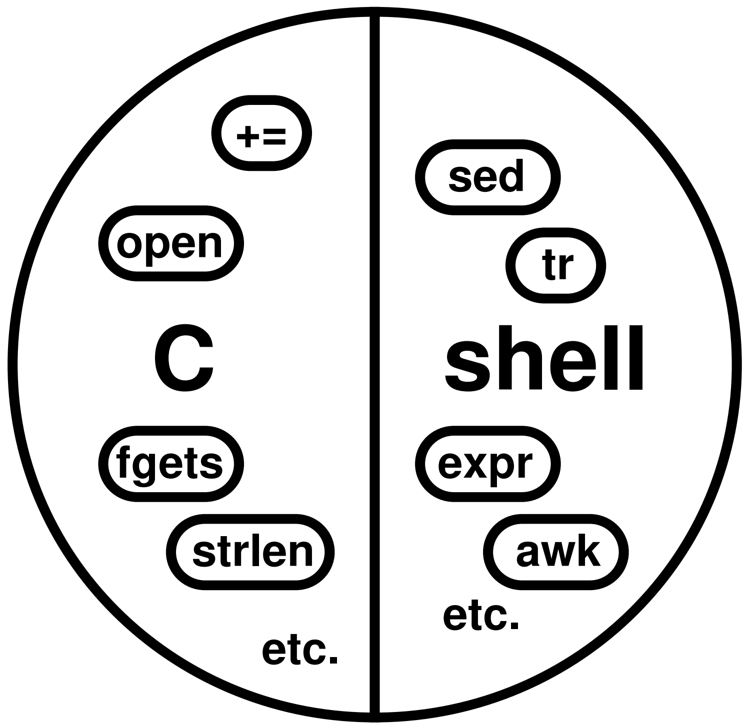
The Taming of the Camel

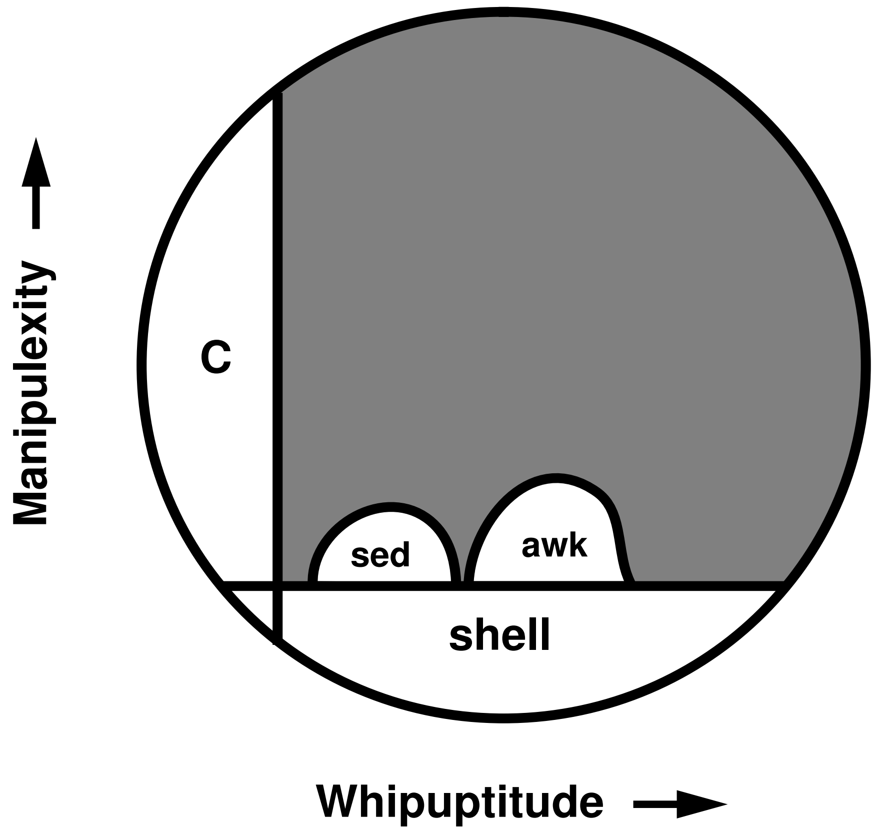
An Overview of Perl 5.0

Larry Wall
<lwall@netlabs.com>

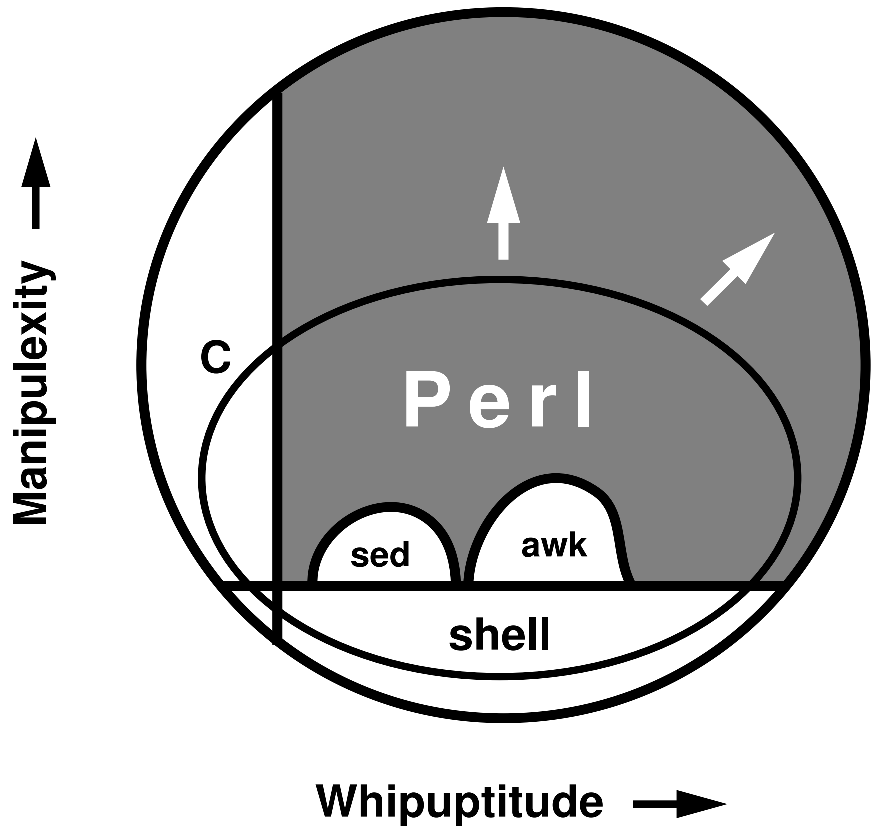
How It Was Back Then
(Sort Of)



