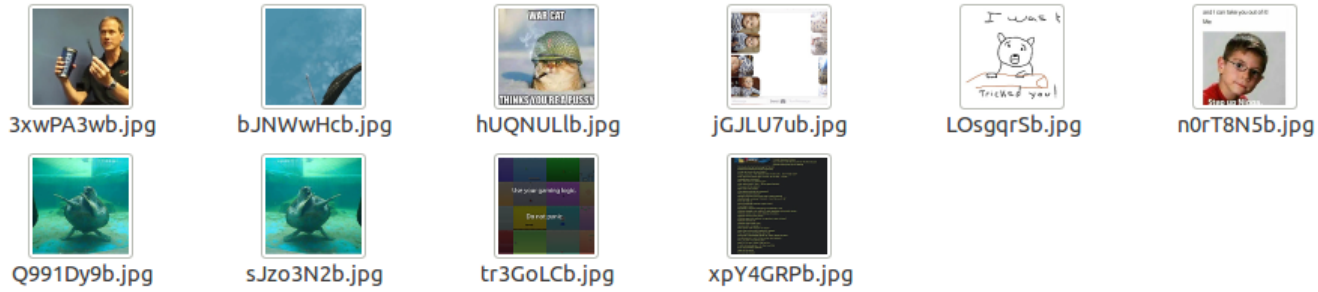
```
Яндекс
['https://mail.yandex.ru', '//www.yandex.ru']
```

Скачиваем все изображения со страницы

Следующий скрипт скачает все изображения и сохранит их в директории `downloaded_images/`. Только сначала не забудьте создать соответствующий каталог. Чтобы сформировать полный путь по относительным ссылкам используется функция `urljoin()` из стандартного модуля а `urllib.parse`.

```
# standard library
import sys
from pathlib import Path
from urllib.parse import urljoin

# third-party
import requests
from lxml import html

response = requests.get('http://imgur.com/')
parsed_body = html.fromstring(response.text)

# Парсим ссылки с картинками при помощи XPath
images = parsed_body.xpath('//img/@src')
if not images:
    sys.exit("images Not Found")

# Конвертирование всех относительных ссылок в абсолютные
images = [
    urljoin(response.url, url)
    for url in images
]
print('Found {} images'.format(len(images)))

# Скачиваем только первые 10
for url in images[0:10]:
    r = requests.get(url)
    target = Path(
        'downloaded_images/{}'.format(
            url.split('/')[-1]  # file name from URL
        )
    )
    target.write_bytes(r.content)
```

После выполнения скрипта в каталоге окажутся скачанные изображения.



CSS selector

См.также:

https://developer.mozilla.org/ru/docs/Web/CSS/CSS_\T2A\CYRS\T2A\cyre\T2A\cyrl\T2A\cyre\T2A\cyrk\T2A\cyrt\T2A\cyro\T2A\cyrr\T2A\cyrery

Селектор определяет, к какому элементу применять то или иное *CSS* правило. При помощи селекторов можно делать выборки дерева объектов в *HTML* документе.

Пример ниже показывает, как использовать *CSS* селекторы используя дерево элементов *HTML* документа, полученного при помощи библиотеки *lxml*.

```
# standard library
from io import StringIO

# third-party
import requests
from lxml import html

r = requests.get('http://ru.arf.ru/')

print(r.encoding)      # Кодировка по умолчанию ISO-8859-1
r.encoding = 'cp1251' # Указываем настоящую кодировку документа

# Формируем дерево элементов
root = html.parse(
    StringIO(r.text)
).getroot()

# Выбираем ссылки внутри тегов <td> выровненных по центру
links = root.cssselect('td[align=CENTER] > a')

for item in links:
    print(
```

(continues on next page)

(продолжение с предыдущей страницы)

```
        item.get('href')  # печатаем значение атрибута href
    )

print()

# Выбираем все изображения
images = root.cssselect('img')
for item in images:
    print(
        item.get('src'),     # печатаем значение атрибута src
        html.tostring(item)  # и сам элемент
    )
```

Результат выполнения:

```
$ python parse.py
ISO-8859-1
/
/Ludi/index.html
/Hrono/index.html
/Svid/index.html
/Links/index.html
mailto:sova@arf.ru
/Ludi/index.html
/Hrono/index.html
/Svid/index.html
/Links/index.html
mailto:sova@arf.ru

/Pict/ru.png b'<img src="/Pict/ru.png" height="80" width="120" alt="RU History Logo
↪" border="0">'
/Pict/line.png b'<img src="/Pict/line.png" height="3" width="550" alt="divider">
↪\r\n          '
/Pict/line.png b'<img src="/Pict/line.png" height="3" width="550" alt="divider">
↪\r\n          '
```

Фильтры

См.также:

http://lxml.de/api/lxml.html-module.html

lxml имеет множество инструментов для обработки полученных данных. Возьмем ради эксперимента один из самых старых сайтов http://infocity.kiev.ua, дата регистрации 1999 год.

По понятным причинам он использует табличную верстку. Пример ниже показывает как выбрать из таблицы «Программирование» все строки, при помощи смежных элементов в *CSS* селекторе.

```
# standard library
from io import StringIO

# third-party
import requests
from lxml import html, etree

r = requests.get('http://infocity.kiev.ua/section51.php')
print(r.encoding)  # ISO-8859-1
r.encoding = 'cp1251'

root = html.parse(
    StringIO(r.text)
).getroot()

# Выбираем нужную таблицу через смежные селекторы и записи этой таблицы начиная
# с 4 строки
rows = root.cssselect('table ~ form ~ table ~ table tr + tr + tr + tr')
```

Далее будем работать только с первой строкой таблицы.

```
item = rows[0]

# Печатаем первый элемент результата поиска
print(
    html.tostring(
        item,
        encoding='unicode'
    )
)
print()

'''
<tr>
    <td>Использование комбинаторных функций в модуле itertools</td>
    <td>Дэвид Мертц</td>
<!--    <td>0kb</td> -->
    <td>12.11.2004</td>

    <td align="center" valign="middle">
        <table cellpadding="0" cellspacing="0" style="margin:0px">
        <tr>
            <td valign="middle">
```

(continues on next page)

(продолжение с предыдущей страницы)

```
                <a href="prog/python/content/python020.phtml" target="_top"><img␣
↪src="images/open.gif" border="0" width="17" height="17"></a>
            </td>
            <td valign="middle">
                <a href="prog/python/content/python020.phtml" target="_blank"><img␣
↪src="images/nopen.gif" border="0" width="17" height="17"></a>
            </td>
        </tr>
        </table>
    </td>
    </tr>
'''
```

Метод `text_content` отбрасывает все тэги и оставляет только содержимое.

```
# Печатаем только текст
print(item.text_content().strip())
print()

'''
Использование комбинаторных функций в модуле itertools
    Дэвид Мертц

    12.11.2004
'''
```

См.также:

http://lxml.de/api/lxml.html.clean.Cleaner-class.html

Для комплексной фильтрации есть мощный класс `Cleaner`, который позволяет задать настройки фильтра и многократно применять их к разным элементам.

```
from lxml.html.clean import Cleaner
cleaner = Cleaner(
    scripts=True,       # Удаляет все js скрипты <script>
    comments=True,      # Удаляет все комментарии
    allow_tags=['br', 'td', 'tr', 'img'],  # Список тэгов которые не нужно удалять
    remove_unknown_tags=False
)
print(
    cleaner.clean_html(  # применяем фильтр
        html.tostring(
            item,
            encoding='unicode'
        )
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    )
)
print()

'''
<tr>
    <td>Использование комбинаторных функций в модуле itertools</td>
    <td>Дэвид Мертц</td>

    <td>12.11.2004</td>

    <td align="center" valign="middle">

        <tr>
            <td valign="middle">
                <img src="images/open.gif" border="0" width="17" height="17">
            </td>
            <td valign="middle">
                <img src="images/nopen.gif" border="0" width="17" height="17">
            </td>
        </tr>

    </td>
    </tr>
'''
```

Более того, наш элемент тоже является деревом элементов, поэтому мы можем производить в нем поиск. Попробуем выбрать все элементы <td> выравненные по центру.

```
# Ищем элемент td выровненный по центру
item_td = item.cssselect('td[align=center]')[0]
print(
    html.tostring(
        item_td,
        encoding='unicode'
    )
)
print()

'''
<td align="center" valign="middle">
        <table cellpadding="0" cellspacing="0" style="margin:0px">
        <tr>
            <td valign="middle">
                <a href="prog/python/content/python020.phtml" target="_top"><img␣
↪src="images/open.gif" border="0" width="17" height="17"></a>
```

