

]

A Tour of Perl Testing

A Tour of Perl Testing

Perl 测试之旅

☺ *agentzh@yahoo.cn* ☺

章亦春 (*agentzh*)

2009.9

Why testing?

☺ When things have gone *wrong* in collaboration...



agentzh: With *my patch* in r2545, Jifty's I18N should work for (at least) standalone servers.

obra: I'm seeing a whole bunch of **new test failures** in the REST tests in the jifty core after your latest tests (Also, a **15% performance penalty**)
Are you seeing the same?

agentzh: No, I'm not. Because I'm using Windows where all the Jifty live tests are *skipped* by default.
:(

😊 When my user file a ticket to *report bugs...*



As seen in my UML::Class::Simple module's
RT ticket queue:

David Favor: UML::Class::Simple-0.10 *fails* on **latest**
perl-5.10.0-34065 stable patch:

PERL_DL_NONLAZY=1 /pkgs/perl-5.10.0-34065/bin/perl "-MExtUtils::Command::MM" "-e"
"test_harness(0, 'inc', 'blib/lib', 'blib/arch')" t/basic.t t/classes-from-runtime.t
t/pod-coverage.t t/pod.t t/umlclass.t
t/basic.t.....invalid width and height at t/basic.t line 68
Renderer type: "gif" not recognized. Use one of: canon cmap cmapx cmapx_np dia dot fig gtk
hpgl imap imap_np ismap mif mp pcl pdf pic plain plain-ext png ps ps2 svg svgz vml vmlz
at t/basic.t line 137

Failed test 'binary GIF data returned'
at t/basic.t line 138.
Looks like you failed 1 test of 36.
Dubious, test returned 1 (wstat 256, 0x100)
Failed 1/36 subtests

agentzh: Hmm, the test failure was due to the *lack* of gif handler support in your graphviz installation.

UML::Class::Simple **should really skip** this GIF test in **basic.t** if no gif handler is found :)

☺ Test suite failures on *various* platforms could also be reported automatically by *bots*! (CPAN Testers)++



CPAN Testers Reports

www.cpan testers.org

Reports for CGI-Cookie-XS

[Blog](#) | [Reports](#) | [Wiki](#) | [Statistics](#) | [PASS Matrix](#) | [Preferences](#) | [Development](#) | [CPAN Daps](#) | [CPAN Matrix](#)

[Home](#) > [Distributions](#) > [C](#) > CGI-Cookie-XS

Reports for CGI-Cookie-XS

These are the test reports that we have for the CPAN distribution [CGI-Cookie-XS](#). This page was last updated on **Tuesday, 15th September 2009 (10:19pm)**.

Module version:

Status	ID	Perl version	OS name	OS version	Architecture
PASS	2247848	5.10.0	Linux	2.6.26.3-fuuka.profvince.com	x86_64-linux-thread-multi
PASS	2235486	5.8.8	Windows (Win32)	5.00	MSWin32-x86-multi-thread
PASS	2235365	5.10.0	Windows (Win32)	5.00	MSWin32-x86-multi-thread
PASS	2230388	5.10.0	Linux	2.6.24-19-generic	i686-linux-thread-multi-64int-ld

Menu

- Home
- Help
- About
- Admin
- Static Website
- Find A Tester

Preferences

Grade:

Availability:

Done

The screenshot shows a Mozilla Firefox browser window titled "CPAN Testers Reports: Reports for CGI-Cookie-XS - Mozilla Firefox". The address bar contains several tabs: "TextF...", "北京朝...", "朝阳区 ...", "[jifty-d...", "A Tour...", "CP..." (active), "things ...", "image...", and "人人网".

The main content area displays two summary sections:

- Report Summary**
- Version Summary:**

To the right of these summaries is a large table listing test results.

