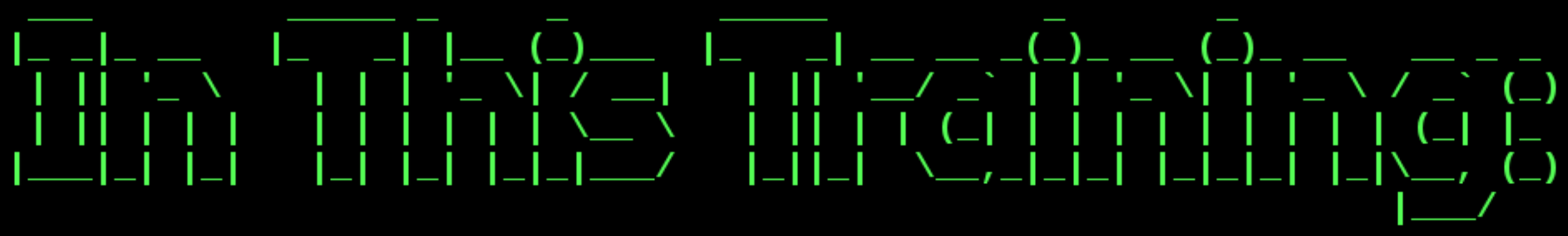


http://modernperlbooks.com/books/modern_perl_2014/

http://unnovative.net/perl.html

Command shell basics

Hack on scan.pl

* OS-specific code

* Interfacing with external programs

* Text handling (finally!)

Finding installed modules (File::Which)

Perl Glue

Perl makes excellent glue code

as it allows you to easily string unrelated commands

and programs together into one comptlete package

Perl has its origins in system administration reporting, and text processing

It draws inspiration from these classic Unix tools:

* Bourne shell
* awk
* sed
* grep
* tr

We can either use Perl's built-in versions of these tools,

or treat Perl like a shell-scripting language and tie these

together through the backtick quote operator (`` or qx{})

Use Perl's built-ins

- # Portable code
- # Better performance
- # Better control over errors

Use the shell's tools

- # R.A.D. to the max
- # Greybeard-friendly

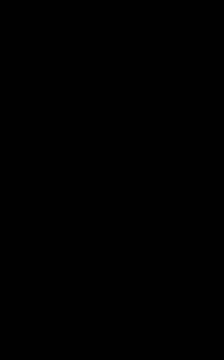
My recommendation:

Where possible you should use Perl's built-in functions

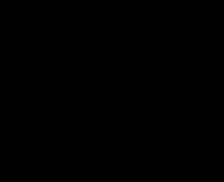
However, we'll do a little bit of both today

Perl Variables

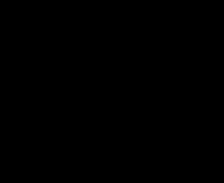
Here are some variables you should know about:



Sub-process exit status

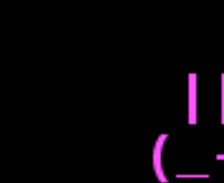


Set by backticks and system()



Status of system call; errno

Set when you run a built-in function



Name of the current OS

Similar to output of "uname" program

The shell's redirection operators control where a command's output goes

And where its input comes from

Today you may see these variations:

Syntax	Meaning
`cmd > /dev/null`	Send STDOUT to the null device Print just the error messages
`cmd 2> /dev/null`	Send STDERR to the null device Only print normal output
`cmd 2>&1`	Combine STDERR with STDOUT
`cmd1 | cmd2`	Make cmd1's STDOUT become cmd2's STDIN

Much of Perl's text processing capabilities are invoked by the

match operators

These operators bind a string to a string-processing function

What they actually do depends on the string-processing function

Function	Meaning
# m//	# Apply a regex match to string, set $1, $2, $3 etc.
# s///	# Substitute matched portion of string
# tr//	# Translate characters in string

Of course, Perl has the usual compliment of text processing functions classics such as:

* substr()
* index()
* rindex()

but the match operators are almost always what you want