

xonsh

A Python-powered, cross-platform, Unix-gazing shell language and command prompt.

Rant on POSIX shell

- Executables are started with a list of strings
opt. env vars, I/O redirection, exit code
- POSIX specifies a lot of awkward features:
 - Non-mnemonic names like \$0, \$?, \$! and \$@
 - Shitty iteration semantics for string splitting
 - How often did you forget to add “\$quotes”?
 - [[How]] \$((to)) [do] arithmetics?
 - How to many nestings of “escapes” are needed?
 - Lack of string operations

Why not IPython?

Two reasons.

- 1) The first is that typing "!" before every subprocess command is extremely tedious.
- 2) The second is that tab completion of subprocess commands after an "!" does not work. These are deal breakers for day-to-day use.

Features

- Python + POSIX shell features
- BSD 2-clause; presented at PyCon 2016
- Requires Python v3.5+; well-documented
- Python and shell commands in terminal
→ batteries included
- tab completion, aliases, env in dict, adjust prompt, subprocesses with `$(echo ...)`

```
lprokop@t495slp: ~ | xonsh
lprokop@t495slp ~ $ 1 + 41
42
lprokop@t495slp ~ $ 1 +
lprokop@t495slp ~ $ import sys
lprokop@t495slp ~ $ sys.version
'3.8.2 (default, Jul 16 2020, 14:00:26) \n[GCC 9.3.0]'
lprokop@t495slp ~ $ print(sys.version)
3.8.2 (default, Jul 16 2020, 14:00:26)
[GCC 9.3.0]
lprokop@t495slp ~ $ echo $USER
lprokop
lprokop@t495slp ~ $ 'HOME' in ${...} # \${...} is ENV
True
lprokop@t495slp ~ $ $XONSH=True
lprokop@t495slp ~ $ ${...}['XONSH']
True
lprokop@t495slp ~ $ with ${...}.swap(XONSH=False):
.....     echo $XONSH
.....
False
lprokop@t495slp ~ $ echo $XONSH
True
lprokop@t495slp ~ $ █
```

```
lprokop@t495slp: ~ | xonsh
lprokop@t495slp ~ $ echo $PATH
/home/lprokop/.cargo/bin:/home/lprokop/.local/bin:/home/lprokop/bin:/usr/local/s
bin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/sn
ap/bin
lprokop@t495slp ~ $ $PATH
EnvPath(
['/home/lprokop/.cargo/bin',
'/home/lprokop/.local/bin',
'/home/lprokop/bin',
'/usr/local/sbin',
'/usr/local/bin',
'/usr/sbin',
'/usr/bin',
'/sbin',
'/bin',
'/usr/games',
'/usr/local/games',
'/snap/bin']
)
lprokop@t495slp ~ $ $PATH.append('/malicious:path')
lprokop@t495slp ~ $ echo $PATH
/home/lprokop/.cargo/bin:/home/lprokop/.local/bin:/home/lprokop/bin:/usr/local/s
bin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/sn
ap/bin:/malicious:path
lprokop@t495slp ~ $
```

```
lprokop@t495slp: ~ | xonsh
lprokop@t495slp ~ $ dir()
['__builtins__', '__name__', '_sha', 'argparse', 'math']
lprokop@t495slp ~ $ help(_sha)
Help on function _sha:

_sha(args, stdin=None)
    Calculate hash of file

lprokop@t495slp ~ $ grep -A 2 "def _sha" ~/.xonshrc
def _sha(args, stdin=None):
    """ Calculate hash of file """
    parser = argparse.ArgumentParser('sha', description="Calc hash sum of file")
lprokop@t495slp ~ $ cmd = !(ls ~/Desktop)
lprokop@t495slp ~ $ print(cmd.returncode)
0
lprokop@t495slp ~ $ print(cmd.out)
crashplan.desktop
task-3
task-4+5
van emde boas

lprokop@t495slp ~ $ print(cmd.err)
None
lprokop@t495slp ~ $
```

```
lprokop@t495slp: ~ | xonsh
lprokop@t495slp ~ $ aliases['g'] = ['rg', '--pretty']
lprokop@t495slp ~ $ g 'sha' ~/.xonshrc
18:def _sha(args, stdin=None):
20:     parser = argparse.ArgumentParser('sha', description="Calc hash sum of file")
22:     parser.add_argument('-a', '--algorithm', default='sha256', help='The hash algorithm used')
37:aliases['sha'] = _sha
lprokop@t495slp ~ $ def example(args, stdin=None, stdout=None, stderr=None, spec=None, stack=None):
.....         print(args, stdin, stdout, stderr, spec, stack)
.....
lprokop@t495slp ~ $ aliases['e'] = example
lprokop@t495slp ~ $ echo 'stdin' | e
[] <_io.TextIOWrapper name=6 encoding='utf-8'> <_io.TextIOWrapper name=9 encoding='utf-8'> <_io.TextIOWrapper name=11 encoding='utf-8'> SubprocSpec([], ProcProxyThread, stdin=6, stdout=<_io.TextIOWrapper name=9 mode='w' encoding='UTF-8'>, stderr=<_io.TextIOWrapper name=11 mode='w' encoding='UTF-8'>, universal_newlines=True, close_fds=False) [FrameInfo(frame=<frame at 0xf89710, file '<xonsh-code>', line 1, code <module>>, filename='<xonsh-code>', lineno=1, function='<module>', code_context=None, index=None), FrameInfo(frame=<frame at 0x7f1387829b20, file '/usr/lib/python3/dist-packages/xonsh/__amalgam__.py', line 2948, code run_compiled_code>, filename='/usr/lib/python3/dist-packages/xonsh/__amalgam__.py', lineno=2948, function='run_compiled_code', code_context=None, index=None), FrameInfo(
```


Not shown here

- History is made of JSON blobs stored in file (default) or SQLite
- Macros!(...) take string and you can rewrite it
- Predefined tab completers for import, bash, xontrib, pip, ... but also programmable
- globbing and regex with `backticks`
- “source” for xonsh and python files
- Virtual environment via “xov”
- Event hooks (e.g. on_chdir)

Disadvantages

- Mixed, inconsistent syntax
- Inappropriate for low-resource servers / servers w/o python
- Total main memory usage via pmap:
 - xonsh: 1.26 GB
 - Zsh: 16.44 MB

Thanks!