Bash Oneliners

David Hashe

IM IN YR GPI, BEING YR PET TAX

The three uses of bash

- Basic file browser and program runner
- Data processing language
- General full-fledged scripting language

Earlier focus

- **Basic file browser and program runner**
- Data processing language
- General full-fledged scripting language

This week's focus

- Basic file browser and program runner
- **Data processing language**
- General full-fledged scripting language

???

- Basic file browser and program runner
- Data processing language
- **General full-fledged scripting language**

How Programs Communicate

Unix is a 2-layered API

- Unix C API
  - Used for “real” programming
- Unix Shell API
  - Subset of C API functionality
  - Used for scripting and interactive use

THE Open GROUP

Portable Operating System Interface (IEEE 1003)

Unix is a 2-layered API

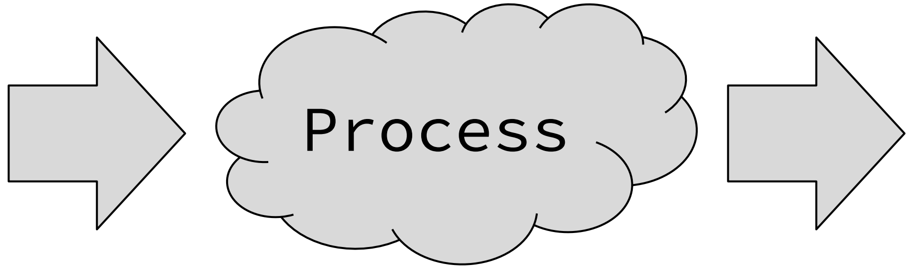
- Unix C API **(15-213)**
  - Used for “real” programming
- Unix Shell API **(GPI)**
  - Subset of C API functionality
  - Used for scripting and interactive use

THE Open GROUP

Portable Operating System Interface (IEEE 1003)

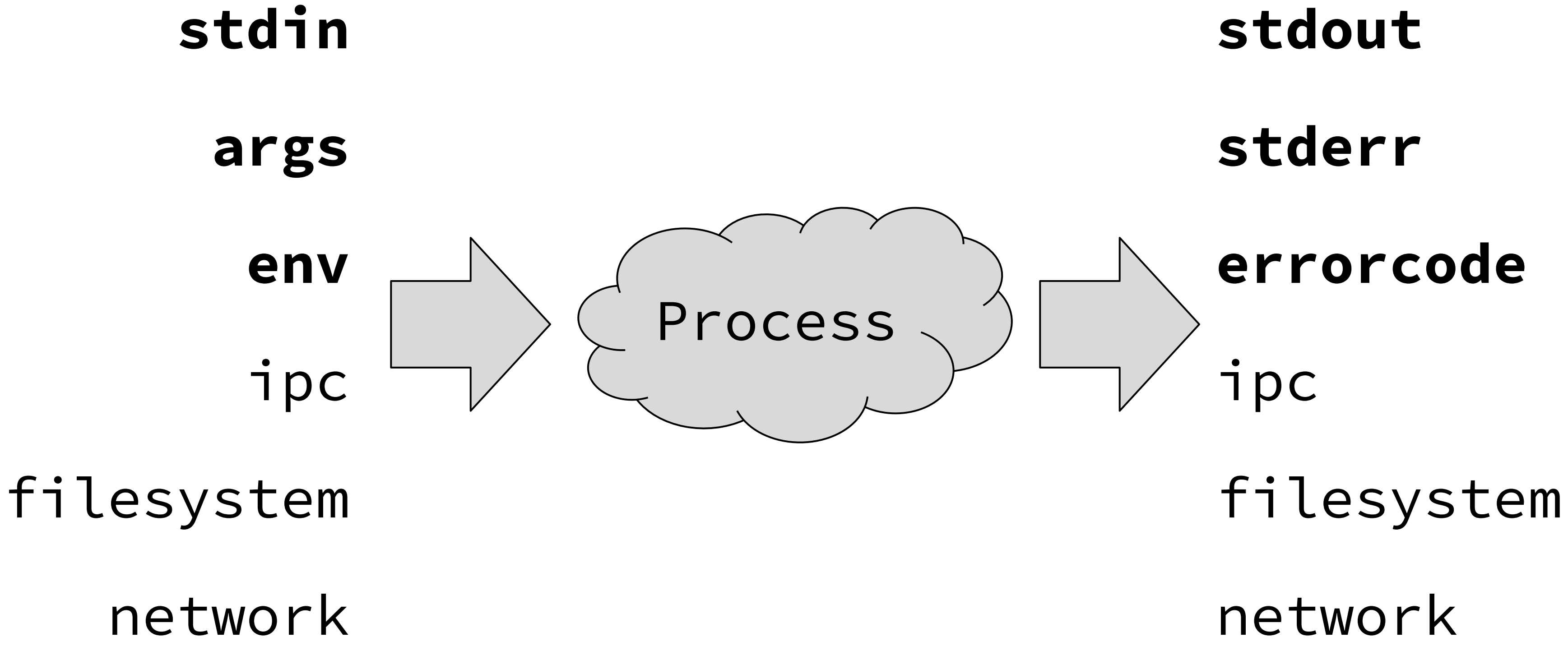
How Unix processes interact with the outside world