(continues on next page)

(продолжение с предыдущей страницы)

```
            </td>
            <td valign="middle">
                <a href="prog/python/content/python020.phtml" target="_blank"><img␣
↪src="images/nopen.gif" border="0" width="17" height="17"></a>
            </td>
        </tr>
        </table>
    </td>

<td>
'''
```

Атрибуты корневого тэга доступны в свойстве `attrib`. Вызовем метод `clear`, чтобы их очистить.

```
# Удаляем все атрибуты корневого тэга
item_td.attrib.clear()
print(
    html.tostring(
        item_td,
        encoding='unicode'
    )
)
print()

'''
<td>
        <table cellpadding="0" cellspacing="0" style="margin:0px">
        <tr>
            <td valign="middle">
                <a href="prog/python/content/python020.phtml" target="_top">
↪src="images/open.gif" border="0" width="17" height="17"></a>
            </td>
            <td valign="middle">
                <a href="prog/python/content/python020.phtml" target="_blank"><img␣
↪src="images/nopen.gif" border="0" width="17" height="17"></a>
            </td>
        </tr>
        </table>
    </td>
'''
```

Элемент содержит много разных свойств, например `tag` позволяет изменить название тэга.

```
# Меняем название тэга
item_td.tag = 'my_custom_tag_td'
print(
    html.tostring(
        item_td,
        encoding='unicode'
    )
)
print()

'''
<my_custom_tag_td>
        <table cellpadding="0" cellspacing="0" style="margin:0px">
        <tr>
            <td valign="middle">
                <a href="prog/python/content/python020.phtml" target="_top"><img␣
↪src="images/open.gif" border="0" width="17" height="17"></a>
            </td>
            <td valign="middle">
                <a href="prog/python/content/python020.phtml" target="_blank"><img␣
↪src="images/nopen.gif" border="0" width="17" height="17"></a>
            </td>
        </tr>
        </table>
    </my_custom_tag_td>
'''
```

Модуль `etree` содержит функции преобразования дерева. `strip_tags` удаляет список тэгов но оставляет их содержимое.

```
# Удаляем список тегов, не трогая их содержимое
etree.strip_tags(item_td, 'a', 'b', 'c')
print(
    html.tostring(
        item_td,
        encoding='unicode'
    )
)
print()

'''
<my_custom_tag_td>
        <table cellpadding="0" cellspacing="0" style="margin:0px">
        <tr>
            <td valign="middle">
                <img src="images/open.gif" border="0" width="17" height="17">
```

(continues on next page)

(продолжение с предыдущей страницы)

```
            </td>
            <td valign="middle">
                <img src="images/nopen.gif" border="0" width="17" height="17">
            </td>
        </tr>
        </table>
    </my_custom_tag_td>
'''
```

`strip_elements` удаляет список тэгов вместе с содержимым.

```
# Удаляем тэг со всеми потрахами
etree.strip_elements(item_td, 'td')
print(
    html.tostring(
        item_td,
        encoding='unicode'
    )
)
print()

'''
<my_custom_tag_td>
        <table cellpadding="0" cellspacing="0" style="margin:0px">
        <tr>
            </tr>
        </table>
    </my_custom_tag_td>
'''
```

Полный код:

```
# standard library
from io import StringIO

# third-party
import requests
from lxml import html, etree

r = requests.get('http://infocity.kiev.ua/section51.php')
print(r.encoding)  # ISO-8859-1
r.encoding = 'cp1251'

root = html.parse(
    StringIO(r.text)
```

(continues on next page)

(продолжение с предыдущей страницы)

```
).getroot()

# Выбираем нужную таблицу через смежные селекторы и записи этой таблицы
# начиная с 4 строки
rows = root.cssselect('table ~ form ~ table ~ table tr + tr + tr + tr')
print(len(rows))
print()

item = rows[0]

# Печатаем первый элемент результата поиска
print(
    html.tostring(
        item,
        encoding='unicode'
    )
)
print()

# Печатаем только текст
print(item.text_content().strip())
print()

# http://lxml.de/api/lxml.html.clean.Cleaner-class.html#_tag_link_attrs
from lxml.html.clean import Cleaner
cleaner = Cleaner(
    scripts=True,       # Удаляет все js скрипты <script>
    comments=True,      # Удаляет все комментарии
    allow_tags=['br', 'td', 'tr', 'img'],  # Список тэгов которые не нужно удалять
    remove_unknown_tags=False
)
print(
    cleaner.clean_html(  # применяем фильтр
        html.tostring(
            item,
            encoding='unicode'
        )
    )
)
print()

# Ищем элемент td выровненный по центру
item_td = item.cssselect('td[align=center]')[0]
print(
    html.tostring(
```

(continues on next page)

(продолжение с предыдущей страницы)

```
        item_td,
        encoding='unicode'
    )
)
print()

# Удаляем все атрибуты корневого тэга
item_td.attrib.clear()
print(
    html.tostring(
        item_td,
        encoding='unicode'
    )
)
print()

# Меняем название тэга
item_td.tag = 'my_custom_tag_td'
print(
    html.tostring(
        item_td,
        encoding='unicode'
    )
)
print()


# Удаляем список тегов, не трогая их содержимое
etree.strip_tags(item_td, 'a', 'b', 'c')
print(
    html.tostring(
        item_td,
        encoding='unicode'
    )
)
print()

# Удаляем тэг со всеми потрахами
etree.strip_elements(item_td, 'td')
print(
    html.tostring(
        item_td,
        encoding='unicode'
    )
)
print()
```

1.4.4 aiohttp

См.также:

- http://aiohttp.readthedocs.org/
- http://lxml.de/cssselect.html
- https://pythonhosted.org/cssselect/

```
import lxml.html as lhtml

import aiohttp
import asyncio


URLS = {
    'imgurl': 'http://imgur.com/',
    'flickr': 'https://www.flickr.com/photos/tags/pretty'
}

HEADERS = {
    'User-Agent':
    "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:40.0) Gecko/20100101 Firefox/41.2"
}


def get_images(html):
    doc = lhtml.document_fromstring(html)
    return doc.cssselect('img')


async def print_img_src(name, url):
    async with aiohttp.request('GET', url, headers=HEADERS) as response:
        if response.status == 200:
            images = get_images(await response.read())
            for img in images:
                if 'src' in img.attrib:
                    print(img.attrib['src'])

tasks = [print_img_src(name, url)
         for name, url in URLS.items()]

loop = asyncio.get_event_loop()
loop.run_until_complete(asyncio.wait(tasks))
loop.close()
```

1.4.5 Red lang

См.также:

- https://gist.github.com/maximvl/c6335b52ac3a4ee1d780afbf3da636c4
- http://www.red-lang.org/2013/11/041-introducing-parse.html

```
Red []
{
grammar HTML
    document          <-   (doctype / text / tag)*
    tag               <-   open_tag (text / tag)* close_tag
    open_tag          <-   "<" [0-9a-zA-Z \"'=-]+ ">"
    close_tag         <-   "</" [0-9a-zA-Z]+ ">"
    doctype           <-   "<!DOCTYPE " [0-9a-zA-Z]+ ">"
    text              <-   [^<]+
}

ws: charset reduce [newline space tab]
digits: charset {0123456789}
chars: union charset [#"a" - #"z"] charset [#"A" - #"Z"]
alphanum: union digits chars
alphanum-with-specials: union ws union alphanum charset {"'=-}

tags-stack: copy []

handle-open-tag:  func [name] [
  append tags-stack name
  ;print ["open" name]
  print tags-stack
]
handle-close-tag: func [name] [
  take/last tags-stack
  ;print ["close" name]
  print tags-stack
]

document: [any [ahead "<" [ tag | doctype ] | text]]
tag: [whitespace open-tag any [ahead not "<" text | tag] close-tag]
open-tag: ["<" copy name tag-name (handle-open-tag name) any tag-parameter ">"]
tag-name: [some alphanum]
tag-parameter: [whitespace some alphanum opt ["=" "^"" some [not "^"" skip] "^""] ]
close-tag: ["</" copy name tag-name (handle-close-tag name) ">"]
doctype: ["<!DOCTYPE " some alphanum ">"]
text: [any [not "<" skip]]
whitespace: [any ws]
```

