# Efficient I/O with zero-copy & psutil

利用零拷贝和 psutil 来高效的进行 I/O 操作

Giampaolo Rodola
Pycon China 2019, Shanghai

# Who am I?

- Giampaolo Rodola
- Python core-developer since 2010
- Author of **psutil** library
- Author of **pyftpdlib** (Python FTP server) library
- https://github.com/giampaolo

# Agenda

- Part 1:
  - basic UNIX concepts
  - basic socket operations
  - send files efficiently
  - copy files efficiently
- Part 2:
  - psutil

- 第1部分
  - 基础的 Unix 概念
  - 基础的 Socket 操作
  - 高效的传输文件
  - 高效的复制文件
- 第2部分
  - psutil

# UNIX concepts (oversimplified)

[简单聊聊 Unix 的相关概念]

# System call / 系统调用

- A way for a user-space application to interact with the kernel
- (mostly) exposed in the **os** module

- 用户空间中的应用程序用于与内核交互的手段
- 在 Python 中相关的 API 由 **os** 模块提供

# System calls / 系统调用

**I/O**

- open()
- read()
- write()

**Processes / 进程**

- fork()
- kill()
- wait()

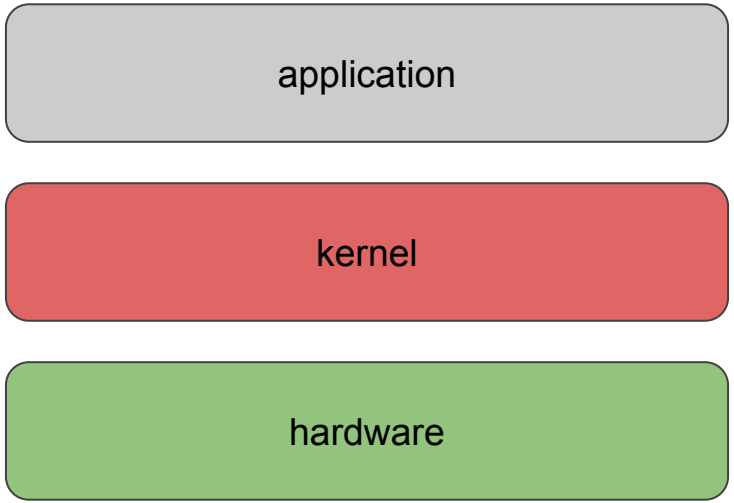
**Filesystem / 文件系统**

- chmod()
- mkdir()
- getcwd()

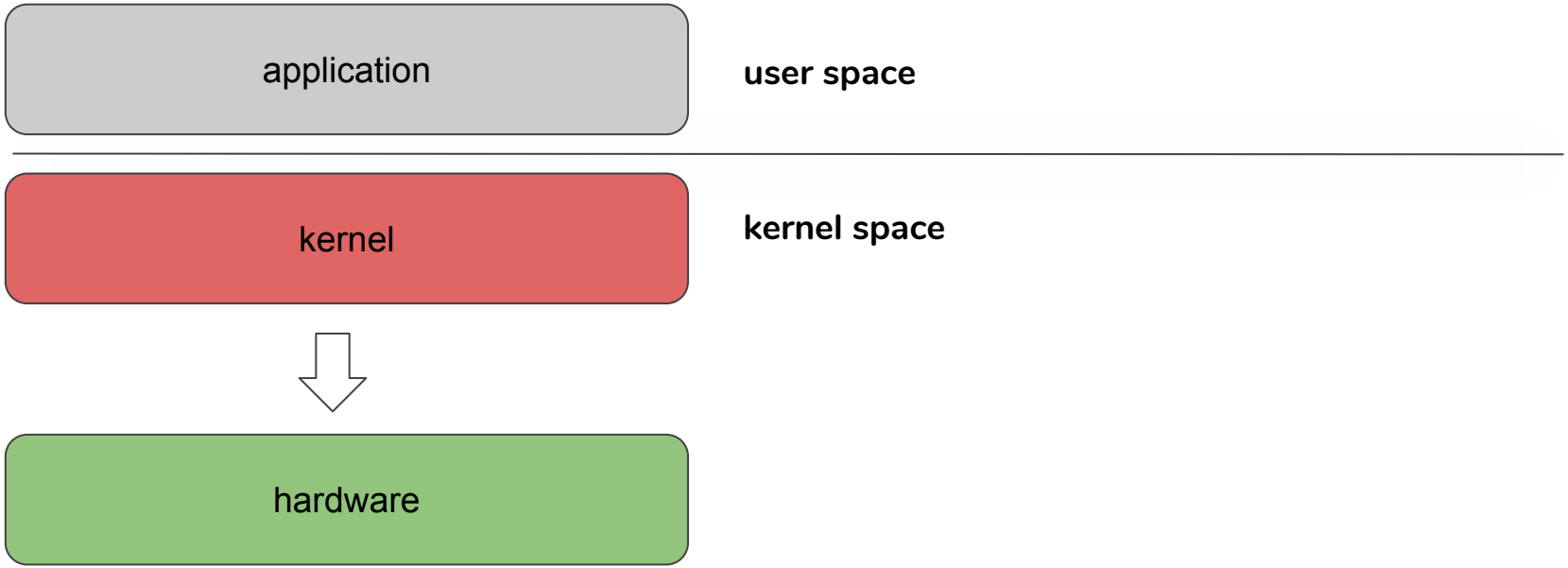
**Communication / 通信**

- pipe()
- splice()
- mmap()