How It Really Was Back Then



The Hatching of an Idea



Humble (?) Beginnings

- **Scalars to Represent Values**

  To give Perl a good memory.
- **Filehandles to Represent Files**

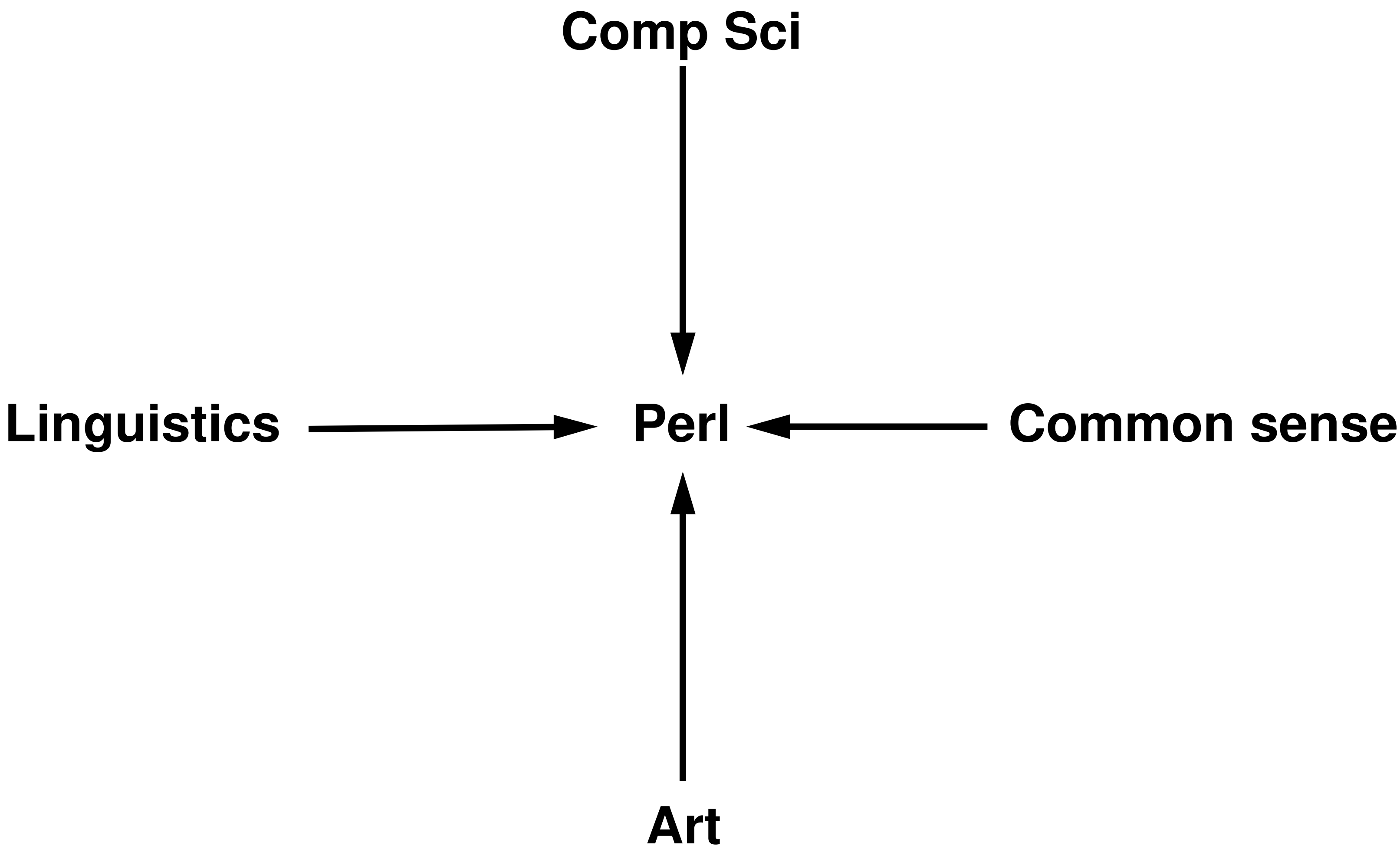
  To give Perl good "legs".
- **Regular Expressions for Extraction**

