mandoc

from scratch to the standard BSD documentation toolkit in 6 years

EuroBSDCon, Stockholm, October 4, 2015

Ingo Schwarze <schwarze@openbsd.org>

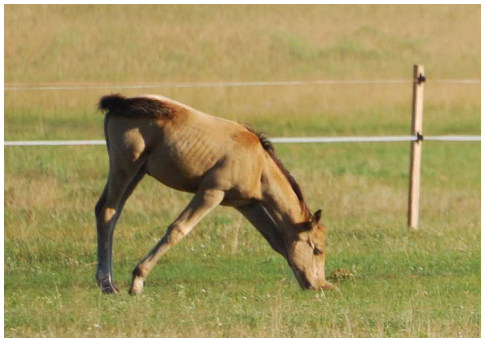
using some material from:

Training a foal to replace a venerable workhorse: mandoc in OpenBSD BSDCan 2011

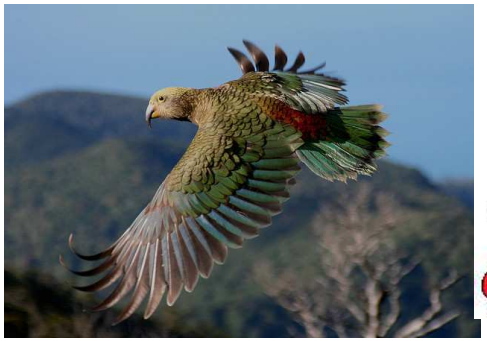
Enhancing the modern toolbox for the classic documentation formats: new trends in mandoc BSDCan 2014

Let’s make manuals more useful! EuroBSD-Con 2014

mandoc: becoming the main BSD manual toolbox BSDCan 2015

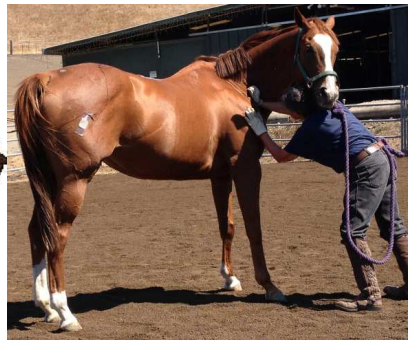


Csikó — Foal © 2010
Adam Tomkó @flickr (CC)



Kea juvenile © 2007
Brent Barrett @flickr (CC)

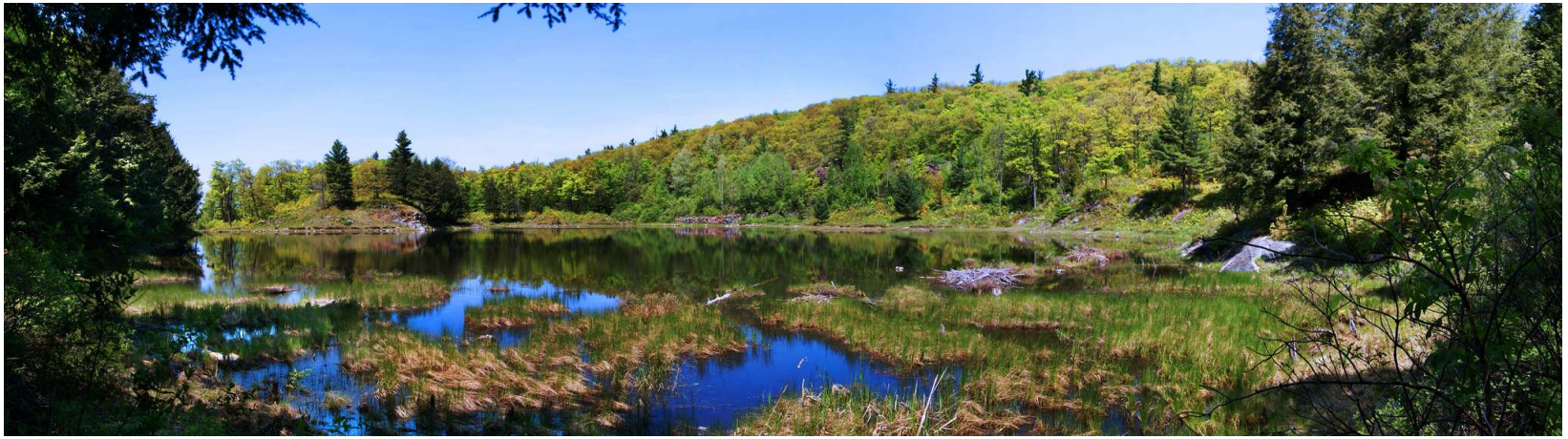
София © 2014
Алица Димитрова



“Bedifferent” © 2013
Cynthia Livingston

Contents

http://mdocml.bsd.lv/press.html has all the slides of these talks

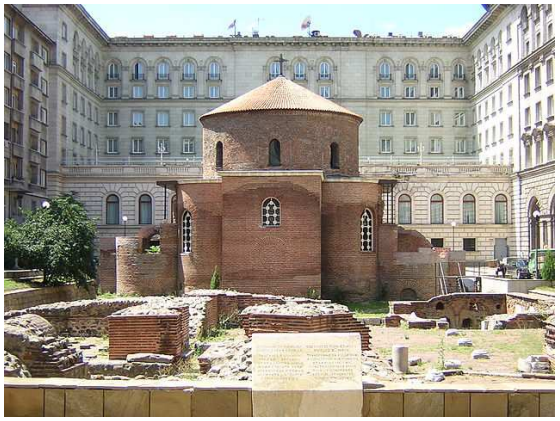


Black Lake near King Mountain, Gatineau Park, Quebec, Canada © 2012 Lezumbalaberenjena@flickr (CC)

Let’s make manuals more useful!

Requirements for good documentation

- correct
- complete
- concise
- easy to find and access,
  all in one local place
- not just plain text: function of words
  must be marked up for display and search



Ротонда Свети Георги, София
© 2006 Преслав @wikimedia (PD)

- easy to read: in particular, uniform display markup and style
- easy to write: in particular, one simple, standard input language

The formatted documentation must seem simple to end users.

The language to write documentation must seem simple to programmers.

Remember:
Without documentation, code is unusable,
and bad documentation is about as bad as bad code.

How does the user experience manuals? →

< >

Let’s make manuals more useful - the user’s perspective.

Let’s read a manual!

One comprehensive tool:
man(1), the manual viewer

1. Find one or more manuals
   in the database or file system.
2. Transparently call a formatter on them.
3. Display the formatted text,
   typically in a pager.

Мальовица 2729м, Рила, Bulgaria
© 2007 Стелиян Касабов @flickr (PD)

Progress during the last year:

- The mandoc toolbox now provides one single tool for all this.
- Unified, simple user interface available since August 26, 2014.
- The traditional way required multiple programs: apropos for searching, man for steering, nroff or mandoc for formatting, ...
- Semantic searching of pages with apropos(1) in production since April 14, 2014.
- Semantic searching within a page in ctags(1) style since July 17, 2015.
- New man.cgi(8) is online on www.openbsd.org since July 12, 2014.

What do authors use? →

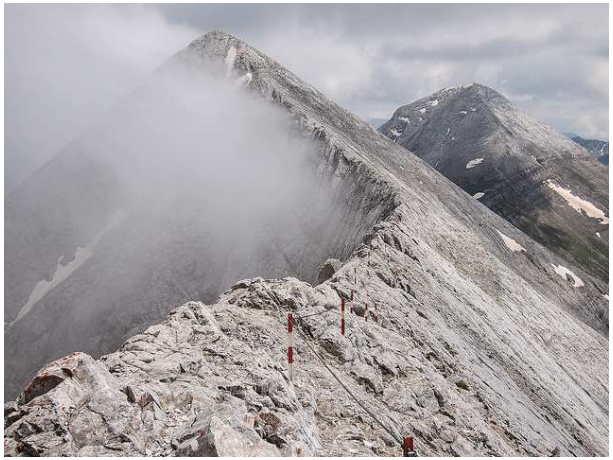
< >

Let’s make manuals more useful - the author’s perspective.

Let’s write manuals!