# Kernel / 内核



# User & kernel space / 用户空间 & 内核空间



| User time | Kernel time |
| --- | --- |
| `x = 0`<br>`while x != 10000000:`<br>`    x += 1` | `# generate random string of N length`<br>`import os`<br>`os.urandom(200000000)` |
| $ time python3 script.py<br>real 0m0,752s<br>**user 0m0,752s**<br>sys 0m0,000s | $ time python3 script.py<br>real 0m1,123s<br>user 0m0,012s<br>**sys 0m1,099s** |

# File descriptors

文件描述符

# File descriptors / 文件描述符

- it's a reference to "something" (usually a file)
- it can be mixed with system calls

- 是对文件/套接字等资源的引用
- 可以和系统调用连用

# Print

```
>>> import sys, os
>>> sys.stdout.fileno()
1
>>> os.write(1, b'hello world')
hello world
```

# Disk

```
>>> import os
>>> fd = os.open('file', os.O_WRONLY | os.O_CREAT)
>>> os.write(fd, b'hello')
5
>>> os.close(fd)
>>>
>>> fd = os.open('file', os.O_RDONLY)
>>> os.read(fd, 11)
b'hello'
```

# Terminal

```
>>> # terminal size
>>> import sys, struct, fcntl, termios
>>> s = struct.pack('HHHH', 0, 0, 0, 0)
>>> t = fcntl.ioctl(sys.stdout.fileno(), termios.TIOCGWINSZ, s)
>>> struct.unpack('HHHH', t)
(55, 105, 0, 0)
```

This is why “everything is a file in UNIX”

所以这就是所谓“Unix 下，一切皆文件”的由来

# Summary

- syscall: a gateway to the kernel
- kernel: a gateway to the hardware
- syscalls cause a context switch
- context switches consume time
- syscalls and file descriptors can be mixed together

- 系统调用：与内核交互的途径
- 内核：与硬件交互的途径
- 系统调用将会触发上下文切换
- 上下文切换将会消耗时间
- 系统调用和文件描述符可以连用

# Basic socket operations

基础的 socket 操作

# Server

```
from socket import socket, AF_INET, SOCK_STREAM
sock = socket(AF_INET, SOCK_STREAM)           # IPv4, TCP
sock.bind(("", 8080))                         # all interfaces, port 8080
sock.listen(5)                                # 监听队列
while True:
    conn, addr = sock.accept()                # accept 连接
    # handle connection
```

# Server: IPv4 + IPv6 (Python 3.8)

```
from socket import create_server, AF_INET6
sock = create_server(("", 8080), family=AF_INET6, dualstack_ipv6=True)
while True:
    conn, addr = sock.accept()
    # handle connection/处理连接
```

# Client

```
from socket import socket, AF_INET, SOCK_STREAM
sock = socket(AF_INET, SOCK_STREAM)
sock.connect(("127.0.0.1", 8080))
sock.send(b"hello")
sock.recv(8196)
```

# Sending files

传输文件

# sending a file

```
from socket import create_server, AF_INET6
sock = create_server(("", 8080), family=AF_INET6, dualstack_ipv6=True)
conn, addr = sock.accept()

with open('somefile', 'rb') as file:
    while True:
        chunk = file.read(65536)
        if not chunk:
            break  # EOF
        conn.sendall(chunk)
```

# sending a file

```
from socket import create_server, AF_INET6
sock = create_server(("", 8080), family=AF_INET6, dualstack_ipv6=True)
conn, addr = sock.accept()

with open('somefile', 'rb') as file:
    while True:
        chunk = file.read(65536)  # 2 context switches
        if not chunk:
            break  # EOF
        conn.sendall(chunk)  # 2 context switches
```

# sending a file

```
from socket import create_server, AF_INET6
sock = create_server(("", 8080), family=AF_INET6, dualstack_ipv6=True)
conn, addr = sock.accept()

with open('somefile', 'rb') as file:
    while True:
        chunk = file.read(65536)  # 1 memory copy
        if not chunk:
            break  # EOF
        conn.sendall(chunk)  # 1 memory copy
```

| | read() / send() |
|---|---|
| **system calls** | 2 |
| **context switches** | 4 |
| **memory copies** | 2 |

How can we avoid that?

怎么样去避免这些问题？

# Zero-copy syscalls
# 支持零拷贝的系统调用

- sendfile()
- copy_file_range()
- mmap()
- splice() / vmsplice() / tee()
- KTLS (kernel-space TLS)