  To give Perl good "eyes".
- **Literals that Allow Interpolation**

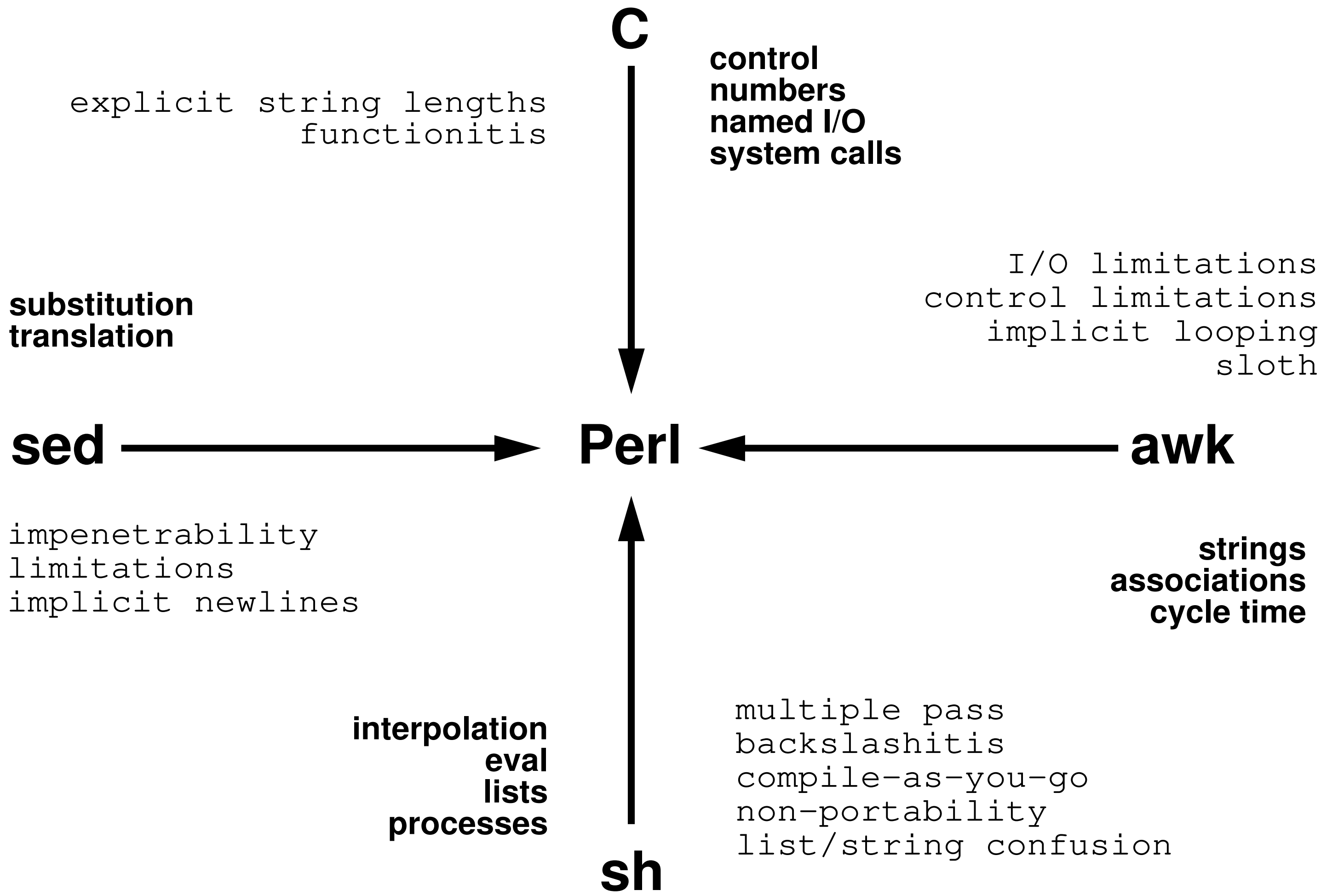
  For easy variable-width formatting.
- **Formats for Reporting**

  For easy fixed-width formatting.

The Inputs



What to Take, What to Leave Behind



Bastardization

or

Hybrid Vigor?

My Irrationalities

- Syntax shouldn’t dangle in the wind
- Ordinary people like me hate abstraction
- C is wonderful
- C is awful
- awk is neither
- Language is an amoral medium
- Ugly can be beautiful
- Beautiful can get ugly real quick
- Visual metaphors are for more than just poetry
- I don’t care what other people think
- I care what other people think
- I think God has free will