So we need a markup language that is:

- easy to write
- easy to diff(1)
- easy to read and change
- easily supports semantic markup
- readily produces various output formats
- easily supports portability



Кутело 2908м и Вихрен 2914м, Пирин
© 2010 Кирил Русев @flickr (CC)

One simple, versatile language: **mdoc(7)** — with a long tradition:

- Based on the roff(7) language (Jerome Saltzer, 1964).
- Successor of the man(7) language (AT&T v7 UNIX, 1979).
- Designed for 4.4BSD at Berkeley (Cynthia Livingston, 1989/90).
- Implemented by mandoc(1) (2008) and groff(1) (1989).
- Used by default in OpenBSD, NetBSD, FreeBSD, DragonFly and illumos.

Was there any progress during the last 35 years? →

Enhancing the modern toolbox for the classic documentation formats:

Origin of semantic markup in manuals

4.2BSD Beastie

- The mdoc(7) semantic markup macro language.
- First in 4.3BSD-Reno by Cynthia Livingston, USENIX, 1990.
- Formatted with Brian Kernighan's device independent troff,
- written in K&R C, running on BSD UNIX on DEC VAX.

Advantages of the mdoc(7) language

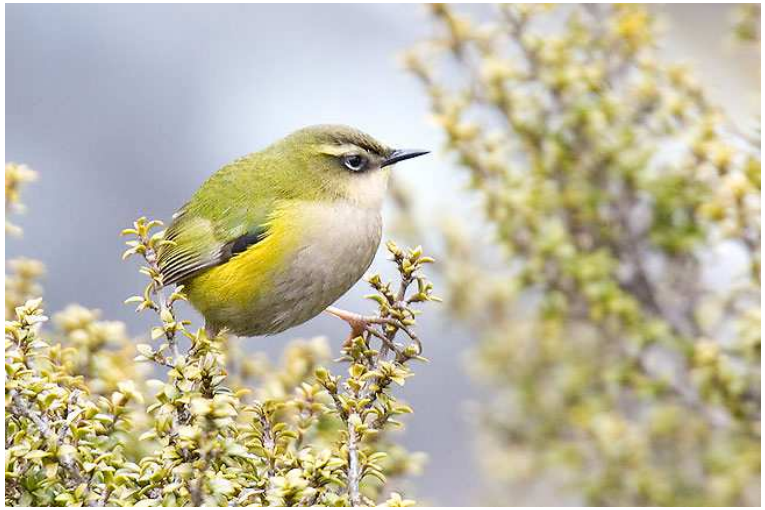
- Considerable expressive power for semantic markup —
  while man(7) is a presentation level language only.
- Works in practice as a standalone language —
  while man(7) regularly requires resorting to low-level roff features.
- Consequently, more uniform appearance, easier to read and write than man(7).
- Portability is no longer an issue: for legacy systems still not having mdoc(7),
  mandoc(1) can be used to convert to man(7).
- Facilitates semantic searching — see this talk.

Classic documentation formats (summary)

- The roff(7) input syntax,
- the mdoc(7) semantic markup,
- and the man(1) presentation format

have proven timeless by their simplicity and efficiency:

- Nobody has come up with a better basic concept yet,
- even though many have tried,
- and regarding the formats, there is indeed little one could wish.



Rock Wren / Hurupounamu (Xenicus gilviventris) New Zealand © 2006 57Andrew@flickr (CC)

So we need modern tools for all this! →

Enhancing the modern toolbox for the classic documentation formats:

Advantages of mandoc(1)

- Functional — all in one binary:
  - searching by filename, page name, word, substring, regular expression
  - searching by semantic keys in and across pages
  - mdoc(7), man(7), tbl(7), eqn(7), and some roff(7) input
  - ASCII, UTF-8, HTML 5, PostScript, PDF output
  - mdoc(7) to man(7) conversion
  - includes mandoc(1), man(1), apropos(1), whatis(1), makewhatis(8)
- Free — ISC/BSD-licensed, no GPL code.
- Lightweight — ANSI C, POSIX, no C++ code.
- Portable — includes compat_*.c files for missing functions on older systems.
- Small — source tarball (uncompressed) is 8% of groff, executable binary 50%.
- Fast — for mdoc(7), typically 5 times faster than groff, typically about a hundred times faster than an AsciiDoc/DocBook toolchain.

Basic functionality already presented during BSDCan 2011.

How does searching work? →

< >

Searching for manual pages

Traditional functionality is preserved

apropos *keywords* ...
: Search case-insensitively for substrings in names and descriptions.

man −k *arguments*
: As before, an alias for: apropos *arguments*
But now also supports new-style arguments, see below.

apropos [**−C** *file*] [**−M** *path*] [**−m** *path*] [**−S** *arch*] [**−s** *section*] *keywords* ...
: Traditional options are all supported.

whatis *keywords* ...
: Search case-insensitively for complete words in page names only.

makewhatis
: Rebuild all configured databases.

makewhatis −d *directory files* ...
: Update entries for the given *files* in one database.

backward compatible

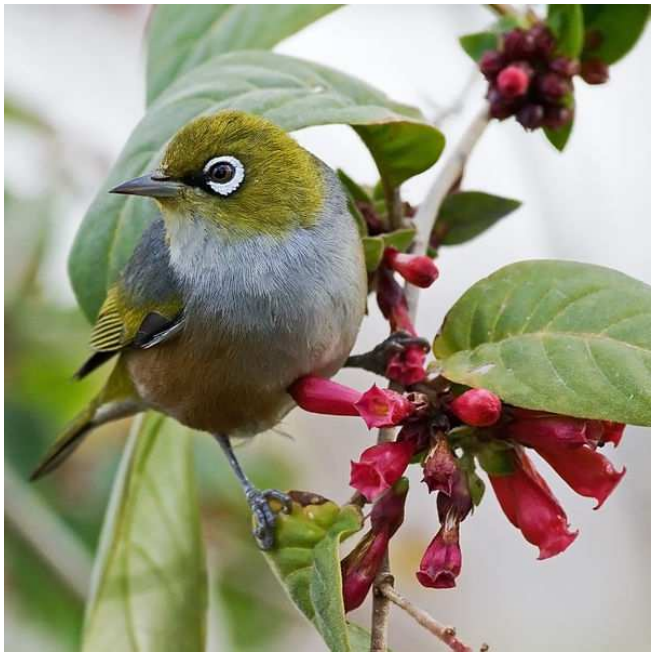
Southern Kiwi / Tokoeka (Apteryx australis)

Markup-sensitive search

```
 $ apropos Ev=USER
Mail, mail, mailx(1) – send and receive mail
csh(1) – a shell (command interpreter) with Clike syntax
login(1) – log into the computer
logname(1) – display user's login name
slogin, ssh(1) – OpenSSH SSH client (remote login program)
su(1) – substitute user identity
[...]
```

Macro keys that can be searched for (examples, ordered by frequency):

Nm	manual page names
Nd	manual page descriptions
sec	manual section numbers
arch	machine architectures
Xr	cross references
Ar	command argument names
Fa	function argument types and names
Dv	preprocessor constants
Pa	file system paths
Cd	kernel configuration directives
Va	variable names
Ft	function return types
Er	error constants
Ev	environment variables
In	include file names
St	references to standards documents
An	author names
...	and so on ...



Silvereye / Tauhou (Zosterops lateralis)
© 2008 J. J. Harrison @wikimedia (CC)

Markup-sensitive search features

Search keys can be OR'ed:

```
 $ apropos Fa,Ft,Va,Vt=timespec
EV_SET, kevent, kqueue(2) – kernel event notification mechanism
clock_getres, clock_gettime, clock_settime(2) – get/set/calibrate date and time
futimens, futimes, utimensat, utimes(2) – set file access and modification times
nanosleep(2) – high resolution sleep
parse_time(3) – parse and unparse time intervals
poll, ppoll(2) – synchronous I/O multiplexing
pselect, select(2), FD_CLR, FD_ISSET, FD_SET, FD_ZERO(3) – synchronous I/O multiplexing
sem_timedwait, sem_trywait, sem_wait(3) – decrement (lock) a semaphore
tstohz, tvtohz(9) – translate time period to timeout delay
[...]
```

Searching across all keys is possible:

```
 $ apropos any=ulimit
ksh, rksh(1) – public domain Korn shell
sh(1) – public domain Bourne shell
getrlimit, setrlimit(2) – control maximum system resource consumption
```

Regular expressions are supported ("~" instead of "="):