(continues on next page)

(продолжение с предыдущей страницы)

```
html: {
<html>
<body>
<img src="picture1.jpg" alt="<title>"></img>тут точно не тайтл<img src="picture2.
↪jpg" alt="</title>"></img>
<img src="picture1.jpg" alt="<u>"></img>тут точно не подчеркнуто<img src="picture2.
↪jpg" alt="</u>"></img>
<u>а тут подчеркнуто</u>
</body>
</html>
}

probe parse html document
```

1.4.6 Qt

1.5 Низкоуровневое программирование

См.также:

- Книги Стивенса
- Модуль socket в Python
- https://docs.python.org/3/howto/sockets.html
- Сетевое программирование в python
- http://beej.us/guide/bgnet/output/html/multipage/index.html

1.5.1 Введение

IP

См.также:

- https://ru.wikipedia.org/wiki/IP-\T2A\cyra\T2A\cyrd\T2A\cyrr\T2A\cyre\T2A\cyrs

IP-адрес (айпи-адрес, сокращение от англ. Internet Protocol Address) — это уникальный сетевой адрес узла в компьютерной сети, построенной по протоколу IP. В сети Интернет требуется глобальная уникальность адреса; в случае работы в локальной сети требуется

уникальность адреса в пределах сети. В версии протокола IPv4 IP-адрес имеет длину 4 байта, в IPv6 — 16 байт.

Для сети класса A...
(один байт - поле сети, следующие за ним - номер хоста)

10.0.0.0 адрес сети класса A, потому что все биты адреса узла равны 0
10.0.1.0 адрес узла этой сети
10.255.255.255 широковещательный адрес этой сети, потому что все биты адреса узла равны 1

Для сети класса B...
(два байта - поле сети, следующие за ним - номер хоста)

172.17.0.0 адрес сети класса B
172.17.0.1 адрес узла этой сети
172.17.255.255 широковещательный адрес этой сети

Для сети класса C...
(три байта - поле сети, следующие за ним - номер хоста)

192.168.3.0 адрес сети класса C
192.168.3.42 адрес узла этой сети
192.168.3.255 широковещательный адрес этой сети

Порт

См.также:

- http://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml
- Wiki Порт

Номер порта является 16-разрядным целым двоичным числом, таким образом, порты возможны в диапазоне от 1 до 65535 (для TCP, номер порта 0 зарезервирован и не может быть использован). Для UDP порт источника не является обязательным и нулевое значение означает отсутствие порта.

- FTP: 21 для команд, 20 для данных
- SSH: 22 — удалённое управление хостом с ОС UNIX/Linux/BSD (если разрешено)

- telnet: 23 — подключение для удалённого управления сетевыми устройствами (если такой сервис поддерживается)
- SMTP: 25
- SMTP с SSL: 465, 587
- DNS: 53 (UDP) — необходимо для работы большинства служб Интернета на ПК обычного пользователя
- DHCP: 67, 68/UDP
- TFTP: 69/UDP
- HTTP: 80, 8080 — если закрыть, браузер не будет отображать страницы
- SNMP: 161/UDP, 162/UDP - стандартный интернет-протокол для управления устройствами в IP-сетях на основе архитектур TCP/UDP. К поддерживающим SNMP устройствам относятся маршрутизаторы, коммутаторы, серверы, рабочие станции, принтеры, модемные стойки и другие.

```
$ cat /etc/services
```

```
$ getent services http
http                  80/tcp www
```

Поиск сервисов на Python, при помощи `socket.getservbyname()`, `socket.getservbyport()`:

```
>>> import socket
>>> socket.getservbyport(80)
'http'
>>> socket.getservbyport(21)
'ftp'
>>> socket.getservbyport(53, 'udp')
'domain'
>>> socket.getservbyname('http')
80
```

На C:

См.также:

https://www.opennet.ru/man.shtml?topic=getservbyname&category=3

Компиляция:

```
$ gcc getservbyname.c -o getservbyname
```

```
#include <stdlib.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <stdio.h>

int main(int argc, char *argv[])
{
   struct servent *serv;

   if (argc < 3) {
      puts("Incorrect parameters. Use:");
      puts("   gsbnm service-name protocol-name");
      return EXIT_FAILURE;
   }

   /* getservbyname() - opens the etc.services file and returns the */
   /* values for the requested service and protocol.                */

   serv = getservbyname(argv[1], argv[2]);
   if (serv == NULL) {
      printf("Service \"%s\" not found for protocol \"%s\"\n",
         argv[1], argv[2]);
      return EXIT_FAILURE;
   }

   /* Print it. */
   printf("Name: %-15s  Port: %5d    Protocol: %-6s\n",
             serv->s_name,ntohs(serv->s_port),serv->s_proto);
   return EXIT_SUCCESS;
}
```

Пример использования:

```
$ ./getservbyname http tcp
Name: http             Port:    80    Protocol: tcp
```

Сокет

См.также:

- http://citforum.ru/internet/articles/art_12.shtml

- https://ru.wikipedia.org/wiki/\T2A\CYRS\T2A\cyro\T2A\cyrk\T2A\cyre\T2A\cyrt_(\T2A\cyrp\T2A\cyrr\T2A\cyro\T2A\cyrg\T2A\cyrr\T2A\cyra\T2A\cyrm\T2A\cyrm\T2A\cyrn\T2A\cyrery\T2A\cyrishrt_\T2A\cyri\T2A\cyrn\T2A\cyrt\T2A\cyre\T2A\cyrr\T2A\cyrf\T2A\cyre\T2A\cyrishrt\T2A\cyrs)
- https://ru.wikipedia.org/wiki/\T2A\CYRS\T2A\cyro\T2A\cyrk\T2A\cyre\T2A\cyrt\T2A\cyrery_\T2A\CYRB\T2A\cyre\T2A\cyrr\T2A\cyrk\T2A\cyrl\T2A\cyri
- https://ru.wikipedia.org/wiki/\T2A\CYRS\T2A\cyro\T2A\cyrk\T2A\cyre\T2A\cyrt_\T2A\cyrd\T2A\cyro\T2A\cyrm\T2A\cyre\T2A\cyrn\T2A\cyra_UNIX

Для обеспечения сетевых коммуникаций используются `сокеты`. `Сокет` это конечная точка сетевых коммуникаций. Каждый использующийся сокет имеет тип и ассоциированный с ним процесс.

Сокеты существуют внутри коммуникационных доменов. Домены это абстракции, которые подразумевают конкретную структуру адресации и множество протоколов, которое определяет различные типы сокетов внутри домена. Примерами коммуникационных доменов могут быть: `UNIX` домен, `Internet` домен, и т.д.

Обычно клиент явно подсоединяется к слушателю, после чего любое чтение или запись через его файловый дескриптор будут передавать данные между ним и сервером.

Файловый дескриптор

См.также:

- https://ru.wikipedia.org/wiki/\T2A\CYRF\T2A\cyra\T2A\cyrishrt\T2A\cyrl\T2A\cyro\T2A\cyrv\T2A\cyrery\T2A\cyrishrt_\T2A\cyrd\T2A\cyre\T2A\cyrs\T2A\cyrk\T2A\cyrr\T2A\cyri\T2A\cyrp\T2A\cyrt\T2A\cyro\T2A\cyrr

Ко всем потокам ввода-вывода (которые могут быть связаны как с файлами, так и с каталогами, сокетами и FIFO) можно получить доступ через так называемые файловые дескрипторы. Файловый дескриптор — это неотрицательное целое число. Когда создается новый поток ввода-вывода, ядро возвращает процессу, создавшему поток ввода-вывода, его файловый дескриптор.

В программе, сокет идентифицируется дескриптором - это просто переменная типа int.