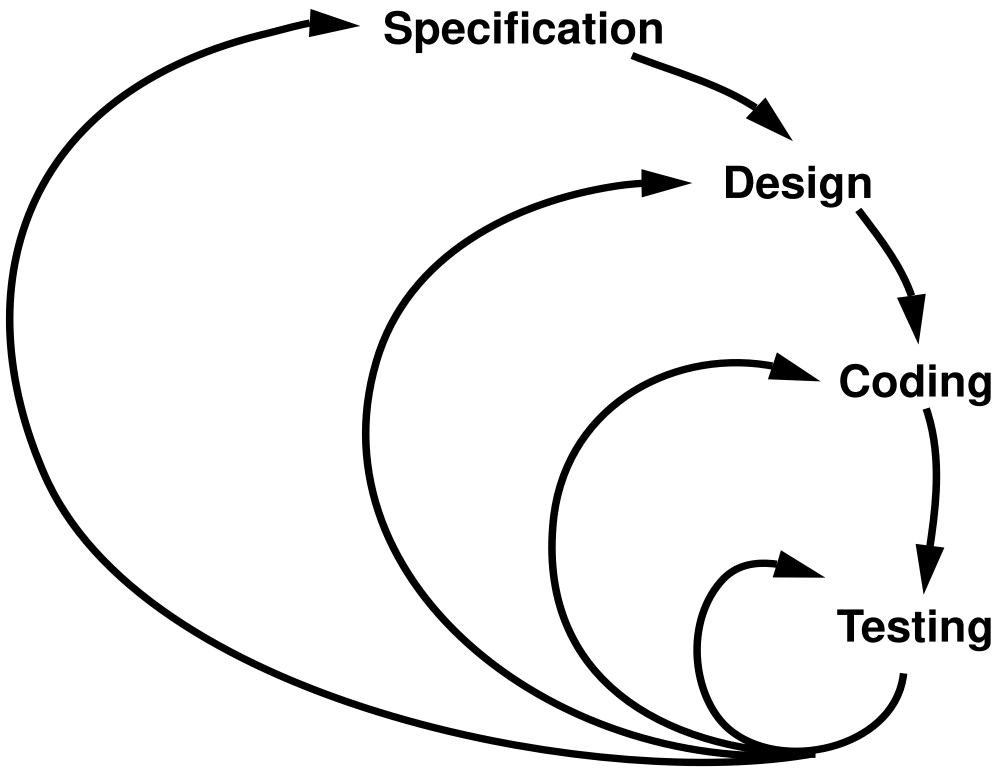
Common Fallacies of Language Design

- “We need to start over from scratch.”
- “If we put in English phrases, that makes it readable.”
- “Simple languages produce simple solutions.”
- “If I wanted it fast, I’d write it in C.”
- “I thought of a way to do it, so it must be right.”
- “This is a VHLL. Who cares about bits?”
- “You can do anything with NAND gates.”
- “Users care about elegance.”
- “The specification is good enough.”
- “Abstraction equals Usability.”
- “The common core should be as small as possible.”
- “Let’s make this easy for the computer.”
- “Most programs are designed top down.”
- “Text processing doesn’t matter.”
- “People should never have been given free will.”

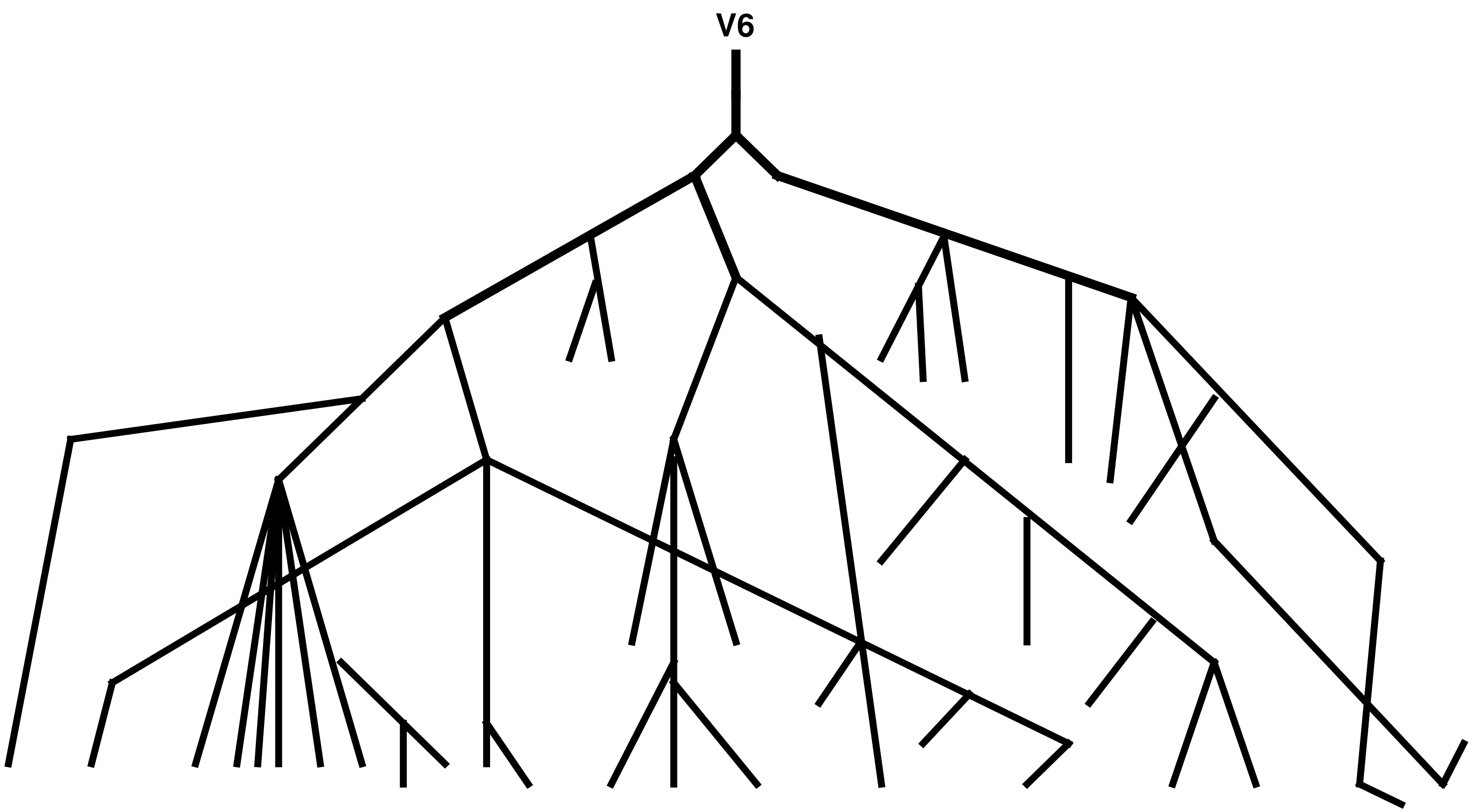
Larry's Conjecture

For most people, the perceived usefulness of a computer language is <u>inversel</u>y proportional to the number of theoretical axes that the language attempts to grind.

