Unsuitable 1.0

The World's First Blog Engine
Written in Forth.

Problems (July 2009)

- Web Hosting Provider Folding in Few Weeks
- Using Serendipity with PostgreSQL Database
- Need to install new blog, retaining content
  - Local Installation of Serendipity with MySQL
- Sounds simple to migrate!

Problems

- Local Serendipity Failed to Import from Remote Serendipity Instance
  - Only first 20 or so articles succeeded
  - No comments were imported

Problems

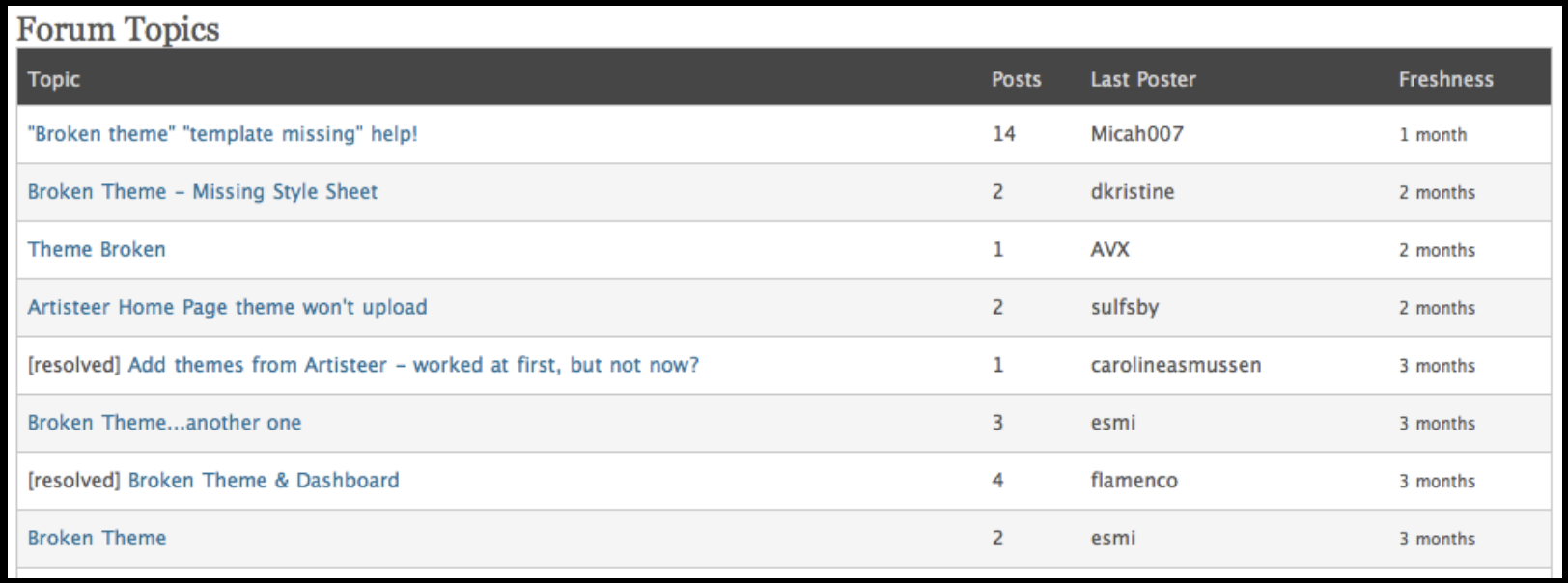
- Serendipity Suffers Numerous Page-Rendering Bugs too.
  - Code listings were always double-spaced.
  - Sometimes you needed \ to escape its wiki-inspired format codes, sometimes not.

Problems

- Decided to try Wordpress.
  - Themes broken out of the box!! *Ouch!*
  - Could not fix them by changing themes. *Ouch!*
  - Failed to import comments from Serendipity.

Problems

- Decided to try Wordpress.
  - I am not alone: 29 complaints in last 3 months



Forum Topics

Topic	Posts	Last Poster	Freshness
"Broken theme" "template missing" help!	14	Micah007	1 month
Broken Theme – Missing Style Sheet	2	dkristine	2 months
Theme Broken	1	AVX	2 months
Artisteer Home Page theme won't upload	2	sulfsby	2 months
[resolved] Add themes from Artisteer – worked at first, but not now?	1	carolineasmussen	3 months
Broken Theme...another one	3	esmi	3 months
[resolved] Broken Theme & Dashboard	4	flamenco	3 months
Broken Theme	2	esmi	3 months

Problems

- Decided to try Movable Type
  - Could not import *anything* from Serendipity.
  - Configuration was a pain.
    - Intended for large-scale sites with dedicated support personnel.
    - I had to pay for the theme I wanted. *Ouch!*

Problems

- It Becomes Personal!

