Desktop Search Engine for Linux

by

Ahmad Razif bin Musa@Mahmud

supervised by

Mr Justin Dinesh Devaraj

A project dissertation submitted in a partial fulfillment of

the requirement for the

Bachelor of Technology (Hons)

(Business Information Systems)

July 2005/Jan 2006

Universiti Teknologi PETRONAS
Bandar Seri Iskandar
31750 Tronoh
Perak Darul Ridzuan

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

CERTIFICATION OF APPROVAL

Desktop Search Engine for Linux

by

Ahmad Razif bin Musa@Mahmud

A project dissertation submitted in a partial fulfillment of

the requirement for the

Bachelor of Technology (Hons)

(Business Information Systems)

Approved by,

(Mr Justin Dinesh Devaraj)

UNIVERSITI TEKNOLOGI PETRONAS

TRONOH, PERAK

JULY 2005/JAN 2006

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and the acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.

(AHMAD RAZIF BIN MUSA@MAHMUD

ABSTRACT

Desktop Search Engine become more popular for personal and enterprise after some difficulties occurs when dealing with the huge amount of files and in a multi-user environment. Desktop Search Engine for Linux is a desktop search tool, integrated between Namazu with web-based interface that can search text format files in the hard drive of personal computer. As increasingly demand for an effective and efficient desktop search tools especially for the Linux environment where there were just a few tools have been developed for Linux compared for Windows although the usage of Linux operating system are increased from days to days. This system is just for Linux (Debian platform) operating system and just search for a text format files. This system index the entire words of files in the hard drive and create one index files that contains all details about the files in the disk. The system just refers to this index files when processing the searching process for a fast and effective results. From the studies and analysis that has been done during the development of this system, there have a benchmark criterion for desktop search tools that can be use as a reference and also a lot of indexer that can be used to index the files. Only the best indexer was be taken to integrate with this system. This system still can be improved with the support, effort and deep knowledge about desktop search tools and technical skills.

ACKNOWLEDGEMENT

For the mutual understanding, motivation and co-operative action the author would like to thank to supervisor Mr Justin Dinesh Devaraj and co-supervisor, Mr Lo Hai Hiung who were always encourage and guide the author as their student in completing this project. In addition, the author would like to acknowledge to the Final Year Project Board that was accepting this project in the early stage of the proposal of this project and also to Mr Albert that is an external examiner that has evaluating and gave good comments during final presentation.

The author would wish to thank to UTP IT lecturers that were evaluate and give a support with the good comments from the presentation. The wisdom that has been bestowed will never be forgotten. Besides, thanks to Mr Rasky, FYP lab technician who lend his hand in helping the testing process of the Linux operating system on the FYP lab during the project development. Also a special thanks to Mohd Hakim for their guide and support of JavaScript coding, Ahmad Fikri for his logical understanding help, Wan Anas for his comment about usability and as a tester for this system, Firdaus Tan for the support of interface.

Last but not least, a million thanks to the MIMOS Open Source Software staffs that give a support and idea for this project and also for introducing me to the best ever operating system, Linux. Besides, thanks to FYP GIS group student and all colleagues that support and give the author a spirit to finish this Desktop Search Engine for Linux.

CHAPTER 1

INTRODUCTION

1.1 BACKGROUND STUDY

The capacity of our PC hard drives has increased tremendously over the past decade, and so has the number of files we usually store in our personal computer. It is no wonder that sometimes we cannot find a document any more; even when we know we saved it somewhere. Ironically, in quite a few of these cases nowadays, the document we are looking for can be found faster on the World Wide Web search than our personal computer.

Would not it be great if a computer could search all files that you want faster and accurate without knowing the file names? If, for instance, it would anticipate what files you need, search for it, and then automatically deliver the right files that contain the relevant information. You'd never have to guess the right keywords of filename or open every folder in the drive and open it one by one using a try and error method just to find a required file. That will waste your time for just doing the search method manually.

Desktop Search Engine for Linux is a desktop search tool that can search text format files in the hard drive of personal computer. This system is an integrated system of Namazu Desktop Search tool with the web-base interface. This system is an improvement of Namazu Desktop Search tools. The description of integration will be discussed in methodology part of this report.

1.2 PROBLEM STATEMENT

1.2.1 Problem Identification

As the increase of hard drive size, there will be more difficult to find the files if we do not know the location of files and the actual name of that files. We can just enter the name of file format such as .doc, .sxw, etc in the search tools that provide by the current operating system but the problem now there were so many file that have the same format. Another alternative is by open one by one folder in the drives to find the required files. That approach is not effective and efficient and will waste time. Below are other problems that related to the searching files in the hard drive that usually occurs to the users:

1.2.1.1 Difficult to find specific files in the hard drive

People are willing to find a file faster and accurately. If the users must waiting for the system a few minutes to completely search a required file, it will waste the time especially when they are in urgent. The old system must explore one by one folder in the hard drive to find a required file. The big problems occurred when the users have a large size of hard drives and contain many partitions and folders inside.

1.2.1.2 Difficult to recall the name of files saved in the hard drive

There are worst thing if we do not remember the name of a file especially when in the urgent time and it's an important file. This is always happen to the users that have a short-term memory.

1.2.1.3 Misplaced files in the hard drive

Users will loss the location of files if other person was change the location (cut and paste files to other partition or folder) of a certain file or they cannot remember the location of files when some changes has been done by them. That case makes users difficult to recall or find back the file they want.

1.3 SIGNIFICANT OF THE PROJECT

Desktop Search Engine for Linux hopefully can ease users to find the required file by just key in the related keywords in provided textbox on the interface of the system. So, users no longer need to search files manually using operating system search files that required users to wait for a few minutes to see the results. The results that displayed by the operating system search tools are in the form of locations and it is difficult for users to find the files by just giving the address of files.

Besides, users also can find the files although they cannot remember the name of the files. That means, although users save the files using any file name (but in the text file formats) and forgets about the name of files they saved, they still can trace that files as long they remember the content of the files. In this world, users will know although one word in the files that they want to search so to search the files that just require strings/words that contain in the files is not possible.

Desktop Search Engine for Linux also displays all possible files and words that have been filling as a result to the files searched. This system will display all possible files that contain the files that have the required words/strings. Besides display the possible files name, this system also display a short phrase of file content on the screen to make sure users can identify the required files easily.

1.4 OBJECTIVE

1.4.1 Objectives of the project

i. To develop a desktop search engine for Linux user to find any text format files in personal computer, no matter what type of text files format and information they stored.
ii. To integrate Namazu with web-based environment to ease the users to use the systems.
iii. To compare the performance of desktop search tools in searching files using different tools of desktop search.

1.5 SCOPE OF STUDY

1.5.1 Personal computer hard drive

Desktop Search Engine for Linux developed just for searching files within a personal computer hard drive. It can search all files in the hard drive faster and efficient. Desktop Search Engine for Linux will create an index (play as a database) of all files in the hard drive and the querying and matching process will refer to the data contains in that index. This system cannot search the others files via network or outside the personal computer hard drive.

1.5.2 Linux (Debian platform) Operating System

Linux is an operating system that is growing in popularity. More and more businesses are recognizing the possibilities absolute customization can provide. The open source code gives Linux an edge that just can't be obtained from a corporate proprietary program. Indeed, Linux has an edge over anything else on the market simply because these corporations are trying to please everyone all the time which

just can't be done. The open source code in Linux that can be change to anything desired is something that closed source code can never have. Desktop Search Engine for Linux developed under Linux Debian platform environment. This system cannot operate in other than this operating system platform.

1.5.3 Search for text format files

Desktop Search Engine for Linux is a desktop search engine that can just search any text format files in the hard drive. It is just limit to the search process of text files that mean not search other that text files format. Below is the text files format that can search by this system:

- OpenOffice file
    - .sxw
    - .sxi
    - etc
- Microsoft Words files
    - .doc
    - .ppt
    - .xls
    - etc
- .html
- .shtml
- .htm
- .css
- .sql
- .odg
- .diz
- .zip
- .pdf
- .java
- .js
- .mf
- .php
- .txt

CHAPTER 2

LITERATURE REVIEW

2.1 DESKTOP SEARCH ENGINE

Nowadays desktop search engine become more popular in the search engine industry. Many big IT or Web organization are running into producing the best desktop search engine such as Google with Google Desktop Search, Yahoo! with Yahoo! Desktop Search, MSN with MSN Toolbar Suite and so on. Desktop search engine is an application used to find data in a user's local system. In order to provide quick access to files that contain certain text no matter the format, desktop search programs index the content on the hard disk. All desktop search programs provide a search for Microsoft Office files such as Word, Excel, PowerPoint and Outlook. Some programs support non-Microsoft formats as well as locally stored Web pages.

The recent arrival of desktop search applications, which index all data on a personal computer, promises to increase search efficiency on the desktop [1]. The index method can search the required files faster when the system does not have to search folder by folder in the hard drive. Desktop search features built into current operating systems, e-mail programs, and other applications have far fewer capabilities than Web search engines. They generally offer only simple keyword searched of a set of files, usually of a single file type [2]. On the Web, search engines can exploit information organized into a common HTML format with standardized ways of identifying various document elements. The engines can use this information, along with links to other documents, to make statistical guesses that increase the likelihood of returning relevant results.

In the current approach to Knowledge Management and enterprise search, capturing the structured data in an enterprise or government agency is a complicated and difficult task since information resides in a variety of formats, systems, and locations [6]. That mean, it requires an efficient and powerful search tool to solve all this matters. Incorporates semantics is a approaches use by Beagle Desktop Search where it uses explicit information, such as file size, creator, last modification date, metadata embedded into specific files[11]. This approach can be use as an advance for the metadata search in a system. The systems that provide an advance features can have a good demand for enterprise usage.

The success of search and retrieval applications deployed in enterprises can be limited by their ability to process unstructured business documents that represent as much as 80% of an enterprise's information [8]. In an effort to help understand the differences between the latest desktop search tools on the market, the UW E-Business Consortium recently conducted a benchmark study of 12 popular desktop search tools. The benchmark criteria that were used for the evaluation included usability, versatility, accuracy, efficiency, security, and enterprise readiness [4].

A new generation of desktop search tools is emerging that allows users to quickly find relevant documents in computers across the enterprise the same way search engines help locate information on the internet. Companies expect that this technology will boost employee productivity and creativity and allow them to compete successfully in today's knowledge-driven economy [6]. In an effort to help understand the differences between the latest desktop search tools on the internet, the UW E-Business Consortium was conducting a benchmark study of 12 popular desktop search tools. The benchmark criteria that were used for the evaluation included usability, versatility, efficiency, security, and enterprise readiness.

Table 1 shows the benchmark criteria that was perform by UW E-Business Consortium [4] for desktop search tools:

Table 1: Benchmark Criteria for Desktop Search Tools

Usability	Versatility
Good desktop search tools must be easy to use, have a lower learning curve, have professional aesthetics, and require fewer steps to reach desired output.	Versatility describes how wide and deep the tool allows you to search. This includes factors such as supported document types, we/e-mail integration, and multi-language support.
Accuracy	**Efficiency**
"Can you find what you are looking for?" This criterion addresses accuracy of search results as well as other factors that help users find the desired information.	This criterion assesses the tool's technically efficiency including memory usage, indexing time or indexed file sizes. The best tool should not jeopardized overall PC performance.
Security	**Enterprise Readiness**
Security and privacy are big concerns, especially in an enterprise environment. This criterion considers how well vendors have incorporates security mechanisms.	While most tools are designed for the consumer/home PC environment, some are ready to be used in an enterprise. This criterion may be especially helpful for IT managers.

Each criterion was quantified and was given a rating, ranging from 1 (worst) to 5 (best). The rating is based on sub criteria, which align with the main criterion's

objective. The Desktop Search Engine for Linux that have been developed used this benchmark criterion as a guideline to make sure this system still in the right track.

2.2 SEARCH ENGINE (Namazu INDEXER)

Desktop search tool required an index to make sure the system can search require file faster and accurate. Indexer is an important component to generate an index. Namazu indexer is a full-text search engine indexer intended for easy use. For searching a great amount of document quickly, Namazu makes an index in advance. The concept of index is just similar to an index of book [13]. The language that used by this software is Perl and C. This indexer only can index local files that are not including files in the networks.

This software has been chosen because this it is easy to use and support Debian GNU/Linux, suit with an operating that been used that is Ubuntu 5.10. The reason of using a freeware indexer rather than develop the new indexer was about the time consuming and limited expertise to develop the own indexer. Namazu indexer is a free software that is distributed via internet and users can redistribute or modify it under the term of the GNU General Public License a as published by the Free Software Foundation.

Namazu is being developed by Namazu Project. Filters enable Namazu to index various formats of files. Mail/News filter works with no additives, some other type requires third partie's filter executable although the calling capabilities included in Namazu package.

2.3 INFORMATION RETRIEVAL

Information retrieval (IR) is the art and science of searching for information in documents, searching for documents themselves, searching for metadata which describe documents, or searching within databases, whether relational stand-alone databases or hypertext networked databases such as the Internet or intranets, for text, sound, images or data. There is a common confusion, however, between data retrieval, document retrieval, information retrieval, and text retrieval, and each of

these have their own bodies of literature, theory, praxis and technologies [16]. Automated information retrieval (IR) systems were originally used to manage information explosion in scientific literature in the last few decades. Many universities and public libraries use IR systems to provide access to books, journals, and other documents. IR systems are often related to object and query. Queries are formal statements of information needs that are put to an IR system by the user. An object is an entity which keeps or stores information in a database. User queries are matched to documents stored in a database. A document is, therefore, a data object. Often the documents themselves are not kept or stored directly in the IR system, but are instead represented in the system by document surrogates.

Methods to support dynamically changing text collections can be divided into two categories: Support for document insertions and support for document deletions. Techniques to support document insertions into an existing index have been studied by many researchers over the last decade. Most of them follow the same basic scheme. They maintain both an on-disk and an in-memory index. Postings for new documents are accumulated in main memory until it is exhausted, and then the data in memory are somehow combined with the on-disk index. Tomasic et al. present an in-place update scheme for inverted files, based on a distinction between short lists and long lists. They also discuss how different allocation strategies for the long lists affect index maintenance and query processing performance. Lester et al. give an evaluation of three different methods to combine the in-memory information with the on-disk data. Kabra et al. present a hybrid IR/DB system with delayed update operations through in-memory buffers. All of these solutions have in common that the entire on-disk index has to be read (or written) every time main memory is exhausted, which causes performance problems for large collections. We show how the number of disk operations can be significantly reduced, at minimal cost for query performance.

In contrast to the case of document insertions, a thorough evaluation of techniques for document deletions is not available. Chiueh and Huang present a lazy invalidation approach that keeps an in-memory list of all deleted documents and performs a post-processing step for every query, taking the contents of that list into account. The approach to document deletions presented in this paper is similar to theirs, but more general, and is not done as a post-processing step, but integrated into the actual query processing.

None of this related work provides a general discussion of how different index maintenance strategies affect query processing performance and how this implies opportunities for indexing versus query processing performance trade-offs.

The Wumpus Search System [12]

Wumpus is similar to other file system search engines, such as Google Desktop Search2, Apple Spotlight3, or Beagle4. Unlike most desktop search systems (except Spotlight), it is a true multi-user search system; only a single index is used for all files in the file system, and security restrictions are applied at query time in order to guarantee that the query results are consistent with all file permissions.

File system search is different from the traditional information retrieval task. The search engine not only has to deal with a large heterogeneous document collection, but a file system is also a truly dynamic environment: files are constantly created, modified, and deleted. The expected number of index update operations is much greater than the number of queries to be processed. Using Wumpus, one of the authors counted more than 4,000 index update operations (document insertions and deletions) on his laptop computer during a typical work day.

Furthermore, when an e-mail arrives, or a new file is created, the user expects the search system to reflect this change immediately. Delays greater than a few seconds are not acceptable. This, together with the great number of update operations that have to be performed, suggest that indexing performance plays a much greater role than query processing performance in this particular domain. Wumpus supports fast instantaneous updates (i.e., changes to the file system are reflected by the search system within fractions of a second).

In addition to being a dynamic environment, file system search is a multi-user application. In order to avoid wasting disk space due to indexing the same file many times, a single index has to be used for all users in the system. Special care has then to be taken so as to guarantee file system security.

2.4 LINUX

Nowadays, the usage of Linux as an operating system is tremendously increased with the support from government and from awareness of publics. Linux is a computer operating system and its kernel. It is one of the most prominent examples of free software and of open-source development; unlike proprietary operating systems such as Windows, all of its underlying source code is available to the public for anyone to freely use, modify, improve and redistribute [15]. The freedom of use, modify and redistribute the source of system make many people interested in joining the Linux and open-source community.

CHAPTER 3

METHODOLOGY

3.1 PROJECT FRAMEWORK

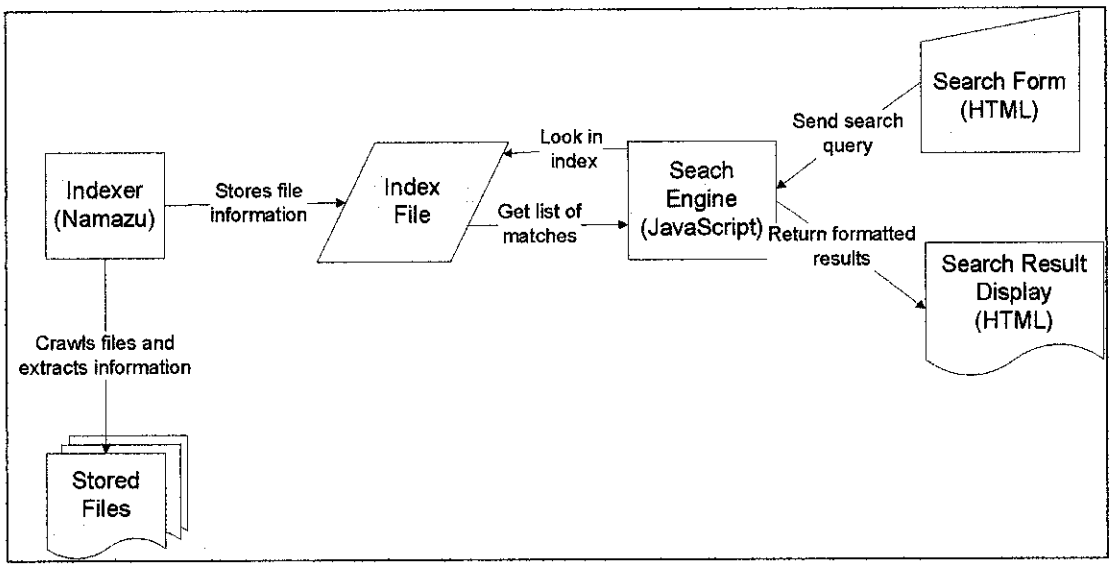

Figure 1: Project Framework

Figure 1 show, the system use an indexer to create an index files; their location on a hard drive's hierarchical tree file structure; file names, types, and keywords. Once existing files are indexed, the indexer indexes new documents in real time. The indexer also collects metadata, which let the engine access files more intelligently by providing additional search parameters.