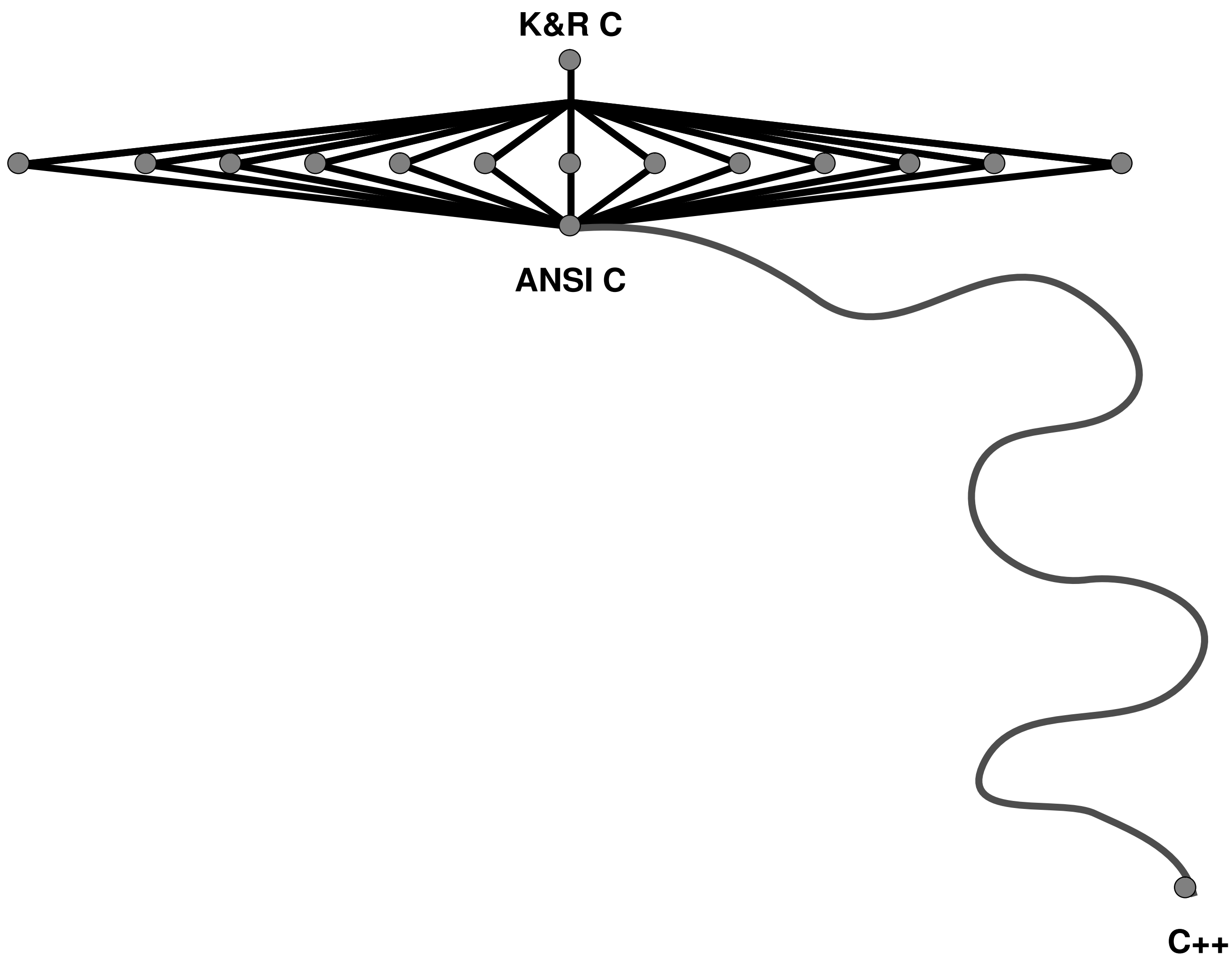
Waterfall or Whirlpool?



