New trends in mandoc:

Enhancing the modern toolbox for the classic documentation formats

Ingo Schwarze <schwarze@openbsd.org>

BSDCan 2014, May 17, Ottawa

Kea juvenile (Nestor notabilis) exploring new horizons (c) 2007 Brent Barrett @flickr (CC)

Enhancing the modern toolbox for the classic documentation formats:

Tools for handling manual pages

Manual viewer: man(1)

- Find manuals in the file system.
- Call a formatter and a pager on them.
- Not the topic of this talk...

Manual formatter: **mandoc(1)**

- Read source code written in markup languages.
- Write formatted text for the user.
- Classical tool for this task: nroff(1).

Manual page search tool: **apropos(1)** = `man -k`

- Do database lookups.

Manual page database tool: **makewhatis(8)**

- Build databases.

Enhancing the modern toolbox for the classic documentation formats:

Classic roff markup syntax

Classic = literally half a century of history!

The concept of text and macro lines:

RUNOFF (later became “roff”)

by Jerome H. Saltzer, MIT, 1964

in MAD for CTSS on the IBM 7094

inspired by Memo, Modify, and Ditto
by Lowry/Corbato/Steinberg 1963.

Requests still in use today:

```
.ad  .ce  .fi  .in  .ll  .nf
.br  .sp
```

IBM 7094 (c) 1965 Columbia Univ. Archives (Courtesy)

Macro lines contain a period (‘.’), a macro name, and optional arguments.

All other lines are text lines.

Sample roff(7) source code

```
.TITLE "Advantages of the roff macro syntax"
.S +2
.BL
.LI
Can easily be hand-edited with minimal typing overhead.
.LI
Looks unobtrusive, does not muddle the actual text.
.LI
Harmonizes very well with diff(1).
.LI
Allows high quality output in multiple output formats,
.br
in particular for terminal output and typesetting.
.LI
Works with simple, fast, portable, readily available tools.
.LI
Does not need any heavyweight or cumbersome toolchains,
.br
in particular, does not require XML.
.LE
.sp 2v
.ce 2
\&... or to say it in one word:
.sp 2v
.S +4
.GPE_EM "KISS!"
.S -6
.GPE_NEXT 50 "Application to software manuals?"
```


OK, try again: Why do we still use that syntax 50 years later? →

Advantages of the roff macro syntax

- Can easily be hand-edited with minimal typing overhead.
- Looks unobtrusive, does not muddle the actual text.
- Harmonizes very well with diff(1).
- Allows high quality output in multiple output formats, in particular for terminal output and typesetting.
- Works with simple, fast, portable, readily available tools.
- Does not need any heavyweight or cumbersome toolchains, in particular, does not require XML.

... or say it in one word:

KISS!

Enhancing the modern toolbox for the classic documentation formats:

Origin of the basic manual structure

AT&T Version 1 UNIX manual

formatted with roff

by Ken Thompson
and Dennis M. Ritchie,
Bell Labs, 1971

written in assembler
for UNIX on the DEC PDP-11

inspired by the CTSS manuals

Section headers in use since v1:
NAME, SYNOPSIS, DESCRIPTION, FILES, SEE ALSO, DIAGNOSTICS, BUGS

Ritchie, Thompson, PDP-11 (c) 1971 Bell Labs
Reprinted with permission of Alcatel-Lucent USA Inc.

Precursors to man(7) and mdoc(7) macros, used in v4–v6 (1973-1975):

`.th` (→ .TH/.Dt) `.sh` (→ .SH/.Sh) `.bd` (→ .B/.Sy) `.it` (→ .I/.Em)

The man(7) language first appeared in Version 7 AT&T UNIX (1979).

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) — see this talk.
- Facilitates semantic searching — see this talk.