Файловая система UNIX

1.5.2 Межпроцессное взаимодействие

См.также:

- https://ru.wikipedia.org/wiki/\T2A\CYRM\T2A\cyre\T2A\cyrzh\T2A\cyrp\T2A\cyrr\T2A\cyro\T2A\cyrc\T2A\cyre\T2A\cyrs\T2A\cyrs\T2A\cyrn\T2A\cyro\T2A\cyre_\T2A\cyrv\T2A\cyrz\T2A\cyra\T2A\cyri\T2A\cyrm\T2A\cyro\T2A\cyrd\T2A\cyre\T2A\cyrishrt\T2A\cyrs\T2A\cyrt\T2A\cyrv\T2A\cyri\T2A\cyre
- http://heap.altlinux.org/tmp/unix_base/ch01s03.html
- https://docs.python.org/3/library/ipc.html
- Презентация с лекций

Межпроцессное взаимодействие (англ. Inter-Process Communication, IPC) — набор способов обмена данными между множеством потоков в одном или более процессах. Процессы могут быть запущены на одном или более компьютерах, связанных между собой сетью. IPC-способы делятся на методы обмена сообщениями, синхронизации, разделяемой памяти и удаленных вызовов (RPC).

Методы IPC зависят от пропускной способности и задержки взаимодействия между потоками и типа передаваемых данных.

Передача данных через файл

См.также:

- https://ru.wikipedia.org/wiki/\T2A\CYRP\T2A\cyre\T2A\cyrr\T2A\cyre\T2A\cyrn\T2A\cyra\T2A\cyrp\T2A\cyrr\T2A\cyra\T2A\cyrv\T2A\cyrl\T2A\cyre\T2A\cyrn\T2A\cyri\T2A\cyre_\T2A\cyrv\T2A\cyrv\T2A\cyro\T2A\cyrd\T2A\cyra-\T2A\cyrv\T2A\cyrery\T2A\cyrv\T2A\cyro\T2A\cyrd\T2A\cyra

Конвейер

См.также:

- https://ru.wikipedia.org/wiki/\T2A\CYRK\T2A\cyro\T2A\cyrn\T2A\cyrv\T2A\cyre\T2A\cyrishrt\T2A\cyre\T2A\cyrr_(UNIX)
- http://pymotw.com/2/pipes/index.html

При помощи конвейера можно передавать вывод одной команды на ввод другой. В примере показан конвейер команд: fortune, cowsay, sed, shuf.

```
$ fortune | cowsay -f `cowsay -l | sed '1,1d' | sed 's/ /\n/g' | shuf -n 1`
 ------------------------------------
/ Лучше ничего не делать, чем делать \
| ничего.                            |
|                                    |
\ -- Л.Н.Толстой                     /
```

(continues on next page)

(продолжение с предыдущей страницы)

```
------------------------------------
    \         __------~~-,
     \      ,'            ,
           /               \
          /                :
         |                  '
         |                   |
         |                   |
          |   _--            |
          _| =-.     .-.    ||
          o|/o/       _.    |
          /  ~          \ |
        (____@)  ___~     |
           |_===~~~.`     |
        _______.--~      |
        \_______         |
                \        |
              __/-___-- -__
             /             _ \
```

Именованный канал

См.также:

- https://ru.wikipedia.org/wiki/\T2A\CYRI\T2A\cyrm\T2A\cyre\T2A\cyrn\T2A\cyro\T2A\cyrv\T2A\cyra\T2A\cyrn\T2A\cyrn\T2A\cyrery\T2A\cyrishrt_\T2A\cyrk\T2A\cyra\T2A\cyrn\T2A\cyra\T2A\cyrl

В программировании именованный канал или именованный конвейер (англ. named pipe) — один из методов межпроцессного взаимодействия, расширение понятия конвейера в Unix и подобных ОС.

Именованный канал позволяет различным процессам обмениваться данными, даже если программы, выполняющиеся в этих процессах, изначально не были написаны для взаимодействия с другими программами.

Пример передачи «Hello World»

Создаем именованный канал при помощи утилиты mkfifo:

```
mkfifo pipe
```

Проверяем тип файла:

```
$ file pipe
pipe: fifo (named pipe)
```

Слушаем канал:

```
cat < pipe
```

```
echo "Hello World" > pipe
```

«Hello World» на Python

```
# sender.py

import os

path = "/tmp/my_program.fifo"
os.mkfifo(path)

fifo = open(path, "w")
fifo.write("Hello World!\n")
fifo.close()
```

```
# receiver.py

import os
import sys

path = "/tmp/my_program.fifo"
fifo = open(path, "r")
for line in fifo:
    print("Получено: %s" % line)
fifo.close()
```

```
Полученно: Hello World!
```

Пример сжатия полученных данных

Можно создать канал и настроить gzip на сжатие того, что туда попадает:

```
mkfifo pipe
gzip -9 -c < pipe > out
```

```
cat file > pipe
```

В файле `out` запишутся переданные данные в сжатом виде.

Обычный файл как транспорт

В отличии от каналов, обычные файлы используют жесткий диск, а не ОЗУ что гораздо медленнее.

Создадим файл, через который будет происходить обмен.

```
$ touch pipe.txt
```

Будем получать данные (смотреть изменение) с помощью команды tail.

```
$ tail -f pipe.txt
```

Отправим данные обычным редактированием файла.

```
$ echo 'Привет' >> pipe.txt
$ echo 'файловая труба!' >> pipe.txt
```

Результат:

```
$ # Полученные данные
$ tail -f pipe.txt
Привет
файловая труба!

$ # Записанные данные в файле
$ cat pipe.txt
Привет
файловая труба!
```

Реализация *tail -f* на Python

```
import time

# Open a file
file = open("pipe.txt", "r")
print("Name of the file: %s" % file.name)

while True:
    where = file.tell()
    line = file.readline()
```

(continues on next page)

(продолжение с предыдущей страницы)

```
10      if not line:
11          time.sleep(1)
12          file.seek(where)
13      else:
14          print(line)  # already has newline
```

Сокеты

См.также:

- https://ru.wikipedia.org/wiki/\T2A\CYRS\T2A\cyro\T2A\cyrk\T2A\cyre\T2A\cyrt_(\T2A\cyrp\T2A\cyrr\T2A\cyro\T2A\cyrg\T2A\cyrr\T2A\cyra\T2A\cyrm\T2A\cyrm\T2A\cyrn\T2A\cyrery\T2A\cyrishrt_\T2A\cyri\T2A\cyrn\T2A\cyrt\T2A\cyre\T2A\cyrr\T2A\cyrf\T2A\cyre\T2A\cyrishrt\T2A\cyrs)
- https://ru.wikipedia.org/wiki/\T2A\CYRS\T2A\cyro\T2A\cyrk\T2A\cyre\T2A\cyrt\T2A\cyrery_\T2A\CYRB\T2A\cyre\T2A\cyrr\T2A\cyrk\T2A\cyrl\T2A\cyri
- http://rsdn.ru/article/unix/sockets.xml
- http://pymotw.com/2/socket/index.html
- http://masandilov.ru/network/guide_to_network_programming

`Сокеты` (англ. socket — разъём) — название программного интерфейса для обеспечения обмена данными между процессами. Процессы при таком обмене могут исполняться как на одной ЭВМ, так и на различных ЭВМ, связанных между собой сетью. Сокет — абстрактный объект, представляющий конечную точку соединения.

Принципы сокетов

Каждый процесс может создать слушающий сокет (серверный сокет) и привязать его к какому-нибудь порту операционной системы (в UNIX непривилегированные процессы не могут использовать порты меньше 1024). Слушающий процесс обычно находится в цикле ожидания, то есть просыпается при появлении нового соединения. При этом сохраняется возможность проверить наличие соединений на данный момент, установить тайм-аут для операции и т.д.

Каждый сокет имеет свой адрес. ОС семейства UNIX могут поддерживать много типов адресов, но обязательными являются INET-адрес и UNIX-адрес. Если привязать сокет к UNIX-адресу, то будет создан специальный файл (файл сокета) по заданному пути, через который смогут сообщаться любые локальные процессы путём чтения/записи из него (см. Доменный сокет Unix). Сокеты типа INET доступны из сети и требуют выделения номера порта.

Обычно клиент явно подсоединяется к слушателю, после чего любое чтение или запись через его файловый дескриптор будут передавать данные между ним и сервером.