When a user fills out a search form and sends a query, the engine searches the index, identifies the appropriate files, finds their locations on the drive, and displays the results. During searches, the engine matches queries to indexed items to find relevant files faster. The result will be display on the screen appropriately. The

users will give options whether to display the normal result or the result with the descriptions.

First, users will choose the type of files they want to search whether it's a documents files, images files, audio sound etc. The search form also provides the textbox that required the user to fill in the keywords of the related files they want to search. Users will give options whether to use a default search or an advance search that will search in an advanced.

3.1.1 Data Flow Diagram (DFD)

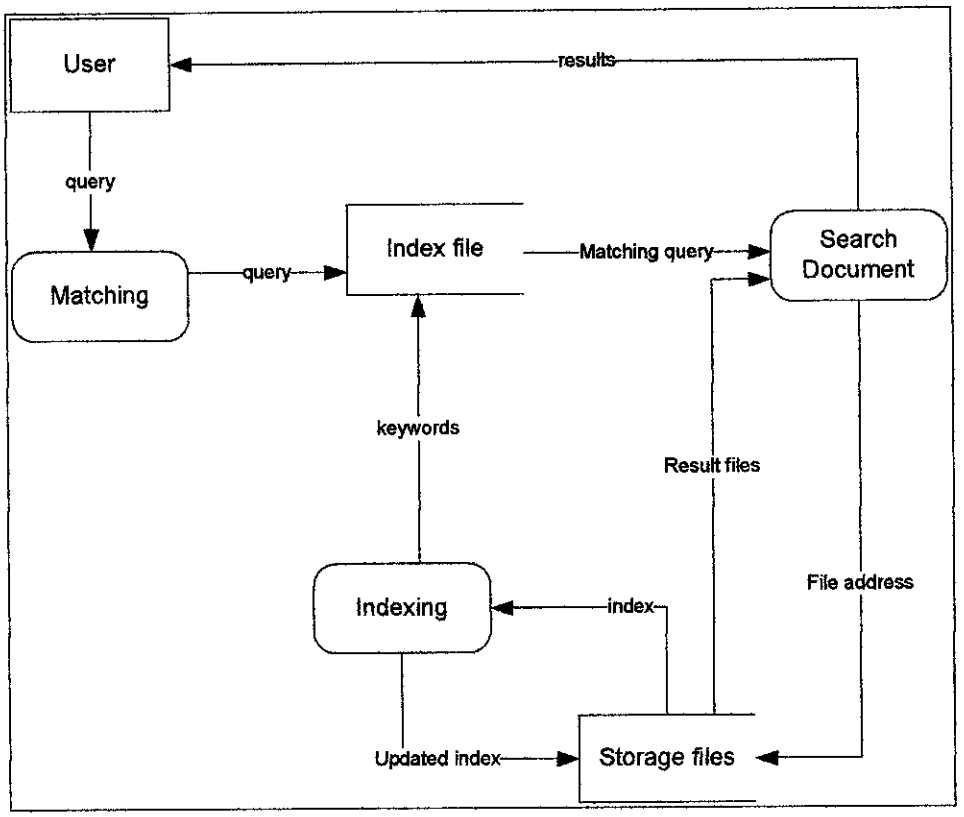

Figure 2: Data Flow Diagram

3.1.2 Use Case Model

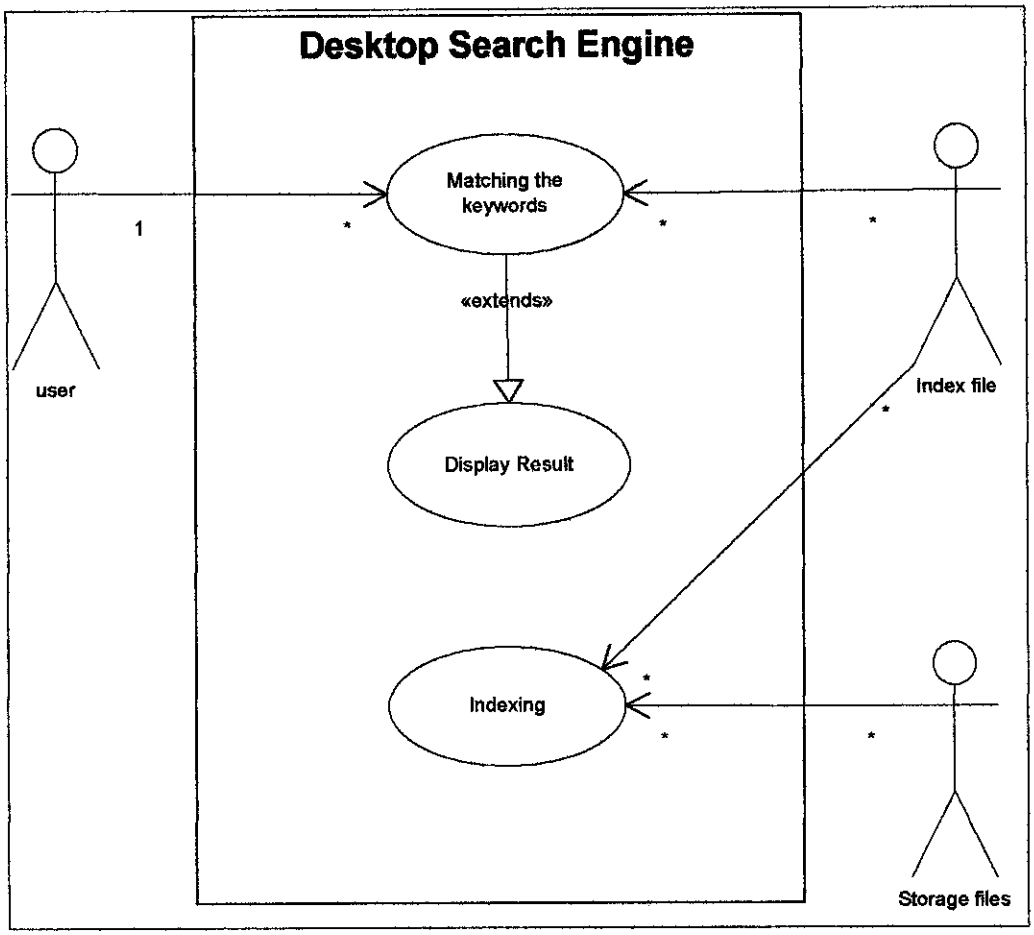

Figure 3: Use Case Diagram

Description:

1. Indexing index the storage files into an index file.
2. Indexing updates the updated files into an index file.
3. Index file store an index as a database of keywords.
4. User types required keywords to search in the system.
5. The keywords entered by users will be match with the keywords stored in the index file.
6. The system will display the related files that contain the keyword entered by user.

3.1.3 Flow of indexing the files (Namazu)

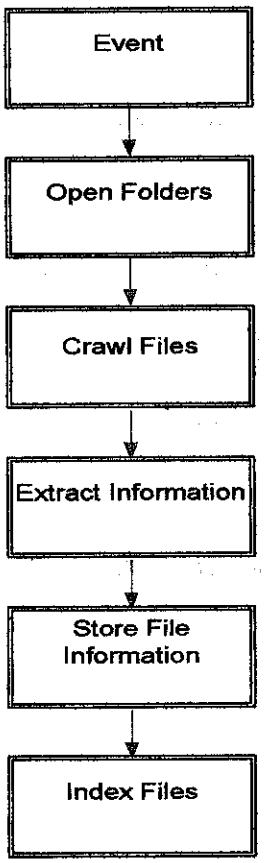

Figure 4: Flow of process indexing the files

Figure 4 shows the flow of the Indexer indexing the files. Firstly, an indexer will open and enter a required folders or partition and crawls files with the text type of files. The indexer will open the folder until the last folder one by one and extract the file information. File information will be store into an index files. The information that crawls by an indexer is filename, author, location (URI), date, words inside files, and date of modification and so on.

3.1.4 Flow of searching the files using JavaScript

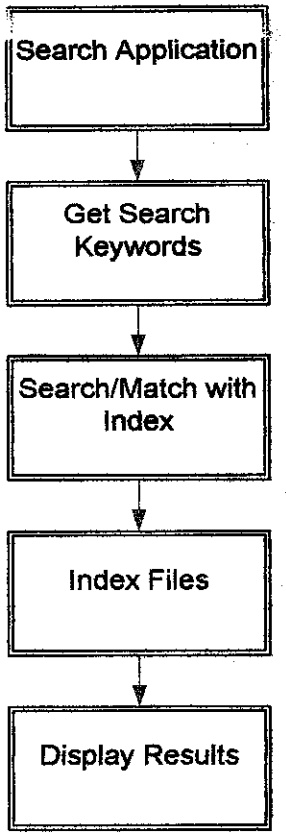

Figure 5: Flow of searching the files

Figure 5 shows a flow of searching the files. A search application that is a Desktop Search Engine for Linux will require a user to fill the required query that consist of related word(s) that contain in the required files. After users key in the word(s), the search engine will get the word(s) and search and match it with the words in the index files. After finish the process of matching keywords, the system will display the results that are related files that consist of keywords that have been filled by the user.

3.2 INTEGRATION OF NAMAZU WITH THE WEB-BASED INTERFACE

Namazu is an indexer that provides index and searching the related files. It is an internal process that makes users difficult to interact with the system. A convenience and user friendly interface can ease the users to interact with the system. Namazu is not a user friendly application that makes a normal user quite difficult to use. Namazu is working in the terminal/console that requires users to type the command to run the searching process. The normal users that do not know about the command cannot use this system. The command that use require user to know the location of index file and also another options that related with the command.

Users are well known and easy with the web base platform because this platform is most familiar platform with to users. From the above problem, developer was taking an initiative to develop a system that can ease the users to use the system. Desktop Search Engine for Linux is an integration web-based system of Namazu and its interface was developed using a web-based platform by using HTML and Javascript. Below is the reason why this system using this platform as an interface:

- easy to develop and maintain the interface
- can create an attractive and nice-looking appearance
- user are more familiar with the web base environment

The system consists of a text box for the users to fill the words/strings of the files or the related files that they want to search. Then the button is a submit button that will execute the process of searching the data. So, users no longer need to remember or type a complex command in the terminal to search for a file.

3.3 PROGRAMMING LANGUAGES

3.3.1 C programming language

- C language is a popular language preferred by professional programmers. Because of its powerful and flexible language, this language has been use by Namazu developer to develop the indexer for Namazu [13].
- This language actually will be a language that will operate the system such as to index the words into the index files, to query the index and so on.
- This language will operate with HTML language that is it will provide an output source and then the system will use HTML language to display the results.

3.3.2 HTML

- This system is a web base system that shows the results in a web base appearance. This system will also include a lot of HTML code as a code for interface of the system.
- This language is chosen because it is an easy learning language and no need to learn another new language that required a lot of time to master.

3.3.3 JavaScript

- JavaScript code has been used to support the system in handling an advance search features for this system.
- This code has been writing on the advance search files that connected with the main or search files for the searching input.

3.3.4 Perl

- Perl programming language has been used as by the namazu indexer installer to install the indexer in the system.

3.4 DEVELOPMENT TOOLS

3.4.1 Anjuta (IDE)

Anjuta is a versatile Integrated Development Environment (IDE) for C and C++. It has been written for GTK+/GNOME, and features a number of advanced programming facilities. It is a graphical interface to the collection of command line programming tools available for Linux and UNIX systems. These are usually run via a text console, and can be unfriendly to use.

Anjuta is an effort to marry the flexibility and power of command line tools with the ease-of-use of the GNOME graphical user interface. It has been made as user-friendly as possible.

3.4.1.1 Starting Anjuta:

To start Anjuta, open the GNOME Main Menu. Anjuta is found on the Development submenu. In RedHat 8.0 or later anjuta can be found in Extra submenu and then Programming submenu. The manual for starting and use Anjuta is attached in the appendices part (Refer to Appendix 2).

Click on the Anjuta icon to start. The IDE (Integrated Development Environment) will open. Alternatively, anjuta can be started from a terminal by issuing the command anjuta. If anjuta is started for the first time, it will address the users with a welcome message.

3.4.2 VIM Editor

VIM Editor is stand for **Vi IM**proved, is an open-source, multiplatform text editor extended from vi. This editor is helpful in editing program source code. This editor is used to support an Anjuta IDE during the development phase of this project.

3.5 OTHER TOOLS

3.5.1 Hardware:

Personal Computer

- Processor: Pentium M 1.4
- RAM: 640 Mhz
- Hard drive: 40Gb

3.5.2 Operating system:

- Linux Ubuntu 5.10 (or others Debian platform Linux distro)

3.5.3 Softwares:

- NAMAZU(Freeware) as an indexer

3.5.4 Programming tools

- VI Editor
- CGI Application
- Anjuta IDE
- etc

CHAPTER 4

RESULTS AND DISCUSSION

4.1 RESULTS

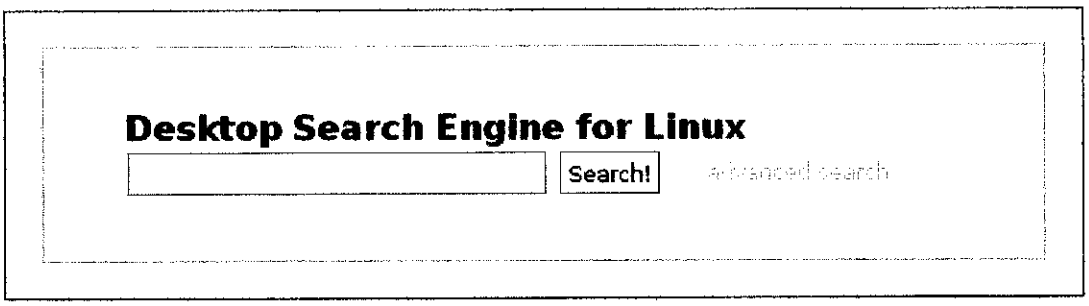

Figure 6: Front page of the system

Figure 6 is a screen shot front page of the Desktop Search Engine for Linux. This page consists of text box, search button, and also advanced search button. Text box is a space that is provided to users to fill the keywords for searching the files. Search button Search! is a button for system to process the action. Advance search button is a link to an advance search page that can ease the users to search for an advance in this system.

4.1.1 Simple Search Query

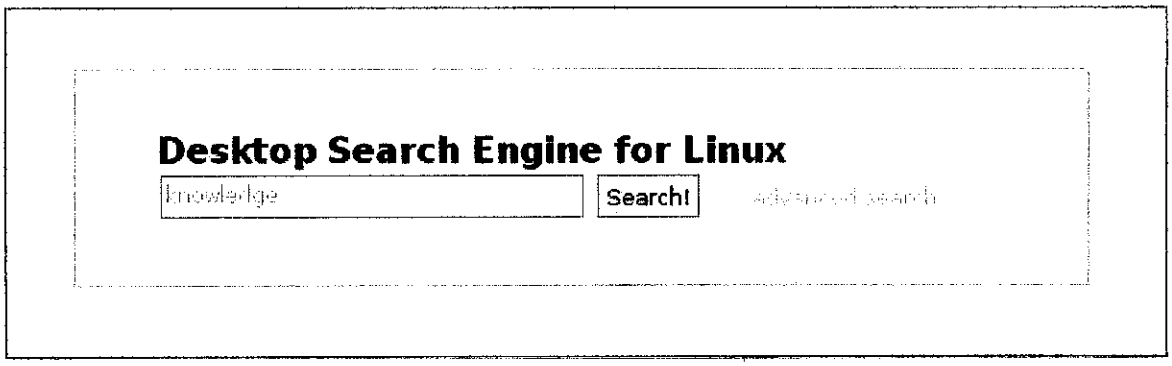

Figure 7: Simple Search Query

Figure 7 shows a screen shot for a simple search query that entered by the users with the keyword "*knowledge*". This system will match the keyword entered by users with the index files and display the result in the result page (Figure 8). If users enter two or more keywords in the text box, then system will search all keywords match in the index files. That mean, all result files displayed contain any keywords that has been entered in that files.