stdin
args
env
ipc
filesystem
network



stdout
stderr
errorcode
ipc
filesystem
network

Only some of these interactions are scriptable in shell



Command line arguments (args)

- We’ve seen these before
  - `echo hello world`
- Evaluated for globs and then sent to the program as `argc` / `argv`
  - `argc = 3, argv = [“echo”, “hello”, “world”]`

Streams (stdin, stdout, stderr)

- `stdin`: standard input (`0`)
  - `raw_input()`, `scanf()`
- `stdout`: standard output (`1`)
  - `print()`, `printf()`
- `stderr`: standard error (`2`)
  - `print(file=sys.stderr)`, `fprintf(stderr)`

The Environment

A list of key-value pairs that changes how programs operate, behind-the-scenes.

Basically the Illuminati of Unix shell-scripting.

```
$ printenv
LANG=en_US.UTF-8
DISPLAY=:0.0
SHLVL=1
LOGNAME=dhashe
LANGUAGE=en_US
ZSH=/home/dhashe/.oh-my-zsh
PAGER=less
LESS=-R
LC_CTYPE=en_US.UTF-8
LSCOLORS=Gxfxcxdxbxegedabagacad
EDITOR=vim
```

The Environment

A list of key-value pairs that changes how programs operate, behind-the-scenes.

Basically the Illuminati of Unix shell-scripting.

```
$ printenv
LANG=en_US.UTF-8
DISPLAY=:0.0
SHLVL=1
LOGNAME=dhashe
LANGUAGE=en_US
ZSH=/home/dhashe/.oh-my-zsh
PAGER=less
LESS=-R
LC_CTYPE=en_US.UTF-8
LSCOLORS=Gxfxcxdxbxegedabagacad
EDITOR=vim
```

Error Codes

These are used to programmatically indicate success (=0) or some kind of failure (>0).

```
#include <stdio.h>
int main(int argc char** argv)
{
    printf(“Hello world!”);
    return 0;
}

$ ./hello
$ echo $?
0
```

Error Codes

These are used to programmatically indicate success (=0) or some kind of failure (>0).

```
#include <stdio.h>
int main(int argc char** argv)
{
    printf(“Hello world!”);
    return 0;
}

$ ./hello
$ echo $?
0
```

Scripting Communication

Scripting Streams: Redirection

	Input	Output	Error
Append	*	`[cmd]>>[file]`	`[cmd]2>>[file]`
Read/ Overwrite	`[cmd]<[file]`	`[cmd]>[file]`	`[cmd]2>[file]`

*The implied syntax creates a heredoc, or “inline file”

Scripting Streams: Tricks and Tips

- Redirect one stream into another
  - `[cmd] 2>&1`
- Ignore a stream
  - `[cmd] > /dev/null`

Scripting Streams: Unix Pipes

- Pipes connect two processes
- The `stdout` of the first process is linked to the `stdin` of the second process
- Allows for unidirectional communication and function composition

Pipe Syntax

```
<cmd> [ARGS] [REDIRECTS] | <cmd> [ARGS] [REDIRECTS]
```

```
cat doggos.txt |
sed ‘s/dogs/cats/g’ 2> /dev/null
```

The pipe character is shift-\ (above the enter key)

Execution Model for Pipes

- All programs in a pipe are run in parallel
- At the pipe boundaries, results are buffered
- Implication: Pipes are fast
- Implication: This **doesn’t work**:
  - `echo “some text” > tmp.txt | cat tmp.txt`
  - Cat may see an empty or non-existent tmp.txt

Pipe Example

```
find . -name "*pdf" | grep -v "written.pdf" | xargs open
```

Open all PDF files in the subtree rooted at the current working directory that are not named written.pdf.