Основные функции

Общие	
Socket	Создать новый сокет и вернуть файловый дескриптор
Send	Отправить данные по сети
Receive	Получить данные из сети
Close	Закрыть соединение
Серверные	
Bind	Связать сокет с IP-адресом и портом
Listen	Объявить о желании принимать соединения. Слушает порт и ждет когда будет установлено соединение
Accept	Принять запрос на установку соединения
Клиентские	
Connect	Установить соединение

socket()

См.также:

- http://unixhelp.ed.ac.uk/CGI/man-cgi?socket+2
- https://docs.python.org/3.5/library/socket.html#socket.socket

Создаёт конечную точку соединения и возвращает файловый дескриптор. Принимает три аргумента:

1. **domain** указывающий семейство протоколов создаваемого сокета
   - **AF_INET** для сетевого протокола IPv4
   - **AF_INET6** для IPv6
   - **AF_UNIX** для локальных сокетов (используя файл)
2. **type**
   - **SOCK_STREAM** (надёжная потокоориентированная служба (сервис) или потоковый сокет)

- **SOCK_DGRAM** (служба датаграмм или датаграммный сокет)
- **SOCK_RAW** (Сырой сокет — сырой протокол поверх сетевого уровня).

3. **protocol**

   Протоколы обозначаются символьными константами с префиксом **IPPROTO_*** (например, **IPPROTO_TCP** или **IPPROTO_UDP**). Допускается значение protocol=0 (протокол не указан), в этом случае используется значение по умолчанию для данного вида соединений.

Примечание: Функция возвращает 1 в случае ошибки. Иначе, она возвращает целое число, представляющее присвоенный дескриптор.

Пример на Си

```
#include <sys/types.h>
#include <sys/socket.h>

int socket(int domain, int type, int protocol);
```

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>

void error(const char *msg)
{
    perror(msg);
    exit(0);
}

int main(int argc, char *argv[])
{
    int sockfd, portno, n;
    struct sockaddr_in serv_addr;
    struct hostent *server;

    char buffer[256];
    if (argc < 3) {
       fprintf(stderr,"usage %s hostname port\n", argv[0]);
       exit(0);
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    }

    // Задаем номер порта
    portno = atoi(argv[2]);

    // Создаем сокет
    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd < 0)
        error("ERROR opening socket");

    // Конвертирует имя хоста в IP адрес
    server = gethostbyname(argv[1]);
    if (server == NULL) {
        fprintf(stderr,"ERROR, no such host\n");
        exit(0);
    }

    // Указываем тип сокета Интернет
    bzero((char *) &serv_addr, sizeof(serv_addr));
    serv_addr.sin_family = AF_INET;

    // Указаваем адрес IP сокета
    bcopy((char *)server->h_addr,
         (char *)&serv_addr.sin_addr.s_addr,
         server->h_length);

    // Указываем порт сокета
    serv_addr.sin_port = htons(portno);

    // Устанавливаем соединение
    if (connect(sockfd, (struct sockaddr *) &serv_addr, sizeof(serv_addr)) < 0)
        error("ERROR connecting");

    // Вводим сообщение из консоли
    printf("Please enter the message: ");
    bzero(buffer, 256);
    fgets(buffer, 255, stdin);

    // Отправляем данные
    n = write(sockfd, buffer, strlen(buffer));
    if (n < 0)
         error("ERROR writing to socket");

    // Сбрасываем буфер
    bzero(buffer, 256);
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    // Читаем ответ
    n = read(sockfd, buffer, 255);
    if (n < 0)
         error("ERROR reading from socket");
    printf("%s\n", buffer);

    close(sockfd);
    return 0;
}
```

Пример на Python

```
import socket

# Создание объекта сокета.
sock_obj = socket.socket(socket.AF_INET, socket.SOCK_STREAM, 0)

# AF_INET, SOCK_STREAM и 0 используются по умолчанию при создании сокета.
# Поэтому можно просто писать:
sock_obj = socket.socket()
```

bind()

См.также:

- http://unixhelp.ed.ac.uk/CGI/man-cgi?bind+2
- https://docs.python.org/3.5/library/socket.html#socket.socket.bind

Связывает сокет с конкретным адресом. Когда сокет создается при помощи socket(), он ассоциируется с некоторым семейством адресов, но не с конкретным адресом. До того как сокет сможет принять входящие соединения, он должен быть связан с адресом. bind() принимает три аргумента:

1. **sockfd** — дескриптор, представляющий сокет при привязке
2. **serv_addr** — указатель на структуру sockaddr, представляющую адрес, к которому привязываем.
3. **addrlen** — поле socklen_t, представляющее длину структуры sockaddr.

Примечание: Возвращает 0 при успехе и 1 при возникновении ошибки.

Пример на Си

```
#include <sys/types.h>
#include <sys/socket.h>

int bind(int sockfd, const struct sockaddr *my_addr, socklen_t addrlen);
```

Пример на Python

```
server_address = ('localhost', 8080)
sock_obj.bind(server_address)  # Привязка адреса и порта к сокету.
```

Автоматическое получение имени хоста.

```
host = socket.gethostname()      # Получить имя локальной машины.
server_address = (host, 8080)
sock_obj.bind(server_address)    # Привязка адреса и порта к сокету.
```

listen()

См.также:

- http://unixhelp.ed.ac.uk/CGI/man-cgi?listen+2
- https://docs.python.org/3.5/library/socket.html#socket.socket.listen

Подготавливает привязываемый сокет к принятию входящих соединений. Данная функция применима только к типам сокетов SOCK_STREAM и SOCK_SEQPACKET. Принимает два аргумента:

1. **sockfd** — корректный дескриптор сокета.
2. **backlog** — целое число, означающее число установленных соединений, которые могут быть обработаны в любой момент времени. Операционная система обычно ставит его равным максимальному значению.

Примечание: После принятия соединения оно выводится из очереди. В случае успеха возвращается 0, в случае возникновения ошибки возвращается 1.

Пример на Си

```
#include <sys/socket.h>
int listen(int sockfd, int backlog);
```

Пример на Python

```
sock_obj.listen(5)  # Ждем соединение клиента.
```

accept()

См.также:

- http://unixhelp.ed.ac.uk/CGI/man-cgi?accept+2
- https://docs.python.org/3.5/library/socket.html#socket.socket.accept

Используется для принятия запроса на установление соединения от удаленного хоста. Принимает следующие аргументы:

1. **sockfd** — дескриптор слушающего сокета на принятие соединения.
2. **cliaddr** — указатель на структуру sockaddr, для принятия информации об адресе клиента.
3. **addrlen** — указатель на socklen_t, определяющее размер структуры, содержащей клиентский адрес и переданной в accept(). Когда accept() возвращает некоторое значение, socklen_t указывает сколько байт структуры cliaddr использовано в данный момент.

Примечание: Функция возвращает дескриптор сокета, связанный с принятым соединением, или 1 в случае возникновения ошибки.

Пример на Си

```
#include <sys/types.h>
#include <sys/socket.h>
int accept(int sockfd, struct sockaddr *cliaddr, socklen_t *addrlen);
```

Пример на Python

```
conn, addr = sock_obj.accept()  # Установление соединения с клиентом.
```

connect()

См.также:

- http://unixhelp.ed.ac.uk/CGI/man-cgi?connect+2
- https://docs.python.org/3.5/library/socket.html#socket.socket.connect

Устанавливает соединение с сервером.

Некоторые типы сокетов работают без установления соединения, это в основном касается UDP-сокетов. Для них соединение приобретает особое значение: цель по умолчанию для посылки и получения данных присваивается переданному адресу, позволяя использовать такие функции как send() и recv() на сокетах без установления соединения.

Загруженный сервер может отвергнуть попытку соединения, поэтому в некоторых видах программ необходимо предусмотреть повторные попытки соединения.

Примечание: Возвращает целое число, представляющее код ошибки: 0 означает успешное выполнение, а 1 свидетельствует об ошибке.

Пример на Си

```
#include <sys/types.h>
#include <sys/socket.h>
int connect(int sockfd, const struct sockaddr *serv_addr, socklen_t addrlen);
```

Пример на Python

```
server_address = ('192.168.1.100', 8080)
sock_obj.connect(server_address)
```

Передача данных

Для передачи данных можно пользоваться стандартными функциями чтения/записи файлов read и write, но есть специальные функции для передачи данных через сокеты:

- send
- recv
- sendto
- recvfrom
- sendmsg
- recvmsg

Нужно обратить внимание, что при использовании протокола TCP (сокеты типа SOCK_STREAM) есть вероятность получить меньше данных, чем было передано, так как ещё не все данные были переданы, поэтому нужно либо дождаться, когда функция recv возвратит 0 байт, либо выставить флаг MSG_WAITALL для функции recv, что заставит её дождаться окончания передачи. Для остальных типов сокетов флаг MSG_WAITALL ничего не меняет (например, в UDP весь пакет = целое сообщение).

send()

См.также:

- http://unixhelp.ed.ac.uk/CGI/man-cgi?send+2

- https://docs.python.org/3.5/library/socket.html#socket.socket.send

send, **sendto** - отправка данных.

Пример на Си