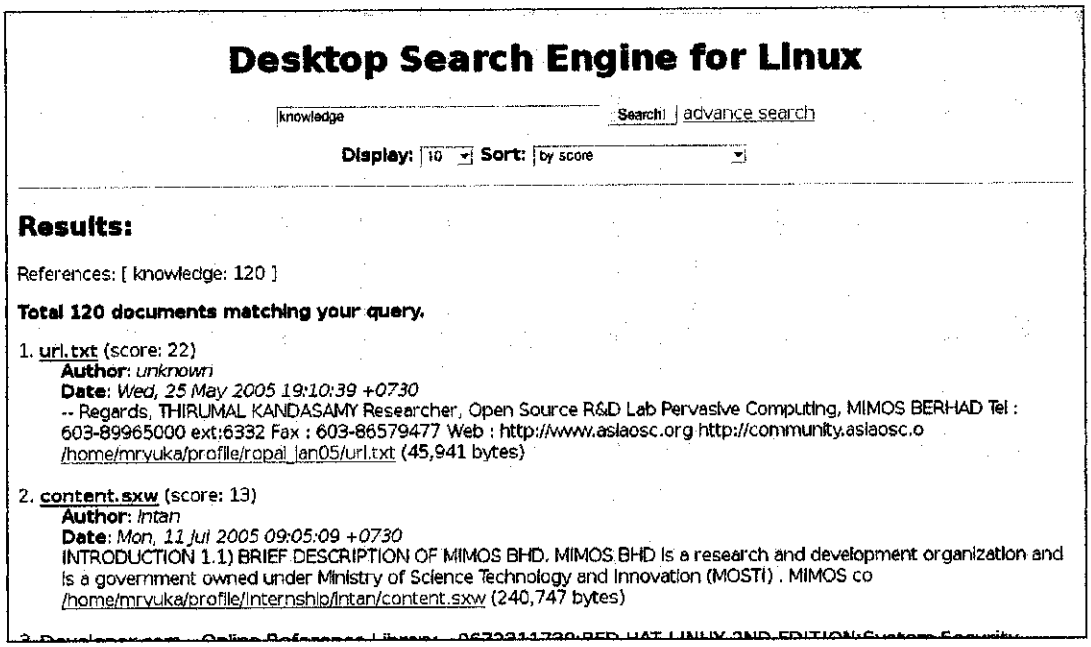

Figure 8: Results page

Figure 8 shows a results page that display results of searching processed. From this page we can see that users are displayed with the keyword that they filled, total documents that match with the query, and also the files name and its details such as:

i. Filename

ii. Author

iii. Date and last update time

iv. Short description or content of file

v. URI of file

vi. Size of file

4.1.2 Advanced Search Features

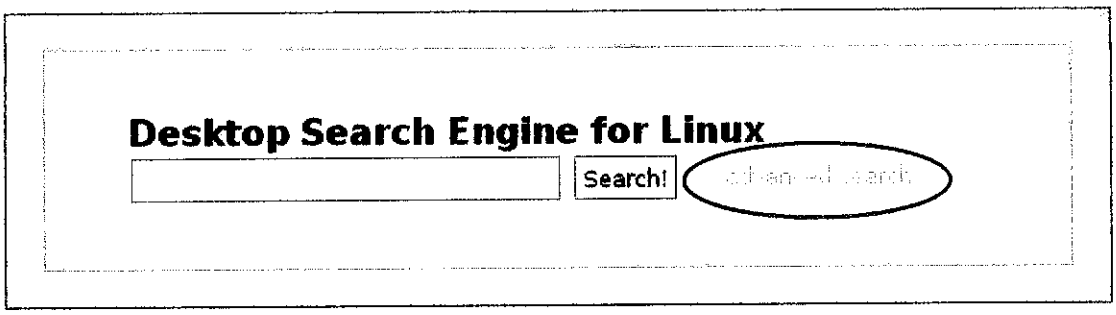

Figure 9: Advanced Search Page Link

Desktop Search Engine for Linux has advance search functions that will ease and help users to search in advance files that they want to search. Below (**Figure 10**) is the screenshot of advance search page for Desktop Search Engine for Linux. The system will display an advance search page when the user clicks the link to the page. Link to advance search page are provided besides the text box search as shown as **Figure 9** (red oval).

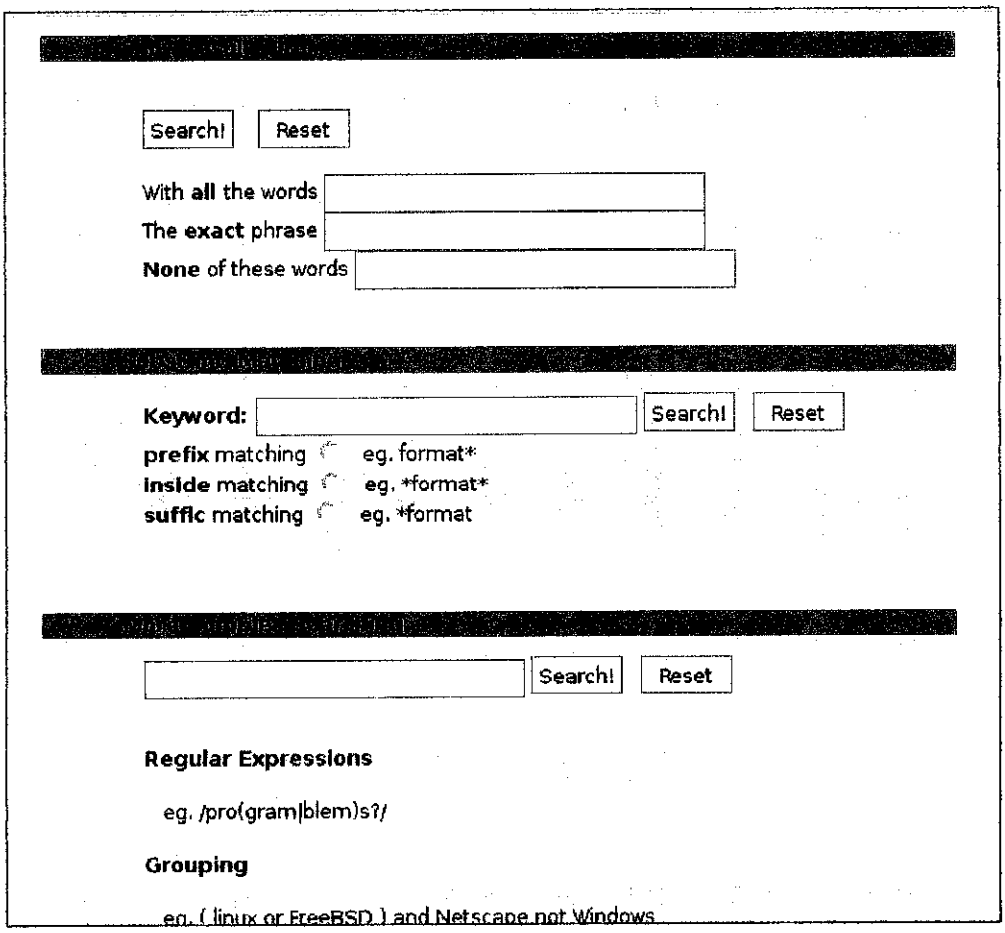

Figure 10: Advanced Search Page

Figure 10 shows a screenshot of an advance search page that will help users to search in advance the required file they want to search. These advance searches features provide three criteria of advance searching that are:

i. Normal search that consist of:
   a. searching with all words
   b. the exact phrase of text
   c. the exclude function

ii. Substring matching that consist of:
   a. prefix matching
   b. inside matching
   c. suffix matching

iii. Complex searching that consist of:
   a. regular expression
   b. grouping capabilities

4.1.2.1 Search with all the words

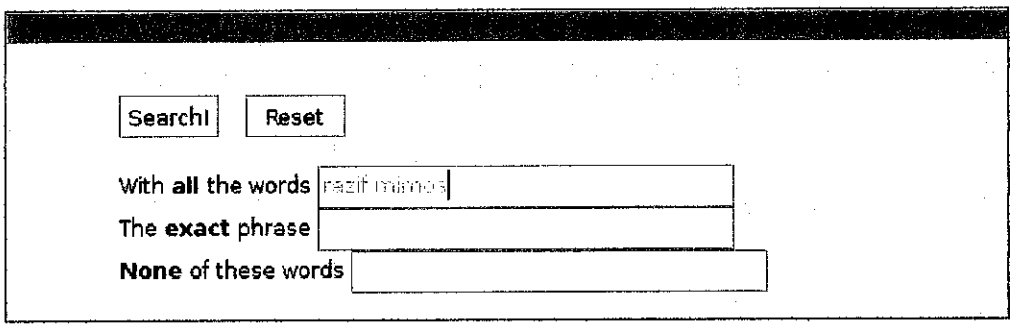

Figure 11: Search with all the words

The system will search keyword(s) that enter by user in this text box and display the result in the result page. For example if user enter keyword "razif mimos" then the system will search all files that contain this "razif mimos" keyword in the hard drive. Figure 11 shows user entered "razif mimos" in the *with all the words* text box. If users entered three keywords in the text box, (eg. knowledge acquisition performance) then the system will search all files that contain these keywords in the hard drive.

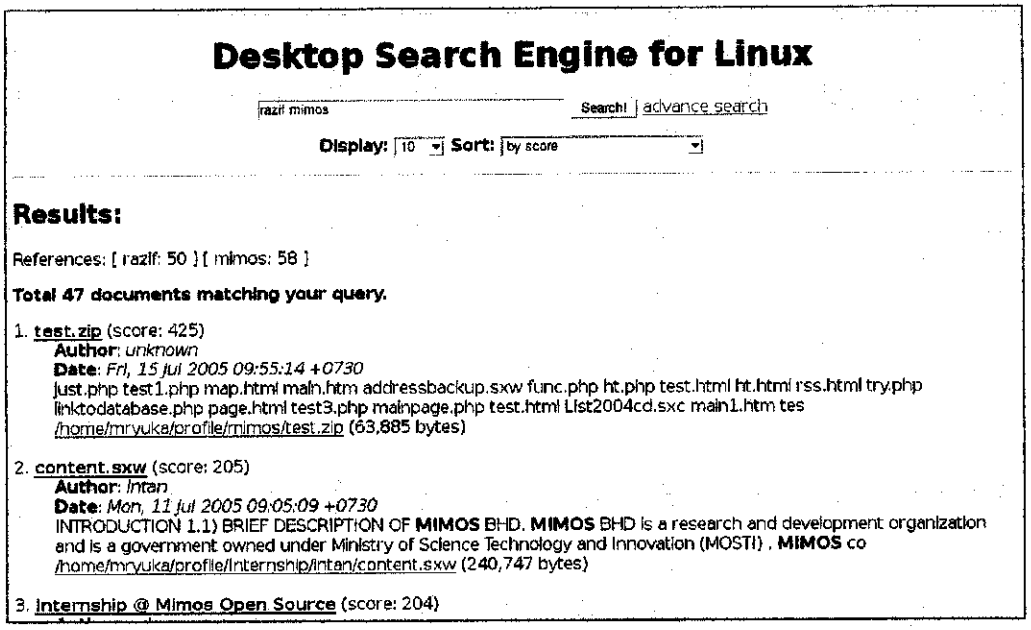

Figure 12: Result of with all the words

Figure 12 shows a screen shot that display the result of keywords that was entered to search the file that contains words *razif* and *mimos*. There are 47 files (documents) that match with the query or match with these two words.

4.1.2.2 Search the exact phrase

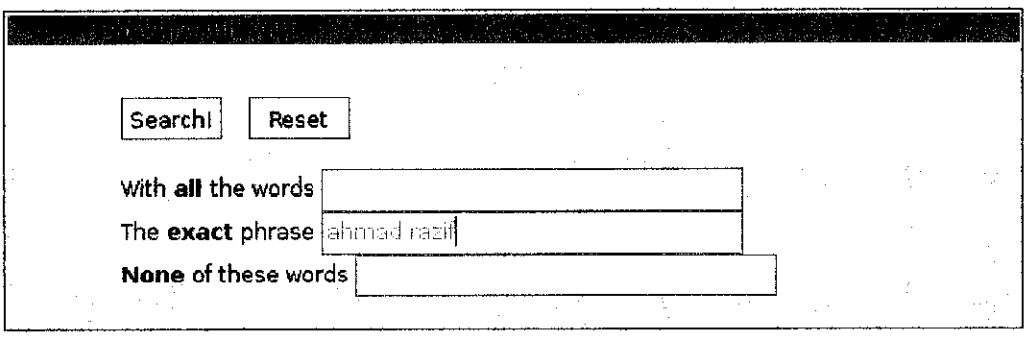

Figure 13: Search the exact phrase

This system also can ease the users to search file that contains an exact phrase in hard drive. **Figure 13** shows the screen shot of the interface that can process this function. Users just fill in the exact phrase of words they want into the text box and then the system will match that exact phrase with the index file and display all file that contains that phrase as a results. For example, the users enter

"*ahmad razif*" then system will search the files that contain this phrase from index files and display the result on the result page.

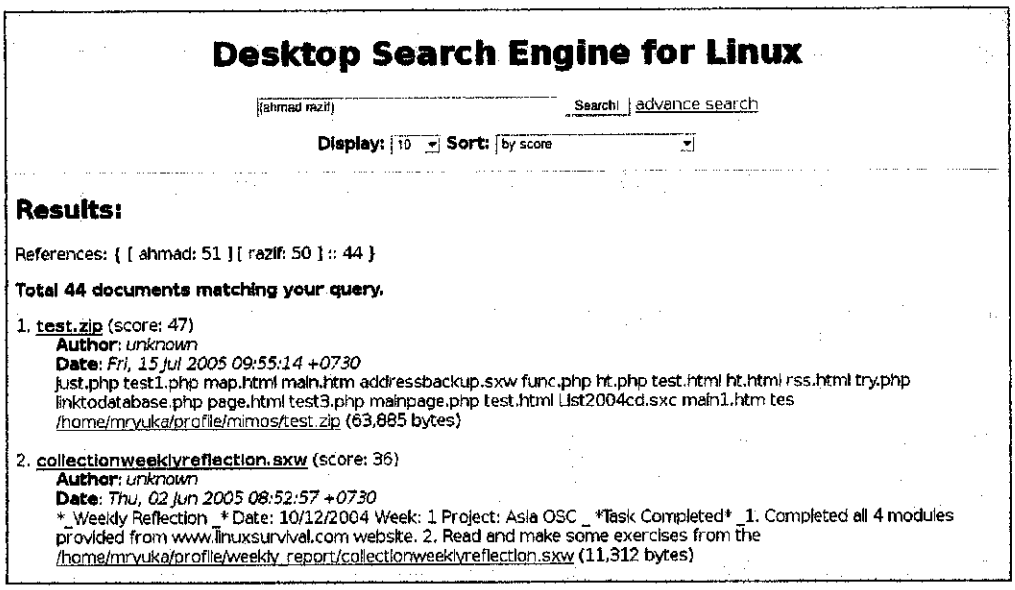

Figure 14: Result of exact phrase search

Figure 14 show a result of exact phrase search that entered by user with keywords "*ahmad razif*". The system enclosed the words with a curly bracket "*{ahmad razif}*" to search this type of matching. Users are not required to enter this curly bracket to perform this kind of search. They just enter the require words in the exact phrase text box provided.

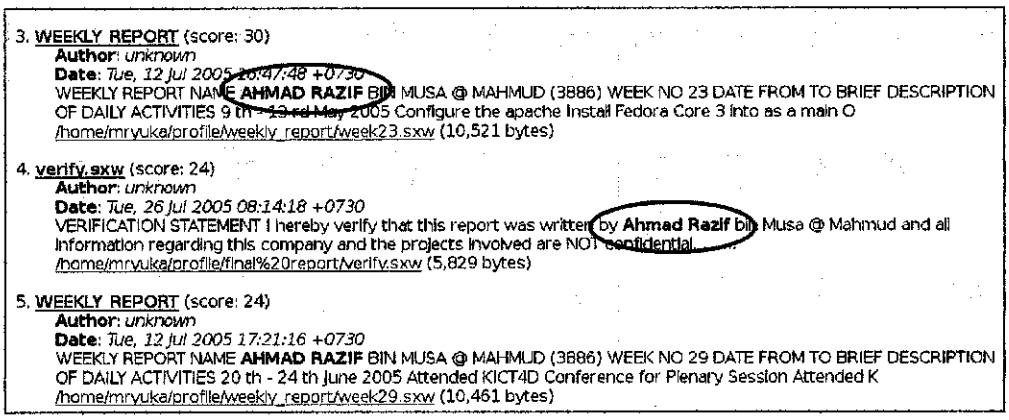

Figure 15: Result with the bold words

4.1.2.3 None of these words

If users have an unwanted word that they do not want to include in searching the file, then this function is the correct function to do so. This function is working if both of *all words function* and *exact phrase function* are filled with the keywords or either *all words function* or *exact phrase function* are filled in with the keyword. For example if user enters the keyword in *none of these words function* text box, then the system will unable its function.

4.1.2.4 All words and exact phrase search

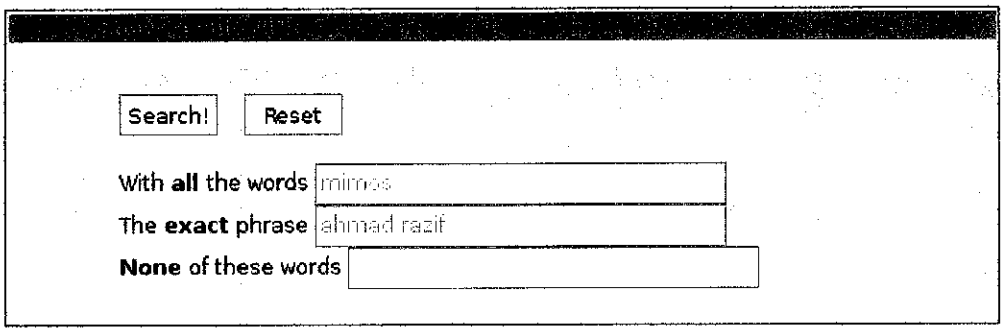

Figure 16: Search all words and exact phrase

Figure 16 shows a screenshot of searching with all words and the exact phrase. Users can use this combination of searching when they want to search a usual words and a word that in exact phrase. For example as show in the **Figure 16**, user filled all words with *mimos* and *ahmad razif* as an exact keyword that want to search. This system will match string mimos and a phrase ahmad razif with the index in the index file and display the result on the result page.

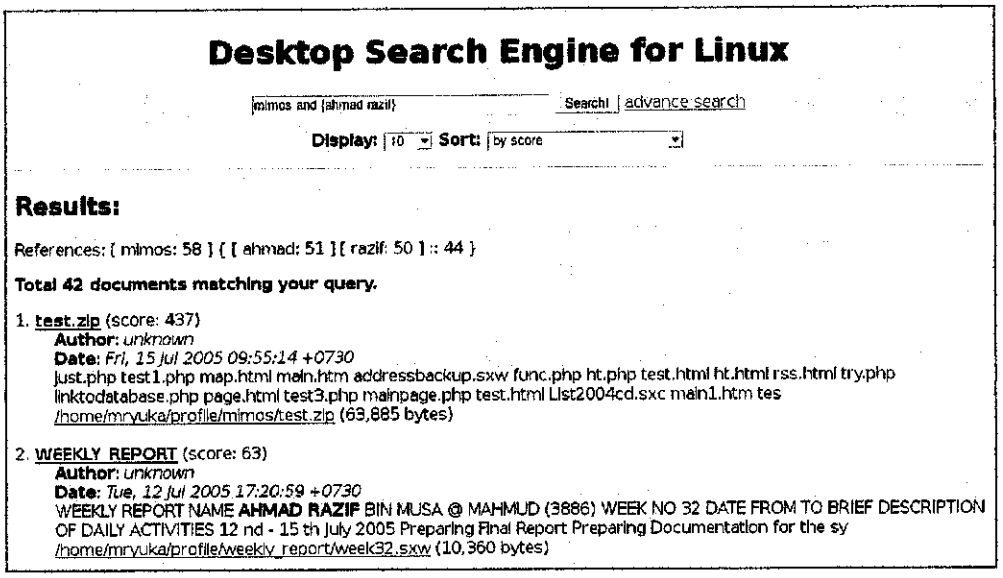

Figure 17: Result of all words and exact phrase search

Figure 17 shows a result page of all words and exact phrase search that has been done by users with the keywords *mimos* and *"ahmad razif"*. The system will make a query *mimos and {ahmad razif}* as a query to process. This query means to search all words that have string mimos and phrase ahmad razif in the file.