The Unix Family Tree



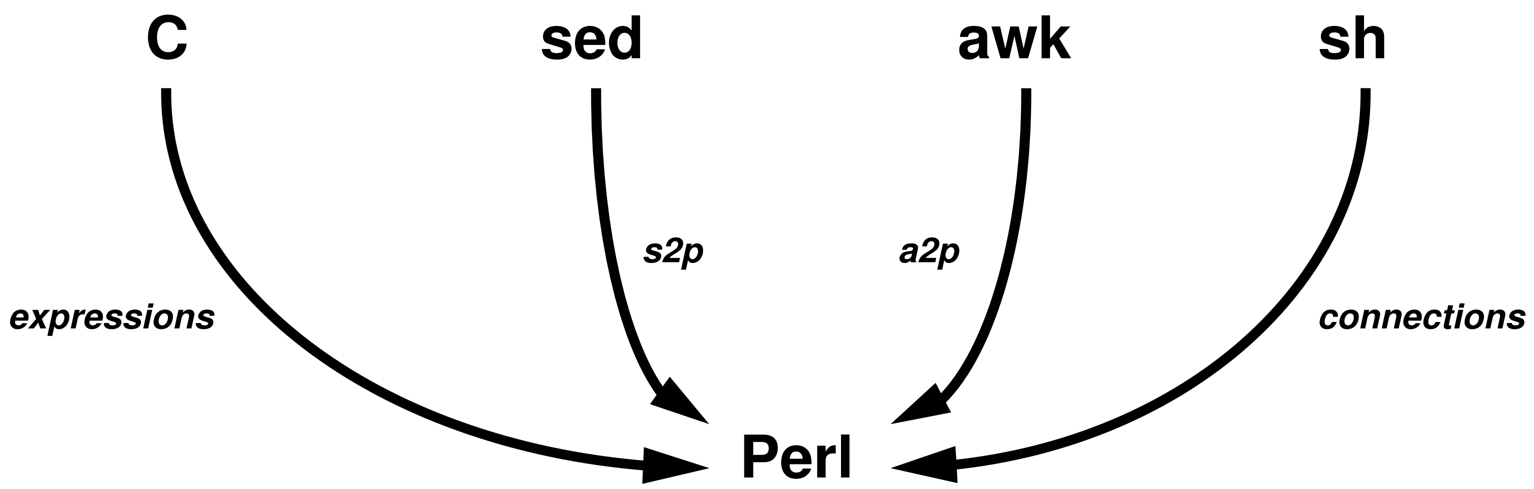
The C Family Tree



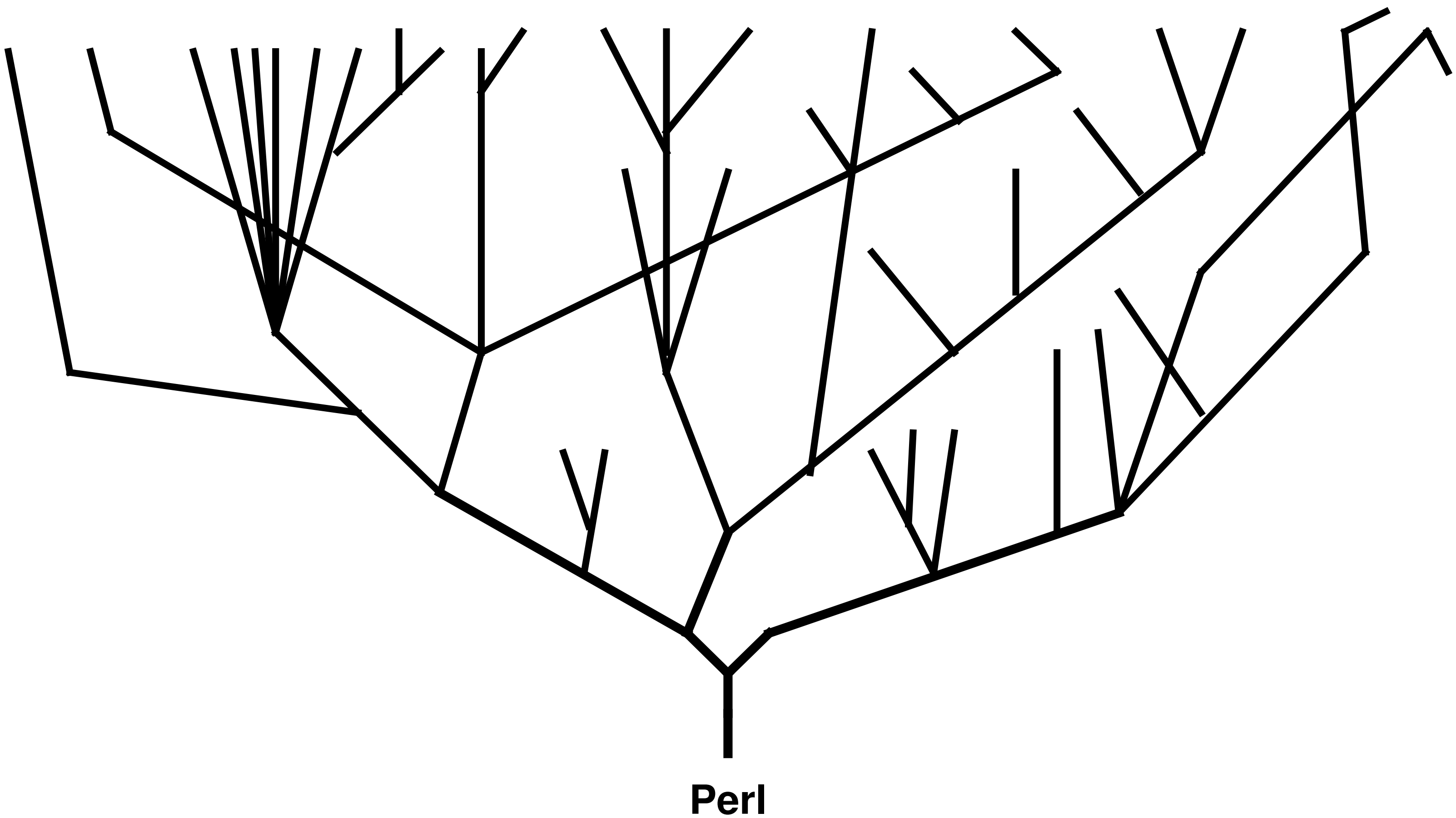
The Perl Family Tree

(under construction)

Easy Migration



The *Other* Perl Family Tree



Visual Distinctions

- **You-pick-'em quotes**

```
s/foo/bar/, m//, tr///, q//, qq//, qx//
tr[a-z][A-Z]
line-oriented
```

- **Classes of operators**

```
== vs eq vs -e
```

- **Formats**

```
@<<<<<<<<<<<<< @||||||||| @>>>>>>>>>>>>>
```

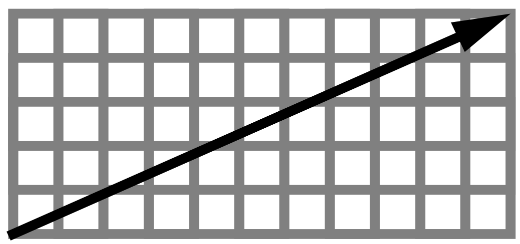
- **Types**

```
$foo @bar %baz *beep
```

- **Mumbles**

```
print "ENTERING FINAL PHASE!!!!\n" if $verbose;
open(HANDLE, "myfile") || die "Can't open: $!";
```

- **Diagonalization**



- **All you need to know is that...**

  `just_do_it()`

- **Landmarks**

  "I'll know it when I see it."