How does mdoc(7) code look like? →

< >

Sample mdoc(7) source code

```
.Dd $Mdocdate: March 31 2014 $
.Dt MDOC 7
.Os
.Sh NAME
.Nm mdoc
.Nd semantic markup language for formatting manual pages
.Sh DESCRIPTION
The
.Nm mdoc
language supports authoring of manual pages for the
.Xr man 1
utility by allowing semantic annotations of words, phrases,
page sections and complete manual pages.
Such annotations are used by formatting tools to achieve a uniform
presentation across all manuals written in
.Nm ,
and to support hyperlinking if supported by the output medium.
.Pp
This reference document describes the structure of manual pages
and the syntax and usage of the
.Nm
language.
The reference implementation of a parsing and formatting tool is
.Xr mandoc 1 ;
the
.Sx COMPATIBILITY
section describes compatibility with other implementations.
```

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) (c) 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:
  - ASCII, UTF-8, HTML, XHTML, PostScript, PDF output
  - mdoc(7) to man(7) conversion
  - makewhatis(8)/apropos(1)
- 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.

Table of contents

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

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)
(c) 2008 Glen Fergus @wikimedia (CC)

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)
(c) 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 "=") since October 19, 2013:

```
 $ 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
[...]
```

Complex search queries

By default, multiple search terms are joined with OR,
but the **−s** and **−S** options attach to the rest of the search expression with AND:

```
 $ apropos −s 1 tbl Nm=eqn
deroff(1) – remove nroff/troff, eqn, pic and tbl constructs
eqn(1) – format equations for troff or MathML
eqn2graph(1) – convert an EQN equation into a cropped image
neqn(1) – format equations for ascii output
tbl(1) – format tables for troff
```

Explicit logical AND and OR are supported:

```
 $ apropos Nd=gigabit −a Cd=sbus
gem(4) – GEM 10/100/Gigabit Ethernet device
ti(4) – Alteon Networks Tigon I and II Gigabit Ethernet device
```

Precedence can be changed with parantheses:

```
 $ apropos −s 1 terminal −a \( At~[1−6] −o Bx~^[12] \)
clear, tput(1) – terminal capability interface
lock(1) – reserve a terminal
reset, tset(1) – terminal initialization
script(1) – make typescript of terminal session
stty(1) – set the options for a terminal device interface
tty(1) – return user’s terminal name
```

Complex search queries are working since January 4, 2014.

Morepork / Ruru (c) 2005
(Ninox novaeseelandiae)
Aviceda@wikimedia (CC)

Flexible output format

- The names, section numbers, and architectures of the search results are always shown because these are needed to access the results with man(1).
- By default, apropos(1) also shows the one-line descriptions.

With the **−O** option, any other macro key can be shown instead:

```
 $ apropos -O Cd wireless
acx(4) – acx* at pci? # acx* at cardbus?
an(4) – an* at isapnp? # an* at pcmcia? # an* at pci?
ath(4) – ath* at pci? dev ? function ? # ath* at cardbus? dev ? function ? # gpio* at ath?
athn(4) – athn* at cardbus? # athn* at uhub? port ? # athn* at pci?
atu(4) – atu* at uhub? port ?
atw(4) – atw* at pci? # atw* at cardbus?
```

The **−O** option is available since December 31, 2013.

Bellbird / Korimako (Anthornis melanura) (c) 2012 Sid Mosdell @flickr (CC)

Database implementation

- The old `whatis.db` was a plain text file.
- Now we need a structured database.
- But a client-server model would be overkill and merely a hindrance.
- So SQLite was the logical choice.

- Four tables. They contain, respectively, one record ...

  mpages ... per physical page, containing the description
  mlinks ... per file system entry, containing section, architecture, filename
  names ... per page name; for all manual page names, not just file names
  keys ... per key=value pair

- The mlinks table has full support for:
  - hard links since December 27, 2013
  - symbolic links since April 18, 2014
  - redirections using the roff .so request since March 19, 2014 (used by X.org)

Search algorithm

Two or three queries:

1. `SELECT FROM mpages`
   to find the pages to be displayed
   - Conveniently, searching for descriptions is fastest —
     in the first step, access one single table only, finding the `pageid`.
   - Searching for names is the second in speed —
     it requires only a simple JOIN to a small table.
2. `SELECT FROM names`
   to find the page names to be displayed
   This is very fast because it is just a simple SELECT
   in a small table using the indexed `pageid`.
3. `SELECT FROM keys`
   to find the values to be displayed
   - Only needed when **−O** is given.
     Otherwise, we already have the description from search step 1.
   - Very fast, too, just another simple SELECT indexed by `pageid`.

Optimization

As usual, optimization is not a well-defined task in the mathematical sense.