4.1.2.4 Complex search (grouping)

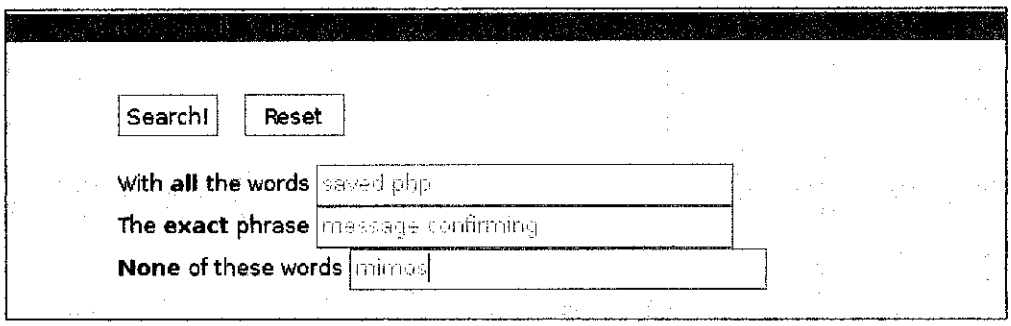

Figure 18: All words, exact phrase and none of the word search

Users also can make a complex search using this advance search function. User can combine these three functions to search a file. For example, users want to search file that contain words "saved" and "php" with the exact phrase of "message confirming" and not contain word "mimos". Then users just fill "saved php" in the all words function text box, "message confirming" in exact phrase function text box and "mimos" in the none of these word function

text box and then click the "Search !" button at above of the function. **Figure 18** show a screenshot of interface with the content that filled by user to search files by using these three features for an accurate searching.

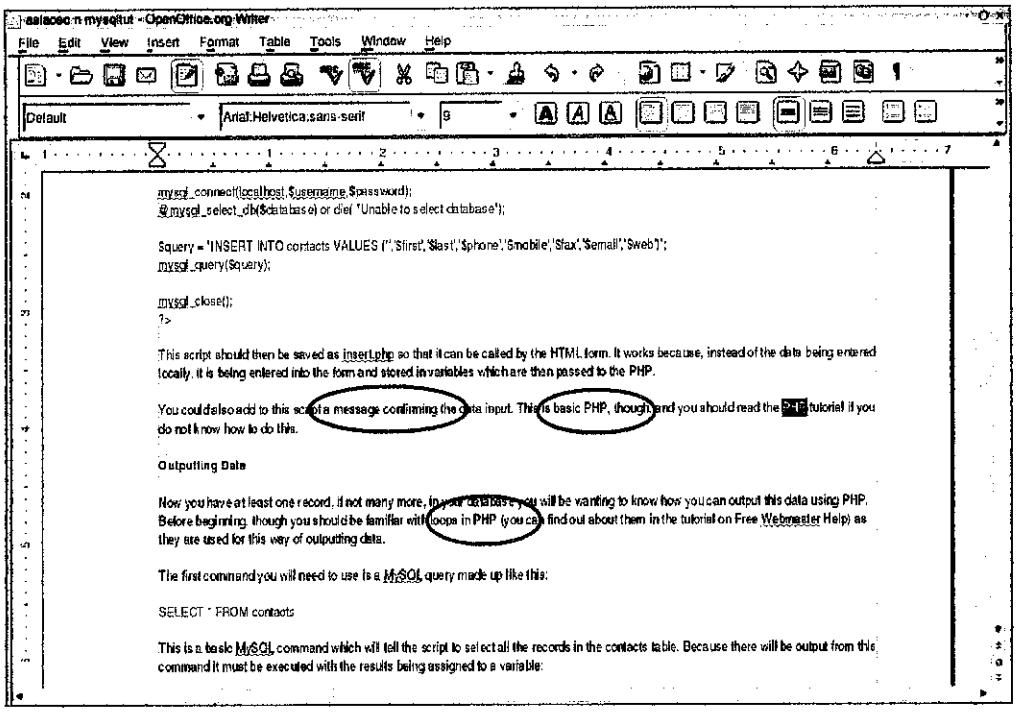

Figure 19: File searched by system

Figure 19 shows a file (OpenOffice.org) that contains words that required by user from the query. This file has all words that was entered by user that are saved, php and message confirming and not contain mimos keyword. The words have been mark with a red ovul on the above figure (**Figure 19**).

4.1.2.5 Prefix matching

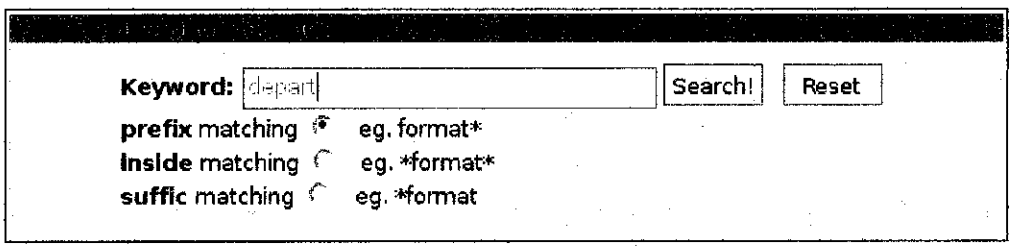

Figure 20: Prefix matching search

This function is to find the files with the terms that begin with the keyword that entered by user. For example if user wants to search for file that contains word begin with "depart", then they just fill in the keyword "depart" in the prefix matching text box and the system will search all files contain words begin with "depart" such as department, departure etc. **Figure 20** shows a screenshot of a keyword filled by user to search a prefix matching that begin with the word *depart*. User must click the prefix matching radio button to search for prefix matching. If not, the system will search by default that is an inside matching.

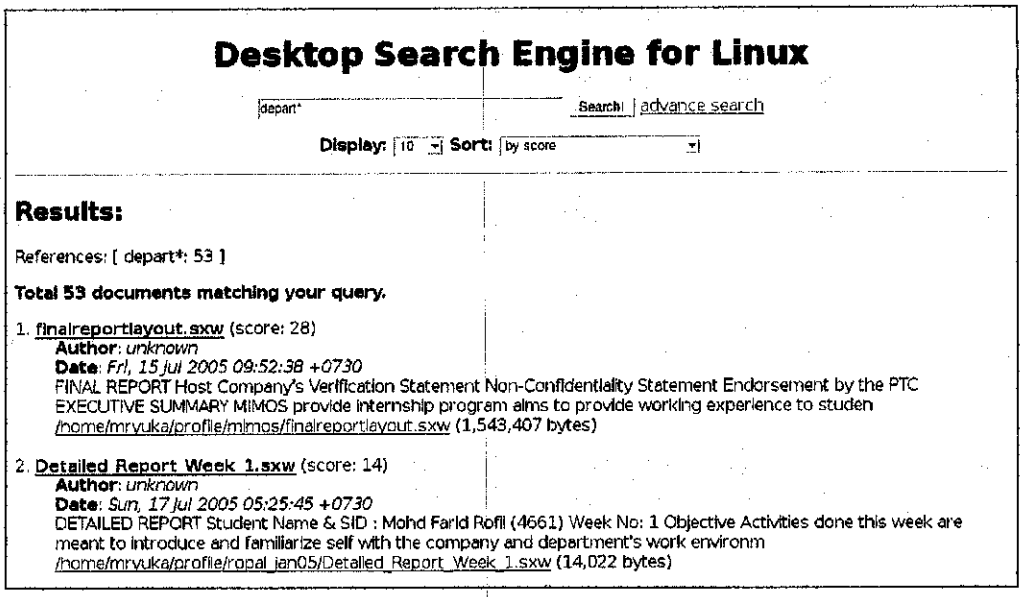

Figure 21: Result of prefix matching

Figure 21 shows a result of prefix matching that was entered by user with the keyword *depart*. System will search all string that begin with depart keywords in a hard drive. **Figure 22** shows an example of file that contain the keyword that begin with *depart* keyword that is department.

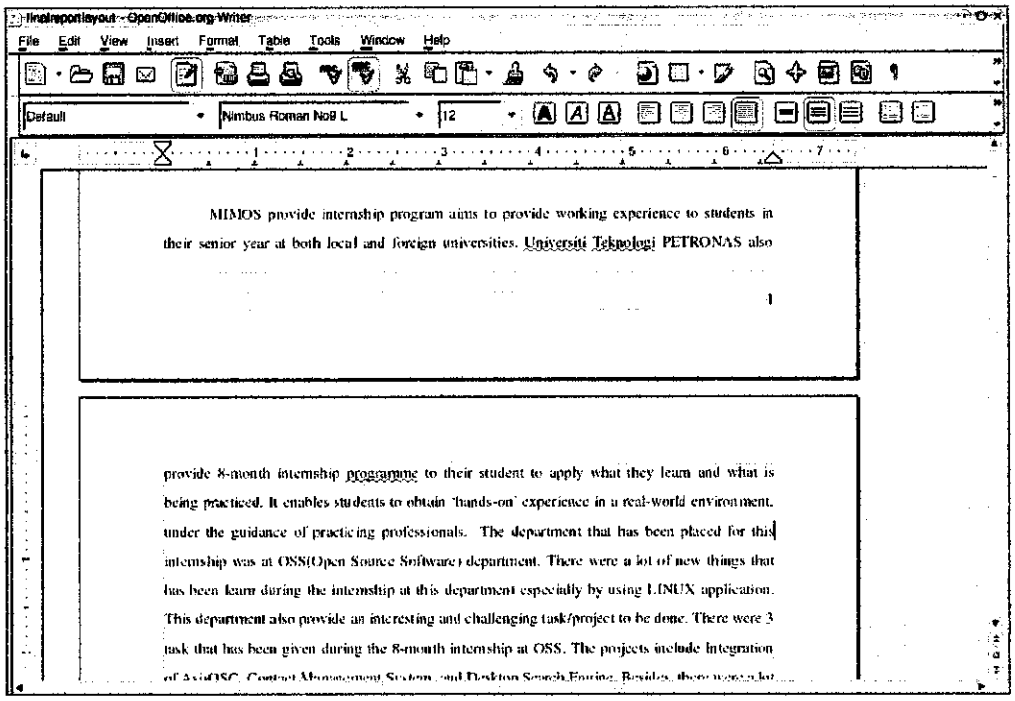

Figure 22: File contains with the prefix macthing keyword

4.1.2.6 Inside matching

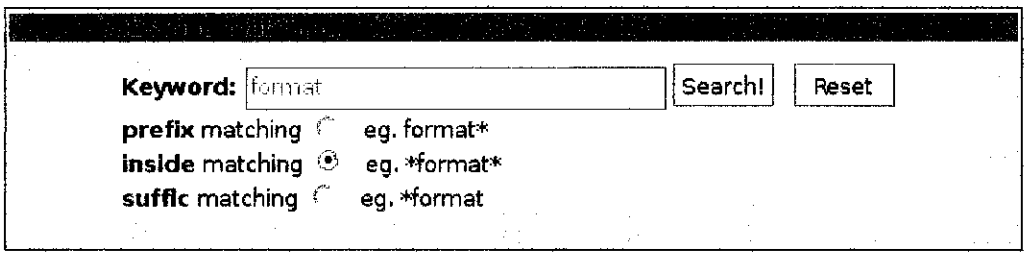

Figure 23: Inside matching search

This function is to find the files that have terms which contains with the keyword that entered by user. For example, if user wants to search for file that have a word that contain "*format*" keyword, then they just fill in the keyword *format* in the Inside matching text box. The system will search all files that contains word that contain a keyword "*format*" such as information, transformation, etc. **Figure 23** shows a screenshot of keyword that has been filled by user and clicked with inside matching radio button.

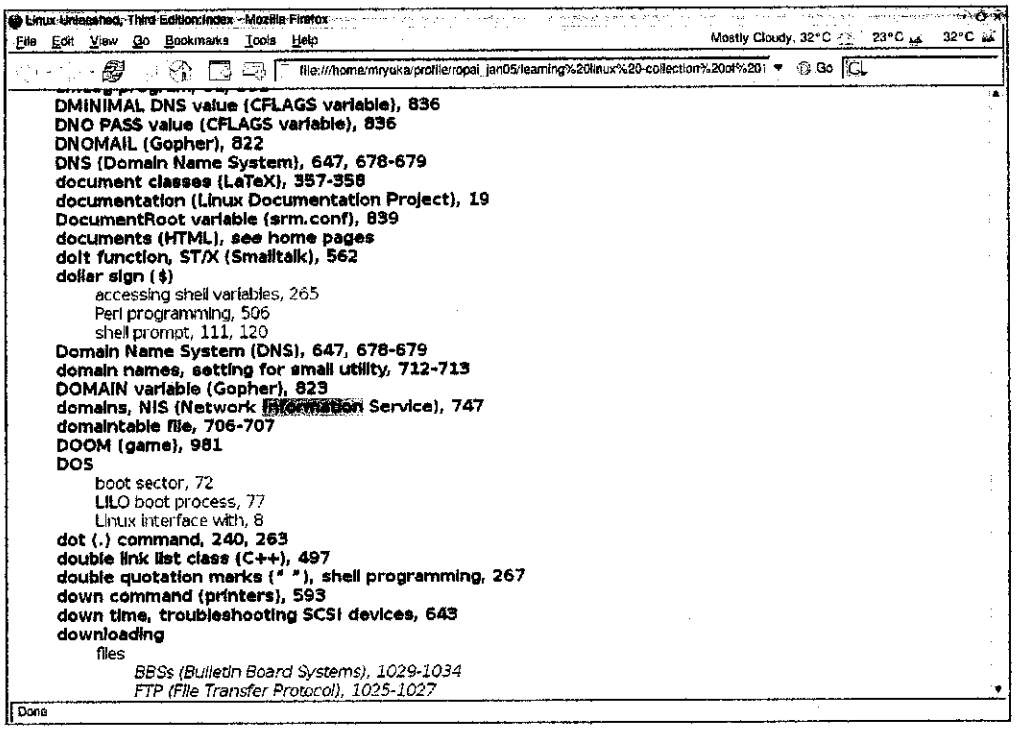

Figure 24: Result file of Inside matching

Figure 24 shows a result page file that contains keywords of inside matching of the keyword format. This file is a .html format and contain the require keyword. This mean, system also can search a file in the web base files such as .html, .htm etc.

4.1.2.7 Suffix matching

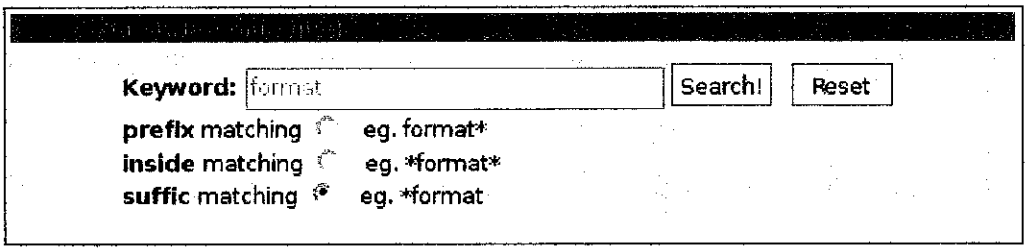

Figure 25: Suffix matching search

This function is to search files with terms that terminate with the filled word. For example if user want to search files that contain term "format" at the end of word, then they just enter keyword "format" in the suffix matching text box. The system will search all files that contain the keywords end with term "format" such as reformat, etc.

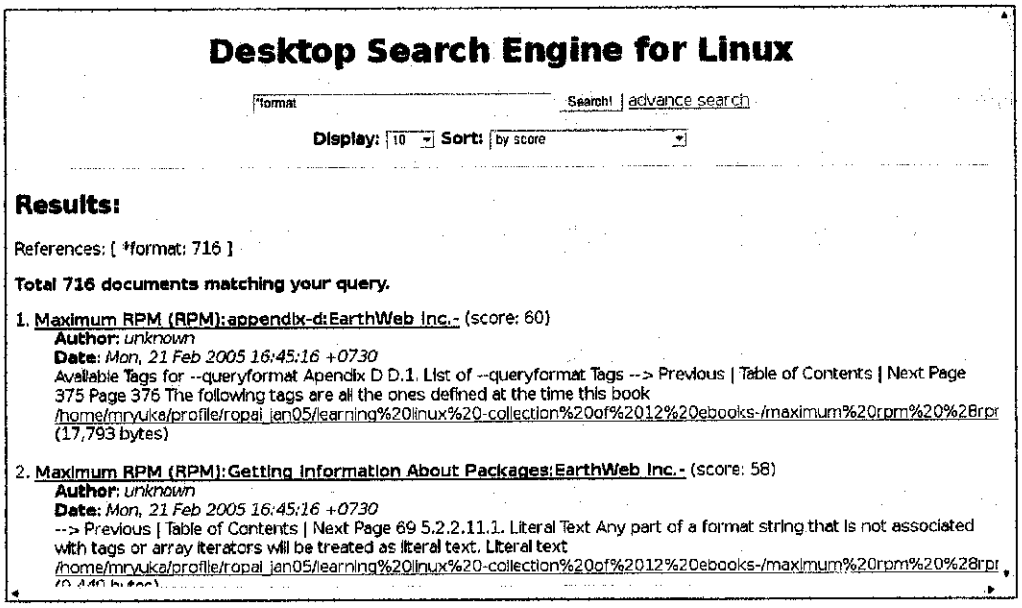

Figure 26: Result of Suffix matching

4.1.2.7 Regular expression

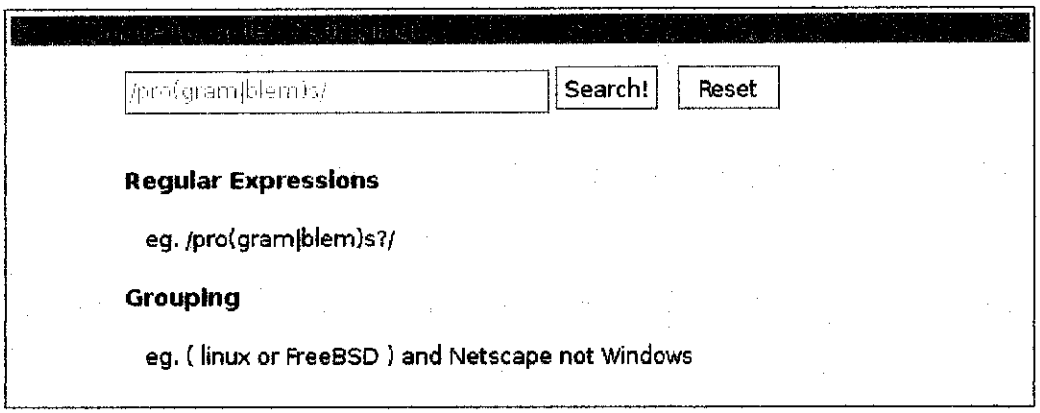

Figure 27: Regular expression searches

This function can search files for pattern matching. This function required users to enter the word surrounded by backslashes /.../. For example, if user want to search files that contain words program or problem then they just enter a keyword /pro(blem|gram)/. The system will search all files that contain keywords problem and program. **Figure 27** shows a screenshot with the string that has been filled by user.