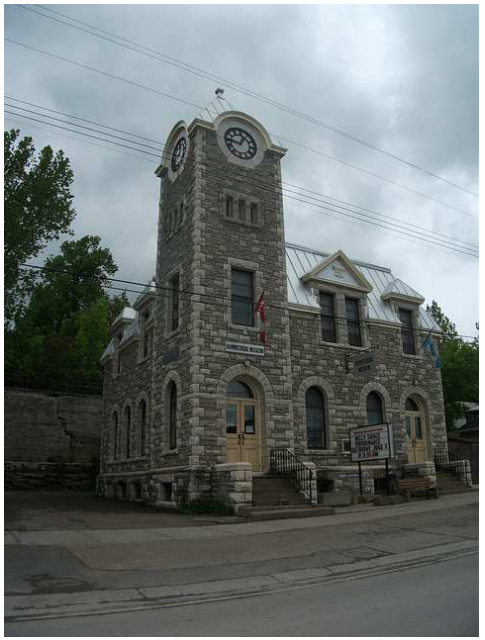
```
 $ apropos "Nm~^[gs]et.*gid"
endgrent, getgrent, getgrgid, getgrgid_r, getgrnam, getgrnam_r, setgrent, setgrfile, setgroup
getegid, getgid(2) – get group process identification
getpgid, getpgrp(2) – get process group
getresgid, getresuid, setresgid, setresuid(2) – get or set real, effective and saved user or
setegid, seteuid, setgid, setuid(2) – set user and group ID
setpgid, setpgrp(2) – set process group
setregid(2) – set real and effective group IDs
[...]
```


Why was man(1) replaced? →

A better manual viewer

Current advantages

- Unified interfaces:
  man can use: -W -T -O -I from mandoc(1)
  apropos can use: -w -h -a from man(1)
- Allow much simpler man.conf(5) format
- Database priority now overrides section priority
- Use additional names from .Dt/.TH and NAME .Nm (250 new entries in OpenBSD, compared to 10200 existing ones)
- One less userland program to maintain
  (that had rather old code)



Bonnechere Museum, Eganville

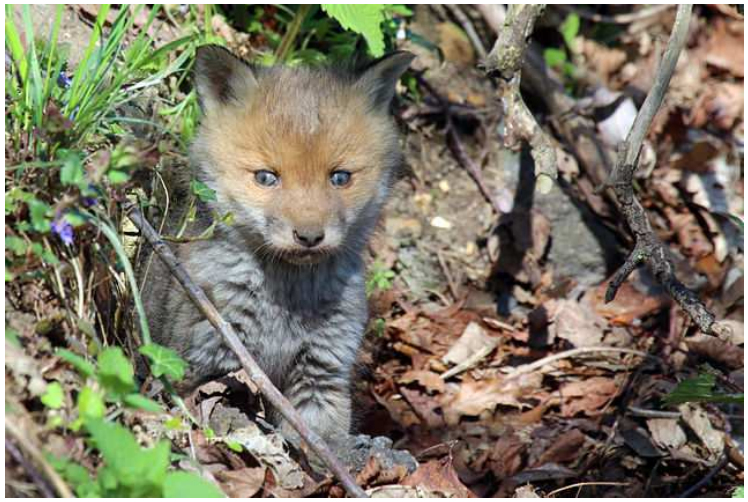
Possible future advantages

- Possibility to get rid of multiple ln(1) links to the same manual
- Possibility to implement an interactive chooser (-i)

Are manuals only available at the console? →

Web interface for manual search and display

- Same user interface as on the command line: man(1) and whatis(1) mode, same query syntax
- Main additional feature: hyperlinks
- Additional potential due to preserved semantic markup, not really used yet for anything except (simple) CSS formatting
- Same directory structure and database format on the server
- Configuration instructions in man.cgi(8)



Red fox kit (Vulpes vulpes)

- Useful mainly for providers of operating systems and large software packages.
- Running your own copy of the manuals of your favourite system is a bad idea:
- It will get outdated and confuse people.

Special thanks to *Sébastien Marie*:
He did an extensive security audit of the code and reported a considerable number of security-relevant bugs that have all been fixed by now.

Internal searching and deep linking

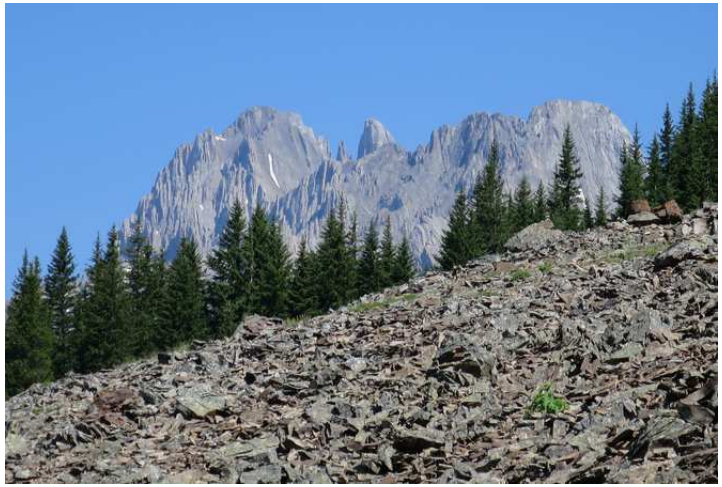
Why?

- Open a large manual page, like ksh(1) or perlfunc(1).
- Where is the description of a keyword like **set**, **if**, **print**?
- Searching with "/" in a pager is almost hopeless.

Imagine just saying: Go to the description of "set"!

Multi-dimensional task:

- link targets
- link sources
- link distance
- jumping mechanism
- output media



Elpoca Mountain 3029m from the Rock Glacier 2100m
Kananaskis Trail, Alberta © 2015 Ingo Schwarze

Dimensions of internal searching and deep linking

link targets:	— sections and subsections: .Sh .Ss — syntax element descriptions: .Fn .Ic .Cm .Ev .Er ... — anchors specified by the document author (sparingly) *The less explicit, the better: avoid requiring new syntax.*
link sources:	— explicitly specified by the document author (sparingly) — implicit: mention of syntax element → description — or just from the reader's mind (= internal searching)
link distance:	— same document (local deep links) — different document (remote deep links)
jumping mechanism:	— search style: type in the target name — hyperlink style: click the link (or type ID or #?)
output media:	terminal, HTML, PDF, ... We don't want mouse-clicking on a server serial console. We don't want to type link numbers in a graphical browser. Yet, we do want consistent functionality across all media!

It's crucial to not jump to an implementation:
The first specific proposal (by Steffen Nurpmeso) would have doubled the number of macros and required substantial rewriting of all manuals... — Yikes!

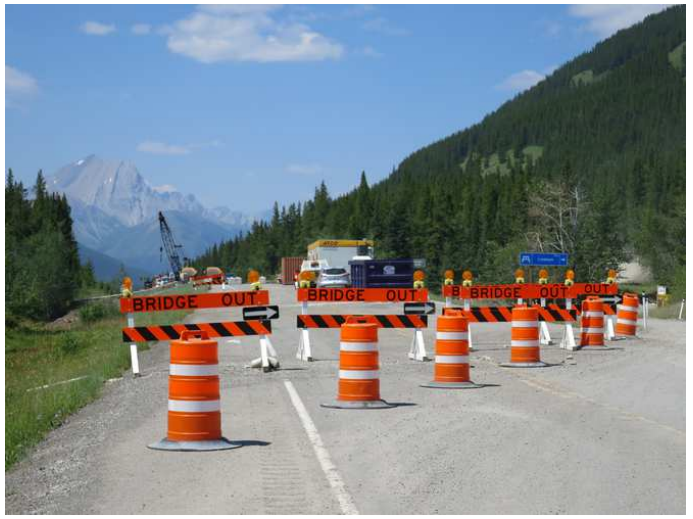
Done: local searching for implicit targets

State before i started:

- local explicit links (.Sx) to sections and subsections (.Sh .Ss)
- only working in HTML output mode
- only linking, no search functionality (except /^SECNAME<enter>)

What i did this summer:

Designed a way to work with
a list of local targets.



Mist Mtn. (3138m) from Lineham Creek Bridge (1680m)
Elbow-Sheep Wildland Provincial Park, Kananaskis,
Alberta, Canada © 2015 Ingo Schwarze

So far used for implicit targets only:

Definitions of syntax elements (functions, keywords, flags, variables, errors)
So far used for local searching only (no linking/nothing remote yet; both non-trivial)

Having a way to handle targets *at all* is a crucial step, and logical as a first step!