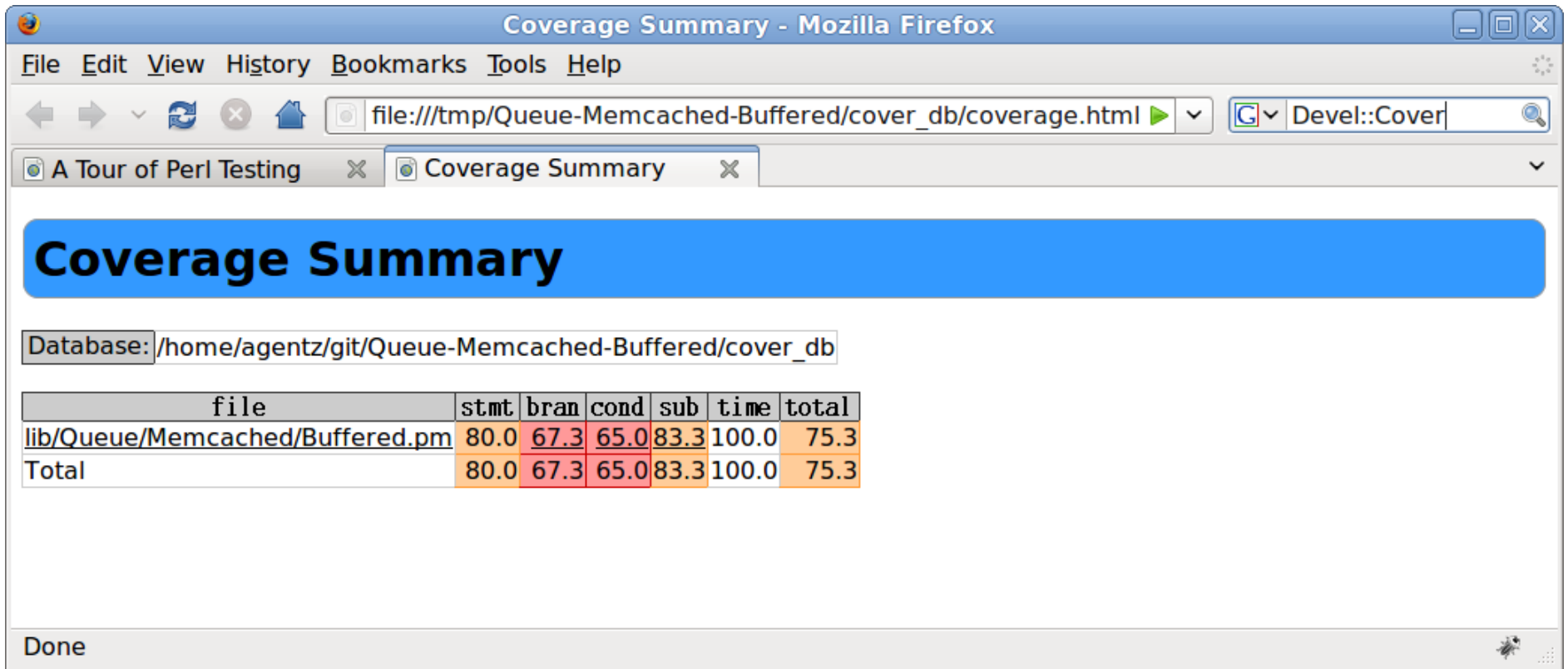
Status	ID	Platform	Architecture	Distribution	Configuration	Description
PASS	2197887	SunOS/Solaris	2.10	i86pc-solaris-thread-multi-64int		
PASS	2197093	SunOS/Solaris	2.9	sun4-solaris-thread-multi		
PASS	2197066	SunOS/Solaris	2.10	i86pc-solaris-thread-multi-64int		
FAIL	2194791	Linux	2.6.24.5-smp	i686-linux-thread-multi-64int		
PASS	2192988	FreeBSD	6.2-release	i386-freebsd-thread-multi		
PASS	2190356	Linux	2.6.24.5-smp	i686-linux-thread-multi-64int-lid		
FAIL	2187626	Linux	2.4.27-3-686	i686-linux		
PASS	2187608	Linux	2.4.27-3-686	i686-linux		
PASS	2187607	NetBSD	2.1.0_stable	alpha-netbsd		
PASS	2187581	Mac OS X	8.10.1	darwin-thread-multi-2level		
PASS	2187572	IRIX	6.5	IP35-irix-64int		
PASS	2185726	NetBSD	3.1	i386-netbsd-thread-multi-64int		
PASS	2185725	NetBSD	3.1	i386-netbsd-thread-multi-64int		
PASS	2185724	NetBSD	3.1	i386-netbsd-thread-multi-64int		

At the bottom left, there are five horizontal bars representing version coverage from 0.14 to 0.18, all shown as green segments indicating full or near-full coverage.