Scripting Args: Globs

- We’ve seen this before!
    - Syntax for pattern-matching on filenames

Scripting Args: Xargs

- Xargs turns the stdout of one process into the args for another

```
program1 | xargs program2

find . | xargs cat
```

Streams vs Args

`echo`: args $\rightarrowtail$ streams

`xargs`: streams $\rightarrowtail$ args

Scripting the Environment

- Only useful in specific situations
    - Not required for the lab / exam, just for completeness

```
VAR1=value VAR2=value <cmd> [args]

EDITOR=emacs git commit
```

Scripting the Exit Code

- Only useful in specific situations
  - Not required for the lab / exam, just for completeness

```
$ false # Unix program that always fails

$ echo $?

1
```

Bash as a General-Purpose Scripting Language

Bash has all the standard features of a scripting language

- Functions with arguments / return codes
- If-then-else / conditionals
- Looping constructions
- String and array processing

So we shouldn’t we use bash for **everything**?

Bash has some fundamental issues with **correctness**

- It can be surprisingly challenging to write shell scripts that are formally correct in all situations
  - https://mywiki.wooledge.org/BashPitfalls
  - https://dwheeler.com/essays/fixing-unix-linux-filenames.html
- Shell is best used interactively or for simple automation scripts, not for building robust software
  - Cautionary tale: GNU Autotools

The Unofficial Bash Strict Mode

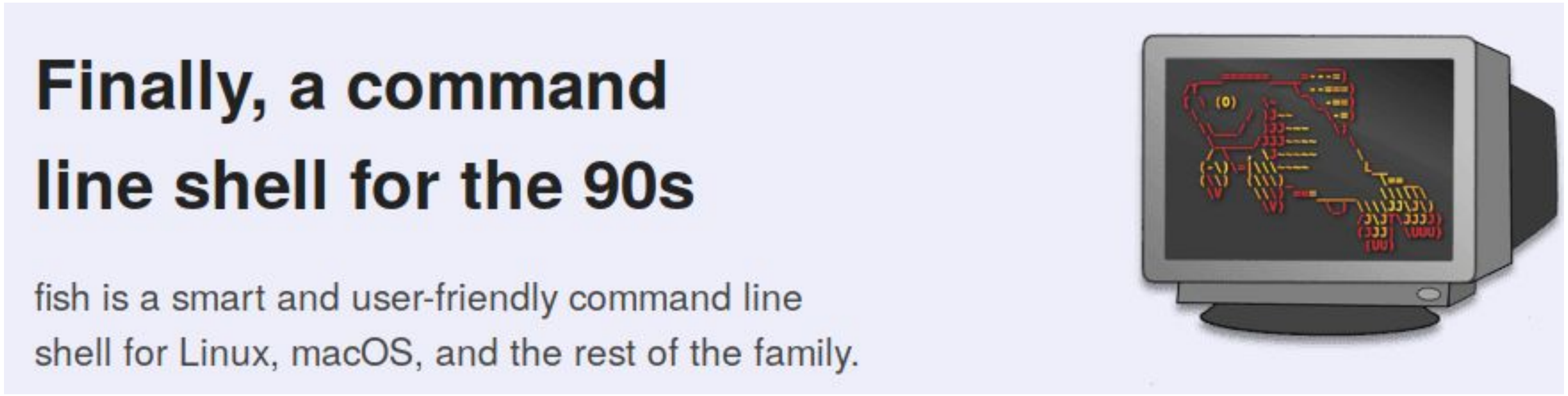
- exit script if any command fails
- error if you reference a variable that has not been defined
- cause a pipe to fail if any of its components also fail
- makes looping behavior more intuitive on lists of items

```
#!/usr/bin/env bash
set -euo pipefail
IFS=$'\n\t'
```

