Creating Agile Unix Command Line Tools With Python

(Using Subprocess, Optparse, Doctest and more)

Noah Gift

PyCon 2008

QUICK WARNING!!!

PLEASE DO NOT SATURATE THE PYCON NETWORK WITH ICMP REQUESTS

WAIT UNTIL YOU GET HOME AND DO IT ON YOUR NEIGHBOR'S OPEN WIRELESS ACCESS POINT

Python Standard Library:
A command line tool's dream come true

- Optparse: Option Handling
- Subprocess: Elegant, One Stop, System Call API
- ConfigParser: Read/Write config files
- Threading/Queue: Concurrency
- Generators: http://www.dabeaz.com/generators/Generators.pdf
- Doctest: Example Driven Testing
- Python Egg's w/Entry Point: Simple way to install scripts

Ideas for Command Line Tools

- Pure Python:
  - No systems calls, but do “Unix stuff”
- Unix Mashups
  - Mix Python + System to solve new problems.
- Network Tools
- Wrapping
  - Use existing tools, change the API, or behavior

The Basics: Subprocess + Optparse

- Subprocess and Optparse:
  - Just like chocolate and peanut butter.
  - Alone they are good, but together they make a tasty treat.

Subprocess: Tips for lazy people

- If you don’t care about stdout, use subprocess.call
- If you want to do something with stdout, use subprocess.Popen

subprocess.call Wrap

Evaluate return code, send output to /dev/null

```
#!/usr/bin/env python
from subprocess import call, STDOUT

def ping(ip="10.0.1.1"):
    ret = call("ping -c 1 %s" % ip,
         shell=True,
         stdout=open('/dev/null', 'w'),
         stderr=STDOUT)
    if ret == 0:
         print "%s: is alive" % ip
    else:
         print "%s: did not respond" % ip

ping()
```

Optparse

Optparse is easy, all the tricky stuff is solved for you

```
#!/usr/bin/env python

import optparse

def main():
    p = optparse.OptionParser()
    p.add_option('--person', '-p', default="world")
    options, arguments = p.parse_args()
    print 'Hello %s' % options.person

if __name__ == '__main__':
    main()
```

Creating a Python/Unix Mashup Tool

Multi-Threaded Ping Sweep

- Wrap system ping command in subprocess
- Create a queue
- Spawn some threads
- It is that easy

Multi-Threaded Ping Sweep

```
#!/usr/bin/env python2.5
from threading import Thread
import subprocess
from Queue import Queue

num_threads = 4
queue = Queue()
ips = ["10.0.1.1", "10.0.1.3", "10.0.1.11", "10.0.1.51"]
#wraps system ping command
def pinger(i, q):
    """Pings subnet"""
    while True:
        ip = q.get()
        print "Thread %s: Pinging %s" % (i, ip)
        ret = subprocess.call("ping -c 1 %s" % ip,
                              shell=True,
                              stdout=open('/dev/null', 'w'),
                              stderr=subprocess.STDOUT)
        if ret == 0:
            print "%s: is alive" % ip
        else:
            print "%s: did not respond" % ip
        q.task_done()
#Spawn thread pool
for i in range(num_threads):

    worker = Thread(target=pinger, args=(i, queue))
    worker.setDaemon(True)
    worker.start()
#Place work in queue
for ip in ips:
    queue.put(ip)
#Wait until worker threads are done to exit
queue.join()
```

Convert It To A Command Line Tool

```
def controller():
    """Runs program and handles command line options"""

    p = optparse.OptionParser(description='Ping sweeps subnet',
                                        prog='ping_thread_cli',
                                        version='ping_thread_cli 0.1',
                                        usage='%prog [10.0.1.1 or 10.0.1.0/24]')

    p.add_option('-n', '--number', help='set number of threads', default=50, type=int)
    options, arguments = p.parse_args()
    if len(arguments) == 1:
        ips = IP(arguments[0])
    else:
        print "subnet/ip required: i.e. 10.0.1.0/24"
        sys.exit(1)
    for i in range(options.number):
        worker = Thread(target=pinger, args=(i, queue, oq))
        worker.setDaemon(True)
        worker.start()
    for ip in ips:
        queue.put(ip)

    queue.join()

if __name__ == "__main__":
    start = time.time()
    controller()
    print "Network Discovered in %s seconds" % str(time.time() - start)
```

Adding SNMP Discovery (another queue, some more threads)

```
num_threads = 50
snmp_threads = 10
queue = Queue()
oq = Queue()

def pinger(i, q, oq):
    """Pings subnet"""
    while True:
        ip = q.get()
        #print "Thread %s: Pinging %s" % (i, ip)
        ret = subprocess.call("ping -c 1 %s" % ip,
                              shell=True,
                              stdout=open('/dev/null', 'w'),
                              stderr=subprocess.STDOUT)
        if ret == 0:
            print "%s: is alive" % ip
            oq.put(ip)
        else:
            pass
        #    print "%s: did not respond" % ip
        q.task_done()

def snmp(i, oq):
    """grabs a valid IP address from a queue and gets macaddr"""
    while True:
        ip = oq.get()
        #print "Thread %s: Doing SNMPQuery %s" % (i, ip)
        ret = subprocess.call("snmpwalk -c public -v 2c %s sysdescr" % ip,
                              shell=True)
        print ret
        oq.task_done()
```

