Cool Code

@KevlinHenney
Art. Craft. Engineering. Science. These are the swirling muses of design patterns. Art and science are stories; craft and engineering are actions.

Craft is midway between art and science; art and craft stand over against engineering and science. Art is the unique example, the first thing, the story as artifact condensing out of talent and desire. Craft is reliable production of quality. A craftsman might be disappointed but rarely fails. A work of craft is the product of a person and materials. Engineering is reliable and efficient production of things for the use and convenience of people. Science is a process of making a story that can be used for engineering.

Wayne Cool

foreword to

Pattern-Oriented Software Architecture, Volume 5: On Patterns and Pattern Languages
プログラマが知るべき97のこと
97 Things Every Programmer Should Know

O'REILLY
WE PROGRAMMERS ARE WEIRD CREATURES. We love writing code. But when it comes to reading it, we usually shy away. After all, writing code is so much more fun, and reading code is hard—sometimes almost impossible. Reading other people’s code is particularly hard. Not necessarily because other people’s code is bad, but because they probably think and solve problems in a different way than you. But did you ever consider that reading someone else’s code could improve your own?
<pre>/*
 * Copyright (c) 1995, 2008, Oracle and/or its affiliates. All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * - Redistributions of source code must retain the above copyright
 * notice, this list of conditions and the following disclaimer.
 *
 * - Redistributions in binary form must reproduce the above copyright
 * notice, this list of conditions and the following disclaimer in the
 * documentation and/or other materials provided with the distribution.
 *
 * - Neither the name of Oracle or the names of its
 * contributors may be used to endorse or promote products derived
 * from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS
 * IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
 * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR
 * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
 * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
 * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
 * PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
 * LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
 * NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
 * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 */

/**
 * The HelloWorldApp class implements an application that
 * simply prints "Hello World!" to standard output.
 */
class HelloWorldApp {
    public static void main(String[] args) {
        System.out.println("Hello World!"); // Display the string.
    }
}
cool, *adjective*
- fashionably attractive or impressive
- excellent
- used to express acceptance or agreement
- used as an intensive
- used when a conversation goes silent
- marked by deliberate effrontery or lack of due respect or discretion
- restrained or relaxed in style

code, *noun*
- a system of words, figures or symbols used to represent others
- a set of instructions for a computer
- a computer program, or a portion thereof
- a set of conventions or principles governing behaviour or activity in a particular domain
- a system or collection of rules or regulations on any subject
- a collection of writings
HERE IS THE PHILOSOPHY OF GUILDESTERNS:
ON EVERY APPEARANCE OR DISAPPEARANCE OF THE MANUAL THROTTLE
DISCRETE TO SELECT P67 OR P66 RESPECTIVELY;
ON EVERY APPEARANCE OF THE ATTITUDE-HOLD DISCRETE TO SELECT P66
UNLESS THE CURRENT PROGRAM IS P67 IN WHICH CASE THERE IS NO CHANGE

# IS UN-AUTO-THROTTLE DISCRETE PRESENT?
# Read CHAN3
 Masc bit3
 ccb a
tcf startp67 # yes

p67now?
tc checkp67 # no; are we in P67 now?
dec 67
tcf stabl? # no

startp66
tc fastcncng # yes

dec 66
extend
dcl moddisp # set desired altitude rate - current
derch vgovern # altitude rate.

startp66a
tc interpret
sload push priask
sload push priasy
sload vgoper
vks
set
biasfact
roflag
stovl vrias
tek
vcomp
stovl oldipax
store
stool delvrdc
store
doscale
store
rocall
pipeline
store
lastpip

exit
cap zero
t s
f

cap vertstart
# page 801
cap two # vertical: P65,66,67
wchpold
wchpase
tc bankcall # temporary, I hope hope hope
cadr stoprate # temporary, I hope hope hope
tc downflag
adres adovflg
tc downflag
adres adredflg

cap zero

t s
f

t cap vertstart

cap bit13 # is un-attitude-hold discrete present?