source: http://redsymbol.net/articles/unofficial-bash-strict-mode/

The New Shells

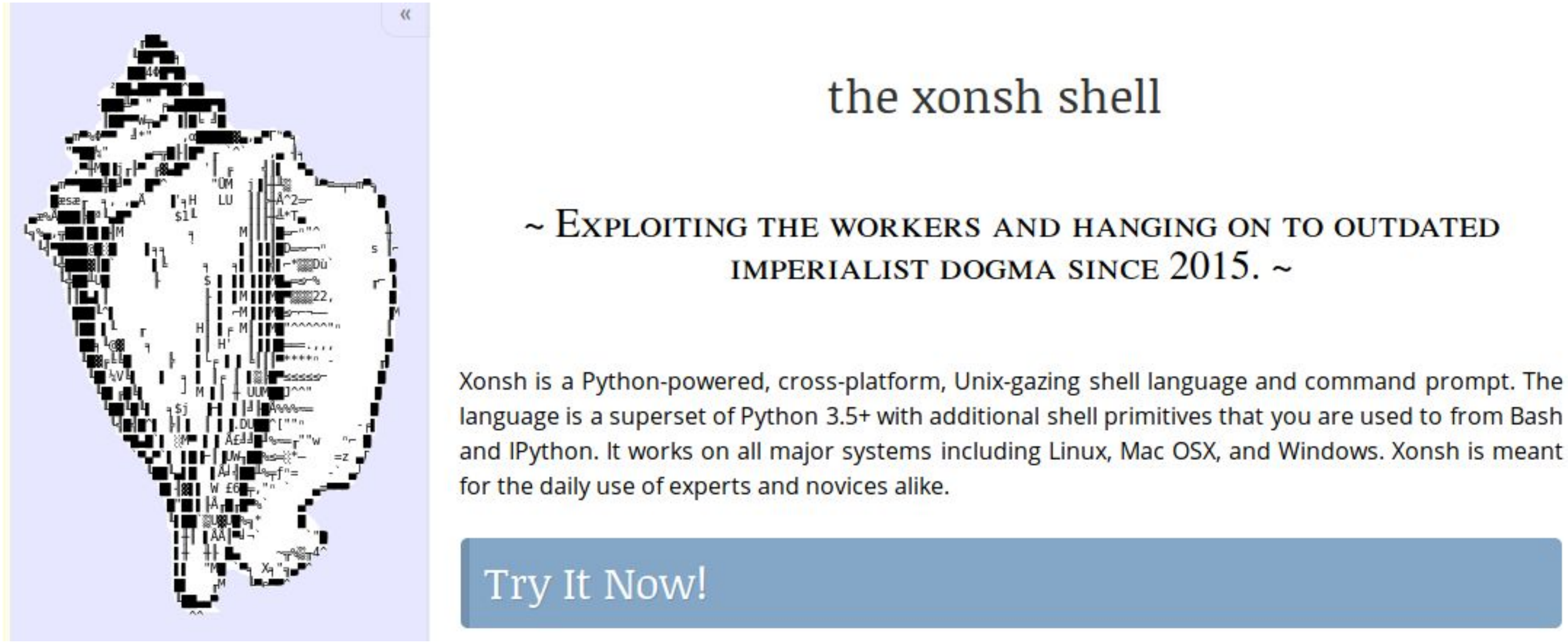
fish, the friendly interactive shell



Finally, a command line shell for the 90s

fish is a smart and user-friendly command line shell for Linux, macOS, and the rest of the family.

xonsh, the python shell



the xonsh shell

~ Exploiting the workers and hanging on to outdated imperialist dogma since 2015. ~

Xonsh is a Python-powered, cross-platform, Unix-gazing shell language and command prompt. The language is a superset of Python 3.5+ with additional shell primitives that you are used to from Bash and IPython. It works on all major systems including Linux, Mac OSX, and Windows. Xonsh is meant for the daily use of experts and novices alike.

