Let’s make manuals more useful!

Tutorial on software documentation

EuroBSDCon 2014, September 26, Sofia

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

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

София BSD Mascot
© 2014 Алица Димитрова

Kea juvenile (Nestor notabilis)
© 2007 Brent Barrett @flickr (CC)

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
- 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.

Let’s read a manual!

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

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

Progress during the last few months:

- 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 in production since April 14, 2014.
- New man.cgi(8) is online on www.openbsd.org since July 12, 2014.

More details on all this later!

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

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

- Based on the roff(7) language (1964).
- Successor of the man(7) language (1979).
- Designed for 4.4BSD at Berkeley (1990).
- Implemented by mandoc(1) (2008) and groff(1) (1989).
- Used by default in OpenBSD, NetBSD, FreeBSD, DragonFly and illumos.

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 © 1965 Columbia Univ. Archives (Courtesy)

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

All other lines are text lines.

Why do we still use that syntax 50 years later? →

< >

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?"
```

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!

Application to software manuals? ⟶

< >

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 © 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

- 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.

4.2BSD Beastie

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) © 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, semantic keys
  - mdoc(7), man(7), tbl(7) and some eqn(7) and roff(7) input
  - ASCII, UTF-8, HTML, XHTML, 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.

What are we going to do? →

< >

Contents of this tutorial

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

The tutorial will officially end at about 17:30.
We can continue work & discussions in a smaller group if any of you want to.

Short introduction of ourselves

Four short sentences per participant:

- Name
- Main role (application programmer, technical writer, operating system developer, system administrator, BSD user, ... ?)
- Main project affiliation
- Most important goal for this tutorial

Let me start:

- Ingo Schwarze
- Current mandoc project lead at mdocml.bsd.lv
- OpenBSD operating system developer (userland)
- Show you the current state of the art in BSD system documentation and try to help with any questions you might have...

Let’s make manuals more useful!

Basic concepts you need right away

- Macro lines (for markup) vs. text lines.

```
.Sh HISTORY
An
.Fn fopen
function first appeared in
.At v1 .
```

- Macro arguments are separated by blanks.
- Quoting allows blanks in arguments.

```
.Ft "FILE *"
.Fo fopen
.Fa "const char *path"
.Fa "const char *mode"
.Fc
```

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

For more details about roff syntax, see the roff(7) manual.

- Every page starts with a prologue.
- Content is organized into sections.

The mdoc(7) prologue

Macro lines only, no text lines!

Always use the following macros in the following order:

Dd Document date: *Month day, year*
Dt Document title: *NAME* (ALL CAPS) and *section* number (see mdoc(7)).
Os Optional operating system name: Just leave it blank.
Sh Section header: Argument must be **NAME**.
Nm Page name: Here use its proper case.
Nd One-line description: No quoting is needed.

If your prologue has *six* lines, it's probably complete.

For example:

```
    .Dd July 16, 2013
    .Dt CAT 1
    .Os
    .Sh NAME
    .Nm cat
    .Nd concatenate and print files
```

Централна минерална баня, София
© 2010 Михал Орела @flickr (CC)

SYNOPSIS for a utility (1)

For utilities and functions, the first section after NAME is always SYNOPSIS.

- Only document syntax, not semantics.
- No free text!
- Formatting is fully automatic.

Required macros:

Nm	Utility name. Usually the same you used in the NAME section.
Op	Optional syntax element.
Fl	Command line options (flags).
Ar	Command line arguments.

Typical example:

```
.Sh SYNOPSIS
.Nm cat
.Op Fl benstuv
.Op Ar
```

Formatted output:

SYNOPSIS
    cat [**−benstuv**] [*file* ...]

Софийски университет
© 2011 Пламен Агов studiolemontree.com (CC)

SYNOPSIS for a utility (2)

Typical example:

```
.Sh SYNOPSIS
.Nm cat
.Op Fl benstuv
.Op Ar
```

Formatted output:

SYNOPSIS
        cat [**−benstuv**] [*file ...*]

Национален дворец на културата, София
© 2012 Bin in Garten @wikimedia (CC)

Important features:

- Most macros can take other macros as arguments.
- The *called* macros don’t have a dot, like **Fl** and **Ar** above.
- The **Op** macro is an example of an *enclosure* macro,
  a special case of a *block* macro,
  having a *scope* that can contain macros and text.
- For **Op**, the scope extends to the end of the input line.
- A few macros have default arguments, like **Ar** above.

What about functions? →

< >

SYNOPSIS for a function

Required macros:

In	include file
Ft	function type
Fo	open function decl.
Fa	function argument
Fc	close function decl.

Хотел Ню Отани 98м и Витоша 2290м
© 2006 Тодор Божинов @flickr (CC)

Typical example:

```
.Sh SYNOPSIS
.In unistd.h
.Ft ssize_t
.Fo read
.Fa "int d"
.Fa "void *buf"
.Fa "size_t nbytes"
.Fc
```

Formatted output:

SYNOPSIS
#include <unistd.h>

ssize_t
read(*int d*, *void *buf*, *size_t nbytes*);

The **Fo** macro is a *block macro* starting a scope
that requires *explicit* closure by the **Fc** companion macro.

DESCRIPTION

- Every manual has a DESCRIPTION section.
- First explain the purpose of the utility or function.
- Then describe the syntax and semantics of all features.
- Be complete, but concise.
- If it gets very long, it can be split into multiple sections.

Remaining sections have conventional names and a conventional order:

- RETURN VALUES (functions only)
- ENVIRONMENT
- FILES
- EXIT STATUS (utilities only)
- EXAMPLES
- DIAGNOSTICS (utilities and drivers)
- ERRORS (functions only)
- SEE ALSO, STANDARDS, HISTORY, AUTHORS, CAVEATS, BUGS.

See the mdoc(7) manual for more information.

Help on mdoc(7)

What was discussed so far should be sufficient to get started.
You can look up all the rest when you need it...

- 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
- Look at the manuals in the OpenBSD base system 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
```

- **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.

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.

Let’s look at an example →

< >

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`.

Goals of quality control for manuals

- Find fatal errors, make sure all manuals actually produce output.
- Make sure all intended content is actually shown.
- Catch severe formatting errors.
- Catch typos
  and stylistic glitches.
- Improve portability
  (but don't overdo this).
- Improve robustness.
- Unify display style.
- Improve mdoc(7) coding style.
- Find formatter bugs.

Тумба 1880м, Беласица
© 2009 Деян Василев @wikimedia (CC)

Various tools are available, and some tasks require manual checking.

The most important tool is **mandoc -Tlint**.

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.

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.

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 cleanups in mandoc: July 2009, May 2010, August 2010, October 2010, January 2011, March 2011, July 2014, ...

Getting started with mandoc -Tlint →

< >

Catching fatal errors

```
mandoc -Tlint -Wfatal *
```

- fatal errors abort the parser, no formatted output at all
- consequently, a manual with a fatal error is not much better than none at all
- in a large tree, handle these first, before anything else
- can be skipped when checking just a few manuals, see next slide

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

Relevant fatal errors in practice — all related to file inclusion:

- use of the unsafe macro **Bd -file**
- use of the **so** file inclusion request with an absolute path
- use of **so** with a path containing ".."
- **so** pointing to a file that doesn’t exist or can’t be opened

See the mandoc(1) manual in the portable mandoc distribution for details, section "FATAL errors".

Catching errors

```
    mandoc -Tlint -Werror *
```

- may cause loss of information or severe misformatting
- almost always needs fixing, trying hard to avoid false positives

Classification:

- unencoded non-ASCII characters in the input
- unknown or mistyped macro or request names
- blocking issues:
  - opening blocks that are never closed again
  - closing blocks that were never opened
  - items outside lists
  - bad nesting of blocks that don’t support it
- severe issues with macro or request arguments:
  - missing essential arguments
  - invalid arguments that cannot be handled adequately
  - excessive arguments that get completely lost during formatting

Again, see mandoc(1), section ERRORS, for a full list.

Mandoc warnings

```
mandoc -Tlint *
```

- everything else mandoc considers problematic
- expect lots of output in a low-quality tree
- false positives are rare, but a few do happen
- use your judgement, do not blindly follow it

Ботев 2376м, Стара Планина
© 2010 Gerovitus91 @wikimedia (CC)

The most important classes are:

- structural errors and syntax errors that only have local effects and do not cause information loss
- low quality syntax like badly nested blocks or macro usage in contexts expecting plain text
- macros that have no effect or are slightly misplaced
- missing arguments or information, if the effect is only local
- violations of usual structural or formatting conventions
- warnings about robustness and portability
- dubious usage of white space and comments

Again, all warnings are listed in the portable mandoc(1) manual.