# sendfile() (zero-copy)

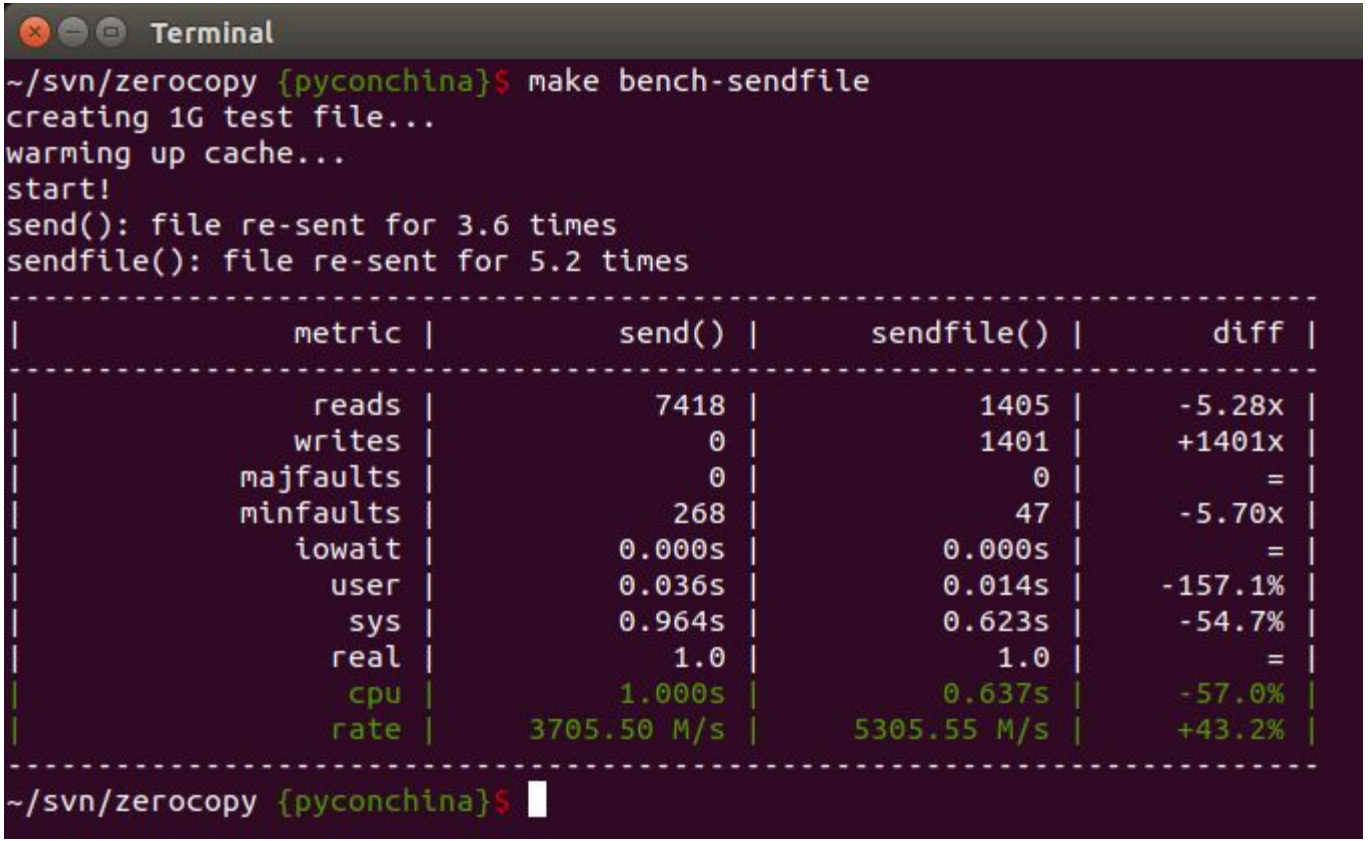
```
import socket, os
sock = socket.create_server(("", 8080))

while True:
    conn, addr = sock.accept()
    with open('somefile', 'rb') as file:
        offset = 0
        while True:
            sent = os.sendfile(conn.fileno(), file.fileno(), offset, 65536)
            if sent == 0:
                break  # EOF
            offset += sent
    conn.close()
```

| | read() / write() | sendfile() |
|---|---|---|
| **system calls** | 2 | 1 |
| **context switches** | 4 | 2 |
| **memory copies** | 2 | 0 |

# How much faster is sendfile()?

Sendfile 到底有多快？


Terminal
~/svn/zerocopy {pyconchina}$ make bench-sendfile
creating 1G test file...
warming up cache...
start!
send(): file re-sent for 3.6 times
sendfile(): file re-sent for 5.2 times
| metric | send() | sendfile() | diff |
| reads | 7418 | 1405 | -5.28x |
| writes | 0 | 1401 | +1401x |
| majfaults | 0 | 0 | = |
| minfaults | 268 | 47 | -5.70x |
| iowait | 0.000s | 0.000s | = |
| user | 0.036s | 0.014s | -157.1% |
| sys | 0.964s | 0.623s | -54.7% |
| real | 1.0 | 1.0 | = |
| cpu | 1.000s | 0.637s | -57.0% |
| rate | 3705.50 M/s | 5305.55 M/s | +43.2% |
~/svn/zerocopy {pyconchina}$

# sendfile() limitations

- can be used with regular files only (no io.BytesIO)
- no SSL (but can use KTLS on Linux 4.13)

- 只能用于常规的文件操作
- 不支持 SSL（比如 Linux 4.13 之后的 KTLS）

# socket.sendfile() utility (Python 3.5)

```
import socket, os
sock = socket.create_server(("", 8080))
while True:
    conn, addr = sock.accept()
    with open('somefile', 'rb') as file:
        conn.sendfile(file)
    conn.close()
```

# Windows TransmitFile (Python 3.9)

- https://bugs.python.org/issue21721

# Copying files (efficiently)

高效拷贝文件

# File copy

```
>>> import shutil
>>> shutil.copyfile('filein', 'fileout')
```

# File copy (Python 3.7)

```
def copyfile(src, dst):
    src = open(src, 'rb')
    dst = open(dst, 'wb')
    while True:
        chunk = src.read(65536)  # 2 ctx switches, 1 memory copy
        if not chunk:
            break                # EOF
        dst.write(chunk)         # 2 ctx switches, 1 memory copy
    src.close()
    dst.close()
```

# File copy on Linux (Python 3.8)

```
# requires Linux >= 2.6.33

def copyfile(src, dst):
    src = open(src, 'rb')
    dst = open(dst, 'wb')
    fsize = os.path.getsize(src)
    offset = 0
    while offset != fsize:
        offset += os.sendfile(dst.fileno(), src.fileno(), offset, fsize)
    src.close()
    dst.close()
```

# sendfile() limitations for files

- regular files only (no io.BytesIO)
- “write” mode only (no “append”)
- files must live on the same filesystem (no NFS)
- no encrypted file-systems (?)