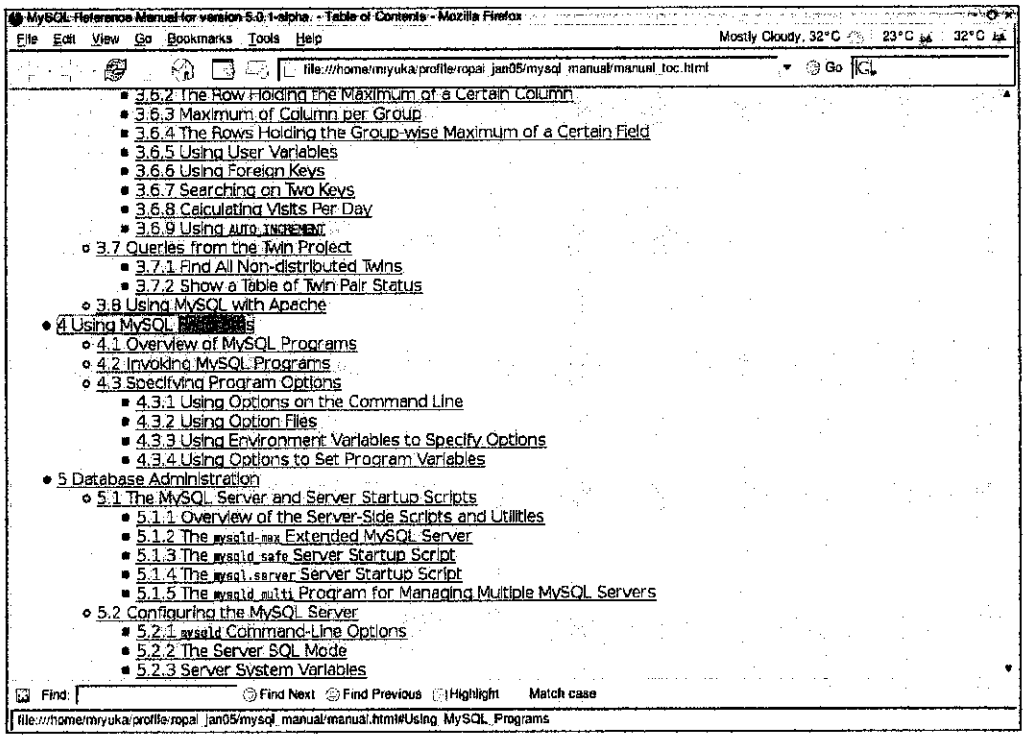

Figure 28: Result of Regular expression (Program)

Figure 28 shows a result page file that contain word *program* that is one of the result after system processing the query. **Figure 29** shows the same result page file that contain word *problem.*

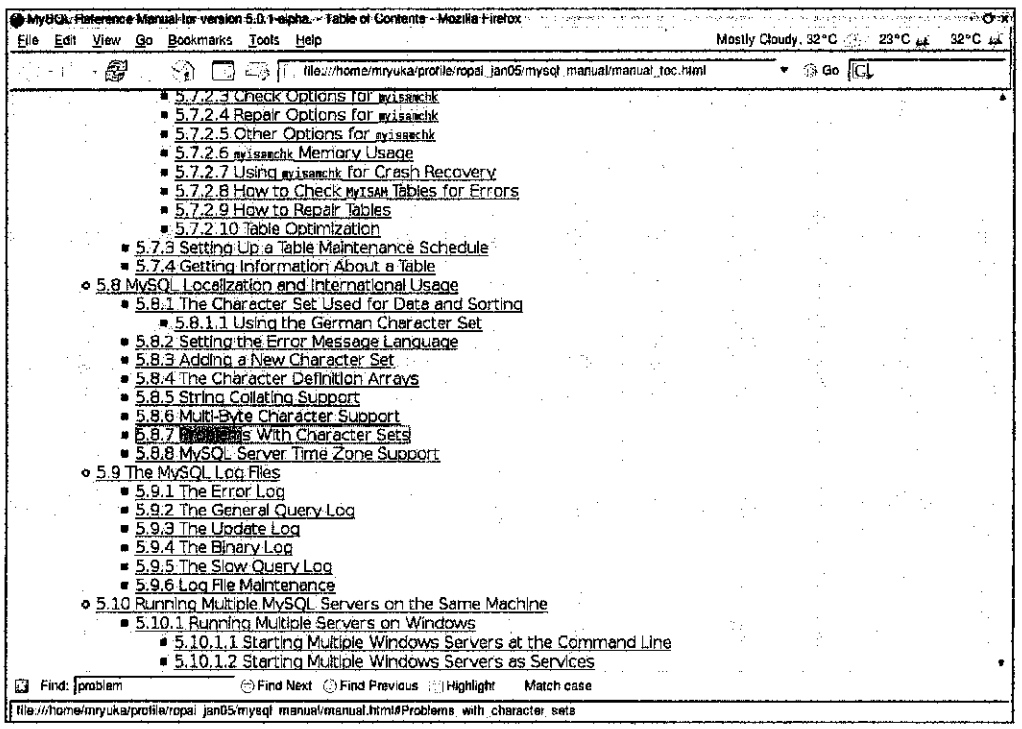

Figure 29: Result of Regular expression (Problem)

This advance features developed using JavaScript programming languages that integrates in HTML web based page to ease users use the functions. This feature was an advance of a Namazu features. Refer **Appendix 2** for the user manual of this system. **Table 2** is a summary of an advance search feature:

Function	**Description**
With **all** the words	search files that consist of all of the filled words
The **exact** phrase	search files with the exact phrase of the filled words
None of the words	search files with none of the word(s) filled
Substring Matching	
Prefix matching	search files with the terms that begin with the filled word Eg. format* = formation, formated
Inside Matching	search files with the terms that contain with the filled word Eg. *format* = information, transformation
Suffix Matching	search files with the terms that terminate with the filled word Eg. *net = internet, bonnet
Complex Searching	
Regular expression	Search files for pattern matching; the words must be surrounded by slashes like /.../ Eg. /pro(gram|blem)s?/ = programs, problems
Grouping	Group queries by surrounding them by parentheses Eg. (Linux or FreeBSD) and Netscape not Windows

Table 2: Summary of Advanced Search Features

4.1.3 Comparison of Usability between DSEforLinux, Ubuntu 5.10 search tool, and Namazu(Terminal) search tool.

Usability testing has been done to get the friendliest tool between Desktop Search Engine for Linux, Namazu Terminal and Ubuntu 5.10 tool. This testing has been conducted by 20 testers that are normal users. All testers were provided with the evaluation form that consist of 4 categories that need to fill that are *ease of use*, *learning curve*, *professional aesthetic* and *steps to reach desired results*. These categories are based on benchmark of usability criteria that was perform by UW E-Business Consortium [4]. These four categories must be filled by testers as marks for each tool and categories that are from 1 (poor) until 5 (best). **Table 3** shows a result of evaluation usability testing for three search tools.

	DSEforLinux	**Ubuntu 5.10**	**Namazu**
Ease of use	85	62	29
Learning curve	90	55	26.5
Professional aesthetic	80	75	23.5
Steps to reach desired output	85	68	31
Total	340	260	110
Average	**17**	**13**	**5.5**

Table 3: Total mark of evaluation for usability testing

The mark for each category is a sum of 20 forms that collected from 20 testers that have been tested these 3 tools that are DSEforLinux, Ubuntu 5.10 and Namazu (Terminal). Total value was divided by 20 as an average for the results. These results are compared as a comparison of usability mark for each tool.

4.2 DISCUSSIONS

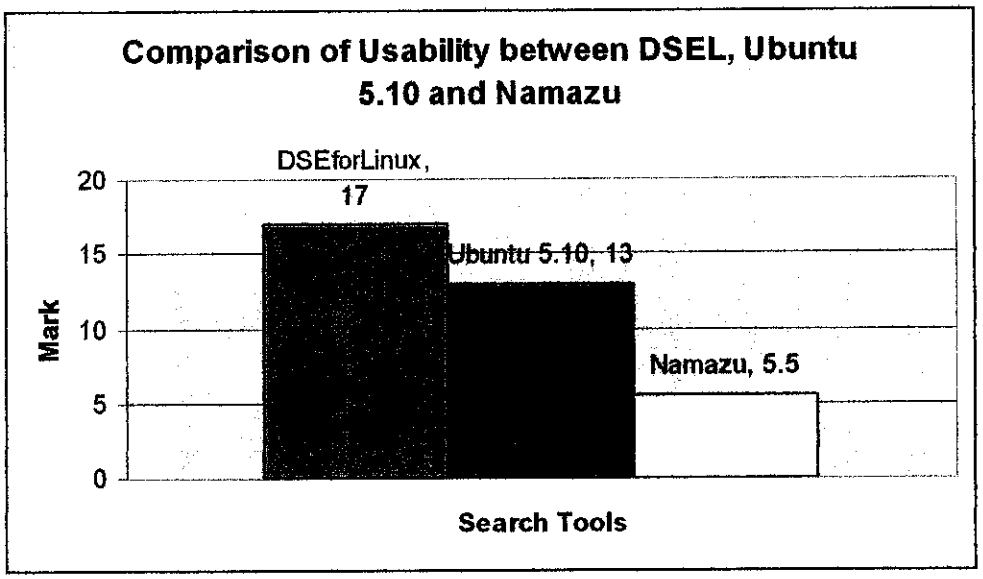

Figure 30: Comparison of Usability between DSEL, Ubuntu 5.10 and Namazu

Figure 30 shows comparison of usability between DSELforLinux, Ubuntu 5.10 and Namazu (Terminal). This chart is based on result of **Table 3**. From this chart we can see that usability mark for DSEforLinux is highest compared with Ubuntu 5.10 and Namazu.

Usability is one of the characteristic that must been focus when dealing with various levels of users. Software developers must make sure they produce a user friendly application that can ease users to deal or using that application. Good desktop search tools must be easy to use, have a lower learning curve, have a professional aesthetic, and require fewer steps to reach desired output [4].

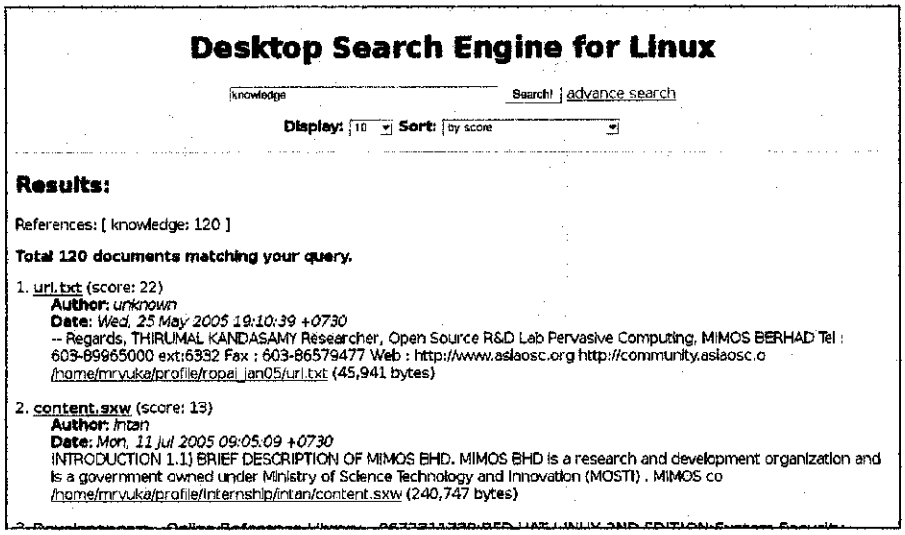

Figure 31: Screenshot of DSEL

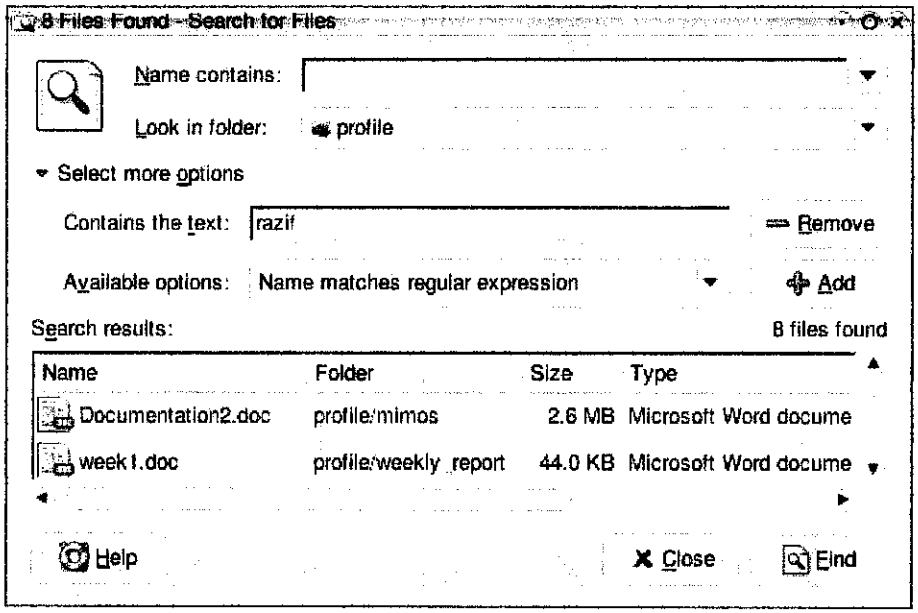

Figure 32: Screen Shot of Ubuntu File Search Tool

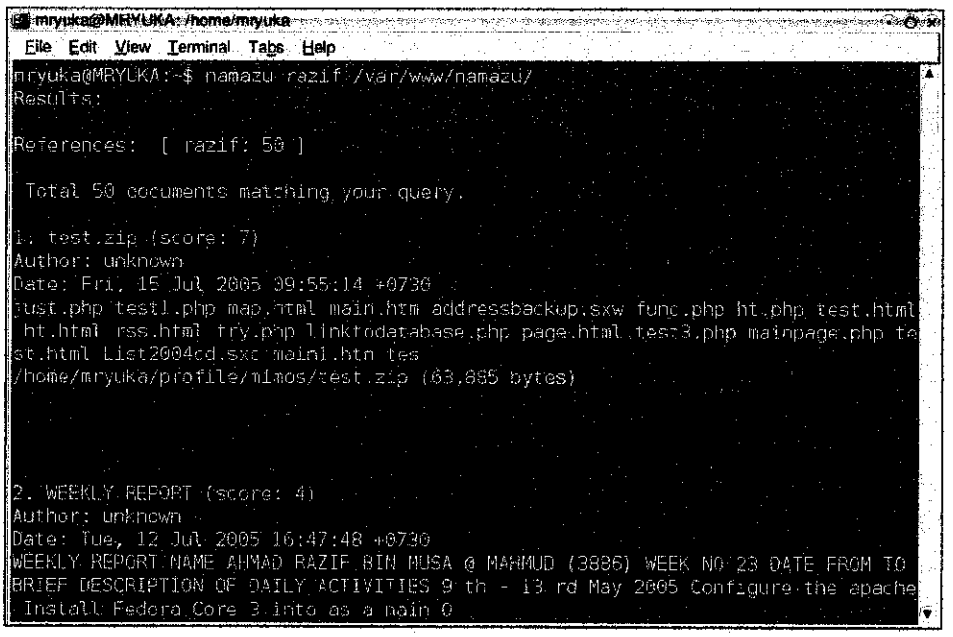

Figure 33: Screen Shot of Namazu(Terminal)

Figure 31 is a screenshot of display page of Desktop Search Engine for Linux. **Figure 32** show a screenshot of Ubuntu File Search Tool that included in the Ubuntu 5.10 distribution and **Figure 33** is a screenshot of Namazu that run in terminal. This three application test by users and comparison has been making to measure the usability, accuracy, versatility and efficiency of the tools.

From this snapshot we can see DSEforLinux has a simple and easy interface that can be use easily by any type of users. DSEforLinux just require user to fill the text box without need to choose another function before proceed with the Search button. An advance function also simple compare to Namazu and Ubuntu 5.10 tool. The application interface for Ubuntu 5.10 is more complex than DSEforLinux that require user to understand many functions that provided there. Namazu (Terminal) search tool are most complicated tool because users need to know the command that must use to execute the system and also users need to know the location of index files to make sure system can search the files.

Table 4 is a comparison table between DSEforLinux, Ubuntu 5.10 and Namazu Terminal that has been evaluated based on the benchmark criteria from UW E-Business Consortium, University of Wisconsin-Madison [4]. This table explains a

brief about the advantages and disadvantages of these tools, the features that had from tools etc.