It’s already quite useful. → live-demo

Implementation of local jump targets

When using a pager and at least one page is formatted
(otherwise, nothing changes, just use stdout; no change for build systems):

1. **tag_init**(): set up two temporary files (output, tags) and one ohash table
2. during formatting:
   - count output lines (depends on **-Owidth**, info cannot be in a database)
   - **tag_put**(*const char *s, int prio, size_t line*) for .Cm .Er .Ev .Fl .Fn .Ic
3. **tag_write**()
4. **spawn_pager**(): call **less -T** *tag_file out_file*
5. **waitpid**() until the user terminates less
6. **tag_unlink**()

Implementation size:

- 40 new lines in existing code (formatter)
- 240 new lines in a one module *tag.h*, *tag.c*

User interface bloat:

- no new macros, no new syntax whatsoever

Yellow Bellied Marmot, Canyon Campground
Kananaskis, Alberta © 2015 Ingo Schwarze

Plans for internal searching and deep linking

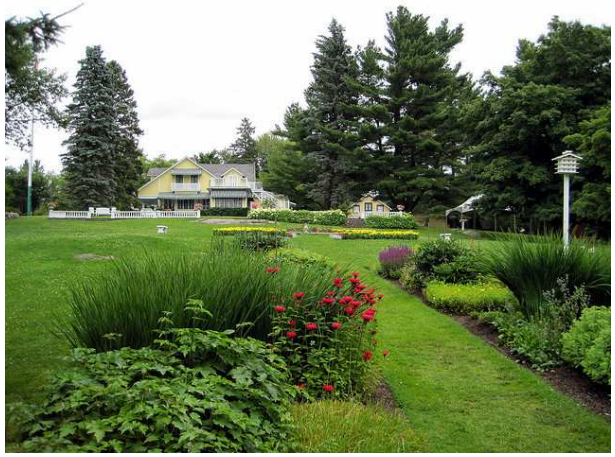
Carefully and slowly extend along the various dimensions

link targets:	add more implicit targets (simple + some smart heuristics) support explicit anchor definitions (straightforward)
link sources:	carefully design explicit internal links to arbitrary targets
jumping mechanism:	carefully design a concept for console hyperlinks — then use that for the existing .Sx — and for implicit links from mentions to descriptions
link distance:	carefully design a concept for remote deep links — then use that for .Xr to specific .Sh/.Ss — then use that for .Xr to other, e.g. implicit targets
output media:	design an analogous concept to :t for HTML (and PDF?)

The Calgary skyline,
looking SE over Kensington
along 7th Street NW,
Alberta, Canada

Auxiliary components in the toolbox

- makewhatis(8) database generation and maintenance
- mandoc(1) -Tlint syntax checker
- mandoc(1) -Thtml polyglot HTML5 generator
- mandoc(1) -Tpdf and -Tps formatters
- mandoc(1) -Tman format converter
- mandoc(1) -Ttree parse tree debugger
- soelim(1) file inclusion resolver
- ...



Moorside Cottage, Mackenzie King Estate, Gatineau Park © 2009 Mac Armstrong @flickr (CC)

What about equations? →

Equations in manual pages

Relevance of the eqn(7) language:
Used less than mdoc(7), man(7), and tbl(7); but: X.org!

Problem was: parsing works well (since 2011), formatting was ugly...
Representing mathematical formula on the terminal is hard.

HTML output:
Rewrite to generate MathML (Kristaps Dzonsons, Oct 10):
straightforward, 1:1 translation of the syntax tree, less than 200 lines of code
quite beautiful in a graphical browser

Terminal output:

- GNU eqn: moves elements up and down to the previous or next line and draws bars out of ASCII characters — results are unintelligible
- mandoc output rewrite (Ingo Schwarze, Oct 12):
linear textual representation: showing fractions like (a + b)/(c + d), matrices like ((a_11 a_12)(b_21 b_22)), and roots like sqrt(x)
- certainly not pretty, but at least you can figure out what the formulas mean
- merely 125 lines of very straightforward code

mandoc eqn now much better than GNU eqn for both terminal and HTML
PostScript/PDF is still the domain of GNU eqn: mandoc just does the same as on the terminal.

What about UTF-8? →

Multibyte character support

Non-english manuals:
rare — hard to maintain — worse than nothing when outdated...

But the tools must not hinder reading them!
That would only aggravate the problem.

Hence, early basic UTF-8 support (Kristaps Dzonsons, May 20-26, 2011):
Like groff, use a preconv(1) preprocessor: UTF-8 → roff escape sequences and -Tutf8 and -Tlocale terminal output modes.

Improvements in 2014

- integrate preconv(1): direct UTF-8 input (Oct 25)
- default to -Tlocale (Dec 2) (formerly -Tascii)
- No more options needed!
- released with 1.13.2 (Dec 13)
- rewrite UTF-8 parser (Dec 19)
- all by Ingo Schwarze

Wasaga Beach, L. Huron © 2013 Joe Mabel @wikimedia (CC)

Why does the design of -Tlint messages matter? →

< >

Why it is critical to get errors and warnings right

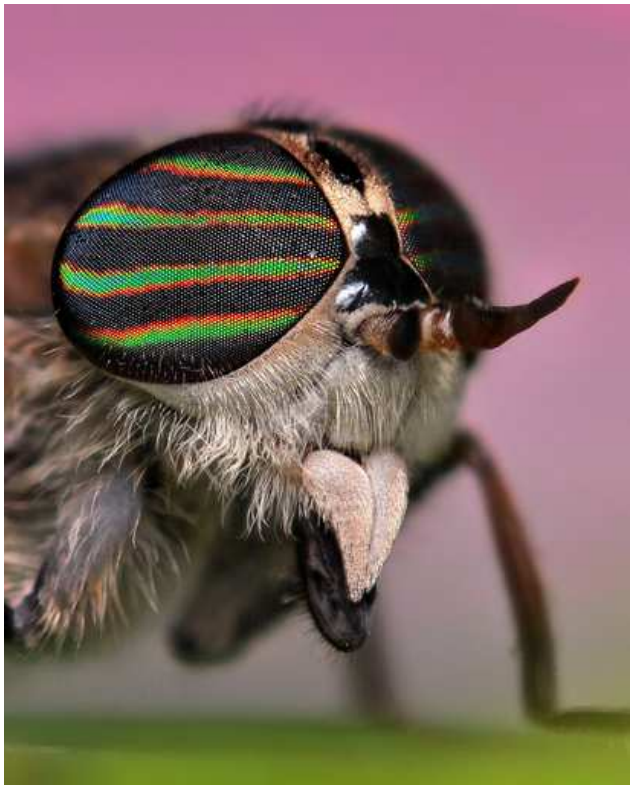
A fatal error gets thrown:
The manual doesn't format at all,
which is very inconvenient.
Can no longer happen since Jan 15, 2015.

An error message is missing or too generic:
Users have a hard time to fix their errors.

A warning message is missing:
Users don't even notice
their dangerous idioms.

A few warnings too many, or too prominent:
Users get annoyed
and switch off all warnings.

More than one or two knobs:
Users don't remember and don't use them.



Horse Fly Portrait

Too few and too many can happen all at once!
It did with mandoc, and it had too many knobs - at first.

There were several major message cleanups in mandoc: July 2009, May 2010, August 2010, October 2010, January 2011, March 2011, July 2014, January 2015, ...

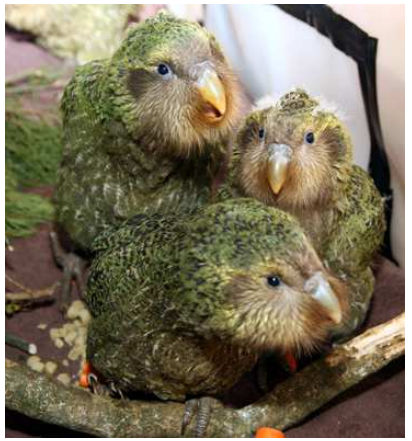
Where can we get more information? →

< >

Help on mdoc(7)

- The mdoc(7) manual is the most important resource.
  - which sections? — MANUAL STRUCTURE section
  - which macros? — MACRO OVERVIEW section
  - usage of a macro? — MACRO REFERENCE section
- Extended mdoc documentation:
  http://mdocml.bsd.lv/mdoc/
- Look at OpenBSD base system manuals for examples.
  This is particularly helpful to find standard wordings
  and customary choices of macro arguments.
  For example, many pages contain this:

```
The options are as follows:
.Bl -tag -width Ds
.It Fl a
```



Kakapo chicks
(Strigops habroptilus)
(c) 2009 NZ DOC @flickr (CC)

- **mandoc -Tlint** catches most syntax errors and provides some hints on style.
- The groff_mdoc(7) manual sometimes helps to resolve ambiguities.
- Kristaps has written a full tutorial: http://manpages.bsd.lv/
- Ask for help on <discuss@mdocml.bsd.lv>
  and provide suggestions to improve the mdoc(7) manual.