☺ How many *code branches* and *conditionals*
we have never run yet?



→ Run your *test suite* with Devel::Cover



The screenshot shows a Mozilla Firefox browser window titled "Coverage Summary - Mozilla Firefox". The address bar displays the file path `file:///tmp/Queue-Memcached-Buffered/cover_db/coverage.html`. The browser has two tabs: "A Tour of Perl Testing" and "Coverage Summary". The "Coverage Summary" tab is active, showing a blue header with the text "Coverage Summary". Below the header, the database path is shown as `/home/agentz/git/Queue-Memcached-Buffered/cover_db`. A table displays the coverage data for the file `lib/Queue/Memcached/Buffered.pm` and the total coverage.

file	stmt	bran	cond	sub	time	total
<code>lib/Queue/Memcached/Buffered.pm</code>	80.0	67.3	65.0	83.3	100.0	75.3
Total	80.0	67.3	65.0	83.3	100.0	75.3

The browser window has a status bar at the bottom that says "Done".

😊 What are they?

File Coverage: lib/Queue/Memcached/Buffered.pm - Mozilla Firefox

File Edit View History Bookmarks Tools Help

file:///home/agentz/git/Queue-Memcached-Buffered/cover_db/lib

Devel::Cover

A Tour of Perl Testing

File Coverage: lib/Que...

File: lib/Queue/Memcached/Buffered.pm

Coverage: 75.3%

line	stmt	bran	cond	sub	time	code
1						package Queue::Memcached::Buffered;
2						
3	1			1	4	use strict;
	1				2	
	1				4	
4	1			1	4	use warnings;
	1				2	
	1				5	
5						
6						our \$VERSION = '0.04';
7						
8	1			1	6	use JSON::XS;
	1				2	
	1				5	
9	1				5	use Memcached::libmemcached qw(
10						memcached_create
11						memcached_server_add
12						memcached_set
13						memcached_get

Done

File Coverage: lib/Queue/Memcached/Buffered.pm - Mozilla Firefox

File Edit View History Bookmarks Tools Help

file:///home/agentz/git/Queue-Memcached-Buffered/cover_db/lit Devel::Cover

A Tour of Perl Testing File Coverage: lib/Que...

19						
20						sub new {
21	8	50		8	89	my \$class = ref \$_[0] ? ref shift : shift;
22	8		100		27	my \$opts = shift {};
23	8				19	my \$memc = delete \$opts->{memc};
24	8				19	my \$servers = delete \$opts->{servers};
25	8	100	67		43	if (!defined \$memc && !defined \$servers) {
26	1				2	die "Neither memc nor servers specified.\n";
27						}
28	7	100			21	my \$queue = delete \$opts->{queue} or
29						die "No queue name specified.\n";
30	6	100			15	my \$item_size = delete \$opts->{item_size} or
31						die "No queue item size specified.\n";
32	5				14	my \$flush_period = delete \$opts->{flush_period};
33						
34	5	50			11	if (%\$opts) {
35	0				0	die "Unrecognized option names: ", join(' ', keys %\$opts), "\n";
36						}
37						
38	5	50			12	if (!defined \$memc) {
		0				
39	5	50			14	if (\$servers) {
40	5	50			11	if (!\$servers) {

Done

Branch Coverage: lib/Queue/Memcached/Buffered.pm - Mozilla Firefox

File Edit View History Bookmarks Tools Help

file:///home/agentz/git/Queue-Memcached-Buffered/cover_db/lib Devel::Cover

A Tour of Perl Testing Branch Coverage: lib/...