codel
crs 3
csf dec66
cextend
bif restart?

c a
rodcount

cif

cextend
bif guilder

cif

cextend
cap startp66

cif

cextend
bif startp66a

c if

cap vertgud

c if: continue with rod
/* grep: search for regexp in file */
int grep(char *regexp, FILE *f, char *name)
{
    int n, nmatch;
    char buf[BUFSIZ];

    nmatch = 0;
    while (fgets(buf, sizeof buf, f) != NULL) {
        n = strlen(buf);
        if (n > 0 && buf[n-1] == '\n')
            buf[n-1] = '\0';
        if (match(regexp, buf)) {
            nmatch++;
            if (name != NULL)
                printf("%s:", name);
            printf("%s\n", buf);
        }
    }
    return nmatch;
}

/* matchhere: search for regexp at beginning of text */
int matchhere(char *regexp, char *text)
{
    if (regexp[0] == '\0')
        return 1;
    if (regexp[1] == '*')
        return matchstar(regexp[0], regexp+2, text);
    if (regexp[0] == '?' && regexp[1] == '\0')
        return *text == '\0';
    if (*text!=='\0' && (regexp[0]=='.' || regexp[0]==*text))
        return matchhere(regexp+1, text+1);
    return 0;
}

/* match: search for regexp anywhere in text */
int match(char *regexp, char *text)
{
    if (regexp[0] == '^')
        return matchhere(regexp+1, text);
    do { /* must look even if string is empty */
        if (matchhere(regexp, text))
            return 1;
    } while (*text++ != '\0');
    return 0;
}

/* matchstar: search for c*regexp at beginning of text */
int matchstar(int c, char *regexp, char *text)
{
    do { /* a * matches zero or more instances */
        if (matchhere(regexp, text))
            return 1;
    } while (*text != '\0' && (*text++ == c || c == '.'));
    return 0;
/**
 * Runs the bare test sequence.
 * @exception Throwable if any exception is thrown
 */

public void runBare() throws Throwable {
    setUp();
    try {
        runTest();
    }
    finally {
        tearDown();
    }
}

/**
 * Override to run the test and assert its state.
 * @exception Throwable if any exception is thrown
 */

protected void runTest() throws Throwable {
    Method runMethod = null;
    try {
        // use getMethod to get all public inherited
        // methods. getDeclaredMethods returns all
        // methods of this class but excludes the
        // inherited ones.
        runMethod = getClass().getMethod(fName, null);
    } catch (NoSuchMethodException e) {
        fail("Method "+fName+" not found");
    }
    if (!Modifier.isPublic(runMethod.getModifiers())) {
        fail("Method "+fName+" should be public");
    }
    try {
        runMethod.invoke(this, new Class[0]);
    } catch (InvocationTargetException e) {
        e.fillInStackTrace();
        throw e.getTargetException();
    } catch (IllegalAccessException e) {
        e.fillInStackTrace();
        throw e;
    }
}
Would you do anything differently in the development of AWK looking back?

One of the things that I would have done differently is instituting rigorous testing as we started to develop the language. We initially created AWK as a 'throw-away' language, so we didn't do rigorous quality control as part of our initial implementation.

I mentioned to you earlier that there was a person who wrote a CAD system in AWK. The reason he initially came to see me was to report a bug in the AWK compiler. He was very testy with me saying I had wasted three weeks of his life, as he had been looking for a bug in his own code only to discover that it was a bug in the AWK compiler! I huddled with Brian Kernighan after this, and we agreed we really need to do something differently in terms of quality control. So we instituted a rigorous regression test for all of the features of AWK. Any of the three of us who put in a new feature into the language from then on, first had to write a test for the new feature.
#!/usr/bin/perl