```
#include <sys/types.h>
#include <sys/socket.h>

ssize_t send(int s, const void *buf, size_t len, int flags);
ssize_t sendto(int  s, const void *buf, size_t len, int flags, const struct␣
↪sockaddr *to, socklen_t tolen);
```

Пример на Python

```
IP = '192.168.1.100'
PORT = 8080

sock_obj.send('Hello World!')
sock_obj.sendto('Hello World!', (IP, PORT))
```

resv()

См.также:

- http://unixhelp.ed.ac.uk/CGI/man-cgi?recv+2
- https://docs.python.org/3.5/library/socket.html#socket.socket.recv

recv, **recvfrom** - чтение данных из сокета.

Пример на Си

```
#include <sys/types.h>
#include <sys/socket.h>

ssize_t recv(int s, void *buf, size_t len, int flags);
ssize_t recvfrom(int s, void *buf, size_t len, int flags, struct sockaddr *from,␣
↪socklen_t *fromlen);
```

Пример на Python

```
BUFFER_SIZE = 1024

data = conn.recv(BUFFER_SIZE)
data, sender_addr = conn.recvfrom(BUFFER_SIZE)
```

SOCK_STREAM vs SOCK_DGRAM

См.также:

- UDP
- TCP

Потоковый (SOCK_STREAM)	Дейтаграммный (SOCK_DGRAM)
Устанавливает соединение	Нет
Гарантирует доставку данных	Нет в случае UDP
Гарантирует порядок доставки пакетов	Нет в случае UDP
Гарантирует целостность пакетов	Тоже
Разбивает сообщение на пакеты	Нет
Контролирует поток данных	Нет

TCP гарантирует доставку пакетов, их очередность, автоматически разбивает данные на пакеты и контролирует их передачу, в отличии от UDP. Но при этом TCP работает медленнее за счет повторной передачи потерянных пакетов и большему количеству выполняемых операций над пакетами. Поэтому там где требуется гарантированная доставка (Веб-браузер, telnet, почтовый клиент) используется TCP, если же требуется передавать данные в реальном времени (многопользовательские игры, видео, звук) используют UDP.

Передача данных через UNIX сокеты

См.также:

- https://ru.wikipedia.org/wiki/\T2A\CYRS\T2A\cyro\T2A\cyrk\T2A\cyre\T2A\cyrt_\T2A\cyrd\T2A\cyro\T2A\cyrm\T2A\cyre\T2A\cyrn\T2A\cyra_UNIX

Сокет домена Unix (англ. Unix domain socket, UDS) или IPC-сокет (сокет межпроцессного взаимодействия) — конечная точка обмена данными, подобная Интернет-сокету, но не использующая сетевой протокол для взаимодействия (обмена данными). Используется в операционных системах, поддерживающих стандарт POSIX, для межпроцессного взаимодействия.

Доменные соединения Unix являются по сути байтовыми потоками, сильно напоминая сетевые соединения, но при этом все данные остаются внутри одного компьютера (то есть обмен данными происходит локально). UDS используют файловую систему как адресное пространство имен, то есть они представляются процессами как иноды в файловой системе. Это позволяет двум различным процессам открывать один и тот же сокет для взаимодействия между собой. Однако, конкретное взаимодействие, обмен данными, не использует файловую систему, а только буферы памяти ядра.

Пример передачи в одну сторону

Сервер

```
import os
import socket

SOCKET_FILE = './echo.socket'

if os.path.exists(SOCKET_FILE):
    os.remove(SOCKET_FILE)

print("Открываем UNIX сокет...")
server = socket.socket(socket.AF_UNIX, socket.SOCK_DGRAM)
server.bind(SOCKET_FILE)

print("Слушаем...")
while True:
    datagram = server.recv(1024)
    if not datagram:
        break
    else:
        print("-" * 20)
    print(datagram)
    if b"DONE" == datagram:
        break
print("-" * 20)
print("Выключение...")
server.close()
os.remove(SOCKET_FILE)
print("Выполнено")
```

Клиент

```
import os
import socket

SOCKET_FILE = './echo.socket'

print("Подключение...")
if os.path.exists(SOCKET_FILE):
    client = socket.socket(socket.AF_UNIX, socket.SOCK_DGRAM)
    client.connect(SOCKET_FILE)
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    print("Выполнено.")
    print("Ctrl-C чтобы выйти.")
    print("Отправьте 'DONE' чтобы выключить сервер.")
    while True:
        try:
            x = input("> ")  # for py2 use raw_input
            if "" != x:
                print("ОТПРАВЛЕНО: %s" % x)
                client.send(x.encode('utf-8'))
                if "DONE" == x:
                    print("Выключение.")
                    break
        except KeyboardInterrupt as k:
            print("Выключение.")
            break
    client.close()
else:
    print("Не могу соединиться!")
print("Выполнено")
```

Схематичное отображение

Передача данных через INET сокеты

TCP пример

См.также:

- https://wiki.python.org/moin/TcpCommunication

Это простой пример эхо-сервера при помощи TCP.

TCP клиент

```
import socket

TCP_IP = '127.0.0.1'
TCP_PORT = 5005
BUFFER_SIZE = 1024
```

(continues on next page)

(продолжение с предыдущей страницы)

```
MESSAGE = b'Hello, World!'

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((TCP_IP, TCP_PORT))
s.send(MESSAGE)
data = s.recv(BUFFER_SIZE)
s.close()

print("received data: {}".format(data))
```

В роли клиента может выступать утилита *telnet*

```
$ telnet localhost 5005
```

TCP сервер

```
import socket

TCP_IP = '127.0.0.1'
TCP_PORT = 5005
BUFFER_SIZE = 20  # Normally 1024, but we want fast response

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind((TCP_IP, TCP_PORT))
s.listen(1)

conn, addr = s.accept()
print("Connection address: {}".format(addr))
while 1:
    data = conn.recv(BUFFER_SIZE)
    if not data:
        break
    print("received data: {}".format(data))
    conn.send(data)  # echo
conn.close()
```

Способы определения длины сообщения:

1. Передать отдельно
2. Читать до разделителя (в http это пустая строка)
3. Передать в заголовке (в http это content-length)
4. Договориться что размер будет фиксированным (как в примере)

5. Читать данные пока не вернется 0

UDP пример

См.также:

- https://wiki.python.org/moin/UdpCommunication

Это простой пример приемо-передачи сообщений при помощи UDP.

UDP клиент

```
import socket

UDP_IP = "127.0.0.1"
UDP_PORT = 5005
MESSAGE = b"Hello, World!"

print("UDP target IP: {}".format(UDP_IP))
print("UDP target port: {}".format(UDP_PORT))
print("message: {}".format(MESSAGE))

sock = socket.socket(socket.AF_INET,      # Internet
                     socket.SOCK_DGRAM)   # UDP
sock.sendto(MESSAGE, (UDP_IP, UDP_PORT))
```

В роли клиента может выступать утилита *netcat*

```
$ nc 127.0.0.1 5005 -u
```

UDP сервер

```
import socket

UDP_IP = "127.0.0.1"
UDP_PORT = 5005

sock = socket.socket(socket.AF_INET,      # Internet
                     socket.SOCK_DGRAM)   # UDP
sock.bind((UDP_IP, UDP_PORT))

while True:
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    data, addr = sock.recvfrom(1024)  # buffer size is 1024 bytes
    print("received message: {}".format(data))
```

Сырые сокеты

См.также:

- https://ru.wikipedia.org/wiki/\T2A\CYRS\T2A\cyrery\T2A\cyrr\T2A\cyro\T2A\cyrishrt_\T2A\cyrs\T2A\cyro\T2A\cyrk\T2A\cyre\T2A\cyrt
- https://github.com/YingquanYuan/raw_sockets
- http://stackoverflow.com/questions/24415294/python-arp-sniffing-raw-socket-no-reply-packets
- http://www.binarytides.com/python-packet-sniffer-code-linux

Сырой сокет - разновидность сокетов Беркли, позволяющий собирать TCP/IP-пакеты, контролируя каждый бит заголовка и отправляя в сеть нестандартные пакеты.