"My crusade in this post is that

Forth is emphatically not the right tool for the job." – lars_ (reddit.com)

Light Bulb!

- By this time, I had only one day left of hosting. I grabbed an SQL dump, intending to import material into whatever blog engine I eventually settled on.

- But then I remembered all those previous articles where I said, “I should write my own blog in Forth someday.”

It was *now*

or *never!*

Simplifications

- Comments
  - Not worth the effort to implement!
  - 99% spam by volume.
  - Serendipity required non-stop administrative expense on my part.
  - Most contemporary blogs moving towards hiding comments by default, because let's face it, *they're just plain ugly and distracting.*

Simplifications

- Comments
  - I'm not alone!
    - Laughing Squid (removed)
    - LispCast (hidden)
    - WSJ (never used; comments on Facebook now!)
    - Science Daily (never existed)
  - Therefore, Unsuitable doesn't support comments.

Simplifications

- Searches
  - Google site:foo.com search works great!
  - It's almost like owning a GSA, but cheaper.
  - Faster than any blog search engine I've used.
  - Therefore, Unsuitable does not provide search capability.

Simplifications

- Falvotech.com belongs only to me.
  - I have no guest authors.
  - No theme: I discuss whatever I'm thinking at the moment.
  - Supporting multiple authors implies user and rights management complexities.
  - Therefore, Unsuitable hardcodes the author of all articles.

Simplifications

- Visual Appearance
  - Different page layouts *really* requires different article flows.
  - I *expect* to make no money from my blog; material no longer of interest to the world at large.
  - Automatic flow management, a la T_EX, is hard!!
  - Therefore, Unsuitable hardcodes much of the visual layout, while accepting essentially raw HTML as input.

Simplifications

- Article Submission Procedure
  - I'm the sole author for all content.
  - Multiple authors -> user management -> management user interfaces.
  - SSH already provides spectacularly awesome authentication.
  - Therefore, Unsuitable requires article submission from command-line.

Deliberate Complexification

- Don't use files for persistence
- Would not be "Forth-like" if I did.
- Instead, use 1970s-era, natively-hosted Forth BLOCK system (DASD).
- Reminds me of allocating data-sets on IBM mainframes. Only way, way, *way* easier and cooler.
- Surprising results!!

The Result

- Only the essentials:
  - Organizes catalog of articles.
  - Provides RSS feed.
  - Permits user to drill down into an article.
- 371 lines of code total.
  - Includes comments and blank lines!
  - Not all compiled at once, though.
- Pure CGI Interface for Lighttpd.

The Result

```
$ siege -c 250 -i http://www.falvotech.com/
  …
Transactions:                  39709 hits
Availability:                 100.00 %
Elapsed time:                 290.39 secs
Data transferred:             291.71 MB
Response time:                  1.31 secs
Transaction rate:             136.74 trans/sec
Throughput:                     1.00 MB/sec
Concurrency:                  179.17
Successful transactions:       39709
Failed transactions:               0
Longest transaction:           14.02
Shortest transaction:           0.01
```

The Result

```
$ siege -f urls.txt -c 250 -I
   …
Transactions:                  47613 hits
Availability:                 100.00 %
Elapsed time:                 366.20 secs
Data transferred:             325.27 MB
Response time:                  1.39 secs
Transaction rate:             130.02 trans/sec
Throughput:                     0.89 MB/sec
Concurrency:                  180.16
Successful transactions:       47613
Failed transactions:               0
Longest transaction:           19.04
Shortest transaction:           0.02
```

The Result

Lighttpd is definitely the bottleneck!

371 Lines of What?

- Forth Machine Model
  - A “Dictionary”
  - Persistent Storage in “blocks”
  - A Data Stack
  - A Return Stack
- Forth code *always* reads left to right.
  - Except when it doesn’t.