# PerlInterpreter must be the first line of the file.
#
# Copyright (c) 1995, Cunningham & Cunningham, Inc.
#
# This program has been generated by the HyperPerl generator. The source hypertext can be found at http://c2.com/cgi/wikibase. This program belongs to Cunningham & Cunningham, Inc., is to be used only by agreement with the owner, and then only with the understanding that the owner cannot be responsible for any behaviour of the program or any damages that it may cause.
#
print "Content-type: text/html\n\n";
$DBM = "/usr/ward/$ScriptName";
dbmopen($db, $DBM, 0666) || &AbortScript("can't open "$DBM");
$CookedInput(browse) && &HandleBrowse;
$CookedInput(edit) && &HandleEdit;
$CookedInput(copy) && &HandleEdit;
$CookedInput(links) && &HandleLinks;
$CookedInput(search) && &HandleSearch;
dbmclose($db);
if ($ENV{REQUEST_METHOD} eq POST) {
    $CookedInput(post) && &HandlePost;
}
# &DumpBinding(*CookedInput);
# &DumpBinding(*old);
# &DumpBinding(*ENV);

# InitialComments

# WikiInHyperPerl
template <int i>
struct D
{
   D(void *);
   operator int();
};

template <int p, int i>
struct is_prime
{
   enum { prim = (p%i) && is_prime<(i-2%p:0), i>::prim };
};

template <int i>
struct Prime_print
{
   Prime_print<i-1> a;
   enum { prim = is_prime<i, i-1>::prim };
   void f() { D<i> d = prim; }
};

struct is_prime<0,0> { enum { prim = 1 }; };
struct is_prime<0,1> { enum { prim = 1 }; };
struct Prime_print<2>
{
   enum { prim = 1 };  
   void f() { D<2> d = prim; }
};

void foo()
{
   Prime_print<10> a;
   
   // output:
   // unruh.cpp 30: conversion from enum to D<2> requested in Prime_print
   // unruh.cpp 30: conversion from enum to D<3> requested in Prime_print
   // unruh.cpp 30: conversion from enum to D<5> requested in Prime_print
   // unruh.cpp 30: conversion from enum to D<7> requested in Prime_print
   // unruh.cpp 30: conversion from enum to D<11> requested in Prime_print
   // unruh.cpp 30: conversion from enum to D<13> requested in Prime_print
   // unruh.cpp 30: conversion from enum to D<17> requested in Prime_print
   // unruh.cpp 30: conversion from enum to D<19> requested in Prime_print
```ruby
class Base
  VERSION = "0.0.2"
  module_method_map = {}
  module_method_remap = {};
  module_method_map_keys = module_method_map.keys;
  module_method_map_values = module_method_map.values;
  module_method_map.each do |module_method_map_key, module_method_map_value|
    if module_method_map_value
      all_modules.each do |mod|
        unless module_method_map_value.key?(mod)
          module_method_map_value[mod] = module_method_map_value[module_method_map_key]
        end
      end
    end
  end
  module_method_remap.each do |module_method_remap_key, module_method_remap_value|
    if module_method_remap_value
      all_modules.each do |mod|
        unless module_method_remap_value.key?(mod)
          module_method_remap_value[mod] = module_method_remap_value[module_method_remap_key]
        end
      end
    end
  end
end
```
(define (eval exp env)
  (cond ((self-evaluating? exp) exp)
         ((variable? exp) (lookup-variable-value exp env))
         ((quoted? exp) (text-of-quotiation exp))
         ((assignment? exp) (eval-assignment exp env))
         ((definition? exp) (eval-definition exp env))
         ((if? exp) (eval-if exp env))
         ((lambda? exp)
           (make-procedure (lambda-parameters exp)
                           (lambda-body exp)
                           env))
         ((begin? exp)
           (eval-sequence (begin-actions exp) env))
         ((cond? exp) (eval (cond->if exp) env))
         ((application? exp)
           (apply (eval (operator exp) env)
                  (list-of-values (operands exp) env)))
         (else
           (error "Unknown expression type - EVAL" exp))))
def eval(x, env=global_env):
    "Evaluate an expression in an environment."
    if isa(x, Symbol):  # variable reference
        return env.find(x)[x]
    elif not isa(x, list):  # constant literal
        return x
    elif x[0] == 'quote':  # (quote exp)
        (_, exp) = x
        return exp
    elif x[0] == 'if':  # (if test conseq alt)
        (_, test, conseq, alt) = x
        return eval((conseq if eval(test, env) else alt), env)
    elif x[0] == 'set!':  # (set! var exp)
        (_, var, exp) = x
        env.find(var)[var] = eval(exp, env)
    elif x[0] == 'define':  # (define var exp)
        (_, var, exp) = x
        env[var] = eval(exp, env)
    elif x[0] == 'lambda':  # (lambda (var*) exp)
        (_, vars, exp) = x
        return lambda *args: eval(exp, Env(vars, args, env))
    elif x[0] == 'begin':  # (begin exp*)
        for exp in x[1:]:
            val = eval(exp, env)
        return val
    else:  # (proc exp*)
        exps = [eval(exp, env) for exp in x]
        proc = exps.pop(0)
        return proc(*exps)

