Apropos Replacement
(Extended Abstract)

Abhinav Upadhyay <er.abhinav.upadhyay@gmail.com>
Joerg Sonnenberger <joerg@netbsd.org>

December 25, 2011

1 Abstract

Unix like operating systems have withstood the test of time for 50 odd years for several technical and philosophical reasons and one of those reasons is the quality documentation they ship in the form of man pages. They are a single point source of consultation for most of our routine work. But there has been a lack of good search tool to accompany this documentation which would increase their usefulness manifolds.

Traditionally *apropos(1)* [1] has been there to act as a search interface but to a large extent it hasn't really lived up to it's expectation. *apropos(1)* was developed in the early days of Unix when computing resources were scarce and therefore we believe it was the primary reason that it's design was kept so minimal and simple.

In this paper we discuss the limitations of the traditional implementation of *apropos(1)(1)*, how did we solve those limitations in our implementation and what other modern implementations are available out there our own implementation.

2 Introduction

The classical version of *apropos(1)* has been implemented by simply indexing the keywords in the NAME section of the man pages in a plain text file (whatis.db) [2] and performing searches on it. The reason for this simple design was most probably lack of computing resources in the early days. The plain text file consisting of keywords would hardly take few hundreds of kilo bytes of disk space and performing search on a plain text file is also not very hard.

This simplified design of *apropos(1)* resulted in limited utility and usability. The searches are limited to the keywords in the NAME section of the keywords. If only a user knows the exact keywords he gets the right results otherwise most of the times the searches are full of bogus results or a dead end. In the modern computing world where hard problems like searching the World Wide Web have been solved [3] to sufficient degree where most of the data is unstructured, then it makes perfect sense that we leverage the advancement in technology to solve the problem of an effective man page search tool. In present time we have powerful machines capable of running the search algorithms efficiently and disk space is cheap enough that we can afford to index additional metadata from man pages, which are a necessary prerequisite for an effective search tool.

3 Limitations of Conventional *apropos(1)*

In this section we will see results of some sample query searches from the classical version of *apropos(1)* and analyze its shortcomings.

3.1 Lack of support for free form queries

The conventional *apropos(1)* is limited to the keywords used in the NAME section of the man pages. If the user specifies keywords relevant to the page that he is looking but probably not used in it's NAME section, *apropos(1)* would not return anything relevant. So in essence the user cannot use free form queries like supported by modern search applications.

```
$ \textit{apropos(1)} ``add new user''
adding new user: nothing appropriate

$ \textit{apropos(1)} ``get process status''
get process status: nothing appropriate

$ \textit{apropos(1)} ``termcap database''
termcap database: nothing appropriate

$ \textit{apropos(1)} ``signal number to string''
signal number to string: nothing appropriate
```

Listing 1: No results for free form queries

3.2 Lack of basic language support

Another major limitation of the classical version of *apropos(1)* is that it has no natural language support. For example in the following listing we can see that while *apropos(1)* returns the correct result for the query *"make directories"* but it fails when we use the keyword *"directory"* in place of *"directories"* although the two words are based on same root word.

```
$ \textit{apropos(1)} ``make directories''
mkdir (1) - make directories
$ \textit{apropos(1)} ``make directory''
make directory: nothing appropriate

$ \textit{apropos(1)} "upgrading software package"
pkg_add (1) - a utility for installing and upgrading software
    package
distributions
$ \textit{apropos(1)} "upgrading software packages"
upgrading software packages: nothing appropriate
```

Listing 2: No support for stems or word with same roots

Similarly, it is very common for users to misspell keywords in a query and the classical *apropos(1)* has no support for it either.

```
$ \textit{apropos(1)} ``copy strings''
stpcpy, stpncpy, strcpy, strncpy (3) - copy strings
$ \textit{apropos(1)} ``coppy strings''
coppy strings: nothing appropriate
```