Desktop Search Tools	Descriptions
DSEforLinux (Desktop Search Engine for Linux)	- easy to use - users easy to understand the functions - results are easy to capture - suitable for non expert or normal user - users no need learn much to use this system - just a few step required to execute the system
Ubuntu 5.10 File Search Tool	- easy to use - functions are difficult to understand - not enough space for displaying result - non expert or normal user take time to learn about the function - user must take more steps to execute the system
Namazu(Terminal)	- difficult to use because require type many words in terminal - functions based on manual; users must remember and know the function keywords - not suitable for non expert or normal users - no attractive interface that can affect users to use this system

Table 4: Comparison of Usability between DSEforLinux, Ubuntu 5.10 and Namazu(Terminal)

4.2.5 Combination flow of Index and Searching

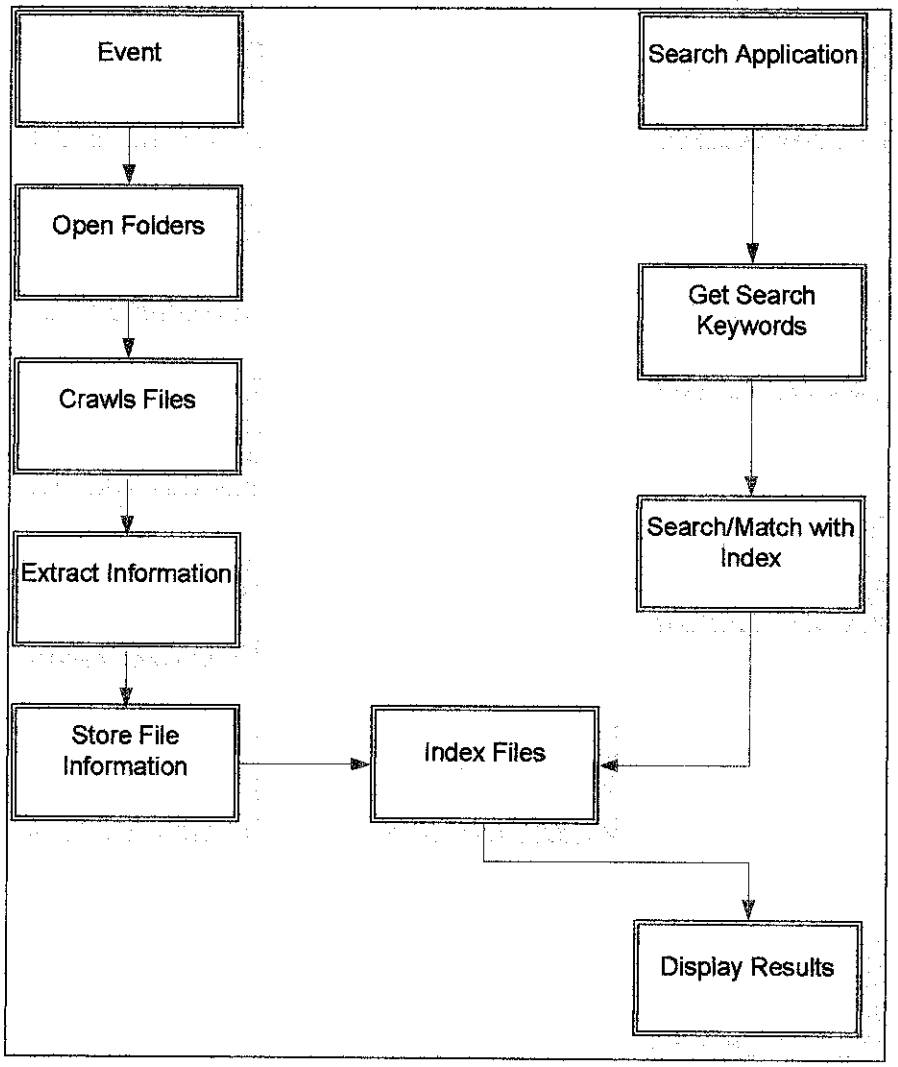

Figure 34: Combination flow of Indexing and Searching files

Figure 34 is a combination flow between indexing and searching files. These two important flows are combining at the index files. The relationship is by the indexing process is producing and index files while the searching process is processing the keywords entered by users by matching it with the data in the index files. The last resort is displaying results process where users can see all files that contain the words that have been filled. Users can select the related files they want by clicking the link to open or save the files.

CHAPTER 5

CONCLUSION AND RECOMMENDATIONS

5.1 CONCLUSION

Desktop Search becomes more popular for personal computer users nowadays. Peoples are demand for the system with the high technology that can provide accurate results, and also faster searching to search the files. Desktop Search Engine for Linux can help users that using Linux operating system (Debian based) to search files in their hard disk faster and accurate.

On the first half of the project implementation, analysis has been made in the sense to understand the process involved in developing this system. Analysis also has been done about the best tools to use to develop this system. The freeware indexer that is Namazu has been chosen as an indexer and a backbone for this system.

On the second half of the project implementation, the integration of interface with an indexer has been done. Interface for displaying the results, front page of the system, and also an advance search feature page have been develop to ease the users using this system.

Desktop Search Engine for Linux can search text format file in the hard disk of personal computer. The user just requires entering the related keyword in the provided text box and then the system will process the transaction and the system will display the results. This system also supported with an advance search features that can help user to search files in more detail and accurate.

5.2 RECOMMENDATIONS

Desktop Search Engine for Linux is still in the development to fulfill the requirement for benchmark criteria Consortium for desktop search tool that was perform by UW E-Business. This system can be improve by adding a feature of automatic index update that is a system always update the index files while the personal computer running or after the user create or modify the new files in the hard disk. This system also suggested by Mr Albert to add a help function to ease the users to use the system.

This system also can be improve by adding a various file searches features that is a features of searching another files such as mp3, pictures, video etc rather than just a text format files. This added feature can upgrade this system to be a multipurpose desktop search engine.

Besides, this system also can be improve by integrate this system with the voice command search. This feature will enable a disable person to use this system. Users also no longer need to type the keywords if this improvement is implemented in this system.

REFERENCES

[1] Chirita A., Gavriloaie R., Ghita S., Nejdl W., Paiua R. (2004). Activity Based Metadata for Semantic Desktop Search. Hanover, Germany.

[2] Cole B. (2005). Industry Trends: Search Engines Tackle the Desktop. www.informatik.uni-trier.de

[3] Stellent (2005). Outside in Technology, A Stellent Product: Outside In Unlocks Business Documents for Search and Retrieval Systems. http://www.outsideinsdk.com

[4] Noda T., Helwig S. (2005). Best Practice Reports: Benchmark Study of Desktop Search Tools. UW E-Business Consortium, University of Wisconsin-Madison.

[5] Lopo E.d.C, Aitken P, Jones B.L (2000). SAMS Teach Yourself (C for Linux Programming in 21 Days).

[6] Nenov D. (2005). X1 Desktop Search Platform: User-Centric Enterprise Retrieval and Action. XI Technologies, Inc.

[7] Clarke C.L.A, Buttcher S. (2005). A Security Model for Full-Text File System Search in Multi-User Environment. School of Computer Science, University of Waterloo, Canada.

[8] Stellent (2005). Outside in Technology, A Stellent Product: Outside In Unlocks Business Documents for Search and Retrieval Systems. http://www.outsideinsdk.com

[9] Johnson M. (July 2005). Personal Tech: Turning the Concept of Search on its Head. http://www.boston.com.

[10] Parker P. (July 2004). Clickz News: Blinkx Plans Ad Model for Desktop/Web Search Tool. http://www.clickx.com/news

[11] Paiu R. (January 2005). L3S Research Center: Beagle Desktop Search and Activity Based Metadata. Hanover, Germany.

[12] Buttcher S. and Clarke L.A.C. (2005) Indexing Time vs. Query Time Tradeoffs in Dynamic Information Retrieval Systems, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada.

[13] Tabayashi S. (2006, Jan 29) Namazu: A Full Text Search Engine. Retrieved February 12, 2006, from http://www.namazu.org

[14] Search Tools Product Reports: Namazu Retrieved February 12, 2006, from http://www.searchtools.com/tools/namazu.html

[15] Advance Linux. Retrieved February 18, 2006 from http://www.linux.com/

[16] Information Retrieval. Retrieved May 17, 2006 from http://www.answers.com/topic/information-retrieval?method=22

APPENDICES

APPENDIX 1

USER MANUAL

DESKTOP SEARCH ENGINE FOR LINUX

DESKTOP SEARCH ENGINE FOR LINUX USER MANUAL

Table of Contents

1. Introduction

1.1 About Desktop Search Engine for Linux

Desktop Search Engine for Linux was developed from November 2005 until Jun 2006 and now still in development. This system is based on requirement for Universiti Teknologi PETRONAS Final Year Project that every student should do as a requirement to finish their study in this institution.

During an internship for 8 month at Open Source R&D team, MIMOS Berhad, the author had introduced with an open source environment. Everything that done were in open source application such as Operating System that used at the workplace are using Linux rather than Microsoft Windows, Apache Server that is one of the best web server, etc. During an internship, the author aware about the open source development in this country are still slow and the awareness about open source and the advantages behind open source environment still blur to the people in the world.

Aware about the advantages of open source and the advantages of using Linux give author a spirit to do a system that can be use by Linux users. With the growth of a desktop search tools technology and a lack of support for Linux environment, that factors give the spirit to author to develop this project. During the planning stage, the author do a lot of studies about the tools that can be used to ease me develop this system. Then, the author found an interesting application that can be use as a based for my project. The author proposed this project to Final Year Project Team about Desktop Search Engine for Linux and they were accepted it.

The idea for this system was about to integrate a freeware Namazu desktop search tool with the web-base environment. Namazu indexer was the main factor that forces author to use this tool. Namazu indexer can crawl text format information from files and create one file that is index file. Then, search engine will process the user query to match the query with information in the index files. The author knows to develop his own indexer will take a long time to finish and this is an alternative way for me to finish this system.

2. Getting Started

2.1 Server Requirement

As we know, Apache is one of the best web server in the world and it is an open source web server so Linux Operating System usually provided with this web server. If the users of Linux environment find out that their Linux do not have this application then they can install it by using a synaptic package manager that provided in the Linux package.

After installing the web server, then copy a namazu folder into web server folder, usually at /var/www/ folder. By using terminal, user must go to namazu folder. The command that can be use by users to go to namazu folder from terminal is /var/www/namazu . Then just type *ls* to list all files contain in that folder.

2.2 Installing Namazu

2.2.1 Test before "make install"

If you wish to test mknmz before make install, do
cd namazu-2.0.x (... where you have unpacked *.tar.gz)
env pkgdatadir=`pwd` scripts/mknmz (in case csh/tcsh)
or
pkgdatadir=. scripts/mknmz (in case with sh/bash).
These will refer adjacent pl,filter,template etc, not exisiting stuff under /usr/local/share/namazu etc).

(To know more about this, see $PKGDATADIR variable in mknmz etc.)

You may try following examples for the first time to see the configuration, help, and to generate indexes for ~/Mail stuff, respectively.

```
./mknmz -C
./mknmz --help
```

./mknmz -O /tmp ~/Mail

2.2.2 Help Menu

If you just type mknmz or namazu with no argument, a short usage will be displayed. If you feed --help as an argument, a long usage will be displayed. The option -C will display the configurations at the time. Useful to remember these 3 option usages.

How to get help menus in command-line

Argument	Meaning	Other Arguments
None	Short Usage	Cannot add any argument
--help	Long Usage	Ignores other arguments
-C	Configurations	Other arguments will have meanings.

2.2.3 Running mknmz

First, create index. Format is changed slightly from versions 1.4.0.8. URI replacement is dealt with by specifying --replace option. URI replacement can be done during namazu/namazu.cgi execution. In this case, run mknmz without --replace option, and setup .namazurc so that URI replacement is performed during namazu/namazu.cgi execution.

Run mknmz as follows.

mknmz [options] target directory

The above example creates index in the current directory. Use -O option to specify the output directory.

For example,

```
mkdir /tmp/index
mknmz -O /tmp/index \
```

```
--replace='s#/foo/bar/doc/#http://foo.bar.jp/software/#' \
/foo/bar/doc
```

mknmz will output the following messages during the creation of index.

```
14 files are found to be indexed.
1/14 - /foo/bar/acrobat3.pdf [application/pdf]
2/14 - /foo/bar/excel97.xls [application/excel]
3/14 - /foo/bar/html.html [text/html]
4/14 - /foo/bar/mail-multipart.txt [message/rfc822]
5/14 - /foo/bar/mail.txt [message/rfc822]
6/14 - /foo/bar/man.1 [text/x-roff]
7/14 - /foo/bar/msg00000.html [text/html; x-type=mhonarc]
8/14 - /foo/bar/plain.txt [text/plain]
9/14 - /foo/bar/plain.txt.Z [text/plain]
10/14 - /foo/bar/plain.txt.bz2 [text/plain]
11/14 - /foo/bar/plain.txt.gz [text/plain]
12/14 - /foo/bar/rfc0000.txt [text/plain; x-type=rfc]
13/14 - /foo/bar/tex.tex [application/x-tex]
14/14 - /foo/bar/word97.doc [application/msword]
Writing index files...
[Base]
Date:            Thu Mar 16 22:14:01 2000
Added Documents:    14
Size (bytes):       58,701
Total Documents:    14
Added Keywords:     95
Total Keywords:     95
Wakati:          module_kakasi -ieuc -oeuc -w
Time (sec):         14
File/Sec:        1.00
System:          linux
Perl:            5.00503
Namazu:          2.0.X
```

- Result (Index) will be in /tmp/index (specified in -O)
- Target documents are /foo/bar/doc
- For URI

  This means "documents under /foo/bar/doc/ will appear as http://foo.bar.jp/software/, so please perform replacement like s#*aaa*#**bbb**# if written in Perl." (In this example, (aaa) corresponds to (/foo/bar/doc/) and (bbb) corresponds to (http://foo.bar.jp/))

- (Depending on $ALLOW_FILE and $DENY_FILE in /usr/local/etc/namazu/mknmzrc) target files may be *.html, *.txt, *.tex, *.pdf, mails in MH format.

2.2.4 Customizing mknmz

Namazu was originally developed for processing HTML documents; Namazu can now deal with various document styles. You will find useful scripts in /usr/local/share/namazu/filter, and detailed explanation will be found in Document filters in Namazu manual.

Mails in MH format

run mknmz

% mknmz ~/Mail/foobar

MHonArc

Namazu will do specific processing for MHonArc HTML.

hnf

.mknmzrc for hnf and guide can be obtained from Hyper NIKKI System

Documents stored in other machines

Cannot search documents using Namazu alone. Need to use other tools (eg. wget, NFS) that transfer the documents in combination.

For mknmz command-line arguments, you get usage information from mknmz --help. With -C option, you get the configurations of the time.

Loaded rcfile: /home/foobar/.mknmzrc

System: linux

Namazu: 2.0.X

Perl: 5.00503

File-MMagic: 1.25

NKF: module_nkf

KAKASI: module_kakasi -ieuc -oeuc -w

ChaSen: module_chasen -i e -j -F "%m "

MeCab: module_mecab -Owakati -b 8192

Wakati: module_kakasi -ieuc -oeuc -w

Lang_Msg: C

Lang: C

Coding System: euc

CONFDIR: /usr/local/etc/namazu

LIBDIR: /usr/local/share/namazu/pl

FILTERDIR: /usr/local/share/namazu/filter

TEMPLATEDIR: /usr/local/share/namazu/template

Supported media types: (42)

Unsupported media types: (2) marked with minus (-) probably missing application in your $path.

application/excel: excel.pl

application/gnumeric: gnumeric.pl

application/ichitaro5: taro56.pl

application/ichitaro6: taro56.pl

application/ichitaro7: taro7_10.pl

application/macbinary: macbinary.pl

application/msword: msword.pl

application/pdf: pdf.pl

application/postscript: postscript.pl

application/powerpoint: powerpoint.pl

application/rtf: rtf.pl

application/vnd.kde.kivio: koffice.pl

application/vnd.kde.kpresenter: koffice.pl

application/vnd.kde.kspread: koffice.pl

application/vnd.kde.kword: koffice.pl

application/vnd.oasis.opendocument.graphics: ooo.pl
application/vnd.oasis.opendocument.presentation: ooo.pl
application/vnd.oasis.opendocument.spreadsheet: ooo.pl
application/vnd.oasis.opendocument.text: ooo.pl
application/vnd.sun.xml.calc: ooo.pl
application/vnd.sun.xml.draw: ooo.pl
application/vnd.sun.xml.impress: ooo.pl
application/vnd.sun.xml.writer: ooo.pl
application/x-apache-cache: apachecache.pl
application/x-bzip2: bzip2.pl
application/x-compress: compress.pl
- application/x-deb: deb.pl
- application/x-dvi: dvi.pl
application/x-gzip: gzip.pl
application/x-js-taro: taro7_10.pl
application/x-rpm: rpm.pl
application/x-tex: tex.pl
application/x-zip: zip.pl
audio/mpeg: mp3.pl
message/news: mailnews.pl
message/rfc822: mailnews.pl
text/hnf: hnf.pl
text/html: html.pl
text/html; x-type=mhonarc: mhonarc.pl
text/html; x-type=pipermail: pipermail.pl
text/plain
text/plain; x-type=rfc: rfc.pl
text/x-hdml: hdml.pl
text/x-roff: man.pl

2.2.5 Targets of index creation

short name	long name	description
-F	--target-list=FILE	read in list of target files for index creation
-t	--media-type=MTYPE	specify the document format of target files
	--allow=PATTERN	specify the regular expression of target file names.
	--deny=PATTERN	specify the regular expression of to-be-excluded file names.
	--exclude=PATTERN	specify the regular expression of to-be-excluded path names.

2.2.6 Running namazu

To search documents, do

% namazu query index

If you omit index, namazu will assume /usr/local/var/namazu/index as target.

Set up for namazu command will be done in namazurc. An example of namazurc can be found in /usr/local/etc/namazu/namazurc-sample in Namazu distribution package.

2.2.7 Namazu components

Namazu is a full-text search engine. Namazu uses the index maker mknmz command and the text searcher namazu command.

For quickly searching through many documents, Namazu generates an index similar to that of a book's.

mknmz command compiles the index. The target directory for indexing is given as an argument for mknmz. For example, if the target directory is /home/foo/public_html, then type

% mknmz /home/foo/public_html

Now documents such as *.html and *.txt under /home/foo/public_html are indexed and NMZ.* files are created in the directory where you run mknmz. NMZ.* files are from Namazu's index.

The namazu command searches the index. For example:

% namazu bar /home/foo/Namazu/foobar

The above searches a keyword "bar" for the index under /home/foo/Namazu/bar.