oil, your upgrade path from bash

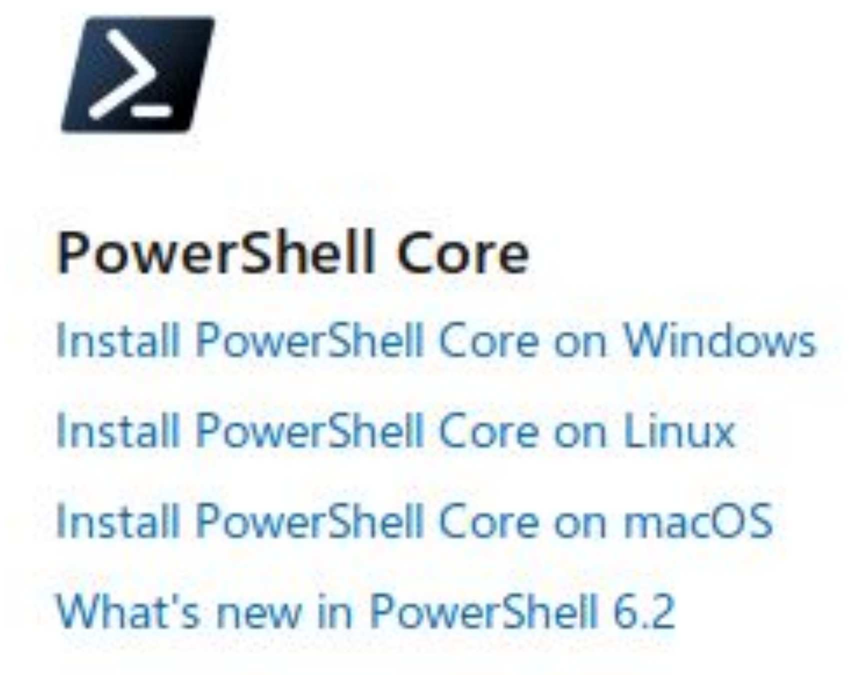
Oil

Oil is a new Unix shell. Why Create a New Unix Shell? / 2019 FAQ

Read the Blog

Download the Latest Release

PowerShell, the object-oriented shell



PowerShell Core

Install PowerShell Core on Windows

Install PowerShell Core on Linux

Install PowerShell Core on macOS

What's new in PowerShell 6.2

Good enough for me

- Honestly, I just use Python + the subprocess module
    - More verbose, but easier to be confident about correctness
- These new shells all have interesting ideas, but you just can't rely on them being installed everywhere
- You should still check them out if curious!
- And Bash is still a good choice for basic interactive use, oneliners, and simple scripting

The Parable of Knuth and McIlory

In which one CS legend asks two others to solve a problem

John “made quicksort faster” Bentley

Read a file of text, determine the n most frequently used words, and print out a sorted list of those words along with their frequencies.

story: http://www.leancrew.com/all-this/2011/12/more-shell-less-egg/
photo: https://www.cse.uconn.edu/uncategorized/cse-colloquium-jon-bentley/

Donald “proves code correct then doesn’t run it” Knuth

Knuth wrote his program in WEB, a literate programming system of his own devising that used Pascal as its programming language. His program used a clever, purpose-built data structure for keeping track of the words and frequency counts; and the article interleaved with it presented the program lucidly.

story: http://www.leancrew.com/all-this/2011/12/more-shell-less-egg/
photo: https://commons.wikimedia.org/wiki/File:KnuthAtOpenContentAlliance.jpg

Douglas “literally invented unix pipes” McIlroy

```
tr -cs A-Za-z '\n' |
tr A-Z a-z |
sort |
uniq -c |
sort -rn |
sed ${1}q
```

story: http://www.leancrew.com/all-this/2011/12/more-shell-less-egg/
photo: https://mg.wikipedia.org/wiki/Sary:Douglas_McIlroy.jpeg

Pipes are really powerful!

Useful programs for the lab

find

```
find <directory> -regex ‘<regex>’

find <directory> -name ‘<glob>’
```

Find is to file names as grep is to file contents. We need find for deep recursive searches (the * glob is shallow).

Usually globs are interpreted by the shell, and regexes are interpreted by the program, but find can do both.

xargs

```
echo arg arg arg | xargs program

xargs program < args.txt
```

Xargs reads in stdin, then executes its argument with arguments constructed from stdin.

curl

```
curl <URL>
```

Curl is a highly versatile tool for making network requests. If you call it with a URL, it will return the file or webpage at that URL.

sed

```
sed ‘<sed_script>’ <files>

sed ‘s/<original>/<replacement>/g’ <files>

echo <text> | sed ‘<sed_script>’
```

Sed is the “streaming editor”. It’s a relative of Vim used for scripting purposes, so it supports some of the same commands. We use sed for the substitute command.

Tips for Writing Oneliners

- Construct oneliners iteratively!
  - Try the first command, see what it outputs
  - Try the first two commands, see what they output
  - and so on …
- Multiple ways/tools do the same thing
  - Choose what you’re familiar with
- “Google is your friend! Your friends are your friends!”

Lab Pro Tips

Helpful commands for pipelab:

- Curl - pulls content from an url
- Sed - Edits text (stream editing) (input can be supplied through stdin)
- Xargs <command> - Transformed newline separated text in stdin to arguments for the given command
- Test locally first! Construct iteratively!
- Small secret:
    - ./driver/driver is a bash script
    - Wow! (you can hack it if you want
    - But its probably easier to do the lab…)