Can we use this for portable software? →

Manual pages for portable software

The problem

- Consider portable software packages like sudo(8), OpenSSH, OpenSMTPd, ...
- Which markup language should be chosen for the manual pages?
- Use mdoc(7) and some legacy systems lose that still don't have mdoc(7) after it has been freely available for more than 20 years (hello, Solaris).
- Use man(7) and everybody loses — that would be a very bad idea indeed.

mandoc(1) to the rescue!

- Write the manual pages using mdoc(7).
- Use mandoc -Tman to convert them to man(7) format. Fully operational since November 19, 2012.
- Include both the mdoc(7) and man(7) versions into distribution tarballs.
- Let `./configure` decide:
- On systems supporting mdoc(7), install the mdoc(7) versions.
- Otherwise, install the man(7) versions.

What's the problem with ports manuals? →

Handling manual pages in ports

The problem

The Law of Feature Creep
: If a software offers some feature, sooner or later somebody will use it.

Porting corollary
: For every feature of the roff language (and for every groff extension), no matter how arcane and how obviously irrelevant for manual pages, sooner or later somebody will want to port a third-party software abusing that feature to format its manual pages.

mandoc(1) is not a complete nroff implementation
: and who knows whether it will ever be...

Not a problem in the base system:
: Given a finite set of manuals, implement in mandoc(1) what is needed, or patch away the worst abuse in the handful of manuals affected.

But in ports, “mandoc or nothing” is not a viable strategy:
: That would inevitably leave you with some seriously misformatted manuals.

Manual pages in ports

By now, 95% of ports manuals just work with mandoc.

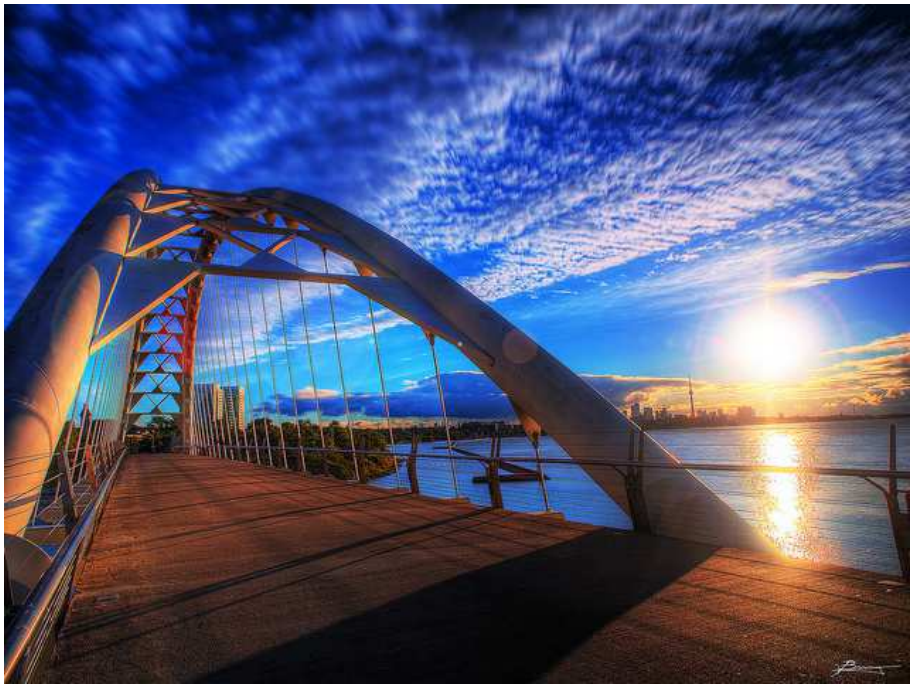
The OpenBSD way

- Mark those that don’t work individually with USE_GROFF (about 200 remain).
- Preformat these manuals at port build time with groff,
  package the preformatted versions of the manuals.

Advantage:
Perfect manuals for every port.

Inconveniences:
Needs support in the
ports infrastructure
(written by Marc Espie@),

and manual maintenance
of the USE_GROFF variable.



Humber Bay Bridge, Toronto © 2010 Paul Bica @flickr (CC)

How does a replacement project work? →

< >

Block nesting

An example of nice mandoc design

Example manual snippet using nested blocks:

```
$ man chgrp
SYNOPSIS
  chgrp [-h] [-R [-H|-L|-P]]
        group file ...
```

Traditional roff/mdoc design:
No block structure!

Low level: roff requests provide

- physical formatting
- registers to store data
- macro expansion

High level: mdoc macros

- call physical formatting
- set registers to keep state
- have no syntax tree

mdoc(7) source code:

```
$ less ‘man -w chgrp‘
[...]
.Sh SYNOPSIS
.Nm chgrp
.Op Fl h
.Oo
.Fl R
.Op Fl H | L | P
.Oc
.Ar group
.Ar
[...]
```

Syntax tree:
(in mandoc, simplified)

```
$ man -Ttree chgrp
Sh (block) SYNOPSIS
  Nm (block) chgrp
    Op (block)
      Fl (elem) h
    Oo (block)
      Fl (elem) R
      Op (block)
        Fl (elem) H
        (text) |
        Fl (elem) L
        (text) |
        Fl (elem) P
    Ar (elem) group
    Ar (elem) file ...
```


What was a typical problem? →

< >

Badly nested blocks and the Xo macro

explicit

```
.Ao ao
.Bo bo
.No ac Ac
.No bc Bc

<ao [bo ac> bc]
```

implicit

```
.Aq aq Bo bo eol
.No bc Bc

<aq [bo eol> bc]
```

in practice: .It Xo

```
$ less ‘man -w find‘
.It Xo
.Ic -exec Ar utility
.Op argument ...
.No ;
.Xc

-exec utility [argument ...] ;
```

- Our first thought: deprecate this abomination, let people use:

  ```
  .It Ic -exec Ar utility Oo argument ... Oc No ;
  ```

- But lots of manuals use ".It Xo" - historical argument limit
- No way to change all manuals, everywhere, or even the habits of authors...
- Reluctant implementation:
  Ao block contains the *whole* Bo block
  Bo block contains an Ao body-end element

More real-world nuisances... →

If ifs & ands were docs & mans...

No way around some low-level roff requests.

- Original design:
  - Forget about the complete bottom layer.
  - Only implement high-level mdoc(7) & man(7) macros.
- Didn’t work out.
  - Real-world manuals use some low-level requests.
  - Realizing that was a slow, painful process.
  - We reluctantly edged in some low-level roff support.
  - Complexity was kept to the bare minimum.
- Both steps have proven to be just right:
  - Starting from the high level gave a clean design.
  - Roff reluctance prevented getting off track.
  - Surprise: Rather little impact in the end.

Very young zebra

What was done about low-level roff? →

< >

Desperation lead to success:

How an afterthought improved the design.

- Mandoc main program:
  - main loop to read input files
  - call the roff preprocessor on each line (not in original version)
  - push line after line into the parser backends
  - parsers look for high-level macros, e.g. mdoc(7)
  - call the formatting frontend on the resulting syntax tree
- Paradigmatic switch:
  - roff: expand high-level macros into low-level requests
    then pass the low-level requests into the formatters
    all structural info lost long before the main parser
  - mandoc: first handle low-level requests (preprocessor)
    then pass the high-level code to the parsers
    structural info kept even when intermixed with low-level roff

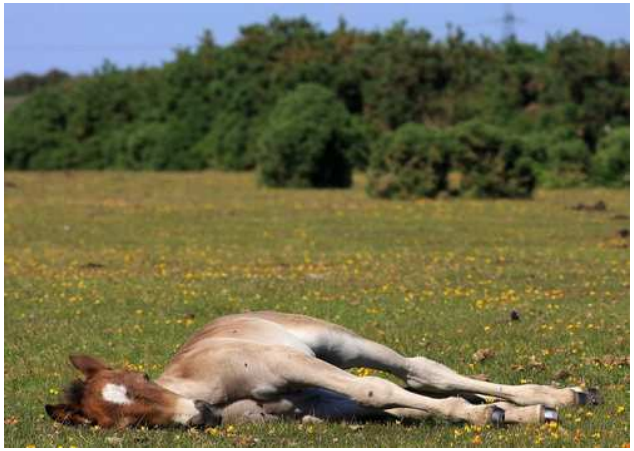
Done in 2011:

Reached the stage: It just works.