Are there other tools? ⟶

< >

Checking with mdoclint

- Author: Thomas Klausner <wiz@NetBSD.org>
- similar focus as **mandoc -Tlint**: syntax
- much fewer tests than mandoc(1)
- used to contain some additional tests
- most of these have recently been added to mandoc(1)
- also tries to avoid false positives, though not quite as strictly **mandoc -Tlint**
- Do not slavishly follow its findings!

- available on NetBSD (pkgsrc/textproc/mdoclint)
  and OpenBSD (src/regress/usr.bin/mdoclint)
- ought to be trivial to port to other systems providing perl(1)

- To run it, give the names of the files you want to check on the command line.
- main purpose of options: suppress some messages

Checking with igor(1)

- Author: Warren Block (FreeBSD)
- completely different focus: style and spelling, knows little about syntax
- not afraid of false positives — bad for bulk checks: noise!
- good to find candidates of bad style or spelling in smaller sets of manuals
- finds whole classes of issues the two linters are completely unaware of

- available as a port in both FreeBSD and OpenBSD (textproc/igor)
- should also be trivial to port to any system having perl(1)
- provide the names of the files to be checked as command line arguments
- command line options to suppress unwanted messages

Groff-mandoc comparisons

most useful to find parser and formatter bugs

- This part is usually straightforward to interpret.

may also provide hints that code is of limited portability

- This part requires wide experience with the mdoc(7) language:
- Just because it formats differently doesn't mean it's bad code!

gmdiff

- very short and simple but handy shell script to run such comparisons
- available from the portable mandoc repository
- provide the names of the files to check as command line arguments

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

Installation checking

with makewhatis(8) -p

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

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

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

Let’s get our hands dirty... →

< >

Exercises: general remarks

1. **Choose** what interests you, you cannot do them all; this phase: 1-3 tasks.
2. Do not waste time reading instructions for exercises you won't work on.
3. Feel free to work alone or with one or two partners at your choice.
4. **Communicate**, don't be shy to interrupt others for asking questions.
5. **Avoid** time-consuming repetitive work during the tutorial.

   For example, don't toil on a long of list options.

   Just write down the first two or three.

   Then move on to the next section.

Squirrel digging © 2010 Phil Davis @flickr (CC)

6. **Present** results *if you want to* (1-3 minutes, tell me while working).

Recommended exercises

Of course, all participants are free to choose any exercise they are interested in.

Software or documentation developers...

- ... working on software lacking one or more manual pages:
  - 8.1.1: (***) **Writing** a manual from scratch
- ... working on software having non-mdoc documentation:
  - 8.1.2: (**) **Translating** a manual to mdoc
- ... working on software having mdoc documentation:
  - 8.3.1: (*) **Checking** one or a few specific pages
- ... maintaining a portable software package:
  - 8.2.1: (****) **Packaging** autogenerated man and cat
  - 8.2.2: (*****) Writing **configuration tests**

Operating system developers and port maintainers:

- 8.3.2: (***) Running **bulk** quality checks
- or 8.3.1 above or 8.4.* below

System administrators:

- 8.4.2: (*** - ****) Checking manual **locations and formats**

End users:

- 8.4.1: (* - ***) Testing makewhatis and **apropos**

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)
© 2008 Glen Fergus @wikimedia (CC)

New functionality →

< >

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)

Intermediate new search features ⟶

< >

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, setgrou
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
[...]
```

Advanced new search features →

< >

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
```

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

Complex search queries are working since January 4, 2014.

New output features →

< >

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) © 2012 Sid Mosdell @flickr (CC)

Can we use all that for man(1), too? →

Unified user interface including man(1)

- man(1) now included in the mandoc toolbox
- unified user interface for mandoc(1), man(1), whatis(1), apropos(1)
- same command line options for all
- still different default behaviour

Мусала 2925м, Рила
© 2007 mattyhike @flickr (CC)

Steps taken by the unified main program:

1. Decide how to interpret the command line arguments.
2. Build a list of manual pages, usually from a database search.
3. Decide which kind of output to provide.
4. Optionally spawn a pager.
5. Loop around the list of manual pages, producing some output for each.