Doctest Example

```
def pinger(i, q, oq):
    """Pings subnet
    >>> q = Queue()
    >>> q.put("127.0.0.1")
    """
    while True:
        ip = q.get()
        #print "Thread %s: Pinging %s" % (i, ip)
        ret = subprocess.call("ping -c 1 %s" % ip,
                              shell=True,
                              stdout=open('/dev/null', 'w'),
                              stderr=subprocess.STDOUT)
        if ret == 0:
            print "%s: is alive" % ip
            oq.put(ip)
        else:
            pass
        #    print "%s: did not respond" % ip
        q.task_done()

def _test():
    """Runs doctests."""
    import doctest
    doctest.testmod(verbose=True)

def controller():
    """Runs program and handles command line options"""

    p = optparse.OptionParser(description='Ping sweeps subnet',
                                        prog='ping_thread_cli',
                                        version='ping_thread_cli 0.1',
                                        usage='%prog [10.0.1.1 or 10.0.1.0/24]')

    p.add_option('-n', '--number', help='set number of threads', default=50, type=int)
    p.add_option('-t', '--test', action="store_true",help='runs doctests')
    options, arguments = p.parse_args()
    #run tests and then exit
    if options.test:
        _test()
        sys.exit(0)
```

A Pure Python Tool Scapy+Storm +Optparse

```
class NetworkRecord(object):
    __storm_table__ = "networkrecord"
    id = Int(primary=True)
    ip = RawStr()
    mac = RawStr()
    hostname = RawStr()

def arping(iprange="10.0.1.0/24"):
    """Arping function takes IP Address or Network, returns nested mac/ip list"""

    conf.verb=0
    ans,unans=srp(Ether(dst="ff:ff:ff:ff:ff:ff")/ARP(pdst=iprange),
              timeout=2)
    collection = []
    for snd, rcv in ans:
        result = rcv.sprintf(r"%ARP.psrc% %Ether.src%").split()
        collection.append(result)
#        print collection
    return collection

def main():
    """Runs program and handles command line options"""

    p = optparse.OptionParser(description='Finds MacAddr of IP address(es)',
                                        prog='pyarping',
                                        version='pyarping 0.1',
                                        usage= '%prog [10.0.1.1 or 10.0.1.0/24]')

options, arguments = p.parse_args()
if len(arguments) == 1:
    database = create_database("sqlite:")
    store = Store(database)
    store.execute("CREATE TABLE networkrecord "
        "(id INTEGER PRIMARY KEY, ip VARCHAR,\
            mac VARCHAR, hostname VARCHAR)")
    values = arping(iprange=arguments)
    machine = NetworkRecord()
    store.add(machine)
    #Creates Records
    for ip, mac in values:
        machine.mac = mac
        machine.ip = ip
    #Flushes to database
        store.flush()
    #Prints Record
    print "Record Number: %r" % machine.id
    print "Mac Address: %r" % machine.mac
    print "IP Address: %r" % machine.ip
```

Integrating Config Files With Your Tool

```
[CENTOS]
IP: 10.0.1.40
[UBUNTU]
IP: 10.0.1.50
[SOLARIS]
IP: 10.0.1.60
[REDHAT]
IP: 10.0.1.51
[FREEBSD]
IP: 10.0.1.80
[CMDS]
ONE: uname
```

```python
#!/usr/bin/env python
import ConfigParser

def readConfig(file="config.ini"):
    Config = ConfigParser.ConfigParser()
    Config.read(file)
    sections = Config.sections()
    for machine in sections:
        #uncomment line below to see how this config file is parsed
        #print Config.items(machine)
        macAddr = Config.items(machine)[0][1]
        print machine, macAddr
readConfig()
```

Using Setuptools To Distribute Tool

```
entry_points="""
      [console_scripts]
      mytool = mytool:main
      """,
      )
```

- Make an egg
- Create an entry point
  - Entry points install your script into Operating System script directory, “for free”
- Create a --upgrade flag and integrate upgrades into your tool by using the Python Package Index

References

- This Presentation and Examples:
  - http://code.noahgift.com/pycon2008
- SNMP and Python:
  - http://www.ibm.com/developerworks/aix/library/au-netsnmpnipython/
- Scapy: http://www.secdev.org/projects/scapy/
- Storm: https://storm.canonical.com/
- Entry Points: http://peak.telecommunity.com/DevCenter/setuptools#automatic-script-creation