2.3 mknmz command

2.3.1 mknmz's options

mknmz 2.0.x, an indexer of Namazu.

Usage: mknmz [options] <target>...

Target files:

-a, --all target all files.

-t, --media-type=MTYPE set the media type for all target files to MTYPE.

-h, --mailnews same as --media-type='message/rfc822'

--mhonarc same as --media-type='text/html; x-type=mhonarc'

-F, --target-list=FILE load FILE which contains a list of target files.

--allow=PATTERN set PATTERN for file names which should be allowed.

--deny=PATTERN set PATTERN for file names which should be denied.

--exclude=PATTERN set PATTERN for pathnames which should be excluded.

-e, --robots exclude HTML files containing

<meta name="ROBOTS" content="NOINDEX">

-M, --meta handle HTML meta tags for field-specified search.

-r, --replace=CODE set CODE for replacing URI.

--html-split split an HTML file with <a name="..."> anchors.

--mtime=NUM limit by mtime just like find(1)'s -mtime option.

e.g., -50 for recent 50 days, +50 for older than 50.

Morphological Analysis:

-b, --use-mecab use MeCab for analyzing Japanese.

-c, --use-chasen use ChaSen for analyzing Japanese.

-k, --use-kakasi use KAKASI for analyzing Japanese.

-m, --use-chasen-noun use ChaSen for extracting only nouns.

-L, --indexing-lang=LANG index with language specific processing.

Text Operations:

-E, --no-edge-symbol remove symbols on edge of word.

-G, --no-okurigana remove Okurigana in word.

-H, --no-hiragana ignore words consist of Hiragana only.

-K, --no-symbol remove symbols.

--decode-base64 decode base64 bodies within multipart entities.

Summarization:

-U, --no-encode-uri do not encode URI.

-x, --no-heading-summary do not make summary with HTML's headings.

Index Construction:

--update=INDEX set INDEX for updating.

-z, --check-filesize detect file size changed.

-Y, --no-delete do not detect removed documents.

-Z, --no-update do not detect update and deleted documents.

Miscellaneous:

-s, --checkpoint turn on the checkpoint mechanism.

-C, --show-config show the current configuration.

-f, --config=FILE use FILE as a config file.

```
-I, --include=FILE       include your customization FILE.
-O, --output-dir=DIR     set DIR to output the index.
-T, --template-dir=DIR   set DIR having NMZ.{head,foot,body}.*.
-q, --quiet              suppress status messages during execution.
-v, --version            show the version of namazu and exit.
-V, --verbose            be verbose.
-d, --debug              be debug mode.
    --help               show this help and exit.
    --norc               do not read the personal initialization files.
    --                   Terminate option list.

Report bugs to <http://www.namazu.org/trac-namazu/trac.cgi>
or <bug-namazu@namazu.org>.
```

2.3.2 mknmzrc settings

Various settings are possible in mknmzrc or .mknmzrc. mknmzrc normally reads configuration files in the order of

1. $(sysconfdir)/$(PACKAGE)/mknmzrc
   Usually, /usr/local/etc/namazu/mknmzrc
2. ~/.mknmzrc
3. file which is specified by -f or --config=FILE --option.

If more than one configuration file is found, they all of the files are loaded.

Installation prepares a sample configuration file $(sysconfdir)/$(PACKAGE)/mknmzrc-sample. You can copy this to $(sysconfdir)/$(PACKAGE)/mknmzrc or to ~/.mknmzrc in your home directory.

The setting details are given as comments in mknmzrc-sample.

2.3.3 Document filters

mknmz automatically identifies target file types and performs the appropriate document filtering. For HTML documents, filtering includes the extraction of <title>

or the deletion of HTML tags. The filtering is dealt with by document filters in $(datadir)/$(PACKAGE)/filter. The standard document filters are described below.

apachecache.pl

Handles an Apache's cache file.

Requirement: None

Note: --replace=apachecache::replacecode option replaces to original URI

bzip2.pl

Handles a bzip2-ed file.

Requirement: bzip2 command.

compress.pl

Handles a compress-ed file.

Requirement: compress command.

deb.pl

Handles a deb package.

Requirement: dpkg command.

dvi.pl

Handles a dvi file.

Requirement: dvi2tty

Suggested software: nkf (only for Japanese documents)

excel.pl

Handles a Microsoft Excel file.

Requirement: xlhtml, (wvSummary, a part of wvWare)

Suggested software:lv (only for Japanese documents)

gnumeric.pl

Handles a Gnumeric file.

Requirement: gzip command or Compress::Zlib perl module.

gzip.pl

Handles a gzipped file.

Requirement: gzip command or Compress::Zlib perl module.

hdml.pl

Handles a HDML file.

Requirement: None

hnf.pl

Handles a file of Hyper NIKKI System Project.

Requirement: the hnf filter is special: it requires namazu_for_hns of Hyper NIKKI System Project.

html.pl

Handles a HTML file.

Requirement: None

koffice.pl

Handles a KOffice KWord, KSpread, KPresenter, Kivio file.

Requirement: unzip, lv(only for Japanese documents)

macbinary.pl

Handles a MacBinary I,II,III file.

Avoid a problem with handle a MacBinary file.

Requirement: None

mailnews.pl

Handles a file of Mail/News and MHTML file.

Requirement: None

Note: To handle MHTML file and Attached base64 bodies, MIME::Base64 and MIME::QuotedPrint are required.(perl5.8 contains them.) --decode-base64 option is required when handling a MHTML file or base64-encoded bodies.

man.pl

Handles a man file.

Requirement: nroff, groff or jgroff

Note: To handle Japanese man, groff supporting -Tnippon is required.

mhonarc.pl

Handles a MHonArc file.

Requirement: None

mp3.pl

Handles an MP3 file's ID3 Tag

Requirement: MP3::Info perl module. (version 1.01 or later are suggested).

msword.pl

Handles a Microsoft Word file.

Requirement: wvWare

Suggested software:lv (only for Japanese documents)

ooo.pl

Handles an OpenOffice.org Writer, Calc, Impress, Draw file.

Requirement: unzip

Suggested software:lv (only for Japanese documents)

pdf.pl

Handles a PDF file.

Requirement: pdftotext, a part of xpdf (version 0.91 or later are suggested).

pipermail.pl

Handles a Mailman/pipermail file.

Requirement: None

postscript.pl

Handles a PostScript file.

Requirement: ps2ascii

powerpoint.pl

Handles a Microsoft PowerPoint file.

Requirement: pptHtml, a part of xlHtml, (wvSummary, a part of wvWare)

Suggested software:lv (only for Japanese documents)

rfc.pl

Handles an RFC file.

Requirement: None

rpm.pl

Handles an RPM package.

Requirement: rpm

rtf.pl

Handles a Microsoft Word file.

Requirement: rtf2html

taro56.pl

Handles a file of Ichitaro, a Japanese word processor, versions 5 and 6.

Requirement: None

taro7_10.pl

Handles a file of Ichitaro, a Japanese word processor, versions 7 through 13.

Requirement: unicode.pl, OLE-Storage_Lite perl module, IO-stringy perl module.

tex.pl

Handles a TeX file.

Requirement: detex

zip.pl

Handles a Zip archive files.

Requirement: unzip

Alternative: Compress::Zlib perl module, Archive::Zip perl module.

The following filters are for Windows only.

ichitaro456.pl

Handles a file of Ichitaro, a Japanese word processor, versions 4, 5 and 6.

Requirement: JSTXT

Note: JSTXT is a tool for MS-DOS.

oleexcel.pl

Handles a Microsoft Excel file.

Requirement: Microsoft Excel 97 SP1 or later, 2000, 2002(XP) or 2003

olemsword.pl

Handles a Microsoft Word file.

Requirement: Microsoft Word 97 SP1 or later, 98, 2000, 2002(XP) or 2003

olepowerpoint.pl

Handles a Microsoft PowerPoint file.

Requirement: Microsoft PowerPoint 97 SP1 or later, 2000, 2002(XP) or 2003

oletaro.pl

Handles a file of Ichitaro, a Japanese word processor, versions 4.

Requirement: Microsoft Word 97 SP1 or later, 98 or 2000

Requirement: and applicable document converter of Microsoft Office attachment.

Handles a file of Ichitaro, a Japanese word processor, versions 5 through 6.

Requirement: Microsoft Word 97 SP1 or later, 98, 2000 or 2002(XP)

Requirement: and applicable document converter of Microsoft Office attachment.

Handles a file of Ichitaro, a Japanese word processor, versions 7 through 13, 2004.

Requirement: Microsoft Word 97 SP1 or later, 98, 2000, 2002(XP) or 2003

Requirement: and applicable document converter of Microsoft Office attachment.

olertf.pl

Handles a Microsoft Word file.

Requirement: Microsoft Word 97 SP1 or later, 98, 2000, 2002(XP) or 2003

olevisio.pl

Handles a Microsoft Visio file.

Requirement: Microsoft Visio 2000, 2002 or 2003

xdoc2txt.pl

Handles a file of Microsoft Word, Excel, Powerpoint, Ichitaro, etc.

Requirement: xdoc2txt.exe

Note: xdoc2txt.exe is a tool for MS-Win32.

NOTE: We believe that mknmz will work well on both the English version and the Japanese version of Microsoft Office, but that is not yet confirmed. We would be grateful if you would notify us how it works. Thanks in advance.

2.4 namazu command

2.4.1 namazu's options

namazu 2.0.x, a search program of Namazu.

```
Usage: namazu [options] <query> [index]...
  -n, --max=NUM        set the number of documents shown to NUM.
  -w, --whence=NUM     set the first number of documents shown to NUM.
  -l, --list           print the results by listing the format.
  -s, --short          print the results in a short format.
    --result=EXT       set NMZ.result.EXT for printing the results.
    --late             sort the documents in late order.
    --early            sort the documents in early order.
```

```
  --sort=METHOD    set a sort METHOD (score, date, field:name)
  --ascending      sort in ascending order (default: descending)
-a, --all          print all results.
-c, --count        print only the number of hits.
-h, --html         print in HTML format.
-r, --no-references  do not display the reference hit counts.
-H, --page         print the links of further results.
                   (This is nearly meaningless)
-F, --form         force to print the <form> ... </form> region.
-R, --no-replace   do not replace the URI string.
-U, --no-decode-uri  do not decode the URI when printing in a plain format.
-o, --output=FILE  set the output file name to FILE.
-f, --config=FILE  set the config file name to FILE.
-C, --show-config  print the current configuration.
-q, --quiet        do not display extra messages except search results.
-d, --debug        be in debug mode.
-v, --version      show the namazu version and exit.
  --help           show this help and exit.
  --norc           do not read the personal initialization files.
  --               Terminate option list.

Report bugs to <http://www.namazu.org/trac-namazu/trac.cgi>
or <bug-namazu@namazu.org>.
```

You can specify one or more target indices in a command-line argument [index dir].... If the target index is omitted, the Default index will be treated as the target index.

By prefixing + such as +foo or +bar, you can specify a target index as a relative path from the default index.

When executed from a command line, Namazu outputs query results in simple text format. The -h option is required in order to display query results in HTML format.

If you want to display query results from the 21st hit through the 40th, type -n 20 -w 20 on the command line. Note that -w is not 21 in this example.

2.4.2 namazurc settings

Various settings are possible in mknmzrc or .mknmzrc. Namazu normally reads configuration files in the following order:

1. $(sysconfdir)/$(PACKAGE)/namazurc
   (Usually, /usr/local/etc/namazu/namazurc
2. ~/.namazurc
3. file which is specified by -f or --config=FILE --option. (In case of CGI, it is .namazurc in the directory namazu.cgi is stored)

If more than one configuration file is found, all of the files are loaded.

Installation prepares a sample configuration file $(sysconfdir)/$(PACKAGE)/namazurc-sample. You can copy this to $(sysconfdir)/$(PACKAGE)/namazurc or to ~/.namazurc in your home directory.

The setting details are given as comments in namazurc-sample.

2.4.3 Default Index

The default index is the index that is used when no other index is specifiedand it follows the rules described below.

- The default is $(localstatedir)/$(PACKAGE)/index Usually, /usr/local/var/namazu/index)
- Otherwise it is the directory which is specified by the Index directive of namazurc.

2.4.4 Template files

Template files explain the display styles of query results in HTML. The details are described below.

NMZ.head

Header of search results.

NMZ.foot

Footer of search results.

NMZ.body

Description of Namazu's query.

NMZ.tips

Tips on searching.

NMZ.result

Format of search results.

These files are available for either language. Files suffixed by `.ja` are for Japanese.

2.4.5 Form settings

Form is defined in NMZ.head. CGI variables are as follows:

query

specify a query expression.

max

specify the maximum number of query results to display at once.

result

specify the display style of query results.

sort

specify the sorting routine.

idxname

specify the name of the index to search.

subquery

specify the sub-query expression.

whence

specify where you wish to display query results.

reference

specify whether or not to display reference hit counts.

lang

specify language of search results.

3. Using Desktop Search Engine for Linux

Figure 3.1 is the screen shot of index page of DSEforLinux. Users just fill the keyword(s) they want to search in the provided text box. After filled the keyword(s), user just need to enter the Search button Search! that provided at the right side of text box. Then, DSEforLinux will process that request and display the result in a result page.

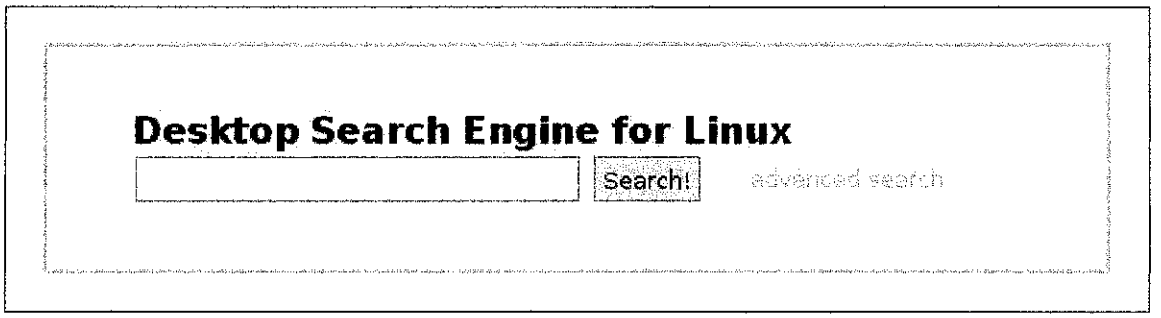

Figure 3.1 Index Search Page

3.1 Advance Search

If users want to search in advance the files that they want to search, then this system provide an advance search functions that can search in details the request. An advance search includes:

iv. Normal search that consist of:
   a. searching with all words
   b. the exact phrase of text
   c. the exclude function

v. Substring matching that consist of:
   a. prefix matching
   b. inside matching
   c. suffix matching

vi. Complex searching that consist of:
   a. regular expression
   b. grouping capabilities

Figure 3.2 is a screenshot of advance search after users click advance search button. To use this features, user just filled the string or word they want to search or do not want to search in the provided text box.

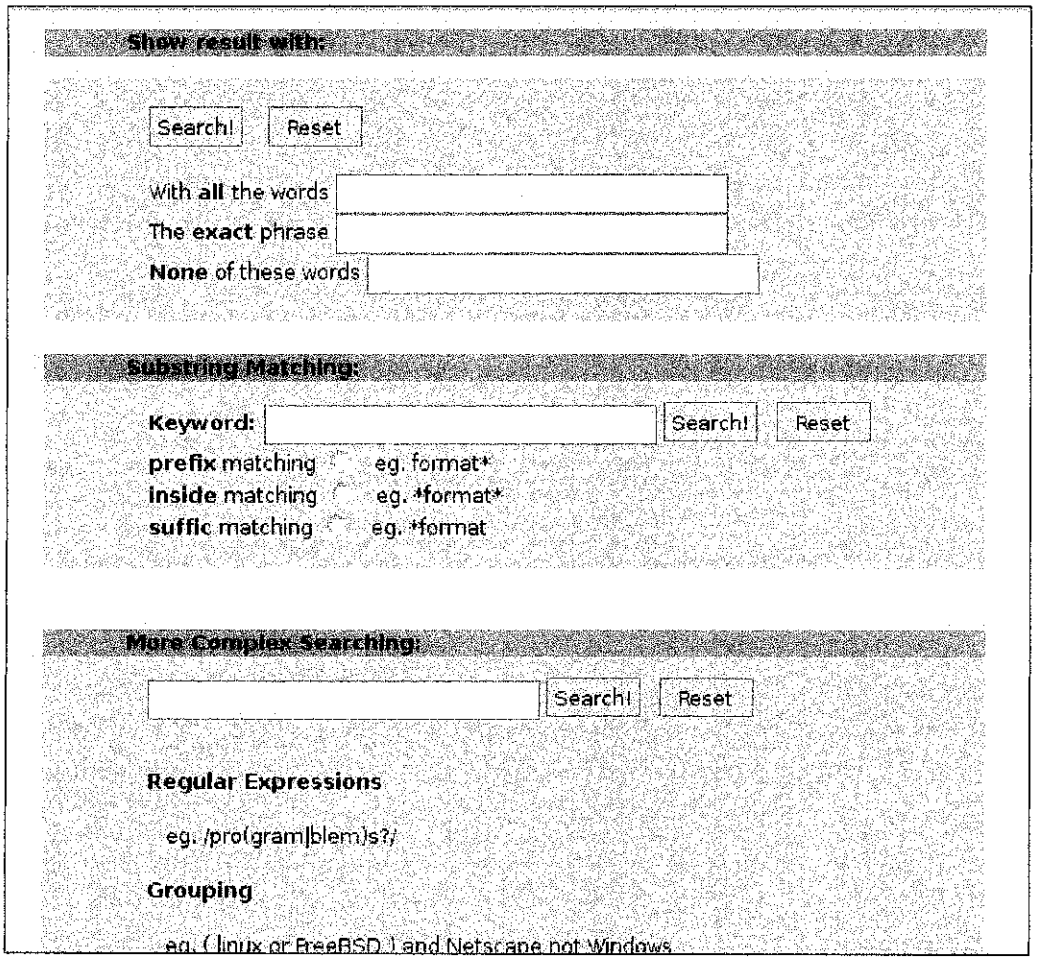

Figure 3.2 Advance Search Page

3.1.1 Search with all the words

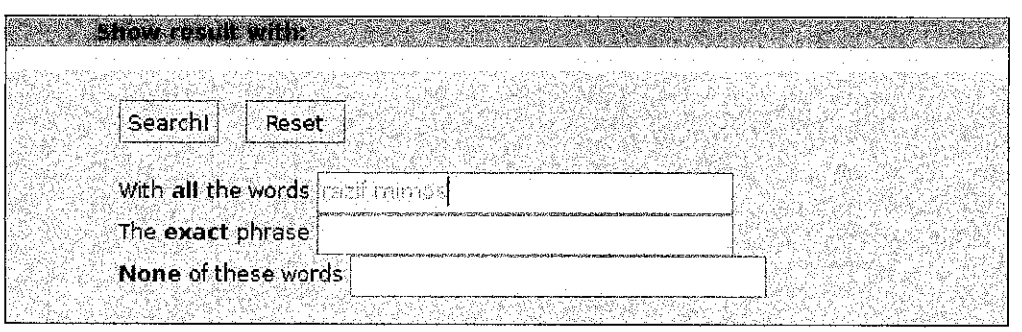

Figure 35.3 All words search

The system will search keyword(s) that enter by user in this text box and display the result in the result page. For example if user enter keyword "razif mimos" then the system will search all files that contain this "razif mimos" keyword in the hard drive. Figure 3.3 shows user entered "razif mimos" in the *with all the words* text box. If users entered three keywords in the text box, (eg. knowledge

acquisition performance) then the system will search all files that contain these keywords in the hard drive.