Unified input options

Specify the meaning of the command line arguments:

−l	command line arguments = filenames (no database search) This is the default when called as **mandoc**.
*	The default when called as **man**: command line arguments = names (for exact matches) default output mode: show exactly one manual (best match)
−f	command line arguments = names (for complete word search) This is the default when called as **whatis**.
−k	Support the full **apropos** search syntax. default output mode: list of title lines

Some **database selection** options only matter when **−l** is not active:

−C	Use the specified *file* instead of the default configuration file.
−M	Use the specified *path* instead of the default one. Do not use any configuration file.
−m	Use the specified *path* in addition to the default one.
−S	Restrict the search to the specified *architecture*.
−s	Restrict the search to the specified *section*.

Unified output options

Specify which kind of output to provide:

−a	Display all matching manual pages, one after the other. This is the default for **−l** input mode.
−h	Display only the SYNOPSIS lines of the matching pages. Implies **−a**.
−O	When showing a list in **−f** or **−k** mode, display the specified macro key instead of the **Nd** one-line descriptions.
−w	Display only the pathnames of the matching manual pages.

Some **parser and formatter options** only take effect when a parser is actually run:

−I	Override the default operating system name for the mdoc(7) **Os** macro.
−m	Specify the input format. Defaults to **−mandoc**, requesting autodetection.
−O	Comma-separated formatter-specific output options.
−T	Select the output format. Defaults to **−Tascii**.
−W	Specify the minimum message *level* to be reported on the standard error output and to affect the exit status. Defaults to **−Wfatal**.

Finally, the **−c** option can be used to suppress the pager and just copy the formatted manuals to standard output.

Nothing planned, just for fun!

Four months back at BSDCan 2014 in Ottawa, i presented a slide "possible future directions". This project was *not* listed. I didn’t expect myself that i would do this.

> “Are there any plans for providing a man(1) command also?
> This would make mdocml a possible, standalone replacement for the groff and man-db combination (typical in Linux distributions).”
> — Paul Onyschuk (Alpine Linux), August 9, 2014

- First impulse: return standard negative answer.
- ...
- Suddenly realized almost all code already there!
- Needed just a bit of code reshuffling...
- Had to do the 1.13.1 and 1.12.4 releases first.
- ... so in the end, it took about two weeks:

Aug 9	idea
Aug 21	first working version
Aug 26	unified interface in OpenBSD

Since man(1) is not production quality yet,
it is not yet used by default in OpenBSD.

Foals Just Wanna Have Fun
© 2009 Gary Tanner @flickr (CC)

Are there any tricky points? →

< >

What a ‘name’ is

- man(1) displays the manual with the given name.
- Traditional man(1) only uses filenames as names in this sense.
- makewhatis(8) adds names to the *names* table in the mandoc.db(5) databases.
- The *names.bits* column tells where the name came from:
  - **Dt/TH** header line
  - **Nm** in the NAME section
  - **Nm** in the SYNOPSIS section
  - filename
- Right now, the mandoc version of man(1) uses all types of names.
- That can and should be tuned!
- Use **Nm** macros from the NAME section only?
- That way, all hard links, symbolic links, and .so link files become obsolete. Consequently, the number of files in a typical operating system installation can be reduced by more than three thousands.
- Continue to use filenames as a fallback?

Remaining issues

option **-i** (interactive)
  not yet integrated — old manpage(1) utility was never used

man.conf(5)

- ignored: **_build**, **_default**, **_subdir**, **_suffix**, *section*
- using **_whatdb** instead of **_default**
- hardcoded subdirectories: `{man,cat}N`
- hardcoded search order: 1, 8, 6, 2, 3, 5, 7, 4, 9

MACHINE environment variable
  not supported (and not really useful?)
  But uname(3) should be used to select the right architecture!

manual selection:
  When finding a formatted and an unformatted manual of the same name in the same section, the old man(1) shows the one that was less recently changed. The mandoc man(1) currently always prefers the unformatted version, even if it's older than the formatted version.

additional results
  for bogus names, see previous slide

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)
© 2014 Charlesjsharp @wikimedia (CC)

- 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.

Coffee break

until 16:30

A Youngster on the Quantocks © 2009 Mark Robinson @flickr (CC)

Getting started with base integration

- Import mandoc into the base tree.
- Collect initial experience with mandoc.
- Report bugs and get them fixed.
- Establish the contact
  to everybody who is interested.
- Get familiar with specific requirements of the system in question and specific topics the developers are concerned about. Do not neglect this aspect or it will bite you later.