- To get there:
  - Re-implemented a small family of languages.
  - Turned the whole paradigm upside down.
  - Remained compatible where it matters.
  - Flouted compatibility that would just hinder.

- Other systems besides OpenBSD can now do the same, if they want.
  - ... without much risk.
  - ... without having to fear major surprises.
  - ... getting support from us in case of need.

  Just mail us!



New Forest Foal

Move fast!

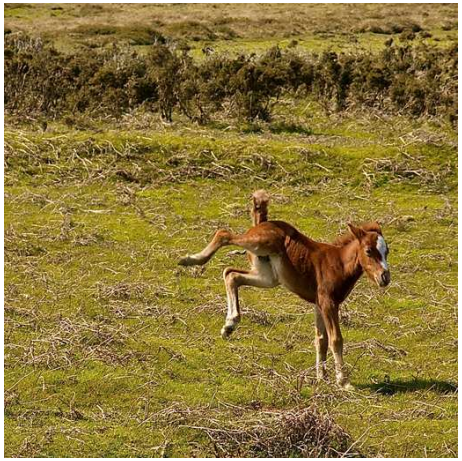
How a replacement project can succeed.

Quickly put your work to real-world use.
Do not let it rot in a corner of the OS.

Keep moving fast, do not fear change.
But don’t trap your users
with incompatible changes.

And keep the balance:

- No need to do it when it makes no difference.
- You won’t find users when you break behaviour.
- You won’t find bugs when you don’t have users.



Foals Just Wanna Have Fun

Bad patches triggering good ones:

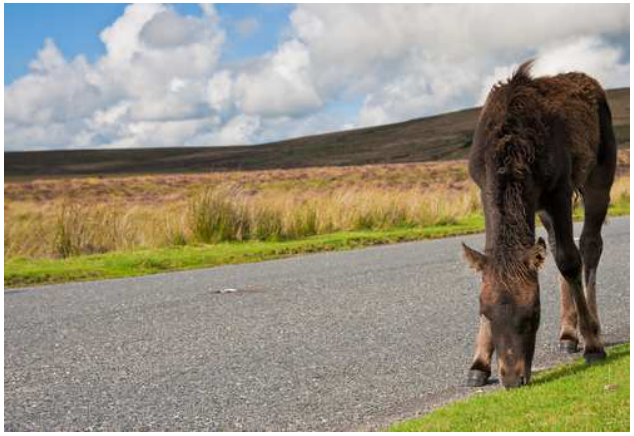
Preliminary code put in and ripped out again.

That may seem inefficient, but actually it’s a perfectly sane approach:

- The first implementation explores the feature.
- The second implementation gets it right.
- Just don’t let the first one sprawl until it can’t be ripped out any more.

This approach got used in at least five cases:

2010 Mar 01 - May 14: end of sentence detection
2010 Apr 25 - May 19: roff conditionals
2010 Mar 01 - Sep 20: pod2man preamble
2010 Jun 16 - Jun 27: roff registers
2010 Oct 15 - Jan 04: tbl integration



On the Road

Newly designed:

Clean implementation of dirty languages.

Started coding with the nice high-level stuff.
That gave us a very clean overall design.

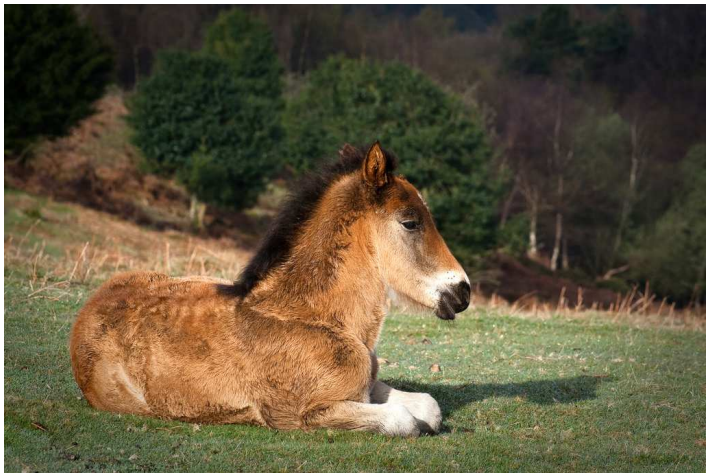
Edged in low-level ugliness later, where required.
Even while the tool was already in production!

Only possible since we kept it small and simple.
Otherwise, such stunts would break the design.

Shun complexity!

Three reasons why simplicity is key:

1. correctness
2. security
3. flexibility



A Youngster on the Quantocks

American Fuzzy Lop inspecting mandoc

Fuzzer: Try to crash or hang a program by feeding it varying input.

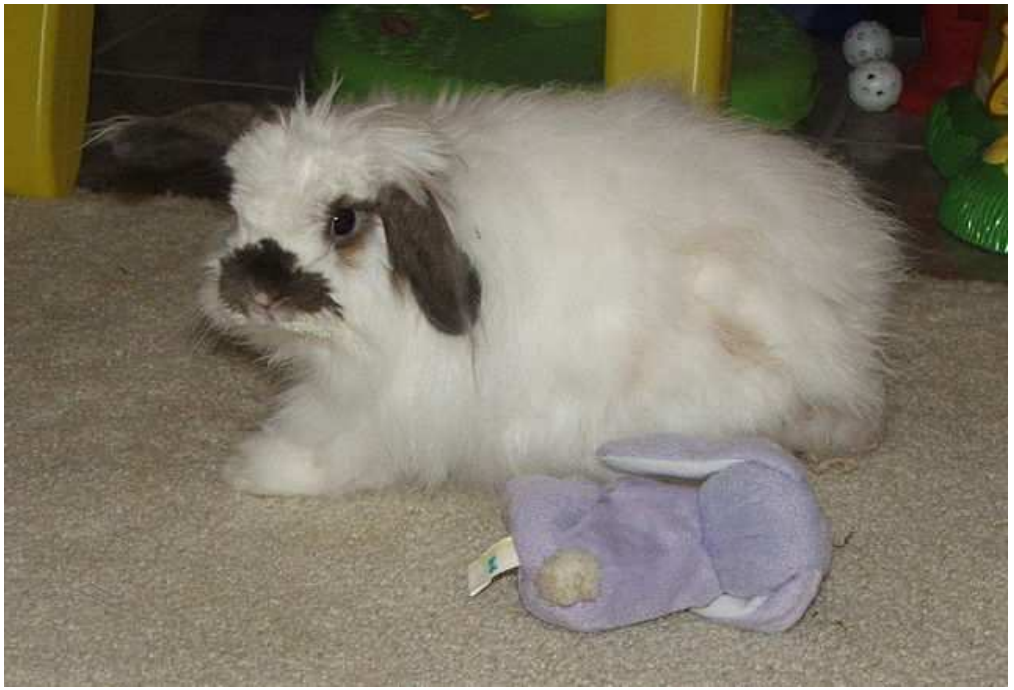
http://lcamtuf.coredump.cx/afl/

"Compile-time instrumentation and genetic algorithms to automatically discover clean, interesting test cases that trigger new internal states."

Goal: Full functional coverage.

For mandoc, afl can be seeded with the extensive test suite.

For mandoc, full functional coverage requires running afl continuously for several days on modern PC hardware.



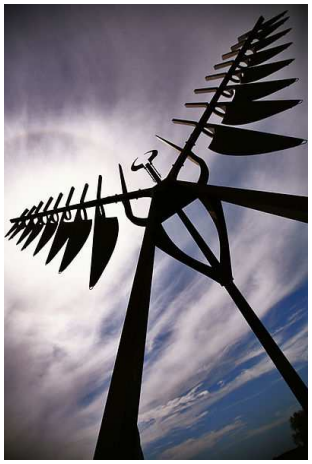
American Fuzzy Lop © 2008 Lithonius@wikimedia (PD)

Run by Jonathan Gray (jsg@) repeatedly since November 21, 2014.

45 issues grand total → Quantitative (but not statistical) analysis.