We want high speed and small size:

- a small database
- a short database build time
- low apropos memory consumption
- short search times

Of course, these optimization goals conflict.

The gprof(1) profiler was used a lot.

Falcon / Karearea (Falco novaeseelandiae)
(c) 2008 Py1jtp@wikimedia (CC)

Search speed optimization

- Moving the descriptions from the keys table directly into the mpages table gained roughly a **factor 4** in speed for searches by description — at no cost, the database shrank, too, due to reduced pageid overhead.
- The dedicated names table gained roughly a **factor 4** in speed for searches by name — at almost no cost.
- Adding an index to the mlinks table sped up the second step in the algorithm, name retrieval, by about a factor of 20, which resulted in an overall **30% economy** for simple searches, and more for searches returning many results. The cost was a 10% growth of the database.
- Adding an index to the keys table sped up the third step in the algorithm, **−O** value retrieval, which is dominant in **−O** searches, resulting roughly in a **factor 4** speedup of such searches. The cost was another 10% grow of the database.
- By providing an SQLITE_CONFIG_PAGECACHE with mmap(3) MAP_ANON, execution time decreased by 20−25% for simple (Nd and/or Nm) queries, 10−20% for non-NAME queries, and even apropos(1) resident memory size decreased by 20% for simple and by 60% for non-NAME queries. Cache size is a compromize to provide nearly optimal speed gain for all queries while limiting additional memory consumption to about 15%.

Database build time optimization

This is relevant because `mandoc.db` is built during regular base system and Xenocara snapshots builds on all architectures, and we don't want to slow down developers during development and testing.

- Quick mode: Abort parsing after the NAME section: factor 2 in speed and factor 4 in size. In the following, all speedups refer to quick mode.
- Do not sync to disk after each individual manual page, only sync to disk one single time when all data is ready: 87%.
- In quick mode, do not clear user-defined macros clashing with mdoc(7) or man(7) standard macros when parsing .Dd or .TH: 25%.
- In quick mode, do not validate and normalize the date format: 18%.
- Do not copy predefined strings into the dynamic string table: 10%.
- Cache uname(3) result: 3%.
- Do not index the keys in the keys table: 12% (and 42% size reduction).
- No primary keys in the mlinks and keys table: 15% (and 3% size).
- Properly handling .so redirections reduced Xenocara database build time by 40% and database size by nearly 50%.

Database size optimizations

All size reductions refer to quick mode.

- Do not store the descriptions twice, once for searching and once for display, at the expense of somewhat more complicated, but not slower search code: 9%.
- Remove the redundant "file" column from the mlinks table: 9%.
- Sort macro keys by frequency: 11%.
- Always store the arch in lower-case only: 1.5%.

Rifleman / Titipounamo (Acanthisitta chloris), NZ’s smallest bird (c) 2008 digitaltrails@flickr (CC)

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)

