Apropos Replacement: Development of a full text search tool for man pages (Initial Outline)

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

December 26, 2011

1 Abstract

Unix like operating systems have withstood the test of time for 40 odd years for several technical and philosophical reasons. One of those reasons is the quality documentation they ship in the form of man pages, which are a single source of consultation for most of our routine work. However, so far, there has been a lack of good search tool to accompany this documentation, which would increase their usefulness manifolds.

Traditionally, *apropos* [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* was developed in the early days of Unix when computing resources were scarce and that is the primary reason for its minimalistic and simple design. The biggest limitation of *apropos* is it's limited search capabilities besides man others.

We worked on fixing this by writing a new implementation of *apropos* As part of *Google Summer of Code 2011* [17]. The main goal of this project was the development of a full text search tool for man pages.

In this paper, the limitations of the traditional implementation of apropos are discussed along with brief details of how these were overcome as part of this project. Additionally, we compare and contrast our approach with other modern implementations available out there.

2 Introduction

The classical version of *apropos* 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 hardly takes few hundreds of kilo bytes of disk space and performing search on a plain text file is quite easy.

This simplified design of *apropos* resulted in limited utility and usability as the searches are limited to the keywords defined in the NAME section of the man pages. Only if a user knows the exact keywords do they get the appropriate results; otherwise most of the times the searches are full of irrelevant results or possibly a dead end. In the modern computing world where hard problems like searching the World Wide Web have been solved [3] to a sufficient degree, it makes perfect sense to leverage the advancement in technology in order to solve the problem. This problem is made even simpler considering that the man pages consist of structured data.

In present times, the machines are powerful and capable of running the

search algorithms efficiently. Disk space is also sufficiently cheap that any extra space incurred by indexing of additional metadata from man pages can be easily afforded. These are necessary prerequisites for building an effective search tool.

3 Limitations of Conventional *apropos*

In this section results of some sample query searches from the classical version of *apropos* are shown and their shortcomings are analyzed.

3.1 Lack of support for free form queries

As noted earlier the conventional *apropos* is limited to the keywords used in the NAME section of the man pages. This means that the users do not have a whole lot of choice for specifying keywords in their queries. Besides they cannot search for keywords which are not usually specified in the title of a man page, for example a query like "EINVAL" would return no results.

```
$ apropos ``add new user''
add new user: nothing appropriate

$ apropos ``get process status''
get process status: nothing appropriate

$ apropos ``termcap database''
termcap database: nothing appropriate

$ apropos ``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* is that it is not smart enough to provide basic natural language processing constructs like stemming and spelling corrections. For example in the following listing it can be seen that while *apropos* returns the correct result for the query *"make directories"*, it fails when the keyword *"directory"* is used in place of *"directories"*, even though the two words are based on same root word.

```
$ apropos ``make directories''
mkdir (1) - make directories
$ apropos ``make directory''
```

```
make directory: nothing appropriate

$ apropos "upgrading software package"
pkg_add (1) - a utility for installing and upgrading software
    package
distributions
$ apropos "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 *apropos* has no support for it either.

```
$ apropos ``copy strings''
stpcpy, stpncpy, strcpy, strncpy (3) - copy strings
$ apropos ``coppy strings''
coppy strings: nothing appropriate
```

Listing 3: No spelling correction

3.3 Unintelligible Output

Because of the way *apropos* works, sometimes it's output can be unintelligible and it can be very hard for the user to identify relevant results.

```
$ \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 way 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 MLINKS mechanism. This approach works fine but it is a mess from maintenance point of view. It should be possible to fix this by utilizing the index already built and maintained by *apropos*, but yet again, the simplistic implementation of *apropos* does not leave any room for improvement.

4 Proposed Solution

A very simple and straightforward solution is proposed to solve these problems as part of this project. 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 the NAME section. For example, queries like *"installing new software package"* now produce relevant 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 pages to support some of the very basic natural language processing functionalities. In this implementation, the Porter stemming algorithm [5] has been used to reduce the individual tokens extracted from the man pages to their root words. This enables support for more flexible searches. For example both *"Installing new packages"* and *"install new package"* will return same results.

Similarly along with the inverted index, a dictionary of the keywords frequently occurring in the corpus of man pages is also built and used to support spelling suggestion.

Bookkeeping of man page metadata

In this implementation, additional metadata related to the man pages, for example an md5 hash, the device id, inode number and modification time of the man page files are indexed and stored so as to support fast and hassle free update of the index as new man pages are installed or the old ones are modified/updated.

Similarly a separate index of all the man page aliases is stored and maintained. This provides an option to get rid of all the hard or symbolic links of the man pages scattered throughout the filesystem, and also for clean up of the MLINKS mess in the makefiles.

4.1 Tools Used

The two main operations that are critical for this project are parsing of the man pages and building of an index of the data obtained from the parsing process. *libmandoc* [6] and *sqlite* [7] have been used for executing these the two tasks.

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. It also provides an 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 reasons for choosing it over the myriad of other possible options is 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 accept user supplied ranking function to suit the needs of the application. Besides that, another advantage of *sqlite* is the RDBMS support, which makes it very easy to store additional metadata in the form of normal database tables without any hassles.

5 Implementation Details

Due to space constraints it is not possible to go into enough implementation details, the most important components of this project are described in brief in this section.

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

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

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

6 Results

This section shows results of some of the sample queries on the version of developed as part of this project *apropos* to demonstrate how it solves many

of the problems cited earlier with the classical version of *apropos*. [1]