```
import socket
import struct
import binascii

ETH_P_ALL = 0x0003

rawSocket = socket.socket(socket.AF_PACKET, socket.SOCK_RAW,
                          socket.htons(ETH_P_ALL))

while True:

    packet = rawSocket.recvfrom(2048)

    ethernet_header = packet[0][0:14]
    ethernet_detailed = struct.unpack("!6s6s2s", ethernet_header)

    arp_header = packet[0][14:42]
    arp_detailed = struct.unpack("2s2s1s1s2s6s4s6s4s", arp_header)

    # skip non-ARP packets
    ethertype = ethernet_detailed[2]
    if ethertype != '\x08\x06':
        continue

    print("***************_ETHERNET_FRAME_***************")
    print("Dest MAC:        ", binascii.hexlify(ethernet_detailed[0]))
```

(continues on next page)

(продолжение с предыдущей страницы)

```
27     print("Source MAC:      ", binascii.hexlify(ethernet_detailed[1]))
28     print("Type:            ", binascii.hexlify(ethertype))
29     print("***********************************************")
30     print("*****************_ARP_HEADER_*****************")
31     print("Hardware type:   ", binascii.hexlify(arp_detailed[0]))
32     print("Protocol type:   ", binascii.hexlify(arp_detailed[1]))
33     print("Hardware size:   ", binascii.hexlify(arp_detailed[2]))
34     print("Protocol size:   ", binascii.hexlify(arp_detailed[3]))
35     print("Opcode:          ", binascii.hexlify(arp_detailed[4]))
36     print("Source MAC:      ", binascii.hexlify(arp_detailed[5]))
37     print("Source IP:       ", socket.inet_ntoa(arp_detailed[6]))
38     print("Dest MAC:        ", binascii.hexlify(arp_detailed[7]))
39     print("Dest IP:         ", socket.inet_ntoa(arp_detailed[8]))
40     print("***********************************************\n")
```

```
# python 1.raw_socket_sniff.py
***************_ETHERNET_FRAME_***************
Dest MAC:         ffffffffffff
Source MAC:       0024b2841922
Type:             0806
***********************************************
*****************_ARP_HEADER_*****************
Hardware type:    0001
Protocol type:    0800
Hardware size:    06
Protocol size:    04
Opcode:           0001
Source MAC:       0024b2841922
Source IP:        192.168.1.1
Dest MAC:         000000000000
Dest IP:          192.168.1.27
************************************************

***************_ETHERNET_FRAME_***************
Dest MAC:         ffffffffffff
Source MAC:       0024b2841922
Type:             0806
***********************************************
*****************_ARP_HEADER_*****************
Hardware type:    0001
Protocol type:    0800
Hardware size:    06
Protocol size:    04
Opcode:           0001
Source MAC:       0024b2841922
```

(continues on next page)

(продолжение с предыдущей страницы)

```
Source IP:        192.168.1.1
Dest MAC:         000000000000
Dest IP:          192.168.1.27
***********************************************
```

HTTP клиент

```
import socket

# TCP/IP socket
sock_obj = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

sock_obj.connect(('httpbin.org', 80))
sock_obj.send(b"GET /ip HTTP/1.1\r\nHost: httpbin.org\r\n\r\n")

while True:
    resp = sock_obj.recv(1024)
    if not resp:
        break
    print(resp)

# Close the connection when completed
sock_obj.close()
```

```
$ python 1.http_socket.py
HTTP/1.1 200 OK
Server: nginx
Date: Thu, 19 Feb 2015 12:50:07 GMT
Content-Type: application/json
Content-Length: 32
Connection: close
Access-Control-Allow-Origin: *
Access-Control-Allow-Credentials: true

{
  "origin": "82.186.14.112"
}
```

1.5.3 Стек протоколов TCP/IP

DNS

В сети используются системы доменных имен (DNS), для преобразования имени сайта (например lectureswww.readthedocs.org) в серии из 4-х цифр (для IPv4). Первое что нужно сделать программисту это преобразовать доменное имя в IP адрес. В языке программирования Python это можно сделать при помощи модуля `socket`.

Переводим имя хоста в IP адрес

См.также:

- https://docs.python.org/3/library/socket.html#socket.gethostbyname
- http://man7.org/linux/man-pages/man3/gethostbyname.3.html
- https://msdn.microsoft.com/en-us/library/windows/desktop/ms738524(v=vs.85).aspx

```
import socket

# результат из hosts файла
print(socket.gethostbyname('localhost'))

# ваша ОС отправит запрос на удаленный DNS сервер
print(socket.gethostbyname('google.com'))
```

```
127.0.0.1
213.180.204.3
```

Расширенное представление:

```
import socket
print(socket.gethostbyname_ex("localhost"))
print(socket.gethostbyname_ex("google.com"))
print(socket.gethostbyname_ex("www.google.com"))
print(socket.gethostbyname_ex("www.python.org"))
```

Вернет (hostname, aliaslist, ipaddrlist) где hostname основное имя хоста по этому IP, aliaslist список (может быть пустым) альтернативных имен на этом IP, ipaddrlist список IPv4 адресов прикрепленных к этому домену (часто не множество IP).

```
('localhost', [], ['127.0.0.1'])
('google.com', [], ['213.180.204.3'])
('www.google.com', [], ['195.64.213.53', '195.64.213.42', '195.64.213.44', '195.64.
↪213.59', '195.64.213.49', '195.64.213.38', '195.64.213.29', '195.64.213.27',
↪'195.64.213.23', '195.64.213.15', '195.64.213.19', '195.64.213.34', '195.64.213.
↪45', '195.64.213.30', '195.64.213.57'])
```

(continues on next page)

(продолжение с предыдущей страницы)

```
('python.map.fastly.net', ['www.python.org'], ['23.235.43.223'])
```

В реальных программах нужно использовать перехват исключений:

```
import socket
name = "www.python.org"
try:
    host = socket.gethostbyname(name)
    print(host)
except socket.gaierror as err:
    print("cannot resolve hostname: %s %s" % (name, err))
```

Пример DNS обращений к текущим лекциям (lectureswww.readthedocs.org)

```
>>> print(socket.gethostbyname('lectureswww.readthedocs.org'))
162.209.114.75

>>> print(socket.gethostbyname_ex('lectureswww.readthedocs.org'))
('lectureswww.readthedocs.org', [], ['162.209.114.75'])

>>> print(socket.gethostbyaddr('162.209.114.75'))
('readthedocs.org', [], ['162.209.114.75'])
```

Локальное имя машины

```
import socket
print(socket.gethostname())
```

```
my-laptop
```

Получаем fqdn (fully qualified domain name)

См.также:

- http://ru.wikipedia.org/wiki/FQDN

```
>>> import socket

>>> print(socket.getfqdn("8.8.8.8"))
google-public-dns-a.google.com

>>> print(socket.getfqdn("193.107.218.31"))
```

(continues on next page)

(продолжение с предыдущей страницы)

```
193.107.218.31

>>> print(socket.getfqdn("127.0.0.1"))
localhost

>>> print(socket.getfqdn("8.8.4.4"))
google-public-dns-b.google.com
```

HTTP

См.также:

- http://www.binarytides.com/python-socket-programming-tutorial/

```
import socket

tcpsoc = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
tcpsoc.bind(('72.14.192.58', 80)) #bind to googles ip
tcpsoc.send('HTTP REQUEST')
response = tcpsoc.recv()
```

ICMP

Ping на чистом Питоне, используя сырые сокеты

ARP

См.также:

- http://stackoverflow.com/questions/24415294/python-arp-sniffing-raw-socket-no-reply-packets
- https://www.phillips321.co.uk/2012/07/24/python-arp-ping-code/

1.5.4 Альтернативные способы связи

NFC

См.также:

- http://book.itep.ru/4/41/nfc.htm
- https://ru.wikipedia.org/wiki/ISO/IEC_14443

- https://en.wikipedia.org/wiki/Smart_card_application_protocol_data_unit

Bluetooth

Примечание:

- http://homepages.ius.edu/RWISMAN/C490/html/PythonandBluetooth.htm
- http://kevindoran1.blogspot.ru/2013/04/bluetooth-programming-with-python-3.html

---