- 只是常规文件 (无 io.BytesIO)
- 只是“改写”模式 (无 “添加”)
- 文件必须存在同一系统中 (无 NFS)
- 无加密的文件系统 (?)

# What about other platforms?

是否适用于其余系统？

# What about other platforms?

- Linux: **sendfile**()
- macOS: **fcopyfile**()
- Windows: **CopyFileEx**()
- https://bugs.python.org/issue33671

# How much faster is sendfile()?

到底有多快？

# Benchmarks

- hot cache
- set highest CPU and disk I/O priority

```
>>> import psutil, os
>>> p = psutil.Process(os.getpid())
>>> p.nice(-20)
>>> p.ionice(psutil.IOPRIO_CLASS_RT, value=7)
```

# shutil.copyfile(): Python 3.7 vs. 3.8

| Size | Linux | Windows | macOS |
|---|---|---|---|
| 128K | +3% | +27% | +8% |
| 8M | +15% | +45% | +47% |
| 512M | +23% | +40% | +50% |

# copy_file_range() (Python 3.9)

- Linux + NFS
- server-side copy
- https://bugs.python.org/issue37159

# Speedup shutil.copytree()

加速 shutil.copytree()

# Copy directory tree

```
>>> import shutil
>>> shutil.copytree('somedir', 'somedir-2')
```

# shutil.copytree()

| **Python 3.7** | **Python 3.8** |
| --- | --- |
| os.**listdir**() + os.**stat**() | os.**scandir**() |
| 7 os.**stat**() calls per file (worst case) | 1 os.**stat**() call per file (best case) |

# 38% less os.stat() syscalls

**8000 files in 4 dirs**

$ strace **python3.7** bench.py  2>&1 | grep "stat(" | wc -l
324808

$ strace **python3.8** bench.py  2>&1 | grep "stat(" | wc -l
198768

# benchmark (8000 files in 4 dirs)

| Platform | Speedup |
| --- | --- |
| Linux | **+8%** |
| Windows | **+20%** |
| Windows (network folder) | **+38%** |

# Part 2: psutil

# psutil

- monitor **system** (CPU, disk, network, temperatures, ...) and **processes**
- cross-platform:
  - Linux
  - Windows
  - macOS
  - FreeBSD, OpenBSD, NetBSD
  - Sun Solaris
  - AIX
- https://github.com/giampaolo/psutil/

# System info

系统信息

# CPU

```
>>> import psutil
>>> psutil.cpu_times()
scputimes(user=17411.7, system=3797.02, idle=51266.57, nice=77.99, iowait=732.58,
          irq=0.01, softirq=142.43, steal=0.0, guest=0.0, guest_nice=0.0)
>>>
>>> psutil.cpu_percent()  # non blocking
2.0
>>> psutil.cpu_percent(interval=1, percpu=True)  # blocking
[2.0, 1.0, 7.6, 8.9]
>>>
```

# CPU

```
>>> psutil.cpu_count()               # with hyper-threading
4
>>> psutil.cpu_count(logical=False)  # physical cores only
2
>>> psutil.cpu_stats()
scpustats(ctx_switches=20455687, interrupts=6598984, soft_interrupts=2134212, syscalls=0)
>>> psutil.cpu_freq(percpu=True)
[scpufreq(current=2394.945, min=800.0, max=3500.0),
 scpufreq(current=2236.812, min=800.0, max=3500.0),
 scpufreq(current=1703.609, min=800.0, max=3500.0),
 scpufreq(current=1754.289, min=800.0, max=3500.0)]
```

# Memory

```
>>> import psutil
>>> psutil.virtual_memory()
svmem(total=10367352832, available=6472179712, percent=37.6, used=8186245120,
      free=2181107712, active=4748992512, inactive=2758115328, buffers=790724608,
      cached=3500347392, shared=787554304, slab=199348224)
>>>
>>> psutil.swap_memory()
sswap(total=2097147904, used=886620160, free=1210527744, percent=42.3, sin=0, sout=0)
```

# Memory

```
import psutil
import time

THRESHOLD = 500 * 1024 * 1024  # 500 MB
last_swap = psutil.swap_memory().sin

def monitor_mem():
    global last_swap
    virt = psutil.virtual_memory()
    if virt.available <= THRESHOLD:
        print("warning: %s bytes of physical mem left" % virt.available)

    swap = psutil.swap_memory().sin
    if swap > last_swap:  # swap activity
        diff = swap - last_swap
        print("warning: %s bytes were swapped to disk since last check" % diff)
    last_swap = swap

while True:
    monitor_mem()
    time.sleep(1)
```

# Disks

```
>>> import psutil
>>> psutil.disk_partitions()
[sdiskpart(device='/dev/sda1', mountpoint='/', fstype='ext4', opts='rw'),
 sdiskpart(device='/dev/sda2', mountpoint='/home', fstype='ext4', opts='rw')]

>>> psutil.disk_usage('/')
sdiskusage(total=21378641920, used=4809781248, free=15482871808, percent=22.5)

>>> psutil.disk_io_counters(perdisk=True)
{'sda1': sdiskio(read_count=988, write_count=2,              # no. of r/w syscalls
                 read_bytes=72972, write_bytes=1024,         # no. of bytes r/w
                 read_time=472, write_time=0,                # time spent r/w from/to disk
                 read_merged_count=0, write_merged_count=0, # no. of merged reads
                 busy_time=8),                               # time spent doing actual I/O
 'sda2': ...}
```