New Forest Foal
© 2010 Krista van der Voorden @flickr (CC)

Don’t be lazy, get moving...

- In OpenBSD, this phase lasted from June to December 2009.
- Because mandoc has matured, it is much easier now.
- In any case, this phase can be shorter.
- Start the next phase early!

What’s the first step for systematic integration? $\longrightarrow$

Identify fatal issues

- Try to build the whole tree.
- Ignore all non-fatal issues for now.
- Report all fatal errors upstream.
- Have them fixed upstream
  or devise workarounds.
  We are glad to provide help!
- Only as a last resort,
  change the affected manuals.

- In OpenBSD, this phase took place
  in January and February 2010.
- Ought to be much easier now:

Very young zebra
© 2009 Tambako @flickr (CC)

mandoc hardly throws any fatal errors any longer

What about manuals that fo build? →

< >

Investigate non-fatal errors

- Run "mandoc -Tlint -Werror".
- Prioritize fallout that ruins content or formatting.
- Distinguish mandoc bugs from markup bugs.
- Report mandoc bugs upstream. Typical areas: Missing low-level roff(7) functionality. Undefined macros.
- Watch out for home-grown non-standard features used in your tree. This also applies to the next phase.
- Watch out for tbl(7). If some of those don’t work yet, as a last resort, you might continue to build those with groff at first.
- Fix ruinous markup bugs in your tree. Do not change manuals to work around mandoc bugs.
- Don’t let non-critical markup bugs overwhelm you. Postpone fixing them if they cause too much slowdown.
- In OpenBSD, the equivalent happened in March 2010. Some parts were postponed to June 2010, after the switch. The OpenBSD tbl(7) pages were switched in October 2010.

Final checks before the switch... →

Check mandoc output

- For the manuals in your tree, systematically compare groff(1) and mandoc(1) output.
- Use gmdiff.
- Look at the patches to textproc/groff in OpenBSD ports.
- The comparison will not be quite easy because there will be some noise, most of it about whitespace.
- Won’t become completely identical, but try to make sure no content gets lost and no formatting is completely garbled.
- If you find any serious issues, report them, in particular if mandoc(1) fails to flag them as ERRORs.
- Patching manuals is usually not the right approach in this phase.
- Quickly move on to the next phase, that is, when you are convinced there are no show-stopper issues, not when you feel everything is perfect. Your system will mature best when it’s enabled by default and when you get and use real-world feedback.

Ready for switching? →

< >

Switch over the tree

- Make sure there is plenty of time until the next release.
- Watch out for bug reports from users of -current.
- Report and fix bugs as quickly as possible.
- It can be done in one or two steps.

On the Road
© 2010 tiny_packages @flickr (CC)

- OpenBSD did it in two steps:
  Change the formatter in April 2010.
  Install source manuals since October 2010.
- NetBSD did it in one step in February 2012.
- One-step process is recommended if you feel confident you can handle it.
- Two-step process is slightly more prudent but causes more work.

More tools to make sure it works? →

< >

Regression testing

- A regression suite does exist.
- Tightly tied into the OpenBSD regression system.
- It would be useful to get it running elsewhere.
- Do *not* copy and change it: A maintenance nightmare would ensue!

Not at all for portable make(1), it requires:

- in assignments: `?= += !=`
- in variable expansion: `:C :M :S`
- control statements: `.if .else .for`
- in conditionals:
  `defined() empty() target()`
- BSD make(1) modules: bsd.subdir.mk bsd.obj.mk bsd.regress.mk bsd.prog.mk bsd.own.nk

- This list incomplete.
- Not sure yet how to proceed...

Horse Fly
© 2010 Jeff Burcher @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.

Is there anything that helps? ⟶

Continuous evolution of mandoc(1)

- 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)
© 2007 Brenda Anderson @flickr (CC)

Matured considerably in addition to growing new features.