Listing 3: No spelling correction

3.3 Unintelligible Output

Because the way *apropos(1)* works, sometimes it's output can be unintelligible and it can be very hard to find the results which are relevant to the user.

```
$ \textit{apropos(1)} power
PCI, pci_activate, pci_bus_devorder, pci_chipset_tag_create,
pci_chipset_tag_destroy, pci_conf_read, pci_conf_write,
pci_conf_print, pci_conf_capture, pci_conf_restore,
    pci_find_device,
pci_get_capability, pci_mapreg_type, pci_mapreg_map,
    pci_mapreg_info,
pci_intr_map, pci_intr_string, pci_intr_evcnt,
    pci_intr_establish,
pci_intr_disestablish, pci_get_powerstate, pci_set_powerstate,
pci_vpd_read, pci_vpd_write, pci_make_tag, pci_decompose_tag,
pci_findvendor, pci_devinfo, PCI_VENDOR, PCI_PRODUCT,
    PCI_REVISION (9)
- Peripheral Component Interconnect
PMF, pmf_device_register, pmf_device_register1,
    pmf_device_deregister,
pmf_device_suspend, pmf_device_resume,
    pmf_device_recursive_suspend,
pmf_device_recursive_resume, pmf_device_resume_subtree,
pmf_class_network_register, pmf_class_input_register,
pmf_class_display_register, pmf_system_suspend,
    pmf_system_resume,
pmf_system_shutdown, pmf_event_register, pmf_event_deregister,
pmf_event_inject, pmf_set_platform, pmf_get_platform (9) - power
management and inter-driver messaging framework
acpi (4) - Advanced Configuration and Power Interface
acpipmtr (4) - ACPI Power Meter
amdpm (4) - AMD768 Power Management Controller and AMD8111
    System
```

```
Management Controller
...
.
.
.
```

Listing 4: Unintelligible output of *apropos(1)*

3.4 Other Problems

Apart from the search related problems there are a few issues related to the man pages are handled in NetBSD. The different aliases of the man pages are stored on the filesystem in the from of hard (or soft) links and these have to be mentioned in the makefiles explicitly using the MKLINK flag. This approach works fine but we feel it is a mess from maintenance point of view. When *apropos(1)* maintains an index of the man page metadata then it should be possible to fix this mess by utilizing that index. But yet again, the simplistic implementation of *apropos(1)* does not leave any room for improvement.

4 Proposed Solution

We propose a simple and straightforward solution. The idea is to parse and index the complete content (or at least most of the content) in the form of an inverted index 4 and use it to build a full text search interface. Having an index based on the complete content of the man pages solves many of the search related problems associated with the conventional *apropos(1)* as noted before and discussed as follows:

Free Form Queries
: The users can express their queries in more natural language form as the searches are no longer limited to the keywords defined in the NAME section of the man pages, the keywords used throughout the body of the page also play a role. For example queries like *"installing new software package"* will get you the results.

Basic Natural Language Processing Support
: When parsing the man pages and building the index, it is also possible to pre-process the tokens extracted from the man page to support some of the basic natural language processing functionalities. We have used stemming [5] to reduce the individual tokens extracted from the man

pages to their root words. This allows us to support more flexible searches. For example both *"Installing new packages"* and *"install new package"* will return same results.

Similarly along with the inverted index, we can also build a dictionary of the keywords frequently occurring in the corpus of man pages and use it to support spelling correction.

We have used both word stemming and spelling correction in our implementation to make the search experience more smooth.

Bookkeeping of man page metadata

In our implementation we have gone ahead and also indexed additional metadata related to the man pages, for example we are indexing an md5 hash of the man page, the device id, inode number and modification time of the man page file so as to support incremental updating of the index as new man pages are installed or the old ones are modified/updated.

Similarly we are also maintaining a separate index of all the aliases of a man page. This will allow us to get rid of all the hard or symbolic links of the man pages which are scattered throughout the filesystem, and also help in getting rid of the MKLINK flags in the makefiles.

4.1 Tools Used

The two main tasks that we needed to perform were parsing of the man pages and building an index of the data, and to do these tasks we have used *libmandoc* [6] and *sqlite* [7] in our implementation.

libmandoc

libmandoc is a library interface to a validating compiler. It provides interface to parse and build an AST (Abstract Syntax Tree) of the man page, and interface for traversing that tree in order to extract the data from nodes which are of our interest.

sqlite

sqlite is an embedded relational database management system providing a relatively small and easy to use C library interface. One of the main reason for choosing it over the myriad of other possible options was that it provides in built support for full text search through it's FTS virtual table module [8]. The FTS module can be accessed using pretty much standard SQL syntax, and it is still flexible enough to allow

us to implement our own ranking function as per our needs. Besides that we have been able to utilize the RDBMS support of *Sqlite* to store additional metadata in the form of normal database tables without any hassles.

5 Implementation Details

Due to space constraints we cannot go into enough implementation details in this abstract but we would like to talk a bit about some important components of this project:

makemandb(1)
: *makemandb(1)* [9] is the key component of this implementation. It traverses the filesystem, reads the raw man page source files, feeds them to them *libmandoc* and then stores the extracted data in the database using *sqlite*.

apropos(1)
: This is the version of *apropos(1)* written from scratch utilizing the full text search functionality of *sqlite*. Unlike the classical version of *apropos(1)* it only displays the top 10 results relevant to the user query and most of the times this is sufficient as we shall see later. Also, it shows a brief snippet for each of the search results, making it more easy to find out the relevant documents.