Results from the afl audit

- Causes of bugs:
  - 1/3 violations of invariants
  - 1/3 excessive complexity
    (in languages; in implementation)
  - 1/9 missing input validation
  - 1/9 buffer overflows
  - 1/9 use after free



Spiritcatcher by Ron Baird (1986) in Barrie

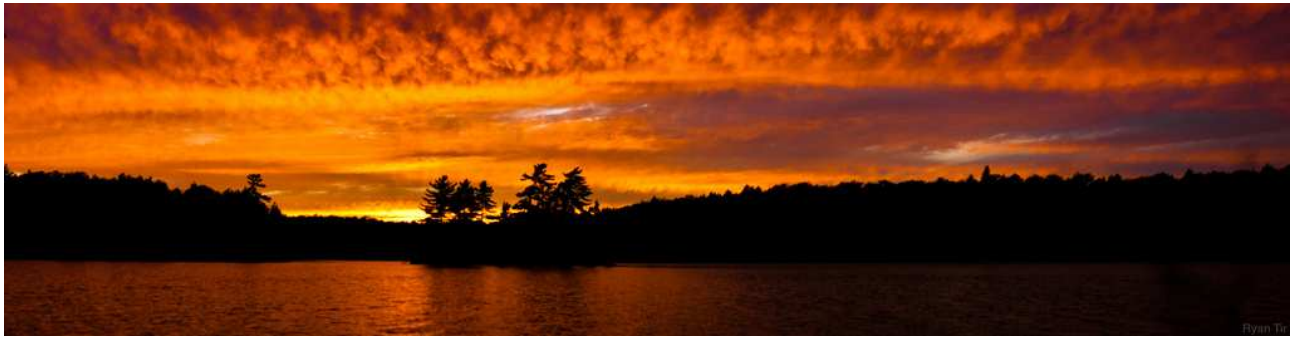
- Lessons:
  - Obviously most important point: Keep code small and simple.
  - Paying attention to the simplest sources of bugs tends to help against the errors having the gravest consequences, even though may no be the most numerous types of errors (in particular buffer and integer overflows, use after free, missing input validation)
  - Explicitly specifying invariants and auditing for their correct implementation is likely to catch a quantitatively important class of bugs (but a lot of work).

Which vulnerabilities were found? →

< >

Security issues found in the web manual viewer

- Invalid configuration → segfault:
  Fix: When there is no MAN_DIR or manpath.conf, log and exit.
- Unvalidated PATH_INFO → access to unrelated files, information disclosure:
  Fix: Reject absolute paths and ascenscion to the parent directory.
- Unvalidated QUERY_STRING manpath → info disclosure in error message:
  Fix: Validate the manpath up front using a whitelist.
- Invalid characters in QUERY_STRING → XSS:
  Fix: Restrict the character set of strings passed into html_alloc().
- Roff escape sequences and quotes in manuals → XSS:
  Fix: HTML-encode quotes and rendered escape sequences.
- Choosing the right encoding is a tricky business...
  Some output needs HTML encoding, some URI encoding, some both.
- Crafted expensive regular expressions → REDoS attacks:
  No full fix possible. Mitigate by limiting CGI process execution time.



Maggie Lake
Algonquin Park
Ontario, Canada

What about performance? →

< >

Search and database performance summary

- With the old, plain text apropos(1), a simple search took about 10 milliseconds on my notebook.
- With the new, SQLite apropos(1), it is unavoidably slower due to the SQL overhead and because the names are now separated from the descriptions. It now takes about 40 milliseconds.
- However, the difference is of no practical relevance even on moderately old hardware (like this notebook).

- Base system database size grows from 250 kB to 900 kB (quick mode) or about 3800 kB (fully featured more). That is not a practical problem for any of our architectures.
- During system builds, database build times are reduced by roughly a factor 3 with respect to the old Perl makewhatis(8).

Spotted Shag / Parekareka (Phalacrocorax punctatus)
(c) 2010 Avenue@wikimedia (CC)

Conclusions

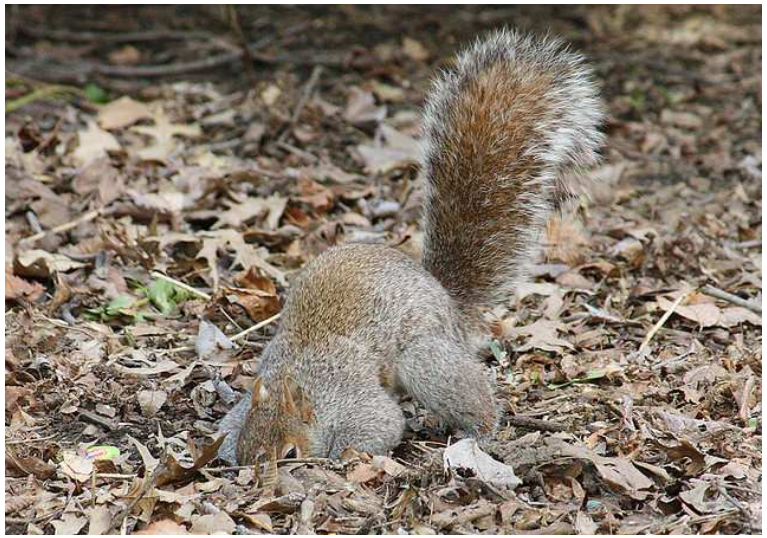
What made the replacement project succeed

- Start with a clean design,
  sparingly add support for less clean, historically grown features where needed.
- Quickly go into production,
  profit from user feedback about bugs and missing functionality to improve.
- Improve step by step aiming for complete functionality but avoid bloat:
  man(1) mdoc(7) man(7) tbl(7) eqn(7) apropos(1) man.cgi(8)

Priorities

1. correctness and security
2. compatibility
3. simplicity
4. performance

... in that order.



Squirrel digging

Let’s summarize the status... →

Status summary

Fully integrated (mandoc, man, apropos, makewhatis)
OpenBSD, Alpine Linux, Void Linux

Almost fully integrated (including apropos but without man)
FreeBSD-current

Default formatter (but less powerful implementations of man and apropos)
NetBSD, illumos

In the base system (but not used by default)
FreeBSD 10, DragonFly BSD, Minix 3

Official packages exist
FreeBSD 9, Arch Linux, pkgsrc

Unofficial packages of useful versions exist
Slackware, Crux Linux, Mac OS X, MinGW

Outdated packages only, probably easy to update
Debian, SUSE, Redhat and derivatives; IBM AIX, Cygwin

Ottawa (Ontario) seen from Gatineau (Quebec) © 2009 G. Baranski @wikimedia (CC)

What shall we do in the future? →

< >

Possible future directions

Work in progress

- Improve pod2mdoc(1) and use it for LibreSSL (first mentioned: BSDCan 2011)
- Unify parsers aiming for better roff(7) support (first mentioned: BSDCan 2014)
- Automatically detect unsupported source code (-Wunsupp) (NEW)
- Delete most ln(1) links to manual pages (NEW)
- texi2mdoc(1) to convert texinfo(1) documentation to mdoc(7) (NEW)
- Help with man(7) to mdoc(7) conversions (first mentioned: BSDCan 2011); docbook2mdoc(1) (first mentioned: BSDCan 2014)
- Continue work on internal search and deep linking (NEW).

Not yet started

- Support automatic semantic enrichment of Perl manuals with pod2mdoc(1). (first mentioned: EuroBSDCon 2014)

Canadian Museum of History, Gatineau

Who contributed? →

< >

Thanks!