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)
© 2010 Sid Mosdell @flickr (CC)

Step-by-step instructions? →

Getting started with ports

- Build and commit a textproc/groff port (version 1.22.2), even if you have groff in base.
- Introduce the USE_GROFF port Makefile variable.
- Trigger a groff build dependency for USE_GROFF.
- Implement format-on-the-fly for USE_GROFF and install formatted.
- Without USE_GROFF, install source manuals.
- Pay attention to get packing lists right.
- Safe way: Turn USE_GROFF on for all ports having manuals. Shortcut: Skip those that don’t have it in OpenBSD.
- Remove groff from base.
- Start removing USE_GROFF on a case by case basis. To be extra safe: Only do it for ports having no mandoc ERRORs and identical output with groff and mandoc.
- This is another critical phase: Stay tuned for bug reports from users and work with upstream to get them resolved.
- In OpenBSD, this happened in October 2010.

Recommended exercises

Of course, all participants are free to choose any exercise they are interested in.

Software or documentation developers...

... working on software lacking one or more manual pages:

8.1.1: (***) **Writing** a manual from scratch

... working on software having non-mdoc documentation:

8.1.2: (**) **Translating** a manual to mdoc

... working on software having mdoc documentation:

8.3.1: (*) **Checking** one or a few specific pages

... maintaining a portable software package:

8.2.1: (****) **Packaging** autogenerated man and cat
8.2.2: (*****) Writing **configuration tests**

Operating system developers and port maintainers:

8.3.2: (***) Running **bulk** quality checks
or 8.3.1 above or 8.4.* below

System administrators:

8.4.2: (*** - ****) Checking manual **locations and formats**

End users:

8.4.1: (* - ***) Testing makewhatis and **apropos**

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 almost four 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.
- Install manual sources, not preformatted manuals since June 23, 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.
- New man.cgi(8) online on openbsd.org since July 12, 2014.
- Unified interface for mandoc(1) and apropos(1) since August 26, 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.
- Enabling it is very difficult because it clashes with makemandb(8).

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.
- Suggested next steps: update to 1.13.1 and use it to format the manuals.

Non-BSD systems →

< >

Status in non-BSD systems

illumos

- Mandoc is contained in the base system
  and used by default for formatting manuals
  since July 21, 2014 (Garrett D'Amore).
- The first and so far only non-BSD system to accomplish this, and the third system grand total.

Minix 3

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

MacOS X

- An mdocml package exists since September 5, 2010.
- Unfortunately, it seems abandoned.

Cygwin

- An mdocml package exists since December 12, 2012.
- Updated to 1.12.2 on November 11, 2013 (Yaakov Selkowitz).

Linux →

< >

Status in Linux distributions

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).
- Updated to 1.12.4 on August 28, 2014 (Peter Bui).
- Potential interest to use mandoc as an alternative to groff plus man-db.

Arch Linux

- An mdocml package exists since October 3, 2010 (Markus M. May).
- Updated to 1.12.3 on April 17, 2014 and to 1.13.1 on August 17, 2014 (new maintainer Jesse Adams).

Slackware Linux

- An mdocml SlackBuild exists since January 7, 2014 (Dániel Lévai).
- Updated to 1.13.1 on August 30, 2014 (Dániel Lévai).

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

CentOS, Debian, Fedora, RedHat, SuSE, Ubuntu Linux

- Unofficial mdocml packages exists since April 19, 2014 (Jesse Adams).

What shall we do in the future? →

< >

Possible future directions

- Replace the traditional BSD man(1) implementation with the mandoc one.
- Switch the default output mode from **−Tascii** to **−Tlocale**.
- Integrate preconv(1) into mandoc(1) for better UTF-8 handling.
- Improve pod2mdoc(1) to better support perlpod(1) to mdoc(7) migration, in particular for the LibreSSL manuals.
- Support automatic semantic enrichment of Perl manuals with pod2mdoc(1). I’m not yet sure this is practicable, it’s just an idea so far.
- Help man(7) to mdoc(7) migration with doclifter(1) and docbook2mdoc(1).
- Unify parsers, allowing further improvement of low-level roff(7) support.

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

Thanks!

Kristaps Dzonsons (bsd.lv)
for writing mandoc

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

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

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

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.)

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

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

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

Thanks!

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

Natanael Copa, Paul Onyschuk (Alpine), 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

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

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

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

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

Thanks!