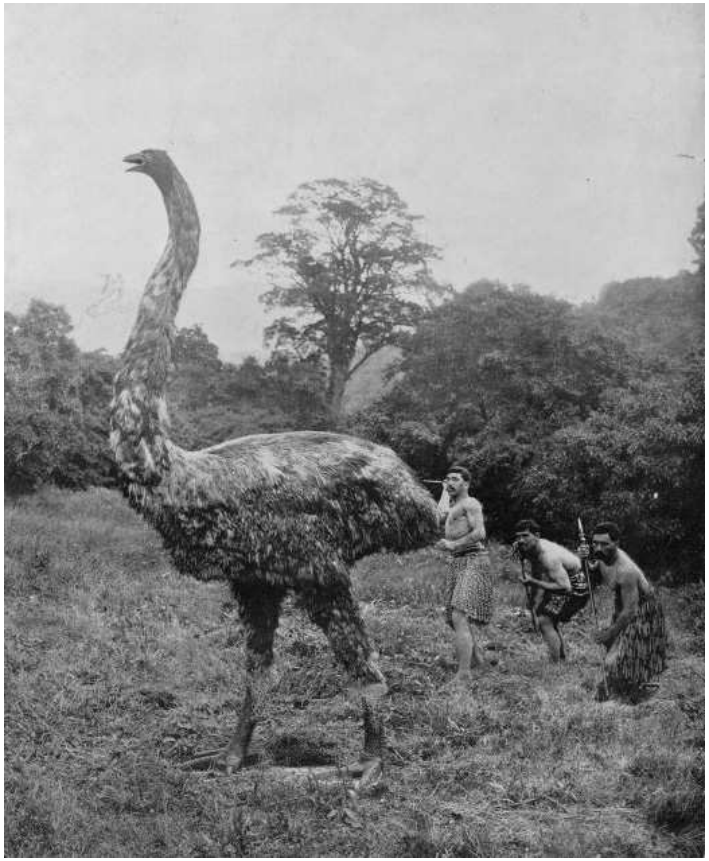
Are there any useful by-products? →

< >

Consistency checking

- Mismatch of the section number given in the manual page with the directory the page is stored in.
- Mismatch of the architecture name given in the manual page with the directory the page is stored in.
- File name does not appear as a name in the NAME section.

- Besides, direct inspection of the database has been used to catch markup errors.
- More can be done later, all this is just a start.



Moa hunt in the Dunedin Public Gardens
(c) 1906 Augustus Hamilton (PD)

Already checked by the old makewhatis(8):

- Missing NAME section, missing name(s) and/or missing description.
- A name in the name section does not appear as an MLINK in the file system.

Status in OpenBSD

- Kristaps@ developed mandoc(1) since November 22, 2008.
- Source code in the base repo since April 6, 2009.
- Schwarze@ maintaining it since June 14, 2009.
- Actively maintained regression suite since October 27, 2009.

- mandoc(1) installed with OpenBSD-current since April 2, 2010.
- Base system manuals built with mandoc(1) since April 3, 2010.
- USE_GROFF framework for ports by espie@ since April 5, 2010.
- Releases fully rely on mandoc(1) since OpenBSD 4.8, November 1, 2010.
- Groff disconnected from base build since October 18, 2010:
  mandoc(1) is the only documentation formatter in base for three years.
- Groff removed from the source tree since March 12, 2011.
- Groff 1.21/1.22 available from the ports tree since March 19, 2011.
- No stable releases contain groff since OpenBSD 4.9, May 1, 2011.

- SQLite-based code in the source tree since December 30, 2013.
- makewhatis(8)/apropos(1) using mandoc since April 18, 2014.
- All will be released with OpenBSD 5.6 on November 1, 2014.

Status in NetBSD

- A pkgsrc mdocml port by Jörg Sonnenberger exists since March 1, 2009 → continued support for *many* platforms.
- The first code patch was sent upstream by Jörg Sonnenberger on June 11, 2009.
- Source code in the base repo and installed by default in NetBSD-current since October 21, 2009.

Big changes in NetBSD on February 7, 2012

- Install source manuals, no longer install preformatted manuals.
- Use mandoc(1) as the default run-time manual formatter instead of groff.
- Use makemandb(8) by Abhinav Upadhyay instead of makewhatis(8) together with versions of apropos(1) and whatis(1) based on it, featuring full text search, but not semantic search.

All this was first released with NetBSD 6.0 on October 17, 2012.

Semantic searching is not yet supported, not even as an option.

Status in FreeBSD

- An mdocml port by Ulrich Spörlein exists since March 9, 2009.
- First code patch sent in by Ulrich Spörlein on July 18, 2009.
- Source code in the base repo and installed by default since October 19, 2012.
- First released with FreeBSD 10.0 on January 20, 2014.

Status in DragonFly BSD

- Source code in the base repo and installed by default since October 27, 2009 (Sascha Wildner)
- First released with DragonFly BSD 3.6.0 on March 28, 2010.
- First code patch sent in by Franco Fichtner on November 25, 2013.

In both:

- mandoc(1) is installed, but not used.
- Semantic searching is not yet supported, not even as an option.

Status in non-BSD systems

Minix 3

- Source code in the base repo since June 26, 2010 (Ben Gras).
- Somewhat apathetic, still using a version that is more than three years old.

Alpine Linux

- A testing aport exists since July 6, 2010 (Natanael Copa).
- The aport moved from testing to main on June 12, 2011 (Natanael Copa).
- Continuously maintained.