- **Geography vs. Orthography**

  "Who put that mountain there during the night?"

- **Get a faster camel**

English as She is Programmed

- **Familiar syntax**

```
redo LINE if $something_left;
chop $head if traitor();
$california or &bust;
do { this() } until $done;
```

- **Indirect objects**

```
give $DOG $bone;
```

- **Verbing nouns**

```
dog($dog)
```

- **Anything as a Boolean**

```
if (@foo)
if (grep(/^#/, @lines)
```

- **Aggressive tokenization**

```
/foo/ / /bar/ . .20 < <STDIN> % %bletch
```

Those Funny Symbols

`$` = "the" (singular)

`@` = "those" (plural)

`%` = "relationship"

`&` = "do"

`*` = "any sort of"

"No, no, hannie in foodie!"

Natural Language Concepts

- Learn it once, use it many times
- Learn as you go
- Many acceptable levels of competence
- Multiple ways to say the same thing
- No shame in borrowing
- Indeterminate dimensionality
- Local ambiguity is okay
- Punctuation by prosody and inflection
- Disambiguation by number, case and word order
- Topicalization
- Discourse structure
- Pronominalization
- No theoretical axes to grind
- Style not enforced except by peer pressure
- Cooperative design
- “Inevitable” Divergence

Perl 5: The Big Nouns

- Compatibility
- Extensibility
- Usability
- Reusability
- Readability
- Scalability
- Maintainability
- Portability
- Responsibility
- Embeddibility
- Respectability?

Deprecations and Depreciations

- **Deprecated for “action at a distance”**

```
vec() to enable bitwise ops
"@literal"
$*
$#
$[
```

- **Deprecated syntax**

```
do verb()
if BLOCK BLOCK
```

- **Optionally deprecatable via compiler directive**

```
Barewords
Symbolic references
Unqualified global variables
```

- **Depreciations (a better way provided)**

```
select(HANDLE)
Verb markers
' as package prefix delimiter
Punctuation as variable names
```

What's New

- Nearly a complete rewrite
- Usually 25-40% Faster
- Simpler grammar
- Much better diagnostics and docs
- Lexical scoping
- Arbitrarily nested data structures
- Anonymous data structures and functions
- Easy objected-oriented programming
- Modules
- External subroutines in C or C++ via XS
- Dynamic linking on many architectures
- Autoloading subroutines
- POSIX and other standard modules
- Improved configuration
- Package constructors and destructors
- Regular expression enhancements

New Operators

- **Ultra-low precedence logicals**

```
and, or, xor, not
```

- **chomp($line)**
- **exists $hash{$key}**
- **tie/untie**
- **abs($num)**
- **chr($num)**
- **access to various internal functions**

```
uc, ucfirst, lc, lcfirst
quotemeta
glob
formline
```

- **qw(foo bar baz)**
- **bless($ref)**
- **pos($string)**
- **goto &subroutine;**

Miscellaneous Perl 5 Stuff

- **Arrow as synonym for comma**

```
%day = (Sun => 0, Mon => 1, ...);

$childmess = mget(
        OC          => $oc,
        OI          => $oi,
        SCOPE       => $scope1,
        ATTR_ID_LIST => $no_attrs);
```

- **Functions as unary or list operators**

```
$age = -M($filename);
@foo = split /^#/, $bar, 3;
```

- **You can now return from an eval**

```
$prog = "return $a ? 1 : 2";
eval $prog;
```

- **Error propagation**

```
die if $@;
```

- **Lexical scoping**

```
my $var = shift;
```

BEGIN and END

- **Explicitly in some pseudo awk:**

```
#!/usr/bin/perl -nl

BEGIN {
    $accum = 0;
}

$accum += $_;

END {
    print $accum;
}
```

- **Implicitly in** `use POSIX qw(setlocale fcntl_h);`

```
BEGIN {
    require POSIX;
    import POSIX qw(setlocale fcntl_h);
}
```

References

- **Creating references from named objects**

```
$ref = \@array;
$sub = \&function;
$attrs = \%MYATTRS;
```

- **Creating references to anonymous objects**

```
$anonarray = [1,12,[57,42], "hike"];
$anonhash = {FOO => BAR, ADAM => EVE, CHIP => DALE};
$anonsub = sub { print "triggered\n" }
```

- **General dereferencing**

```
push(@{$anonarray}, "crunch");
print ${$anonhash}{"CHIP"};
```

- **Syntactic sugar**

```
@list = @$anonarray;
$$attrs{ID}++;
print "The answer is: ", $anonarray->[2]->[1];
$anonhash->{ADAM} = MADAM;
$count[$a]{$b}[$c] += 42;
```

- **Reference type function**

```
recurse($reference) if ref $reference;
ref($arg) eq HASH or die bad "argument";
```