apropos-utils(3)
: *apropos-utils(3)* [10] is a small library interface provided with this implementation. It provides functions for querying the FTS index and processing the results in a user supplied callback function. It's main purpose is to develop different interfaces on top of this implementation for different use cases. For example we have built a small CGI application using it for doing the searches from a web browser, similarly we also built an IRC bot using this

6 Results

We would like to show how our implementation solves the problems cited earlier with the classical version of *apropos(1)*. [1]

[1] Although this implementation of *apropos(1)* returns 10 results in normal case for a query but in the following listings we have snipped the results to save space

```
$ \textit{apropos(1)} ``add new user''
ssh-add(1)      adds private key identities to the
    authentication agent
...on the command line. If any file requires a passphrase, ssh-
    add
asks for the passphrase from the user. The passphrase is read
    from the
user's tty. ssh-add retries the last passphrase if multiple
    identity
files are given...

chpass(1)       add or change user database information
add or change user database information

useradd(8)      add a user to the system
The useradd utility adds a user to the system, creating and
    populating
a home directory if necessary. Any skeleton files will be
    provided for
the new user if they exist in the skel-dir directory (see the k
option). Default...
.
.
.
```

Listing 5: Add new user

```
$ \textit{apropos(1)} ``make directory''
make(1) maintain program dependencies
...CURDIR A path to the directory where make was executed. Refer
     to
the description of PWD for more details. MAKE The name that make
     was
executed with argv[0] . For compatibility make also sets .MAKE
    with
the same value. The...

mkdir(1)        make directories
make directories

ln(1)   make links
...a directory in which to place the link; otherwise it is
    placed in
the current directory. If only the directory is specified, the
    link
will be made to the last component of source_file . Given more
    than
two arguments, ln makes...
```

```
mkfifo(1)        make fifos
make fifos ... of a=rw mkfifo requires write permission in the
    parent
directory. mkfifo exits 0 if successful, and >0 if an...

mkdir(2)         make a directory file
... will contain the directory has been exhausted. EDQUOT The
    user's
quota of inodes on the file system on which the directory is
    being
created has been exhausted. EIO An I/O error occurred while
    making the
directory entry or...
```

Listing 6: make directory

```
$ \textit{apropos(1)} ``get process status''
ps(1)    process status
process status

netstat(1)       show network status
... use the fstat(1) command to find out which process or
    processes
hold references to a socket. The interface display ... show
    network
status

bounce(8)        Postfix delivery status reports
... the master(8) process manager.  The bounce(8) daemon
processes two types of service requests: Append a recipient
(non-)delivery status record to a per message log file. Enqueue
    a
delivery status notification message, with a...

ocsp(1) Online Certificate Status Protocol utility
Online Certificate Status Protocol utility ... it is in client
    mode. The
request(s) the responder processes can be either specified on
    the
command line (using...

fstat(1)         display status of open files
... the owner of the process (effective UID). CMD The command
    name of
the process. PID The process ID. FD The... display status of open
     files
```

Listing 7: get process status

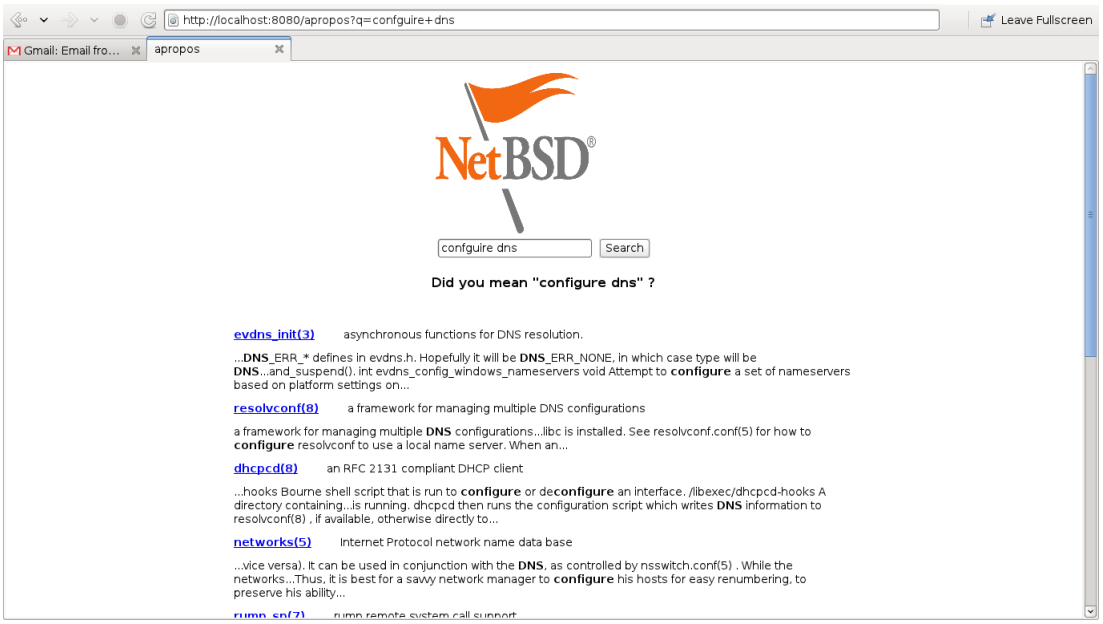

Figure 1: The spell corrector and the web interface in action

7 Related Work

There are at least two projects which are related to this in some way.