21	50	T	F	ref \$_[0] ? :
25	100	T	F	if (not defined \$memc and not defined \$servers)
28	100	T	F	unless my \$queue = delete \$opts{'queue'}
30	100	T	F	unless my \$item_size = delete \$opts{'item_size'}
34	50	T	F	if (%opts)
38	50	T	F	if (not defined \$memc) { }
	0	T	F	elsif (defined \$servers) { }
39	50	T	F	if (\$servers)
40	50	T	F	if (not @\$servers)
47	100	T	F	if (not defined \$port)
51	100	T	F	if (not \$port =~ /\d+\$/)
77	100	T	F	if (length(\$elem_json) + 2 > \$\$self{'item_size'})
85	50	T	F	if (defined \$\$self{'flush_period'} and not defined \$\$self{'flush_time'})
89	50	T	F	if (defined \$\$self{'flush_period'} and \$now - \$\$self{'flush_time'} >= \$\$self{'flush_
97	100	T	F	if (length(\$queue) + length(\$elem_json) + length(\$\$self{'write_buf'}) + 1 > \$\$self{'
105	100	T	F	if (\$\$self{'write_buf'} eq '') { }
116	100	T	F	if (\$\$self{'write_buf'} eq '')
123	50	T	F	unless memcached_set(\$memc, \$queue, \$\$self{'write_buf'} . '')
137	100	T	F	if (@\$buf)
143	100	T	F	if (not defined \$elem_list_json)
144	50	T	F	if (defined \$memc->errstr and \$memc->errstr =~ /NOT FOUND/) { }
155	100	T	F	if (\$@)
158	100	T	F	if (not defined \$elem_list or not ref \$elem_list or ref \$elem_list ne 'ARRAY')

Done

☹ *Poors'* automated testing



preparing expected outputs

\$./my-script.pl < input.txt > output.txt

\$ less output.txt # verity the outputs

\$ mv output.txt *output.txt.expected*

hacking on my-script.pl some...

\$./my-script input.txt > output.txt

\$ *diff* -udT *output.txt.expected* output.txt

☺ Testing is usually about *comparing* outputs
with a set of common inputs.



test.t

use Test::More tests => 3;

is 1 + 1, 2, '1 + 1 == 2';

is 1 - 1, 0, '1 - 1 == 0';

ok defined 1, '1 is defined';

\$ *perl* test.t

1..3

ok 1 - 1 + 1 == 2

ok 2 - 1 - 1 == 0

ok 3 - 1 is defined

☺ Standard *TAP* output could be summarized
by **prove**.

\$ *prove* test.t

test....ok

All tests successful.

Files=1, Tests=3, 0 wallclock secs

(0.04 cusr + 0.00 csys = 0.04 CPU)

☺ Let's *plant* a "bug" intentionally into our tests.



test.t

use Test::More tests => 3;

is 1 + 1, 2, '1 + 1 == 2';

is 1 - 1, *1*, '1 - 1 == 0';

ok defined 1, '1 is defined';

\$ *perl* test.t

1..3

ok 1 - 1 + 1 == 2

not ok 2 - 1 - 1 == 0

Failed test '1 - 1 == 0**'**

at test.pl line 5.

got: '0**'**

expected: '1**'**

ok 3 - 1 is defined

Looks like you failed 1 test of 3.

\$ prove test.t

test....

Failed test '1 - 1 == 0'

at test.pl line 5.

got: '0'

expected: '1'

Looks like you failed 1 test of 3.

test....dubious

Test returned status 1 (wstat 256, 0x100)

DIED. FAILED test 2

Failed 1/3 tests, 66.67% okay

Failed Test Stat Wstat Total Fail List of Failed

test.pl 1 256 3 1 2

Failed 1/1 test scripts. 1/3 subtests failed.

Files=1, Tests=3, 0 wallclock secs

(0.01 cusr + 0.00 csys = 0.01 CPU)

Failed 1/1 test programs. 1/3 subtests failed.

☺ A few tests from the Perl 5 test suite

→ Test **Perl 5** with **Perl 5**



```
# perl-current/t/op/arith.t
print "1..145\n";

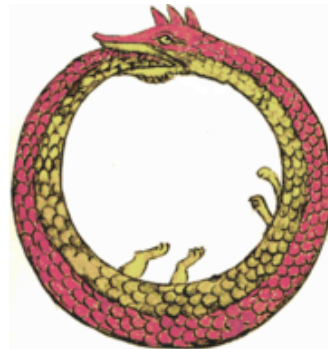
...
sub tryeq ($$$) {
    if ($$_[1] == $_[2]) {
        print "ok $_[0]\n";
    } else {
        print "not ok $_[0] # $_[1] != $_[2]\n";
    }
}

...
my $$T = 1;
tryeq $$T++, 13 % 4, 1;
tryeq $$T++, -13 % 4, 3;

...
```