For bug reports and useful suggestions and discussions:

Alexander Bluhm, Antoine Jacoutot, Bob Beck, Brad Smith, Bret Lambert, Brian Callahan, Bryan Steele, Daniel Dickman, David Coppa, David Gwynne, Doug Hogan, 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, Matthieu Herrb, Matthias Kilian, Miod Vallat, Nicholas Marriott, Nick Holland, Nigel Taylor, Okan Demirmen, Paul Irofti, Paul de Weerd, Peter Hessler, Philip Guenther, Remi Pointel, Reyk Flöter, Stefan Sperling, Ted Unangst, Thordur Bjoernsson, Todd T. Fries, Vadim Zhukov (OpenBSD)

David Holland, Nicolas Joly, Abhinav Upadhyay, Havard Eidnes, Jonathan Perkin (NetBSD), Antonio Huete Jimenez, Sascha Wildner (DragonFly), David Hill (Bitrig), Michael Dexter (bsd.lv), Carsten Kunze (Heirloom Doctools),

Chris Bennett, Chris Hettrick, David Levine, Dmitrij D. Czarkoff, Fabian Raetz, Frantisek Holop, Guy Harris, Igor Zinovik, James Jerkins, Jan Stary, Jörn Clausen, Justin Haynes, Marcus Merighi, Maxim Belooussov, Michel Jansens, Mike Small, Mikolaj Kucharski, Ryan Flannery, Steffen Nurpmeso, 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://2014.eurobsdcon.org/wp-content/uploads/2014/06/BSDSofiaForWeb.png Alica Dimitrova: Sofia BSD Mascot
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:StGeorgeRotundaSofia.JPG Preslav: Rotonda Sveti Georgi, Sophia (pd)
http://www.flickr.com/photos/70058529@N00/5169255238 Mihal Orela: Central Mineral Baths, Sofia (by)
http://commons.wikimedia.org/wiki/File:Sofia_University_panorama_2.jpg Plamen Agov: Sofia University (by-sa) studiolemontree.com
http://commons.wikimedia.org/wiki/File:NDK_Sofia_2012_PD_06.jpg Bin im Garten: National Palace of Culture, Sofia (by-sa)
http://www.flickr.com/photos/podoboq/75568649 Todor Bozhinov: Vitosha 2290m (by)
http://commons.wikimedia.org/wiki/File:Tumba_Belasica_IMG_7972.jpg Deyan Vasilev: Tumba 1880m, Belasica (by-sa)
https://www.flickr.com/photos/everexplore/8572686541 Kiril Rusev: Kutelo 2908m and Vihren 2914m, Pirin (by-nc-nd)
http://commons.wikimedia.org/wiki/File:Botev_Peak.jpg Gerovitus91: Botev 2376m, Stara Planina (by-sa)
https://www.flickr.com/photos/tom-margie/2070704728 Tom Margie (by-sa): still of Marty Feldman as Igor from ‘Young Frankenstein’;
fair use to illustrate the inspiration for the name of the program
http://commons.wikimedia.org/wiki/File:Maliovitsa_54072.jpg Stelian Kasabov: Maljovitsa 2729m, Rila (pd)
http://commons.wikimedia.org/wiki/File:Moa_mock_hunt.jpg Augustus Hamilton: Moa © expired
https://www.flickr.com/photos/eastsidephil/4452171897 Phil Davis: Squirrel digging (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)
https://www.flickr.com/photos/13564559@N06/1382926708 mattyhike: Musala 2925m, Rila (by-nc-sa)
http://www.flickr.com/photos/gazzat/3495392530 Gary Tanner: Foals Just Wanna Have Fun (by-na-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/66176388@N00/3436935367 Mark Robinson: A Youngster on the Quantocks (by-nc)
http://www.flickr.com/photos/kristavandervoorden/4737488285 Krista van der Voorden: New Forest Foal (CC)
http://www.flickr.com/photos/tambako/3578468294 Tambako: Very young zebra (by-nd)
http://www.flickr.com/photos/tiny_packages/5045038219 tiny_packages: On the road (by-nc-nd)
http://www.flickr.com/photos/jimmy-jay/4672901414 Jeff Burcher: Horse Fly Portrait (by-nc-nd)
https://www.flickr.com/photos/curiouskiwi/566823278 Brenda Anderson: Fantail (by-nc-sa)
https://www.flickr.com/photos/sidm/5225410158 Sid Mosdell: Red-crested parakeet (by)
http://commons.wikimedia.org/wiki/File:Little_Penguin_Feb09.jpg Fir0002/Flagstaffotos: Little Penguin (by-nc)
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)