Arch Linux

- An mdocml package exists since October 3, 2010 (Markus M. May).
- Updated to 1.12.3 on April 17, 2014 (new maintainer Jesse Adams).

Little Blue Penguin / Korora
(Eudyptula minor)
(c) 2009 Fir0002/Flagstaffotos
@wikimedia (CC)

Debian GNU/Linux

- An unofficial mdocml package exists since April 19, 2014 (Jesse Adams).

Solaris, MacOS X

- Tested using pkgsrc.

Adoption status (summary)

OpenBSD
: Only documentation formatter in base for three years,
and now also enabled as the manual page search system.

NetBSD
: Default manual formatter for two years.

FreeBSD, DragonFly
: Part of the base system, and both consider switching to it.

Minix 3
: Part of the base system.

Linux, Solaris, MacOS
: Tested.

pkgsrc
: Provides packages
for additional systems.

A lot of traction in all BSDs and beyond.

Blue Duck / Whio in Sussex, England
(Hymenolaimus malacorhynchos)
(c) 2005 Philipp Capper @flickr (CC)

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, and in some cases with no usable manuals at all.

How can this be solved? →

< >

The OpenBSD solution for manual pages in ports

- Make sure that no port tries to preformat manuals during the build target.
- Let every port install manual page sources during the fake install target.
- For the majority of ports, mandoc(1) can handle all manuals:
  That's it, you are done with respect to these.
- For the remaining minority of ports: Set a special boolean make(1) variable in the port Makefile, in OpenBSD called USE_GROFF.
- That variable implies a build dependency on the groff port.
- When building the package, the ports framework runs groff on the fly and packages the preformatted pages instead of the source pages.
- After installing the packages, this will work just fine at run time: The preformatted pages will be diplayed directly by man(1), and man(1) will format the source pages with mandoc(1), with no dedicated configuration.
- Example of OpenBSD: 7952 ports, 1217 still USE_GROFF (15%).
  Some of these probably don't really need it, but there is no hurry.
  Removing USE_GROFF needs a manual check — which was already done for about 3000 ports during the last three years.
- This concept has been designed and implemented by Marc Espie, and it has proven very sturdy and very easy to use.

makewhatis(8) in ports

- Base and X manual databases are essentially static.
- But packages get installed and deinstalled.
- You could wait for the periodic weekly(8) makewhatis(1) rebuild.
- Better idea:
  During pkg_add, run `makewhatis -d /usr/local/man files ...`
  During pkg_delete, run `makewhatis -u /usr/local/man files ...`
- Done routinely on OpenBSD, works seamlessly with the new makewhatis.
- So, right after pkg_add(1), you call apropos(1), and it finds the freshly installed manual pages.

Features that help, in particular for ports

- Handle preformatted manuals (since Nov. 27, 2011).
- Natively support gzip(1)'ed manual pages (since March 26, 2014).
- During makewhatis -d/-u, automatically rebuild missing or corrupt databases (since April 18, 2014).

Red-crested parakeet / Kakariki
(Cyanoramphus novaezelandiae)
(c) 2010 Sid Mosdell @flickr (CC)

What about the reverse perspective? →

< >

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.
- 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.

Case study: the sudo(8) manuals

Build system for the distribution tarball

Simplified code from the Makefile:

```
sudo.man: sudo.mdoc
        mandoc -Tman sudo.mdoc > sudo.man

sudo.cat: sudo.mdoc
        mandoc sudo.mdoc > sudo.cat
```

Installation system

- If `./configure` finds mandoc(1), install the *.mdoc pages
- If `./configure` does not find nroff(1), install the *.cat pages
- If `./configure` successfully tests `nroff -mdoc`, install the *.mdoc pages
- Otherwise, install the *.man pages.
- To override this autodetection logic, provide `--with-mdoc` or `--with-man` options to `./configure`.