☺ The same is also true for the **Perl 6** test suite

➔ **Test Perl 6 with Perl 6**



```
# rakudo/t/01-sanity/07-for.t
```

```
use v6;
```

```
say "1..9";
```

```
my @array = <a b c>;
```

```
my $i = 0;
```

```
for @array -> $item {
```

```
    if @array[$i] eq $item {
```

```
        say "ok ", $i + 1
```

```
    } else {
```

```
        say "not ok ", $i + 1
```

```
    }
```

```
    $i++;
```

```
}
```

```
...
```

```
# rakudo/t/spec/S32-str/ucfirst.t
```

```
use v6;
```

```
use Test;
```

```
plan 5;
```

```
# L<S32::Str/Str/ucfirst>
```

```
is ucfirst("hello world"), "Hello world", "simple";
```

```
...
```

☺ Test::Base ➡ *Data-driven* test framework in Perl

```
use Test::Base;
plan tests => blocks() * 1;
run {
    my $block = shift;
    my $name = $block->name;
    my $val = eval($block->expr);
    if (defined $block->test_eq) {
        is $val, $block->test_eq, "$name - test_eq";
    } elsif (defined $block->test_defined) {
        ok defined $val, "$name - test_defined";
    }
};
__DATA__
```

=== TEST 1: 1 + 1 == 2

--- expr: 1 + 1

--- test_eq: 2

=== TEST 2: 1 - 1 == 0

--- expr: 1 - 1

--- test_eq: 0

=== TEST 3: 1 is defined

--- expr: 1

--- test_defined

\$ *prove* test-base.t

test-base....ok

All tests successful.

Files=1, Tests=3, 0 wallclock secs

(0.03 cusr + 0.00 csys = 0.03 CPU)

- ☺ Let's see some test cases from the *GNU make* official test suite (using PPI to generate tests based on Test::Base from the original Perl 4 code)

```
use t::Gmake;  
plan tests => 3 * blocks;  
run_tests;
```

```
__DATA__
```

```
=== TEST 1: rules with no command
```

```
--- source
```

```
a: b
```

```
b: c
```

```
c;; echo 'hello!'
```

```
--- stdout
```

```
echo 'hello!'
```

```
hello!
```

```
--- stderr
```

```
--- error_code
```

```
0
```

=== TEST 2: command at beginning

--- source

a: b

b: c

c;; echo 'hello!'

