

1 Which Coding Technique is Faster

1.1 Description

This document tries to show more efficient coding styles by benchmarking various styles.

WARNING: This doc is under construction

META: for now these are just unprocessed snippets from the mailing list. Please help me to make these into useful essays.

1.2 backticks vs XS

META: unprocessed yet.

compare the difference of calling an xsub that does `_nothing_` vs. a backticked program that does `_nothing_`.

```
/* file:test.c */
int main(int argc, char **argv, char **env)
{
    return 1;
}

/* file:TickTest.xs */
#include "EXTERN.h"
#include "perl.h"
#include "XSUB.h"

MODULE = TickTest          PACKAGE = TickTest

void
foo()

CODE:

# file:test.pl
use blib;
use TickTest ();

use Benchmark;

timethese(100_000, {
    backtick => sub { `./test` },
    xs => sub { TickTest::foo() },
});
```

Results:

```
Benchmark: timing 100000 iterations of backtick, xs...
backtick: 292 wallclock secs (18.68 usr 43.93 sys + 142.43 cusr 84.00 csys = 289.04 CPU) @ 1597.19/s (n=100000)
xs: -1 wallclock secs ( 0.25 usr +  0.00 sys =  0.25 CPU) @ 400000.00/s (n=100000)
(warning: too few iterations for a reliable count)
```

1.3 sv_catpv vs. fprintf

META: unprocessed yet.

and what i'm trying to say is that if both the xs code and external program are doing the same thing, xs will be heaps faster than backticking a program. your xsub and external program are not doing the same thing.

i'm guessing part of the difference in your code is due to fprintf having a pre-allocated buffer, whereas the SV's SvPVX has not been pre-allocated and gets realloc-ed each time you call sv_catpv. have a look at the code below, fprintf is faster than sv_catpv, but if the SvPVX is preallocated, sv_catpv becomes faster than fprintf:

```
timethese(1_000, {
    fprintf    => sub { TickTest::fprintf() },
    svcat      => sub { TickTest::svcat() },
    svcat_pre => sub { TickTest::svcat_pre() },
});
```

Benchmark: timing 1000 iterations of fprintf, svcat, svcat_pre...

```
fprintf:  9 wallclock secs ( 8.72 usr +  0.00 sys =  8.72 CPU) @ 114.68/s (n=1000)
svcat:   13 wallclock secs (12.82 usr +  0.00 sys = 12.82 CPU) @  78.00/s (n=1000)
svcat_pre:  2 wallclock secs ( 2.75 usr +  0.00 sys =  2.75 CPU) @ 363.64/s (n=1000)
```

```
#include "EXTERN.h"
#include "perl.h"
#include "XSUB.h"
```

```
static FILE *devnull;
```

```
MODULE = TickTest          PACKAGE = TickTest
```

```
BOOT:
```

```
devnull = fopen("/dev/null", "w");
```

```
void
```

```
fprintf()
```

```
CODE:
{
    int i;
    char buffer[8292];

    for (i=0; i<sizeof(buffer); i++) {
        fprintf(devnull, "a");
    }
}
```

```
void
```

```
svcat()
```

```
CODE:
{
```

1.4 Maintainers

```
    int i;
    char buffer[8292];
    SV *sv = newSV(0);

    for (i=0; i<sizeof(buffer); i++) {
        sv_catpvn(sv, "a", 1);
    }

    SvREFCNT_dec(sv);
}

void
svcat_pre()

CODE:
{
    int i;
    char buffer[8292];
    SV *sv = newSV(sizeof(buffer)+1);

    for (i=0; i<sizeof(buffer); i++) {
        sv_catpvn(sv, "a", 1);
    }

    SvREFCNT_dec(sv);
}
```

1.4 Maintainers

Maintainer is the person(s) you should contact with updates, corrections and patches.

Stas Bekman [<http://stason.org/>]

1.5 Authors

- Stas Bekman [<http://stason.org/>]
- Doug MacEachern <dougm (at) covalent.net>

Only the major authors are listed above. For contributors see the Changes file.

Table of Contents:

1	Which Coding Technique is Faster	1
1.1	Description	2
1.2	backticks vs XS	2
1.3	sv_catpv vs. fprintf	3
1.4	Maintainers	4
1.5	Authors	4