# Disks

```
>>> import time
>>> import psutil
>>> from psutil._common import bytes2human
>>> while True:
...     io1 = psutil.disk_io_counters()
...     time.sleep(1)
...     io2 = psutil.disk_io_counters()
...     bytes_read = io2.read_bytes - io1.read_bytes
...     bytes_written = io2.write_bytes - io1.write_bytes
...     print("%-7s/s %-7s/s" % (bytes2human(bytes_read), bytes2human(bytes_written)))
...
  0.0 B/s     0.0 B/s
595.6 M/s   688.0 K/s
451.4 M/s   279.3 M/s
303.1 M/s   502.4 M/s
```

# Network

```
>>> import psutil
>>> psutil.net_io_counters(pernic=True)
{'lo': snetio(bytes_sent=547971, bytes_recv=547971,
              packets_sent=5075, packets_recv=5075,
              errin=0, errout=0,      # number of errors while sending/receiving
              dropin=0, dropout=0),   # number of packets in/outgoing packets dropped
'wlan0': snetio(bytes_sent=13921765, bytes_recv=62162574,
                packets_sent=79097, packets_recv=89648,
                errin=0, errout=0,
                dropin=0, dropout=0)}
>>>
```

# Network

```
>>> import psutil
>>> psutil.net_connections()
[pconn(fd=115,
       family=<AddressFamily.AF_INET: 2>,  # IPv4
       type=<SocketType.SOCK_STREAM: 1>,   # TCP
       laddr=('10.0.0.1', 46788),
       raddr=('93.186.135.91', 80),
       status='ESTABLISHED',
       pid=1254),
 pconn(fd=117,
       family=<AddressFamily.AF_INET: 2>,   # IPv4
       type=<SocketType.SOCK_STREAM: 1>,    # TCP
       laddr=('10.0.0.1', 43761),
       raddr=('72.14.234.100', 80),
       status='CLOSING',
       pid=2987),
 ...]
```

# Network

```
>>> import psutil
>>> psutil.net_if_addrs()
{'wlan0': [snicaddr(family=<AddressFamily.AF_INET: 2>,    # IPv4
                    address='192.168.1.3',
                    netmask='255.255.255.0',
                    broadcast='192.168.1.255',
                    ptp=None),
           snicaddr(family=<AddressFamily.AF_INET6: 10>,  # IPv6
                    address='fe80::c685:8ff:fe45:641%wlan0',
                    netmask='ffff:ffff:ffff:ffff::',
                    broadcast=None,
                    ptp=None),
           snicaddr(family=<AddressFamily.AF_LINK: 17>,    # MAC
                    address='c4:85:08:45:06:41',
                    netmask=None,
                    broadcast='ff:ff:ff:ff:ff:ff',
                    ptp=None)], 'lo': ... }
```

# Sensors

```
>>> import psutil
>>> psutil.sensors_temperatures()
{'acpitz': [shwtemp(label='', current=47.0, high=103.0, critical=103.0)],
 'asus': [shwtemp(label='', current=47.0, high=None, critical=None)],
 'coretemp': [shwtemp(label='Physical id 0', current=52.0, high=100.0, critical=100.0),
              shwtemp(label='Core 0', current=45.0, high=100.0, critical=100.0),
              shwtemp(label='Core 1', current=52.0, high=100.0, critical=100.0),
              shwtemp(label='Core 2', current=45.0, high=100.0, critical=100.0),
              shwtemp(label='Core 3', current=47.0, high=100.0, critical=100.0)]}
>>>
>>> psutil.sensors_fans()
{'asus': [sfan(label='cpu_fan', current=3200)]}
```

# Sensors

```
>>> import psutil
>>>
>>> def secs2hours(secs):
...     mm, ss = divmod(secs, 60)
...     hh, mm = divmod(mm, 60)
...     return "%d:%02d:%02d" % (hh, mm, ss)
...
>>> bat = psutil.sensors_battery()
>>> bat
sbattery(percent=93, secsleft=16628, power_plugged=False)
>>> print("charge = %s%%, time left = %s" % (bat.percent, secs2hours(bat.secsleft)))
charge = 93%, time left = 4:37:08
```

# Load average

```
>>> import psutil
>>> psutil.getloadavg()
(5.14, 3.89, 3.67)
>>> psutil.cpu_count()
10
>>> [(x / psutil.cpu_count() * 100) for x in psutil.getloadavg()]
(51.4, 38.9, 36.7)  # percentage representation
```

# Processes

进程

# Processes

```
>>> import psutil
>>> psutil.pids()
[1, 2, 3, 4, 5, 6, 7, 46, 48, 50, 51, 178, 182, 222, 223, 224, 268,
 1215, 1216, 1220, 1221, 1243, 1244, 1301, 1601, 2237, 2355, 2637,
 2774, 3932, 4176, 4177, 4185, 4187, 4189, 4225, 4243, 4245, 4263,
 4282, 4306, 4311, 4312, 4313, 4314, 4337, 4339, 4357, 4358, 4363,
 4383, 4395, 4408, 4433, 4443, 4445, 4446, 5167, 5234, 5235, 5252,
 5318, 5424, 5644, 6987, 7054, 7055, 7071]
>>>
>>> p = psutil.Process(7055)
>>> p
psutil.Process(pid=7055, name='python', started='09:04:44')
```