Implementation of the mdoc to man converter

- First run the mdoc(7) parser, constructing exactly the same abstract syntax tree in memory as when running -Tascii, -Thtml, or -Tps.
- Then run the mdoc(7)-to-man(7) output module, structured similarly as the mdoc(7)-to-ASCII output module, but sharing no code.
- One file, 1600 lines of very straightforward C source code.
- One macro lookup table containing pre- and post-node action functions and pre- and post-node output strings for each mdoc(7) macro type.
- The module iterates the syntax tree and calls the appropriate action functions for each mdoc(7) node.
- An alternative, slightly more flexible approach would have been to first translate the mdoc(7) syntax tree to a man(7) syntax tree, then provide a non-translating man(7) output module.
- That would have allowed man(7)-to-man(7) code normalization as a by-product.
- However, the direct approach was simpler and has so far proven sufficient for all practical needs.

Tomtit / Miromiro
(Petroica macrocephala)
(c) 2007 Mark Jobling
@wikimedia (PD)

Implementation of the mdoc to man converter (2)

- Typical example of a tool that was technically quite easy to build on top of existing infrastructure, that is the mandoc(3) parser library, but quite useful and powerful in practice.
- At first, I underestimated the importance of this tool, development only proceeded haltingly, nearly exclusively at hackathons:
- I started development on September 17, 2011 (s2k11, Ljubljana).
- The bulk of the work was done around July 10, 2012 (g2k12, Budapest).
- It is ready for production since November 19, 2012 (c2k12, Coimbra).

Tui (Prosthemadera novaeseelandiae) (c) 2011 Sid Mosdell @flickr (CC)

Continuous evolution of mandoc(1)

- Output modes -Tuft8 and -Tlocale since May 20, 2011.
- Indirect references in roff(7) expansion since April 7, 2014.
- Expansion of roff(7) number registers since March 21, 2013 (Christos Zoulas).
- Almost complete support for roff(7) numerical expressions since April 7, 2014.
- Numeric comparison in roff(7) conditionals since April 3, 2013 (Chr. Zoulas).
- String comparison in roff(7) conditionals since March 8, 2014.
- Newly supported roff(7) requests:

  .as (append to string) .cc (control character)
  .it (input line trap) .ll (line length)
  .rr (remove register) .tr (character translation)
- newly supported man(7)-ext macros:

  .EX/.EE (example display)
  .OP (optional element)
  .PD (paragraph distance)
  .UR/.UE (uniform resource identifier)
- Lots of bugfixes and formatting corrections.

Fantail / Piwakawaka (Rhipidura fuliginosa)
(c) 2007 Brenda Anderson @flickr (CC)

Matured considerably in addition to growing new features.

Did we reach our goals last time?

During BSDCan 2011, I presented the following "possible future directions":

- Install manual sources, not preformatted manuals. DONE June 23, 2011
- Implement -mdoc -Tman. DONE November 19, 2012
- Rewrite apropos(1) and makewhatis(8) to use mandoc. DONE April 14, 2014
- Write pod2mdoc(1) in Perl. WIP kristaps@ in C
- Implement -man -Tmdoc. WIP kristaps@ via docbook2mdoc
- Use the mandoc toolbox to replace man.cgi on the websites. OPEN

Paradise Shelduck / Putangitangi (Tadorna variegata) (c) 2008 Dave Young @flickr (CC)

Possible future directions

- Use the new apropos(1) to replace man.cgi on the websites.
- Integrate preconv(1) into mandoc(1) for better UTF-8 handling.
- Support transitions from man(7) to mdoc(7) with doclifter and docbook2mdoc.
- Support semantic enrichment of Perl and libReSSL manuals with pod2mdoc.
- Unify parsers, allowing improvement of low-level roff(7) support.

Buller’s Mollymawk / Toroa-teoteo (Thalassarche bulleri) (c) 2008 Andrew Barclay @flickr (CC)

Thanks!

Kristaps Dzonsons (bsd.lv)
for writing mandoc

Franco Fichtner (DragonFly), Jörg Sonnenberger, Christos Zoulas, Tsugutomo Enami (NetBSD)
for code contributions

Marc Espie (OpenBSD)
for OpenBSD ports integration and lots of important feedback

Jason McIntyre (OpenBSD)
for lots of feedback and countless useful discussions

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

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

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

Anthony J. Bentley (OpenBSD)
for porting related software to OpenBSD and for many bug reports

Natanael Copa (Alpine), Jesse Adams (Arch), Matthias Scheler (Solaris), Ben Gras (Minix 3)
for porting mandoc(1) and providing feedback

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

Thanks!