Objects

- **Object = Reference + Package**
- **Constructors create reference and then “bless” it**

```
sub new { return bless {NAME => FIDO}; }
```

- **Package has one destructor named DESTROY**

```
sub DESTROY {
    my $self = shift;
    print ”Dog $$self{NAME} died\n”;
}
```

- **Methods are ordinary subroutines with special first argument**

```
sub method {
    my $self = shift;
    ref $self eq DOG or die ”Type mismatch”;
```

- **Four ways to call a method**

```
DOG::method($object,@ARGS)
$object->method(@ARGS);
method $object @ARGS;
$object->CLASS::method(@ARGS);
```

- **Multiple Inheritance of methods via @ISA**

```
package DOG;
@ISA = qw(MAMMAL, ANIMAL, FRIEND);
```

Modules

```
package Cwd;
require 5.000;
require Exporter;

@ISA = qw(Exporter);
@EXPORT = qw(getcwd fastcwd);
@EXPORT_OK = qw(chdir);


# By Brandon S. Allbery
#
# Usage: $cwd = getcwd();

sub getcwd
{
    ...
}

sub fastcwd {
    ...
```

Importing

- **The short way:**

```
use Cwd;
```

- **The long way:**

```
BEGIN {
    require Cwd;
    import Cwd;
}
```

- **The short way with a list:**

```
use Cwd qw(getcwd chdir);
```

- **The long way with a list:**

```
BEGIN {
    require Cwd;
    import Cwd qw(getcwd chdir);
}
```

- **The same mechanism is used for “pragmas”**

```
use strict vars, subs, refs;
use integer;
no integer;
use less memory;
use sigtrap qw(ILL TRAP EMT FPE BUS SEGV);
```

The English Module

```
@_   @ARG
$_   $ARG

$&   $MATCH
$`   $PREMATCH
$'   $POSTMATCH
$+   $LAST_PAREN_MATCH

$.   $INPUT_LINE_NUMBER
$.   $NR
$/   $INPUT_RECORD_SEPARATOR
$/   $RS

$|   $OUTPUT_AUTOFLUSH
$,   $OUTPUT_FIELD_SEPARATOR
$,   $OFS
$\   $OUTPUT_RECORD_SEPARATOR
$\   $ORS

$"   $LIST_SEPARATOR
$;   $SUBSCRIPT_SEPARATOR
$;   $SUBSEP
```

```
$?   $CHILD_ERROR
$!   $OS_ERROR
$@   $EVAL_ERROR

$%   $FORMAT_PAGE_NUMBER
$=   $FORMAT_LINES_PER_PAGE
$-   $FORMAT_LINES_LEFT
$~   $FORMAT_NAME
$^   $FORMAT_TOP_NAME
$:   $FORMAT_LINE_BREAK_CHARACTERS
$^L  $FORMAT_FORMFEED

$]   $PERL_VERSION
$^D  $DEBUGGING
$^I  $INPLACE_EDIT
$^T  $BASETIME
$^W  $WARNING
$^X  $EXECUTABLE_NAME

$<   $UID
$>   $EUID
$0   $PROGRAM_NAME
```

Other Standard Modules

- AutoLoader - standard autoloader base class
- Benchmark - run comparitive speed tests
- Carp - report errors outside of current package
- Config - access to all config.sh values
- Cwd - directory processing
- DynaLoader - the dynamic loader
- Env - make %ENV look like regular variables
- Exporter - base class for standard exporters
- Fcntl - common fcntl() definitions
- FileHandle - methods on filehandle objects
- NDBM_File - tie methods for NDBM
- ODBM_File - tie methods for ODBM
- POSIX - POSIX.1 functionality
- SDBM_File - tie methods for SDBM
- Shell - makes undefined functions call programs
- Socket - common socket() definitions

Yet More Standard Modules

- ExtUtils::MakeMaker - extension makefile maker
- File::Basename - portable filename manipulation
- File::CheckTree - verify permissions
- File::Find - walk the directory tree
- Getopt::Long - get long option names
- Getopt::Std - get short option names
- I18N::Collate - sort according to locale
- IPC::Open2 - open two pipes
- IPC::Open3 - open three pipes
- Math::BigFloat - arbitrary precision floating point
- Math::BigInt - arbitrary precision integers
- Math::Complex - complex arithmetic
- Net::Ping - routines to ping the net
- Search::Dict - binary search
- Sys::Hostname - get hostname somehow
- Sys::Syslog - log system messages

Still More Standard Modules

- Term::Cap - termcap
- Term::Complete - command completion
- Test::Harness - run regressions for extension
- Text::Abbrev - abbreviation expansion
- Text::ParseWords - split words like a shell
- Text::Soundex - the one and only
- Text::Tabs - translate tabs
- Time::Local - timelocal() and timegm()

Many others available from the net, including

- Tk
- DBI
- Curses
- Sx
- Msql

Regular Expression Enhancements

- **New options**

```
/m       Assume multiline (like $* = 1)
/s       Assume single line
/x       Extended--allow whitespace
```

- **Minimal (non-greedy) matching**

```
*?       Minimal *
+?       Minimal +
??       Minimal ?
{n,m}?   Minimal {n,m}
```

- **Extension syntax: (?...)**

```
(?sxi)   Embedded options
(?:...)  Non-backref grouping
(?=...)  Positive lookahead assertion
(?!...)  Negative lookahead assertion
(?#...)  Comment
```

- **Example:**

```
s{ (?xgs)
    /\*      (?# match the slashterisk)
    .*?      (?# minimal number of anys)
    \*/      (?# match the asterslash)
}{};
```