# Basic info

```
>>> p.name()
'python'
>>> p.cmdline()
['/usr/bin/python', 'main.py']
>>> p.exe()
'/usr/bin/python'
>>> p.cwd()
'/home/giampaolo'
>>> p.status()
'running'
>>> p.username()
'giampaolo'
>>> p.uids()
puids(real=1000, effective=1000, saved=1000)
>>> p.gids()
pgids(real=1000, effective=1000, saved=1000)
```

# Basic info

```
>>> p.create_time()
1267551141.5019531
>>> p.terminal()
'/dev/pts/0'
>>> p.ppid()
7054
>>> p.parents()
[psutil.Process(pid=4699, name='bash', started='09:06:44'),
 psutil.Process(pid=1, name='systemd', started='05:56:55')]
>>> p.children(recursive=True)
[psutil.Process(pid=29835, name='python2.7', started='11:45:38'),
 psutil.Process(pid=29836, name='python2.7', started='11:43:39')]
>>> p.environ()
{'LC_PAPER': 'it_IT.UTF-8', 'SHELL': '/bin/bash', 'GREP_OPTIONS': '--color=auto',
'XDG_CONFIG_DIRS': '/etc/xdg/xdg-ubuntu:/usr/share/upstart/xdg:/etc/xdg', ...}
```

# CPU

```
>>> p.cpu_times()
pcputimes(user=1.02, system=0.31, children_user=0.32, children_system=0.1, iowait=0.0)
>>> p.cpu_percent(interval=1.0)
12.1
>>> p.cpu_affinity()
[0, 1, 2, 3]
>>> p.cpu_affinity([0, 1])  # set
>>> p.cpu_num()
1
>>> p.threads()
[pthread(id=5234, user_time=22.5, system_time=9.2891),
 pthread(id=5237, user_time=0.0707, system_time=1.1)]
```

# Counters

```
>>> p.io_counters()
pio(read_count=478001, write_count=59371, read_bytes=700416, write_bytes=69632,
    read_chars=456232, write_chars=517543)
>>> p.num_ctx_switches()
pctxsw(voluntary=78, involuntary=19)
>>>
>>> p.num_threads()
4
>>> p.num_fds()
8
```

# Memory

```
>>> p.memory_maps()
[pmmap_grouped(path='/lib/x8664-linux-gnu/libutil-2.15.so', rss=32768, size=2125824,
               pss=32768, shared_clean=0, shared_dirty=0, private_clean=20480,
               private_dirty=12288, referenced=32768, anonymous=12288, swap=0),
 pmmap_grouped(path='/lib/x8664-linux-gnu/libc-2.15.so', rss=3821568, size=3842048,
               pss=3821568, shared_clean=0, shared_dirty=0, private_clean=0,
               private_dirty=3821568, referenced=3575808, anonymous=3821568, swap=0)
 ...]
>>> p.memory_full_info()
pfullmem(rss=10199040, vms=52133888, shared=3887104, text=2867200, lib=0, data=5967872,
         dirty=0, uss=6545408, pss=6872064, swap=0)
>>> p.memory_percent()
0.7823
```

# Find memory leaks

```
import psutil, os
from cext import some_c_function

TOLERANCE = 4096
TIMES = 100000

def check_leaks(fun):
    p = psutil.Process(os.getpid())
    mem_before = p.memory_full_info().uss
    fds_before = p.num_fds()
    for x in range(TIMES):
        some_c_function()
    mem_after = p.memory_full_info().uss
    fds_after = p.num_fds()
    assert mem_after - mem_before < TOLERANCE, "memory leak"
    assert fds_after == fds_before, "unclosed fd"

check_leaks(some_c_function)
```