isa = isinstance
Symbol = str

def to_string(exp):
    "Convert a Python object back into a Lisp-readable string."
    return '(' + '.join(map(to_string, exp)) + ')' if isa(exp, list) else str(exp)

def repl(prompt='lis.py> '):
    "A prompt-read-eval-print loop."
    while True:
        val = eval(parse(raw_input(prompt)))
        if val is not None: print to_string(val)
import re, collections

def words(text): return re.findall('[a-z]+', text.lower())

def train(features):
    model = collections.defaultdict(lambda: 1)
    for f in features:
        model[f] += 1
    return model

NWORDS = train(words(file('big.txt').read()))
alphabet = 'abcdefghijklmnopqrstuvwxyz'

def edits1(word):
    splits = [(word[:i], word[i:]) for i in range(len(word) + 1)]
    deletes = [a + b[1:] for a, b in splits if b]
    transposes = [a + b[1] + b[0] + b[2:] for a, b in splits if len(b)>1]
    replaces = [a + c + b[1:] for a, b in splits for c in alphabet if b]
    inserts = [a + c + b for a, b in splits for c in alphabet]
    return set(deletes + transposes + replaces + inserts)

def known_edits2(word):
    return set(e2 for e1 in edits1(word) for e2 in edits1(e1) if e2 in NWORDS)

def known(words):
    return set(w for w in words if w in NWORDS)

def correct(word):
    candidates = known([word]) or known(edits1(word)) or known_edits2(word) or [word]
    return max(candidates, key= NWORDS.get)
#define __ -F<00||--F-OO--; int F=00,OO=00;main(){F_00();printf("%1.3f\n",4.*-F/00/00);}F_00()
#!/bin/bash

function f() {
    sleep "\$1"
    echo "\$1"
}

while [ -n "\$1" ]
do
    f "\$1" &
    shift
done

wait
/\^1?\$|^(11+\?)\1+\$/
;while [ $? -eq 0 ]; do nc -vlp 8080 -c'\(r=\text{read}; e=\text{echo}; r a b c; z=r; while \[ \{\#z\} -gt 2 \]; do \$r z; done; f=`\$e \$b\{|sed 's/[\^a-z0-9_.-]//gi'\}; h=\"HTTP/1.0\"; o=\"\$h 200 OK\r\n\"; c=\"Content\"; if [ \(-z f \); then(\$e o; ls|\(while \$r n; do if [ \(-f "\$n" \); then \$e "\(<a href=\"/\$n\">ls -gh \$n</a><br>"; fi; done); \)\); elif [ \(-f f \]; then \$e "$o$c-Type: `\text{file -ib} \$f`\n$c-Length: `\text{stat -c%s $f}`; \$e; cat $f; else \$e -e "$h 404 Not Found\n\n404\n"; fi); done
Lions' Commentary on UNIX®
6th Edition
with Source Code
John Lions
Foreword by Dennis Ritchie