```
$ apropos ``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

```
$ apropos ``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
```

[1] Although this implementation of *apropos(1)* returns 10 results in normal cases for a query but in the following listings the output has been snipped to save space

```
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

```
$ apropos ``signal number to string''
psignal(3)        system signal messages
...the signal number is not recognized sigaction(2) , the string
Unknown signal is produced. The psiginfo function produces the
    same
output as the psignal function, only it uses the signal number
information from the si argument. The message strings can...

intro(2)          introduction to system calls and error numbers
...undefined signal to a signal(3) or kill(2) function). 23
    ENFILE Too
many open files in system . Maximum number...shell. Pathname A
    path
name is a NUL -terminated character string starting with an
    optional
slash \&/ , followed by zero or...

groff_mdoc(7)     reference for groff's mdoc implementation
...Qq .Qq string ) , string ) , .Qq string Ns ), string ), .Sq .
    Sq
string string .Em or...UNTITLED is used. The section number may
    be a
number in the range 1...2, 3 and 9 error \&.\e"      and signal
handling only. \&.\e" .Sh ERRORS \&.\e...
```

Listing 7: signal number to string

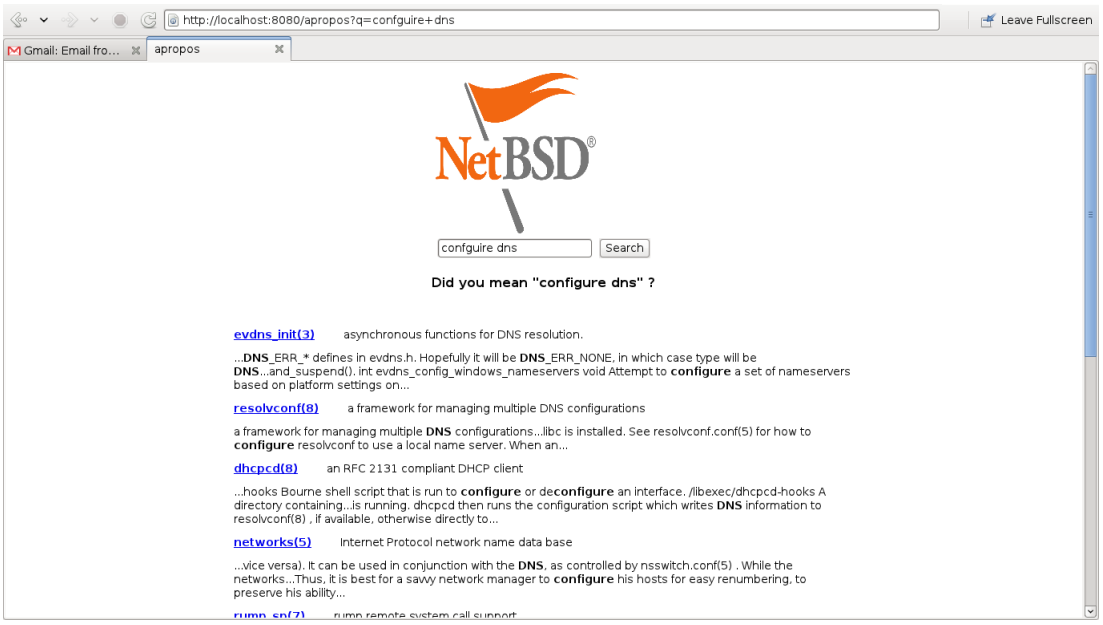

Figure 1: The spell corrector and the web interface in action

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*, 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 this 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* 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

A ranking algorithm based on probabilistic model [14] of information retrieval has been implemented to filter out the most relevant results and show them at the top. 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 certain weight parameters are assigned to different sections of man pages. At the moment these values have been determined manually but one goal is to use some supervised machine learning techniques in the order 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. These issues are to be addressed in the coming days.

9 Availability

The code for this project is hosted on *Github* and it is under active development. It can be obtained from: *https://github.com/abhinav-upadhyay/apropos_replacement*

It can

10 Acknowledgement

This project has been developed as part of *Google Summer of Code 2011* [17], so thanks to **Google** for sponsoring it. Special thanks to **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 like to thank **David Young** who was involved with this project closely and offered useful help and guidance throughout. A special thanks goes to **Thomas Klausner** who helped in writing and reviewing man pages for this project. Thanks to **Petra Zeidler** 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] Porter, M. F. *An algorithm for suffix stripping*, Program, 14(3): 130-137, 1980

[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