```
$ hciconfig
hci0:   Type: BR/EDR  Bus: USB
    BD Address: 40:2C:F4:8E:D5:2D  ACL MTU: 1021:8  SCO MTU: 64:1
    UP RUNNING PSCAN ISCAN
    RX bytes:2125 acl:9 sco:0 events:64 errors:0
    TX bytes:1638 acl:9 sco:0 commands:41 errors:0
```

Рассылка

Подписаться Отказаться

Реклама на сайте

Программирование

»»» Python ««««

Наименование	Автор	Да
Использование комбинаторных функций в модуле itertools	Дэвид Мертц	12.11.
Множественная диспетчеризация	Дэвид Мертц (David Mertz)	18.11.
Управление персистентностью Python	О'Брайен П.	09.07.
Используя Psyco, компилятор обработки Python	tr 473 × 34	26.06.
И опять о функциональном программировании на Python	David Mertz	24.04.
Еще о функциональном программировании на Python	Mertz D.	18.04.
Функциональное программирование на языке Python	Mertz D.	10.04.
Семинар по программированию на Python	Гвидо ван Россум	22.10.
Сборник эссе о языке Pyton	Guido van Rossum, пер. Тезадов С.	18.10.
Неформальное введение в программирование на языке Python		04.10.

Inspector Console Debugger Style Editor Performance Me

5 of 12 form ~ table ~ table tr + tr + tr + tr

```
</td>
<td>Дэвид Мертц</td>
<!--<td>0kb</td>-->
<td>12.11.2004</td>
<td valign="middle" align="center">…</td>
</tr>
<tr>
<td>Множественная диспетчеризация</td>
<td>Дэвид Мертц (David Mertz)</td>
<!--<td>21kb</td>-->
<td>18.11.2003</td>
<td valign="middle" align="center">…</td>
</tr>
<tr>…</tr>
<tr>…</tr>
<tr>…</tr>
<tr>…</tr>
<tr>…</tr>
<tr>…</tr>
<tr>…</tr>
<tr>…</tr>
<tr>…</tr>
</tbody>
</table>
<table width="475" cellspacing="0" cellpadding="8" border="0">…</table>
<table width="475" cellspacing="0" cellpadding="8" border="0">…</table>
```

html > body > table > tbody > tr > td > table > tbody > tr

Глава 2

Закрепление материала

2.1 HTTP Запросы/Ответы на разных языках

2.1.1 Цель работы

Получить практические навыки в написании и отладке программ работающих по HTTP протоколу с использованием системного вызова `socket` на языке программирования, имеющем средства работы с сокетами. Изучить возможности различных языков программирования в сетевых задачах.

2.1.2 Задание

См.также:

Список языков программирования.

Написать программу на любом языке программирования, которого нет в списке *Примеры HTTP запросов на разных языках*.

Программа должна выполнять *HTTP* запрос используя вызов `socket`, получать ответ и выводить его в стандартный поток вывода.

2.1.3 Содержание отчета

На каждое задание создать отчет, который должен быть оформлен в виде репозитария на GitHub или заметок на сервисе Gist. В отчете должно быть: исходный код программы, описание последовательности действий, результат выполнения заданий и выводы по работе.

2.2 Работа с HTTP через сокеты

2.2.1 Цель работы

Получить практические навыки по работе с HTTP протоколом при помощи системного вызова *socket*.

2.2.2 Замечания к выполнению

Address already in use

Данная ошибка может возникать при аварийном завершении программы или если в программе забыли прописать закрытие сокета. Как это исправить см. раздел «*Что делать когда возникает ошибка «Address already in use»*».

В ОС *Windows* также доступна команда *netstat*, которая позволяет увидеть текущие соединения.

```
Microsoft Windows [Version 10.0.14393]
(c) Корпорация Майкрософт (Microsoft Corporation), 2016. Все права защищены.

C:\Users\user>netstat -a

Активные подключения

  Имя    Локальный адрес        Внешний адрес          Состояние
  TCP    0.0.0.0:135            DESKTOP-9JPISDO:0      LISTENING
  TCP    0.0.0.0:445            DESKTOP-9JPISDO:0      LISTENING
  TCP    0.0.0.0:3050           DESKTOP-9JPISDO:0      LISTENING
  TCP    0.0.0.0:7680           DESKTOP-9JPISDO:0      LISTENING
  TCP    0.0.0.0:8889           DESKTOP-9JPISDO:0      LISTENING
  TCP    10.0.2.15:139          DESKTOP-9JPISDO:0      LISTENING
  TCP    10.0.2.15:54628        a172-226-117-113:https  ESTABLISHED
  TCP    10.0.2.15:54629        2.19.78.144:http       ESTABLISHED
  TCP    [::]:135               DESKTOP-9JPISDO:0      LISTENING
```

(continues on next page)

(продолжение с предыдущей страницы)

```
UDP    0.0.0.0:3544           *:*
UDP    0.0.0.0:5050           *:*
UDP    10.0.2.15:137          *:*
UDP    [::]:5353              *:*
UDP    [::]:5355              *:*
UDP    [::1]:1900             *:*
UDP    [::1]:60633            *:*
UDP    [fe80::e0aa:34e:fe8c:d651%2]:546  *:*
UDP    [fe80::e0aa:34e:fe8c:d651%2]:1900  *:*
UDP    [fe80::e0aa:34e:fe8c:d651%2]:60632  *:*
```

2.2.3 Задания

Описание заданий находится в разделе Работа с протоколом HTTP через telnet.

Задание 1

См.также:

- http://ruslanspivak.com/lsbaws-part1/

```
myproject/
├── about
│   └── aboutme.html
└── index.html
```

- Написать socket сервер который отдает статикой файлы по HTTP при обращении по IP адресу.
- Файл `aboutme.html` должен быть доступен по ссылке http://localhost:8000/about/aboutme.html
- Файл `index.html` должен быть доступен по ссылке http://localhost:8000/index.html или http://localhost:8000/

Задание 2, 3

1. Реализовать HTTP запросы при помощи модуля `socket`;
2. Реализовать HTTP запросы при помощи модуля `http.client` (или `urllib.request`, или `requests`).

Задание 4

Делать на сокетах не надо.

Задание 5

См.также:

http://www.tcpdump.org/

Отправить следующие параметры POST запросом на сервис http://httpbin.org/post

```
{
    "github": "MyNickName",
    "Name": "MyName",
    "Surname": "MySurname"
}
```

При помощи утилиты tcpdump перехватить трафик с запросом и выложить результат в виде заметок `Gist` от `GitHub`.

2.2.4 Содержание отчета

На каждое задание создать отчет, который должен быть оформлен в виде репозитария на GitHub или заметок на сервисе Gist. В отчете должно быть: исходный код программы, описание последовательности действий, результат выполнения заданий и выводы по работе.

Глава 3

Справочник

См.также:

- https://docs.python.org/3/library/internet.html

3.1 Что делать когда возникает ошибка «Address already in use»

См.также:

- http://hea-www.harvard.edu/~fine/Tech/addrinuse.html
- http://www.cyberciti.biz/faq/what-process-has-open-linux-port/

Не закрытые соединения занимают порт, даже после завершения программы. Обычно операционная система закрывает их сама по таймауту. Рассмотрим варианты как это сделать вручную.

3.1.1 Netstat

netstat показывает сетевую активность системы. Из списка всех сетевых сервисов мы можем отфильтровать интересующий нас:

```
$ sudo netstat -tulpn | grep :8080
tcp   0    0 127.0.0.1:8080     0.0.0.0:*     LISTEN      11778/python
```

Номер процесса, который занимает порт, 11778. Уничтожим его принудительно:

```
$ sudo kill -9 11778
```

Дополнительную информацию о процессе можно получить из директории `/proc`:

```
$ ls /proc/11778/
```

3.1.2 fuser

Утилита fuser ищет номер процесса по порту:

```
$ fuser 8080/tcp
8080/tcp:            11778
```

Флаг `-k` уничтожит процесс:

```
$ fuser -k 8080/tcp
8080/tcp:            11778
```

3.1.3 lsof

lsof - показывает все процессы связанные с файловыми дескрипторами. Флаг `-i` позволяет фильтровать сетевые сокеты:

```
$ lsof -i :8080
COMMAND   PID     USER   FD   TYPE  DEVICE SIZE/OFF NODE NAME
python  11778 uralbash    7u  IPv4 5754939      0t0  TCP localhost:http-alt␣
↪(LISTEN)
```