“After 20 years, this is still the best exposition of the workings of a 'real' operating system.”
Ken Thompson

“BOOK OF THE YEAR”
Book Review
Summary—what’s most important

To put my strongest concerns in a nutshell:

1. We should have some ways of coupling programs like garden hose—screw in another segment when it becomes necessary to massage data in another way.

   This is the way of IO also.

2. Our loader should be able to do link-loading and controlled establishment.

3. Our library filing scheme should allow for rather general indexing, responsibility, generations, data path switching.

4. It should be possible to get private system components (all routines are system components) for mugging around with.

K. D. Hill
Oct. 11, 1964
While Thompson and Ritchie were laying out their file system, McIlroy was "sketching out how to do data processing by connecting together cascades of processes and looking for a kind of prefix-notation language for connecting processes together."

Over a period from 1970 to 1972, McIlroy suggested proposal after proposal. He recalls the break-through day: "Then one day, I came up with a syntax for the shell that went along with the piping, and Ken said, I'm gonna do it. He was tired of hearing all this stuff." Thompson didn't do exactly what McIlroy had proposed for the pipe system call, but "invented a slightly better one. That finally got changed once more to what we have today. He put pipes into Unix." Thompson also had to change most of the programs, because up until that time, they couldn't take standard input. There wasn't really a need; they all had file arguments. "GREP had a file argument, CAT had a file argument."

The next morning, "we had this orgy of `one liners.' Everybody had a one liner. Look at this, look at that. ...Everybody started putting forth the UNIX philosophy. Write programs that do one thing and do it well. Write programs to work together. Write programs that handle text streams, because that is a universal interface." Those ideas which add up to the tool approach, were there in some unformed way before pipes, but they really came together afterwards. Pipes became the catalyst for this UNIX philosophy. "The tool thing has turned out to be actually successful. With pipes, many programs could work together, and they could work together at a distance."
The WorldWideWeb application is now available as an alpha release in source and binary form from info.cern.ch.

WorldWideWeb is a hypertext browser/editor which allows one to read information from local files and remote servers. It allows hypertext links to be made and traversed, and also remote indexes to be interrogated for lists of useful documents. Local files may be edited, and links made from areas of text to other files, remote files, remote indexes, remote index searches, internet news groups and articles. All these sources of information are presented in a consistent way to the reader. For example, an index search returns a hypertext document with pointers to documents matching the query. Internet news articles are displayed with hypertext links to other referenced articles and groups.

The code is not strictly public domain: it is copyright CERN (see copyright notice is in the .tar), but is free to collaborating institutes.

Also available is a portable line mode browser which allows hypertext to be browsed by anyone with a dumb ascii terminal emulator. Hypertext may be made public by putting on an anonymous FTP server, or by using a HTTP daemon. A skeleton HTTP daemon is also available in source form. A server may be written to make other existing data readable by WWW browsers. Files are

```
/pub/WWWNeXTStepEditor_0.12.tar.Z  NeXT application + sources
/pub/WWWLineMode_0.11.tar.Z     Portable Line Mode Browser
/pub/WWWDaemon_0.1.tar.Z        Simple server
```

Basic documentation is enclosed. Details about our project and about hypertext in general are available in hypertext form on our servers, as are lists of known bugs and features.

This project is experimental and of course comes without any warranty whatsoever. However, it could start a revolution in information access. We are currently using WWW for user support at CERN. We would be very interested in comments from anyone trying WWW, and especially those making other data available, as part of a truly world-wide web.

Tim BL
If you don't have time to read, you don't have the time or the tools to write.

Stephen King