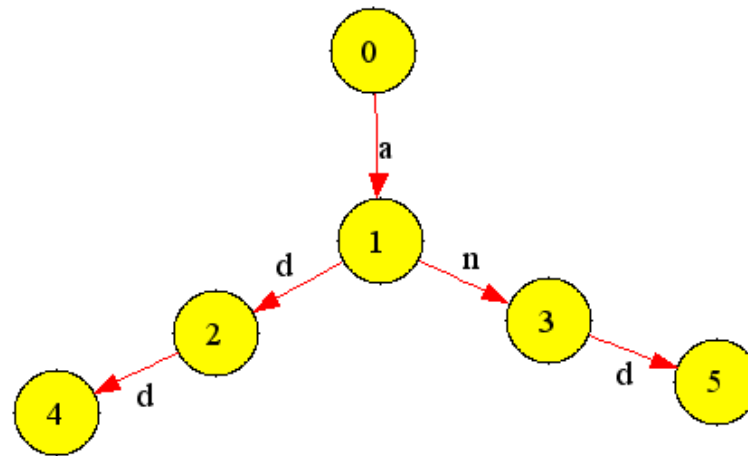
--- stdout

--- stderr_like

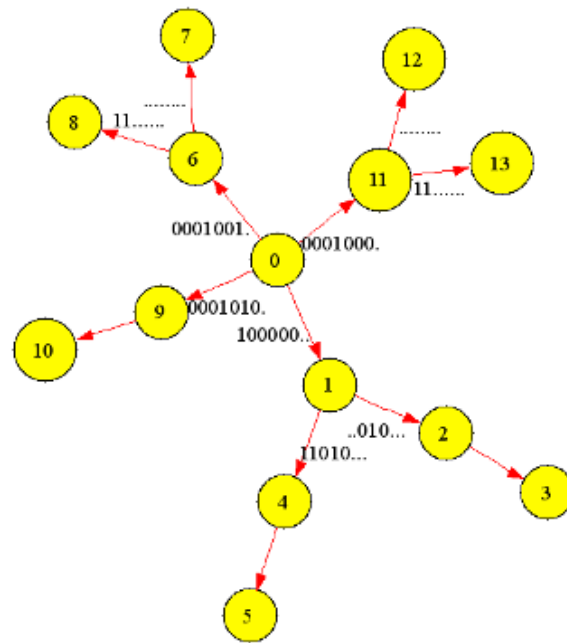
.*?commands commence before first target.*?

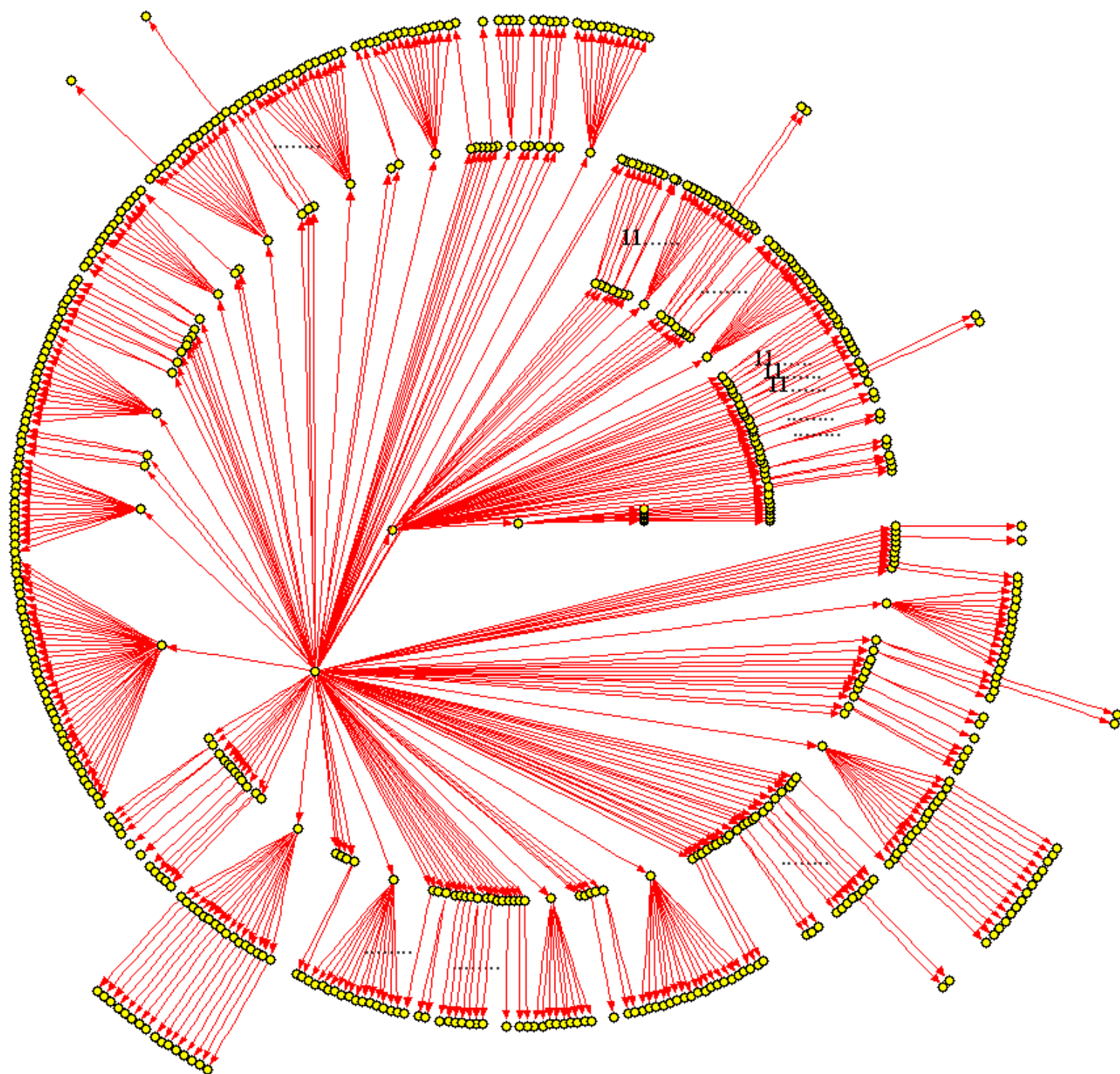
--- error_code: 2

☺ Generate a *random* test suite from **state machines**



☺ State machines *driving* the tests of
an *IA-32* dissambler in my Salent project.