man-db

man-db [11] is a complete implementation of the man page documentation system and it is used on a number of GNU/Linux distributions. *man-db* takes an interesting approach for indexing the man page data. Unlike the classical *apropos(1)*, it uses a Berkley DB database but still it's index is limited to the NAME section only. It adds an option 'K' to man(1) to allow a crude full text search but it is not very efficient nor effective.

mandocdb

mandocdb [12] is more in line with the goals of our project but it takes a novel approach. It indexes the keywords extracted from the man pages in a key-value store using btree(3) [13]. It also comes with it's own implementation of *apropos(1)* which performs search using this key-value store. The main key point of this implementation is that it exploits the semantic structure of the man pages.

8 Future Work

Work on Ranking Algorithm

We have used a ranking algorithm based on probabilistic model [14] of information retrieval. It is essentially based on the Okapi BM25F algorithm [15] and uses certain parameters whose values are usually dependent on the corpus and the search application. For example we have used certain weights for different sections of man pages. At the moment these values have been determined manually but we would like to use some machine learning techniques in the coming days to automate this.

Fix Some Loose Ends In Parsing of man pages

Although the parsing routine is working fine for most purposes on the man pages supplied with NetBSD and also on most of the man pages found in pkgsrc. But there are perhaps a few corner cases still to be fixed. Besides that another issue that is to be addressed is the parsing of the escape sequences which are used extensively in the *man(7)* [16] based man pages. We would like to address these issues in the coming days.

9 Acknowledgement

This project was developed as part of **Google Summer of Code 2011** [17], so thanks to Google for sponsoring it. We would also like to thank **Kristaps Dzonsons** who is the developer of the *mdocml* [18] project, he also helped by pointing out several issues in the parsing related code. We would also like to thank **David Young** who was involved with this project closely and offered useful help and guidance. A special thanks goes to **Thomas Klausner** who helped in writing and reviewing our man pages. Thanks to Petra Zedlier for administering the GSoC program for *The NetBSD Foundation*.

References

[1] *NetBSD manual page for apropos(1)*
http://netbsd.gw.com/cgi-bin/man-cgi?apropos++NetBSD-5.1

[2] *NetBSD manual page for makewhatis(1)*
http://netbsd.gw.com/cgi-bin/man-cgi?makewhatis++NetBSD-5.1

[3] Brin, S.; Page L. *The Anatomy of a Large-Scale Hypertextual Web Search Engine* Computer Networks and ISDN Systems 30:107-117, 1998

[4] Manning; Raghwan; Schutze *Introduction to information retrieval*, 3-9, 2008

[5] Frakes; Baeza-Yates *Information retrieval data structures & algorithms*, 131-161, 1992

[6] *mdocml online manual page for libmandoc* http://mdocml.bsd.lv/mandoc.3.html

[7] *Sqlite home page* http://sqlite.org

[8] *Sqlite FTS3 and FTS4 Extensions* http://sqlite.org/fts3.html

[9] *Online manual page for makemandb(1)* http://netbsd-soc.sourceforge.net/projects/apropos_replacement/makemandb.html1

[10] *Online manual page for apropos-utils(3)* http://netbsd-soc.sourceforge.net/projects/apropos_replacement/apropos-utils.html3

[11] *man-db, the on-line manual database* http://man-db.nongnu.org/

[12] *Online manual page for mandocdb(1)* http://mdocml.bsd.lv/mandocdb.8.html

[13] *NetBSD online manual page for btree(3)* http://netbsd.gw.com/cgi-bin/man-cgi?btree+3+NetBSD-5.1

[14] Fuhr, Norbert *Probabilistic models in information retrieval* The Computer Journal 1992

[15] Zaragoza H.; Craswell N.; Taylor M.; Saria S.; Robertson S. *Microsoft Cambridge at TREC13: Web and HARD tracks* In proceedings of TREC-2004

[16] *Online manual page for man(7)* http://mdocml.bsd.lv/man.7.html

[17] *Google Summer of Code home page* http://code.google.com/soc/

[18] *The mdocml project home page*
http://mdocml.bsd.lv