Cynthia Livingston (USENIX)
for designing and implementing the mdoc(7) language and translating all the BSD manual pages to it

Kristaps Dzonsons (bsd.lv)
for writing mandoc

Jörg Sonnenberger (NetBSD)
for important code contributions and for hosting an excellent mandoc hackathon at BEC.de

Marc Espie (OpenBSD)
for OpenBSD ports integration, lots of important feedback, and for writing the ohash library

Jonathan Gray (OpenBSD)
for extensive testing with afl resulting in over 40 bug reports

Sébastien Marie
for a complete man.cgi(8) security audit

Todd C. Miller (OpenBSD & sudo)
for feedback and multiple patches to the mdoc-to-man converter

Jason McIntyre (OpenBSD)
for excellently and tirelessly maintaining our manuals, for helping with countless bug reports, and for discussing countless questions regarding mdoc(7)

Theo de Raadt (OpenBSD)
for inviting Kristaps and getting mandoc imported. (Otherwise, i might have missed it.)
for ongoing encouragement, in particular to make OpenBSD developers and users our guinea pigs. (None complained, they seemed to enjoy it.)
for many bug reports and some patches

Anthony J. Bentley (OpenBSD)
for porting related software to OpenBSD, many bug reports, and some source code patches

Jeremy Evans (OpenBSD)
for crucial help with SQLite database optimization

Matthieu Herrb, Todd Fries, Miod Vallat (OpenBSD)
for help with Xenocara integration

Christian Weisgerber (OpenBSD)
for continuous work on mandoc issues in OpenBSD ports

Stuart Henderson (OpenBSD)
for help with large numbers of porting issues

Pascal Stumpf (OpenBSD)
for repeated help with difficult groff porting issues

Peter Hessler (OpenBSD)
for help with the regression infrastructure

Thomas Klausner (NetBSD)
for NetBSD and pkgsrc porting work
and lots of feedback and release testing

Baptiste Daroussin and Ulrich Spörlein (FreeBSD)
for FreeBSD porting and many bug reports

Werner Lemberg (GNU troff)
for tireless help with groff-mandoc synchronization

Robin / Toutouwai (Petroica australis)

Natanael Copa, Paul Onyschuk (Alpine), Svyatoslav Mishyn (Crux), Jesse Adams (Arch), Dániel Lévai (Slackware)
for porting mandoc(1) to Linux and providing feedback

Garrett D’Amore, Yuri Pankov (IllumOS), Matthias Scheler (Solaris), Ben Gras (Minix 3)
for porting mandoc(1) to Non-BSD systems and providing feedback

For source code patches:

Franco Fichtner (DragonFly), Christos Zoulas, Tsugutomo Enami (NetBSD), Daniel Dickman, Doug Hogan, Ted Unangst (OpenBSD), Martin Natano

For bug reports and useful suggestions and discussions:

Alexander Bluhm, Antoine Jacoutot, Bob Beck, Brad Smith, Bret Lambert, Brian Callahan, Bryan Steele, David Coppa, David Gwynne, Dmitrij D. Czarkoff, Edd Barrett, Florian Obser, Giovanni Becchis, Gleydson Soares, Henning Brauer, Ian Darwin, Igor Sobrado, Janne Johansson, Jasper Lievisse Adriaanse, Jérémie Courrèges-Anglas, Jonathan Gray, Juan Francisco Cantero Hurtado, Kenji Aoyama, Kenneth R. Westerback, Kurt Miller, Landry Breuil, Martin Pieuchot, Matthew Dempsky, Matthias Kilian, Nicholas Marriott, Nick Holland, Nigel Taylor, Okan Demirmen, Paul Irofti, Paul de Weerd, Philip Guenther, Remi Pointel, Reyk Flöter, Stefan Sperling, Thordur Bjoernsson, Vadim Zhukov (OpenBSD)

Abhinav Upadhyay, David Holland, Nicolas Joly, Havard Eidnes, Jonathan Perkin (NetBSD), Kurt Jaeger, Pedro Giffuni (FreeBSD), Antonio Huete Jimenez, Sascha Wildner (DragonFly), Michael Dexter (bsd.lv), Carsten Kunze (Heirloom Doctools), Alexis Hildebrandt (Homebrew)

Alessandro de Laurenzis, Andreas Vögele, Chris Bennett, Chris Hettrick, Christian Neukirchen, David Hill, David Levine, Fabian Raetz, Frantisek Holop, Guy Harris, Igor Zinovik, James Jerkins, Jan Stary, Jörn Clausen, Justin Haynes, Marcus Merighi, Markus Waldeck, Maxim Belooussov, Michel Jansens, Michael Reed, Mike Small, Mikolaj Kucharski, Patrick Keshishian, Ryan Flannery, Steffen Nurpmeso, Theo Bühler, Tim van der Molen, Tristan Le Guern, Will Backman

Where did the images come from? →

Thank you for sharing your pictures!

http://www.flickr.com/photos/tomkoadam/4778126822 Adam Tomkó: Csikó – Foal (by-nc-nd)
http://www.flickr.com/photos/whereisbrent/461055143 Brent Barrett: Kea juvenile (by-nc-nd)
http://2014.eurobsdcon.org/wp-content/uploads/2014/06/BSDSofiaForWeb.png Alica Dimitrova: Sofia BSD Mascot
Cynthia Livingston's off-track thoroughbred "Bedifferent" (private communication, used with permission)
https://www.flickr.com/photos/14020964@N02/7373733864/ Lezumbalaberenjena: Black Lake, Gatineau Park, Quebec (by-nc-nd)
http://commons.wikimedia.org/wiki/File:StGeorgeRotundaSofia.JPG Preslav: Rotonda Sveti Georgi, Sophia (pd)
http://commons.wikimedia.org/wiki/File:Maliovitsa_54072.jpg Stelian Kasabov: Maljovitsa 2729m, Rila (pd)
http://www.flickr.com/photos/everexplore/8572686541 Kiril Rusev: Kutelo 2908m and Vihren 2914m, Pirin (by-nc-nd)
http://www.mckusick.com/beastie/shirts/bsdunix.html USENIX: 4.2BSD Beastie
http://www.flickr.com/photos/29954808@N00/1300190844 57Andrew: Rock Wren (by)
http://commons.wikimedia.org/wiki/File:Tokoeka.jpg Glen Fergus: Southern Kiwi (by-sa)
http://commons.wikimedia.org/wiki/File:Silvereye.jpg J. J. Harrison: Silvereye (by-sa)
http://www.flickr.com/photos/dougtone/5806473660/ Doug Kerr: Bonnechere Museum, Eganville, Renfrew County, Ontario (by-sa)
http://commons.wikimedia.org/wiki/File:Red_fox_kit_2_%28Vulpes_vulpes%29.jpg Charlesjsharp: Red fox kit (by-sa)
http://www.flickr.com/photos/reiver/3821502286/ Mac Armstrong: Moorside cottage, Mackenzie King Estate, Gatineau Park, Quebec (by-sa)
http://commons.wikimedia.org/wiki/File:Aerial_-_Wasaga_Beach,_Ontario_from_SW_01_-_white_balanced_%289656223451%29.jpg Joe Mabel: Wasaga Beach (by-sa)
http://www.flickr.com/photos/jimmy-jay/4672901414 Jeff Burcher: Horse Fly Portrait (by-nc-nd)
http://www.flickr.com/photos/docnz/8528275645 NZ DOC: Kakapo (by-nc-sa)
http://www.flickr.com/photos/dexxus/5003010775/ Paul Bica: Humber Bridge, Toronto, Ontario (by)
http://www.flickr.com/photos/tambako/3578468294 Tambako: Very young zebra (by-nd)
http://www.flickr.com/photos/kristavandervoorden/4737488285 Krista van der Voorden: New Forest Foal (CC)
http://www.flickr.com/photos/gazzat/3495392530 Gary Tanner: Foals Just Wanna Have Fun (by-na-sa)
http://www.flickr.com/photos/tiny_packages/5045038219 tiny_packages: On the road (by-nc-nd)
http://www.flickr.com/photos/66176388@N00/3436935367 Mark Robinson: A Youngster on the Quantocks (by-nc)
http://commons.wikimedia.org/wiki/File:Rabbit_american_fuzzy_lop_buck_white.jpg Lithonius: American Fuzzy Lop rabbit (pd)
http://commons.wikimedia.org/wiki/File:Spiritcatcher_barrie_wide.jpg Tudor Costache: Spiritcatcher by Ron Baird, Barrie, Ontario (by)
http://www.flickr.com/photos/ryan_tir/6232749531/ Ryan Tir: Maggie Lake, Algonquin Park, Ontario (by)
http://commons.wikimedia.org/wiki/File:Spotted_Shag_%28Phalacrocorax_punctatus%29_in_flight_2.jpg Avenue: Spotted Shag (by-sa)
http://www.flickr.com/photos/eastsidephil/4452171897 Phil Davis: Squirrel digging (by-nc-sa)
http://commons.wikimedia.org/wiki/File:Canada_Ottawa_Panorama.jpg G. Baranski: Ottawa Panorama (by-sa)
http://commons.wikimedia.org/wiki/File:Gatineau_-_QC_-_Museum_of_Civilisation.jpg Wladyslaw: Canadian Museum of History, Gatineau (by-sa)
http://commons.wikimedia.org/wiki/File:070308_Stewart_Island_robin_on_Ulva.jpg Mark Jobling: NZ Robin (pd)

North Tea Lake, Algonquin Provincial Park © 2008 Raul Heinrich @wikimedia (CC)

What would you like to ask? →