371 Lines of What?

- Forth is Interpreted

371 Lines of What?

- Forth is Interpreted

32 180 100 */ 32 +

371 Lines of What?

- Forth is Interpreted

32 180 100 */ 32 +

DATA STACK: 32

371 Lines of What?

- Forth is Interpreted

32 180 100 */ 32 +

DATA STACK: 32 180

371 Lines of What?

- Forth is Interpreted

32 180 100 */ 32 +

DATA STACK: 32 180 100

371 Lines of What?

- Forth is Interpreted

32 180 100 */ 32 +

DATA STACK: (32*180)/100 or 57.6

371 Lines of What?

- Forth is Interpreted

32 180 100 */ 32 +

DATA STACK: 57.6 32

371 Lines of What?

- Forth is Interpreted

32 180 100 */ 32 +

DATA STACK: 89.6

371 Lines of What?

- Forth is Compiled

```
: >F   180 100 */ 32 + ;
```

- : defines new words. Note : is like any other word!
- ; closes the definition.
- Names may contain *any* non-space character.

371 Lines of What?

- Forth recognizes three distinct times: ICE
  - Interpret-time.
  - Compile-time.
  - Edit-time.

371 Lines of What?

- Forth recognizes three distinct times: ICE

```
: >F   180 100 */ 32 + ;
32 >F
```

371 Lines of What?

- Data Stack Manipulation
  - Sometimes, required data isn't where you need it to be.

1 2 3	DUP	1 2 3 3
1 2 3	OVER	1 2 3 2
1 2 3	DROP	1 2
1 2 3	NIP	1 3
1 2 3	ROT	2 3 1

371 Lines of What?

- Return Stack Used to Remember What to Do Next
  - Think, “Return to what I was doing.”
  - May also be thought of as partial continuations.
- Return Stack Manipulators
  >R (to-R)
  R> (R-from)

371 Lines of What?

- Structured Returns Common in Functional Languages Today
  - Forth coders have been doing this since the 60s!!

```
: onlyIfOdd     dup 1 and 0= if r> drop then ;
: testIt        onlyIfOdd ." This number is odd." cr ;

3 testIt   4 testIt
```

- Forth thus allows *factoring control flow!*

371 Lines of What?

- Structured Returns and Deep Stacks Lead to Spaghetti If Abused.
- Used properly, however, they permit Forth to express in 371 lines what Java/Perl/Python/ et. al. require ~2000 for.
- Compression also comes from eliminating redundancy in the source code.

371 Lines of What?

In Forth:

```
: >core
   dup 10 rshift block swap 1023 and + ;
```

In Java (Hypothetical):

```
public Ptr<char>
convertToCoreAddress(int address) {
   int blkAddress;

   blkAddress =
      fillBlockCache(address >> 10);
   return
      new Ptr<char>(blkAddress +
         (address & 1023));
}
```

A Word about Types

- Forth, the language, specifies no types.
  - Moore believes, rightly, that a potentially infinite number of sub-types exist for any non-trivial application.

    Ex: Form-raw strings vs. HTML-escaped strings

    Ex: Does the range [1, 7] correspond to days of the week or a valid menu selection? Both? And, when?

A Word about Types

- Type checking done through Hoare Logic.
  - Prefer edit-time "checking" with Hoare Logic.
  - Compiler almost always ignorant about types. See StrongForth for an exception.
  - Interpret-time checking preferred over run-time checking (c.f. Unsuitable's modules, TDD, etc.)
  - Run-time checking enforceable through execution of guards and preconditions.

Workflow

Workflow

- Unsuitable receives request in PATH_INFO.

```
S" PATH_INFO" getenv
   constant /path-info
   constant &path-info
```

- Convention: Read */x* as "size of *x*," as in bytes *per x*.

Workflow

- Unsuitable receives request in PATH_INFO.
- Basic sanity checks ensure obvious breakage attempts yields an index page.

```
: |url|>=2
   /path-info 2 u< if
     s" m-index.fs" included bye
   then ;
|url|>=2
```

Workflow

- Unsuitable receives request in PATH_INFO.
- Basic sanity checks ensure obvious breakage attempts yields an index page.
- More sophisticated forms of breakage requires compromising Lighttpd itself.
- Based on URL component, dispatch to specific Forth module.