☺ The state machine was constructed *automatically*
from Intel's instruction table in its **manual**.

<http://svn.berlios.de/svnroot/repos/salent/idu/encoding.txt>

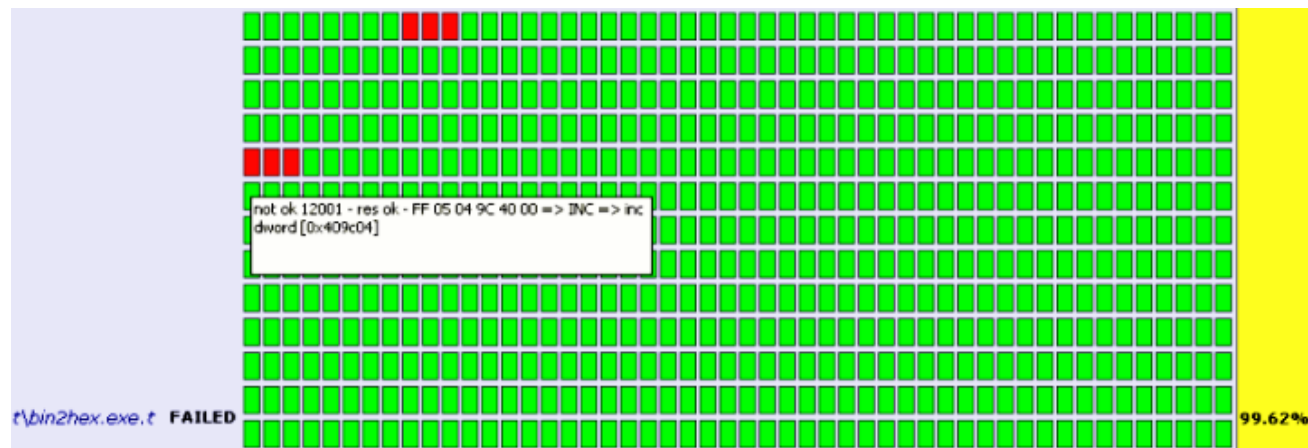
☺ Generate pretty *HTML reports* from TAP outputs

```

---
build_info: Balena v0.12 Instruction Decoding Unit
config:
  concurrent: 1
  exclude:
    - Disabled
  exclude_x86: {perl/segexp:
    REGEXP: (Disabled)}
  cdb: {}
  cdb_x86: {perl/segexp:
    REGEXP: \.t}$
  output-file: tests.yml
  x86cpu: 1
  x86cpu: 0
  test_count: 17
  revision: unknown
  smoke:
    email: agene2002@126.com
    name: Agene Sheng
  timing
  duration: 49
  and: 1125578761
  or: 1125578761

```

Test file	Test cases	%
t\01disasm.t	OK	100.00%
t\02disasm.t	OK	100.00%
t\03disasm.t	OK	100.00%
t\04util.t	OK	100.00%
t\05.t	OK	100.00%
t\bin2asm.t	OK	100.00%
t\bin2hex.t	OK	100.00%
t\catin.t	OK	100.00%



☺ Test *web services* using Test::Base

```
use t::OpenResty;  
plan tests => 3 * blocks() - 3 * 2;  
run_tests;  
__DATA__
```

=== TEST 1: Login w/o password

--- request

GET /=/login/\$TestAccount.Admin

--- response

{"success":0,"error":"\$TestAccount.Admin is not anonymous."}

=== TEST 2: Delete existing models (w/o login)

--- request

DELETE /=/model.js

--- response

{"success":0,"error":"Login required."}

☺ Utilize *multiple* cores while running *big* test suites.

```
$ prove -j4 -r t
```

😊 *Any questions?* 😊

