Plumbum

Presentation by Norman J. Harman Jr. <njharman@gmail.com>

Code and Source: https://github.com/njharman/presentations

Who

Tomer Filiba

- git hub https://github.com/tomerfiliba

What

Plumbum 'Shell Combinators and More'.

Three things mashed into one lib

- Shell-like 'syntax'. Easy path, current working directory, environment manipulation.
- CLI interface - Neat but should be in it's own package.
- Remote execution - Of zero interest to me.

Features

- Python 2.5-3.2 compatible (requires six)
- Microsoft compatible
- MIT license
- Pip installable: `pip install plumbum`
- Issue tracker
- Read the Docs
- Fair number of unittests but not complete
- Travis CI

Why

[Warning very biased opinion ahead]

Bash shell script replacement. 'Quick & Dirty' and esp the Q&D's that evolve into critical infrastructure components. Probably not

Paths

```
from plumbum import local

root = local.path('/')
print root
print repr(root)
root.exists()
root.isdir()
root.isfile()

# Iteration over directory contents. Also .walk()
for path in root:
    print path

# Directory concatination
var = root / 'var' / 'log'
print var

# Globbing
glob = root // '???'
glob
```

Bunch of other methods: basename() dirname() stat() move() copy() mkdir() open() / read() / write().

Working-Directory Context Manager

```
from plumbum import local
from plumbum.cmd import ls

print local.cwd
with local.cwd('/var/log'):
    print ls()
    with local.cwd('/'):
        print ls()
print local.cwd
local.chdir('/')
print local.cwd
```

Environment Manipulation and Context Manager

```
from plumbum import local
from plumbum.cmd import custom_command

local.env['EDITOR'] == 'vim' or 'fail'

print custom_command()
with local.env(HOME='/root'):
    print custom_command()
print custom_command()
```

System Commands

Imports any executable found in $PATH, converts `_` to `-`.

```
from plumbum.cmd import ls, custom_command
from plumbum import CommandNotFound

try:
    from plumbum.cmd import no_such_program
except CommandNotFound as e:
    print 'Woopers', e, '\n'

# Commands are objects. str() is "command line".
print ls
print repr(ls)

# Calling executes command (not in subshell).
print ls()
# Parameters must be listed individually.
print ls('-d', '/')
# Partial application with "indexing" .
ll = ls['-l']
print ll
print repr(ll)
print ll('/tmp')
print ll('/var')

stdout = ls()
exit_code, stdout, stderr = ls.run()
subprocess_popen_obj = ls.popen()
```

Exit Codes

Normally non-zero exit results in `Exception()`. Keyword param `retcode` to change.

```
from plumbum.cmd import ls
from plumbum import ProcessExecutionError

try:
    ls('*')
except plumbum.commands.ProcessExecutionError as e:
    print 'so sad, no glob'
    print 'exit code:', e.retcode
    print 'stdout:', e.stdout
    print 'stderr:', e.stderr.strip()
    print 'argv:', e.argv
    print '\nThe Exception\n%s' % e
ls('missing', retcode=2)       # ignore 2, which is ls's not found
ls('missing', retcode=None)    # ignore every code
ls('missing', retcode=[0,2])   # ignore multiple codes
```

Pipes and Redirection

```
import sys
from plumbum.cmd import ls, grep, wc, cat, head

# Note: Must use partial application []
pipe = ls['-a'] | grep['-v', '\\.py'] | wc['-l']
print pipe
print repr(pipe)
print pipe()

redirect = ((cat < '/etc/hosts') | head['-n', 4])
print redirect
print redirect()

# other redirection examples
(ls['-a'] > 'file.list')()
(cat['file.list'] | wc['-l'])()
with open('delme', 'w') as fh:
    ((grep['foo'] < sys.stdin) > fh)()
```

No >> (append). But there is helpful stdin redirection <<:

```
(cat << 'some python string')()
```

Command Nesting

```
from plumbum.cmd import sudo, ifconfig

cmd = sudo[ifconfig['-a']]
print a
print repr(a)
print a()
```

Foreground and Background

```
from plumbum import FG, BG
from plumbum.cmd import ls

ls['-l'] & FG
ls['-l'] & BG
process = _
process.ready()
process.wait()
process.stdout
```