Workflow

- URL Structure

http://*host-name*/blog.fs/module/parameters

Resolved by POWS

Selects Unsuitable Request Handler

Optional

Workflow

http://*host-name*/blog.fs/module/parameters

```
&path-info /path-info +
     constant end-of-url

&path-info 1+
     constant module
```

Workflow

http://*host-name*/blog.fs/module/parameters

```
: -eou   dup end-of-url >= if r> drop then ;
: -/     dup c@ [char] / = if r> drop then ;
: slash  begin -eou -/ char+ again ;
module slash constant &parameters
```

Workflow

http://*host-name*/blog.fs/module/parameters

```
: path       [char] m c, [char] - c, ;
: base       module &parameters over -
             here swap dup allot move ;
: extension  S" .fs" here swap dup allot
             move ;
: filename   path base extension ;
: dispatch   here  filename  here over -
             included ;
dispatch bye
```

Workflow

- Each module responsible for validating and interpreting URL parameters, if any.
- Modules exhibit bracket structure:
  - First “bracket”: examine URL and make sure it’s valid for this module. Don’t compile rest of the module unless it’s worth it!

Workflow

- Each module responsible for validating and interpreting URL parameters, if any.

```
end-of-url &parameters -
   constant /parameters
: oops
   s" m-index.fs" included bye ;
: |parameters|>=2
   /parameters 2 u< if oops then ;
|parameters|>=2
```

Workflow

- Each module responsible for validating and interpreting URL parameters, if any.
- Modules exhibit bracket structure:
  - Interior: macro definitions and support utilities.

```
: exists:
    dup -1 = if r> 2drop then ;
: .f
    id @ articleWithId!
    execute exists: gob! get, ;
: title    ['] title .f ;
: lead     ['] lead .f ;
: body     ['] body .f ;
```

Workflow

- Each module responsible for validating and interpreting URL parameters, if any.
- Modules exhibit bracket structure:
  - Second “bracket”: render the resulting page using the above-defined macros, if any.

```
variable s
variable end
: mime
    ." Content-type: text/html" cr cr ;
: valid
   mime  here s !
   s" article.html" slurp here end !
   s" response.fs" included bye ;
```

Storage

- Blog written as if it were orthogonally persistent.
- BLOCKs implement software-managed virtual memory with 1024 byte “pages.”
  - @f and !f to fetchs and stores (resp.) to *persistent* space.
  - @ and ! for *normal* core/dictionary storage.

Storage

From	To	Description
00000000	000003FF	Vestigial boot block storage[1]
00000400	000007FF	GOS and DB Metablock
00000800	000107FF	unused[2]
00010800	00010FFF	2K GOS Handle Table[3]
00011000	000137FF	10K Articles Table[3]
00013800	0003FFFF	unused
00040000	0013FFFF	1MB GOS blob storage

1 Used back when Forth implementations doubled as a machine's operating system.
2 Formerly, 64K GOS blob storage.
3 At current blog article posting rate, should suffice for another *five years* of posts!

Storage

- Administering Storage Surprisingly Quick/Easy
  - Filesystems not as useful as I expected for this project!
  - When I ran out of my first GOS (64K), I expanded it to 256K, including relocating it in the block space, in only 10 minutes.
  - GOS now is 1MB in size.
  - I anticipate GOS exhaustion around Feb 2011.

No Libraries!

- Dwindling user-base since Forth eclipsed by C in mid-80s.
- Hence, no libraries exist for collections, networking, etc.
  - That's OK though: Forth philosophy prefers building from scratch for production code.
  - Re-using code considered OK for R&D though!
- Code re-use takes backseat to concept re-use.

Compiles on Every Page View

- Forth compilers work *very* fast!
  - Gforth compiles 45,801 LOC/s on my web server.
  - Gforth-fast compiles 54,500 LOC/s.
- If *every line* of the blog were compiled for a single request, it would introduce only 8ms of latency.

Future Directions

- Interest Expressed in a Forth Web Application Server!
- Adding Features to Unsuitable:
  - Series of Articles (e.g., Unsuitable on Unsuitable)
  - Different Navigation Methods?
  - Automated Article Obsolescence Tracking?
  - Support for Different Kinds of Articles?

Additional Resources

- Blog Address
  - http://www.falvotech.com/blog2/blog.fs
- Unsuitable on Unsuitable
  - http://www.falvotech.com/blog2/blog.fs/articles/1030
- Declarative, Imperative, then Inquisitive Pattern
  - http://www.falvotech.com/blog2/blog.fs/articles/1022

Additional Resources

- Starting Forth
  - http://www.forth.com/starting-forth/
- Thinking Forth
  - http://thinking-forth.sourceforge.net/
- Silicon Valley Forth Interest Group
  - If you don't mind a bunch of disgruntled Forth coders reminiscing of the good-ol' days… ;-)
  - http://www.forth.org/svfig

Thank You!

Q & A