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

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

For bug reports and useful suggestions and discussions:

Antoine Jacoutot, Bob Beck, Brad Smith, Bret Lambert, Bryan Steele, David Coppa, David Gwynne, Gleydson Soares, Florian Obser, Igor Sobrado, Jasper Lievisse Adriaanse, Jérémie Courrèges-Anglas, Jonathan Gray, Juan Francisco Cantero Hurtado, Kenji Aoyama, Kenneth R. Westerback, Landry Breuil, Martin Pieuchot, Matthew Dempsky, Matthias Kilian, Miod Vallat, Nicholas Marriott, Nick Holland, Nigel Taylor, Okan Demirmen, Paul de Weerd, Philip Guenther, Stefan Sperling, Ted Unangst, Theo de Raadt, Todd T. Fries, Vadim Zhukov (OpenBSD)

Nicolas Joly, Abhinav Upadhyay, Havard Eidnes, Jonathan Perkin (NetBSD), Antonio Huete Jimenez, Sascha Wildner (DragonFly), David Hill (Bitrig), Yuri Pankov (IllumOS), Michael Dexter (bsd.lv)

Chris Hettrick, David Levine, Dmitrij D. Czarkoff, Fabian Raetz, Jan Stary, Maxim Belooussov, Mike Small, Mikolaj Kucharski, Steffen Nurpmeso, Tim van der Molen

Robin / Toutouwai (Petroica australis)
(c) 2007 Mark Jobling @wikimedia (PD)

Bye, and thanks for all the birds!

https://www.flickr.com/photos/whereisbrent/461055143 Brent Barrett: Kea juvenile (by-nc-nd)
http://www.columbia.edu/cu/computinghistory/1965.html Columbia University: IBM 7094 (with permission)
Courtesy of University Archives, Columbia University in the City of New York
http://cm.bell-labs.com/who/dmr/picture.html Bell Labs: PDP-11 (with permission)
Reprinted with permission of Alcatel-Lucent USA Inc.
http://www.mckusick.com/beastie/shirts/bsdunix.html USENIX: 4.2BSD Beastie
https://www.flickr.com/photos/29954808@N00/1300190844 57Andrew: Rock Wren (by)
https://www.flickr.com/photos/docnz/8528275645 NZ DOC: Kakapo (by-nc-sa)
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://commons.wikimedia.org/wiki/File:Nz_boobook.JPG Aviceda: Morepork (by-sa)
https://www.flickr.com/photos/sidm/6681523917 Sid Mosdell: Bellbird (by)
http://commons.wikimedia.org/wiki/File:Nz_falcon.JPG Py1jtp: NZ Falcon (by-sa)
https://www.flickr.com/photos/digitaltrails/2214314908 Digitaltrails: Rifleman (by-nc-sa)
http://commons.wikimedia.org/wiki/File:Spotted_Shag_%28Phalacrocorax_punctatus%29_in_flight_2.jpg Avenue: Spotted Shag (by-sa)
http://commons.wikimedia.org/wiki/File:Moa_mock_hunt.jpg Augustus Hamilton: Moa (c) expired
http://commons.wikimedia.org/wiki/File:Little_Penguin_Feb09.jpg Fir0002/Flagstaffotos: Little Penguin (by-nc)
https://www.flickr.com/photos/flissphil/53850663 Philip Capper: Blue Duck (by)
https://www.flickr.com/photos/sidm/5225410158 Sid Mosdell: Red-crested parakeet (by)
http://commons.wikimedia.org/wiki/File:Petroica_macrocephala_macrocephala1.jpg Mark Jobling: Tomtit (pd)
https://www.flickr.com/photos/sidm/6557924841 Sid Mosdell: Tui (by)
https://www.flickr.com/photos/curiouskiwi/566823278 Brenda Anderson: Fantail (by-nc-sa)
https://www.flickr.com/photos/22018552@N08/2944307294 Dave Young: Paradise Shelduck (by)
https://www.flickr.com/photos/electropod/2817879475 Andrew Barclay: Buller’s Mollymawk (by-nc-nd)
http://commons.wikimedia.org/wiki/File:070308_Stewart_Island_robin_on_Ulva.jpg Mark Jobling: NZ Robin (pd)

What would you like to ask? →

<