# File descriptors

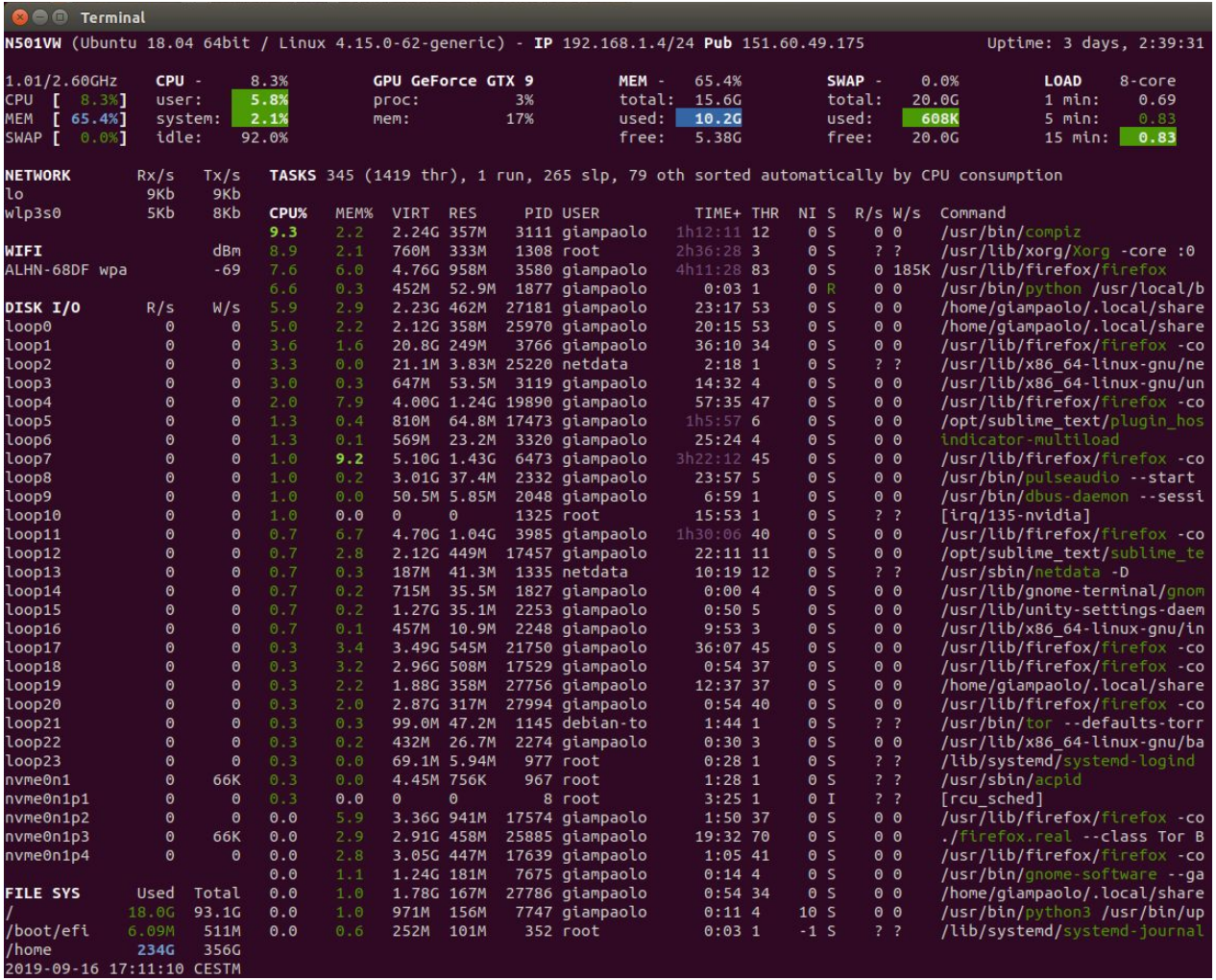
```
>>> p.open_files()
[popenfile(path='/home/giampaolo/monit.py', fd=3, position=0, mode='r', flags=32768),
 popenfile(path='/var/log/monit.log', fd=4, position=235542, mode='a', flags=33793)]
>>>
>>> p.connections()
[pconn(fd=115, family=<AddressFamily.AF_INET: 2>, type=<SocketType.SOCK_STREAM: 1>,
       laddr=addr(ip='10.0.0.1', port=48776), raddr=addr(ip='93.186.135.91', port=80),
       status='ESTABLISHED'),
 pconn(fd=117, family=<AddressFamily.AF_INET: 2>, type=<SocketType.SOCK_STREAM: 1>,
       laddr=addr(ip='10.0.0.1', port=43761), raddr=addr(ip='72.14.234.100', port=80),
       status='CLOSING')]
```

# Signals

```
>>> p.is_running()
True
>>> p.suspend()
>>> p.resume()
>>> p.terminate()
>>> p.kill()
>>> p.wait(timeout=3)
```

# Priority / limits

```
>>> p.nice()
0
>>> p.nice(-20)  # set highest
>>>
>>> p.ionice()
pionice(ioclass=<IOPriority.IOPRIO_CLASS_NONE: 0>, value=4)
>>> p.ionice(psutil.IOPRIO_CLASS_RT, value=7)  # set highest
>>>
>>> p.rlimit(psutil.RLIMIT_NOFILE, (5, 50))  # set resource limits (Linux only)
>>> p.rlimit(psutil.RLIMIT_NOFILE)
(5, 5)
```


Terminal
N501VW (Ubuntu 18.04 64bit / Linux 4.15.0-62-generic) - IP 192.168.1.4/24 Pub 151.60.49.175 Uptime: 3 days, 2:39:31
1.01/2.60GHz CPU - 8.3% GPU GeForce GTX 9 MEM - 65.4% SWAP - 0.0% LOAD 8-core
CPU [ 8.3%] user: 5.8% proc: 3% total: 15.6G total: 20.0G 1 min: 0.69
MEM [ 65.4%] system: 2.1% mem: 17% used: 10.2G used: 608K 5 min: 0.83
SWAP [ 0.0%] idle: 92.0% free: 5.38G free: 20.0G 15 min: 0.83
NETWORK Rx/s Tx/s TASKS 345 (1419 thr), 1 run, 265 slp, 79 oth sorted automatically by CPU consumption
lo 9Kb 9Kb
wlp3s0 5Kb 8Kb CPU% MEM% VIRT RES PID USER TIME+ THR NI S R/s W/s Command
9.3 2.2 2.24G 357M 3111 giampaolo 1h12:11 12 0 S 0 0 /usr/bin/compiz
WIFI dBm 8.9 2.1 760M 333M 1308 root 2h36:28 3 0 S ? ? /usr/lib/xorg/Xorg -core :0
ALHN-68DF wpa -69 7.6 6.0 4.76G 958M 3580 giampaolo 4h11:28 83 0 S 0 185K /usr/lib/firefox/firefox
6.6 0.3 452M 52.9M 1877 giampaolo 0:03 1 0 R 0 0 /usr/bin/python /usr/local/b
DISK I/O R/s W/s 5.9 2.9 2.23G 462M 27181 giampaolo 23:17 53 0 S 0 0 /home/giampaolo/.local/share
loop0 0 0 5.0 2.2 2.12G 358M 25970 giampaolo 20:15 53 0 S 0 0 /home/giampaolo/.local/share
loop1 0 0 3.6 1.6 20.8G 249M 3766 giampaolo 36:10 34 0 S 0 0 /usr/lib/firefox/firefox -co
loop2 0 0 3.3 0.0 21.1M 3.83M 25220 netdata 2:18 1 0 S ? ? /usr/lib/x86_64-linux-gnu/ne
loop3 0 0 3.0 0.3 647M 53.5M 3119 giampaolo 14:32 4 0 S 0 0 /usr/lib/x86_64-linux-gnu/un
loop4 0 0 2.0 7.9 4.00G 1.24G 19890 giampaolo 57:35 47 0 S 0 0 /usr/lib/firefox/firefox -co
loop5 0 0 1.3 0.4 810M 64.8M 17473 giampaolo 1h5:57 6 0 S 0 0 /opt/sublime_text/plugin_hos
loop6 0 0 1.3 0.1 569M 23.2M 3320 giampaolo 25:24 4 0 S 0 0 indicator-multiload
loop7 0 0 1.0 9.2 5.10G 1.43G 6473 giampaolo 3h22:12 45 0 S 0 0 /usr/lib/firefox/firefox -co
loop8 0 0 1.0 0.2 3.01G 37.4M 2332 giampaolo 23:57 5 0 S 0 0 /usr/bin/pulseaudio --start
loop9 0 0 1.0 0.0 50.5M 5.85M 2048 giampaolo 6:59 1 0 S 0 0 /usr/bin/dbus-daemon --sessi
loop10 0 0 1.0 0.0 0 0 1325 root 15:53 1 0 S ? ? [irq/135-nvidia]
loop11 0 0 0.7 6.7 4.70G 1.04G 3985 giampaolo 1h30:06 40 0 S 0 0 /usr/lib/firefox/firefox -co
loop12 0 0 0.7 2.8 2.12G 449M 17457 giampaolo 22:11 11 0 S 0 0 /opt/sublime_text/sublime_te
loop13 0 0 0.7 0.3 187M 41.3M 1335 netdata 10:19 12 0 S ? ? /usr/sbin/netdata -D
loop14 0 0 0.7 0.2 715M 35.5M 1827 giampaolo 0:00 4 0 S 0 0 /usr/lib/gnome-terminal/gnom
loop15 0 0 0.7 0.2 1.27G 35.1M 2253 giampaolo 0:50 5 0 S 0 0 /usr/lib/unity-settings-daem
loop16 0 0 0.7 0.1 457M 10.9M 2248 giampaolo 9:53 3 0 S 0 0 /usr/lib/x86_64-linux-gnu/in
loop17 0 0 0.3 3.4 3.49G 545M 21750 giampaolo 36:07 45 0 S 0 0 /usr/lib/firefox/firefox -co
loop18 0 0 0.3 3.2 2.96G 508M 17529 giampaolo 0:54 37 0 S 0 0 /usr/lib/firefox/firefox -co
loop19 0 0 0.3 2.2 1.88G 358M 27756 giampaolo 12:37 37 0 S 0 0 /home/giampaolo/.local/share
loop20 0 0 0.3 2.0 2.87G 317M 27994 giampaolo 0:54 40 0 S 0 0 /usr/lib/firefox/firefox -co
loop21 0 0 0.3 0.3 99.0M 47.2M 1145 debian-to 1:44 1 0 S ? ? /usr/bin/tor --defaults-torr
loop22 0 0 0.3 0.2 432M 26.7M 2274 giampaolo 0:30 3 0 S 0 0 /usr/lib/x86_64-linux-gnu/ba
loop23 0 0 0.3 0.0 69.1M 5.94M 977 root 0:28 1 0 S ? ? /lib/systemd/systemd-logind
nvme0n1 0 66K 0.3 0.0 4.45M 756K 967 root 1:28 1 0 S ? ? /usr/sbin/acpid
nvme0n1p1 0 0 0.3 0.0 0 0 8 root 3:25 1 0 I ? ? [rcu_sched]
nvme0n1p2 0 0 0.0 5.9 3.36G 941M 17574 giampaolo 1:50 37 0 S 0 0 /usr/lib/firefox/firefox -co
nvme0n1p3 0 66K 0.0 2.9 2.91G 458M 25885 giampaolo 19:32 70 0 S 0 0 ./firefox.real --class Tor B
nvme0n1p4 0 0 0.0 2.8 3.05G 447M 17639 giampaolo 1:05 41 0 S 0 0 /usr/lib/firefox/firefox -co
0.0 1.1 1.24G 181M 7675 giampaolo 0:14 4 0 S 0 0 /usr/bin/gnome-software --ga
FILE SYS Used Total 0.0 1.0 1.78G 167M 27786 giampaolo 0:54 34 0 S 0 0 /home/giampaolo/.local/share
/ 18.0G 93.1G 0.0 1.0 971M 156M 7747 giampaolo 0:11 4 10 S 0 0 /usr/bin/python3 /usr/bin/up
/boot/efi 6.09M 511M 0.0 0.6 252M 101M 352 root 0:03 1 -1 S ? ? /lib/systemd/systemd-journal
/home 234G 356G
2019-09-16 17:11:10 CESTM

# Thanks
# 谢谢

- github: giampaolo
- twitter: grodola
- mail: g.rodola@gmail.com