3.1.2 Search the exact phrase

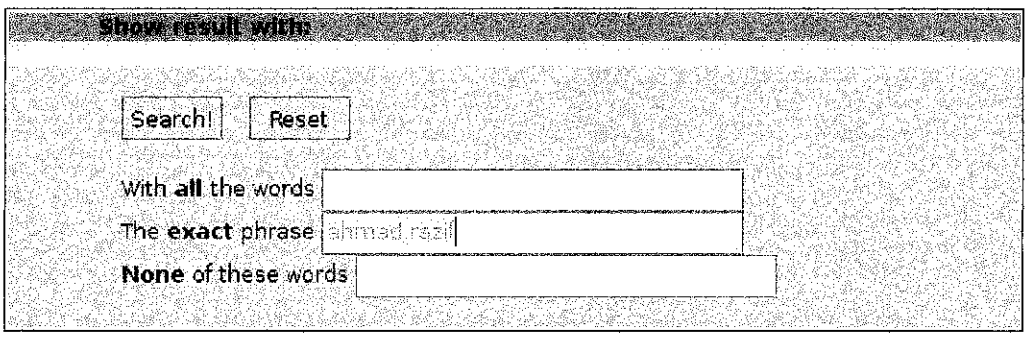

Figure 3.4 Search the exact phrase

This system also can ease the users to search file that contains an exact phrase in hard drive. Figure 3.4 shows the screen shot of the interface that can process this function. Users just fill in the exact phrase of words they want into the text box and then the system will match that exact phrase with the index file and display all file that contains that phrase as a results. For example, the users enter "*ahmad razif*" then system will search the files that contain this phrase from index files and display the result on the result page.

3.1.3 None of these words

If users have an unwanted word that they do not want to include in searching the file, then this function is the correct function to do so. This function is working if both of *all words function* and *exact phrase function* are filled with the keywords or either *all words function* or *exact phrase function* are filled in with the keyword. For example if user enters the keyword in *none of these words function* text box, then the system will unable its function.

3.1.4 All words and exact phrase search

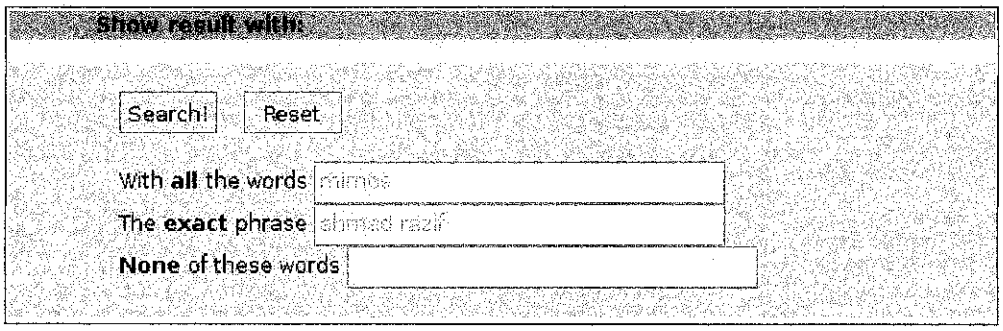

Figure 3.5 Search all words and exact phrase

Figure 3.5 shows a screenshot of searching with all words and the exact phrase. Users can use this combination of searching when they want to search a usual words and a word that in exact phrase. For example as show in the Figure 16, user filled all words with *mimos* and *ahmad razif* as an exact keyword that want to search. This system will match string mimos and a phrase ahmad razif with the index in the index file and display the result on the result page.

3.1.5 Complex search (grouping)

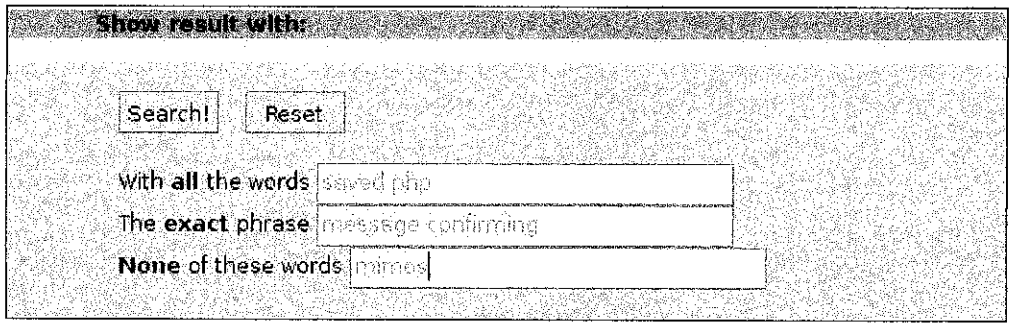

Figure 3.6 All words, exact phrase and none of the word search

Users also can make a complex search using this advance search function. User can combine these three functions to search a file. For example, users want to search file that contain words "saved" and "php" with the exact phrase of "message confirming" and not contain word "mimos". Then users just fill "saved php" in the all words function text box, "message confirming" in exact phrase function text box and "mimos" in the none of these word function text box and then click the "Search !" button at above of the function. Figure 3.6 show a screenshot of interface with the

content that filled by user to search files by using these three features for an accurate searching.

3.1.6 Prefix matching

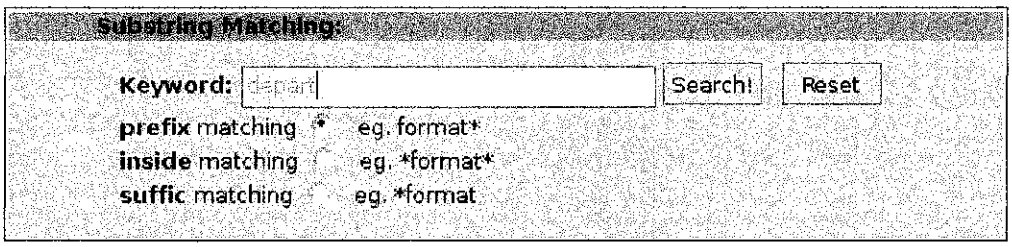

Figure 3.7 Prefix matching search

This function is to find the files with the terms that begin with the keyword that entered by user. For example if user wants to search for file that contains word begin with "depart", then they just fill in the keyword "depart" in the prefix matching text box and the system will search all files contain words begin with "depart" such as department, departure etc. Figure 3.7 shows a screenshot of a keyword filled by user to search a prefix matching that begin with the word *depart*. User must click the prefix matching radio button to search for prefix matching. If not, the system will search by default that is an inside matching.

3.1.7 Inside matching

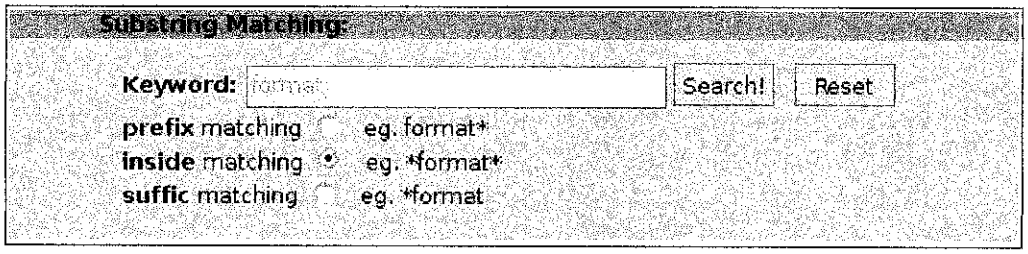

Figure 3.8 Inside matching search

This function is to find the files that have terms which contains with the keyword that entered by user. For example, if user wants to search for file that have a word that contain "*format*" keyword, then they just fill in the keyword *format* in the Inside matching text box. The system will search all files that contains word that contain a keyword "*format*" such as information, transformation, etc. Figure 3.8

shows a screenshot of keyword that has been filled by user and clicked with inside matching radio button.

3.1.8 Suffix matching

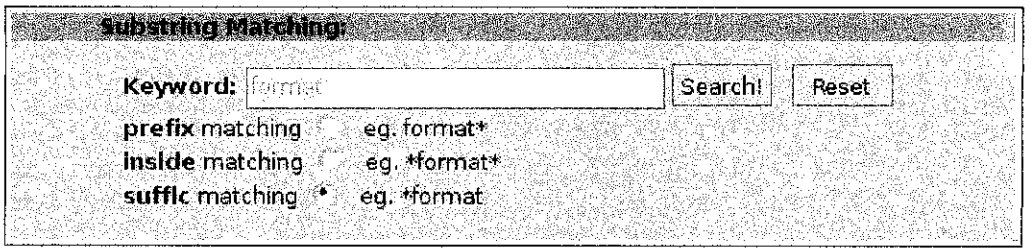

Figure 3.9 Suffix matching search

This function is to search files with terms that terminate with the filled word. For example if user want to search files that contain term "format" at the end of word, then they just enter keyword "format" in the suffix matching text box. The system will search all files that contain the keywords end with term "format" such as reformat, etc.

3.1.9 Regular expression

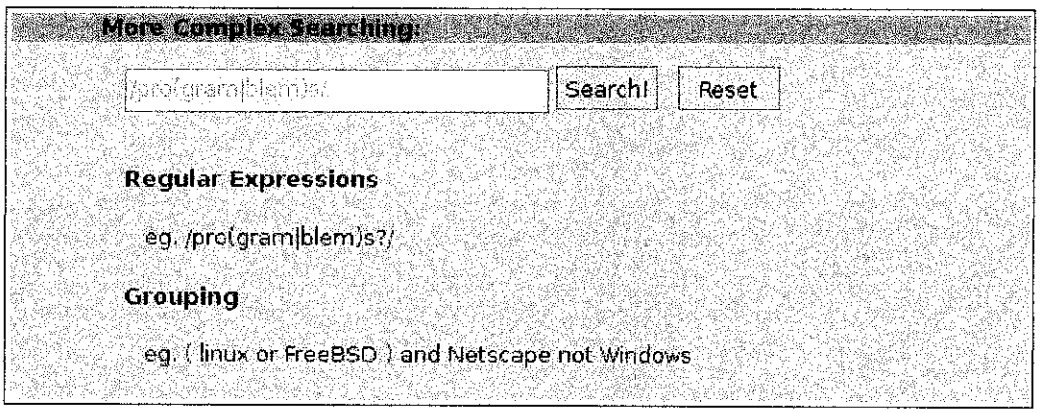

Figure 3.10 Regular expression searches

This function can search files for pattern matching. This function required users to enter the word surrounded by backslashes /.../. For example, if user want to search files that contain words program or problem then they just enter a keyword /pro(blem|gram)/. The system will search all files that contain keywords problem and program. Figure 3.10 shows a screenshot with the string that has been filled by user.

APPENDIX 2

Comparison between DSEforLinux, Namazu and Ubuntu 5.10

A-2 Comparison between DSEforLinux, Ubuntu 5.10 search tool, and Namazu(Terminal) search tool

A simple experiment has been done to measure the Accuracy and Efficiency of this system (DSEforLinux) compared with another two tools that are Ubuntu 5.10 search tool (Ubuntu 5.10) and Namazu desktop search tool using terminal (Namazu Terminal). Below is the step to measure the characteristic for accuracy, efficiency and versatility:

1. Select a folder – One folder has been chosen that consist of 8 files. The folder that has been chosen is cms folder from /home/profile/internship/cms directory.
2. Select the keywords that will be use as a measurement – the keywords that has been chosen are *management, razif* and *page*.
3. Count manually and verify the files that contain the selected keywords. The details of files are recorded (as Table 3).
4. Open desktop search tools that will use to search the files with those 3 keywords – open DSEforLinux (Figure 7), Ubuntu 5.10 (Figure 34), and Namazu(Terminal) (Figure 35).
5. Enter the keyword, and click or execute the system. If using DSEforLinux, click the Search button, Ubuntu 5.10 click Find button and with Namazu type a command *namazu management /var/www/namazu* . Time of the system processing the query and display the result has been taken and record.
6. The result was compared by time, total files, and total result files (related files) with those three search tool.
7. Then an analysis has been done to make a conclusion.

Tools that have been used to measure the time were a stop watch that has a minutes, seconds and millisecond features. Times are record during the execution of system until the system completely displays the results.

Figure 1 is a snapshot of folder and files that contains in that folder. These files used as a measurement for the accuracy, efficiency and versatility of this system.

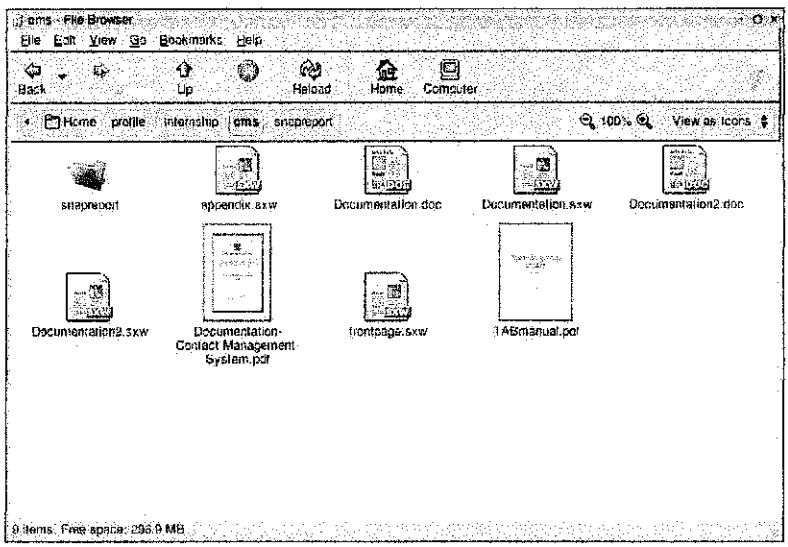

Figure 36: cms folder and files

Table 1 below is a record detail of files in the cms folder from /home/profile/internship/cms directory. This folder consists of 8 files. From folder have 7 files that contain management keyword, 6 files with razif keyword and 6 files with page keyword.

	Files	**Keyword**		
		management	*razif*	*page*
1	appendix.sxw	0	0	0
2	Documentation.doc	95	1	16
3	Documentation.sxw	95	1	16
4	Documentation2.doc	69	2	16
5	Documentation2.sxw	69	2	16
6	Documentation-Contact Management System.pdf	102	2	16
7	frantpage.sxw	6	1	0
8	TABmanual.pdf	6	0	16
	Total file that contains the keyword	7	6	6

Table 1: Details files in the CMS folder

Table 2 shows a results table that has been recorded. From his table we can see the time (in second) that has been taken for each tool to process the query. Besides, this record also shows a number of results that contain in the CMS folder and the total files that have been searched by each tool.

Tools		**Keywords**		
		management	*razif*	*page*
DSEforLinux	*Time(sec)*	2"73	1"30	1"10
	Result	7	6	6
	Total files	206	50	1081
Ubuntu 5.10	*Time(sec)*	1'42"68	1'16"75	2'10"59
	Result	1	1	7
	Total files	178	8	987
Namazu(Terminal)	*Time(sec)*	1"17	0"85	0"78
	Result	7	2	6
	Total files	206	50	1081

Table 2: Results table

A-2.1 Accuracy

A calculation has been made to measure the accuracy of this system compared to the operating system files search tool. To measure the accuracy of the system, results were divided by number of files that is the number of files in the folder that contains that keyword.

Equation to calculate the accuracy of the systems:

Percentage Accuracy of systems = (exact results from folder / total files in folder) x 100

After the value of accuracy calculated, then the average value calculated. **Table 3** shows the results of the accuracy between DSEforLinux (Desktop Search Engine for Linux), Ubuntu 5.10 (Linux Ubuntu 5.10 files search tool), and Namazu Terminal search tool.

Tools		**Keywords**		
		management	*razif*	*page*
DSEforLinux	*Files*	7	6	6
	Result	7	6	6
	Accuracy	100%	100%	100%
	Average result/file	**100%**		
Ubuntu 5.10	*Files*	7	6	6
	Result	1	1	7
	Accuracy	14.29%	16.67%	116.67%
	Average result/file	**49.21%**		
Namazu(Terminal)	*Files*	7	6	6
	Result	7	6	6
	Accuracy	100%	100%	100%
	Average result/file	**100%**		

Table 3: Results of accuracy comparison between DSEforLinux, Ubuntu 5.10 and Namazu(Terminal)

From the results of **Table 3**, we can see that DSEforLinux and Namazu(Terminal) has 100% accuracy when they can search all files that contain in the cms folder. The average result/files of Ubuntu 5.10 just 49.21% because they cannot search certain files in the cms folder.

A-2.2 Efficiency

Table 4 shows results of efficiency comparison between DSEforLinux, Ubuntu 5.10 and Namazu(Terminal). The table shows total files, times recorded for searching process and the efficiency results for these three search tools. The efficiency results are got from the total files divided by time(sec). After got the value of efficiency, then the average of efficiency calculated. Below is all data that got from the record and calculation involved.

Tools		**Keywords**		
		management	*razif*	*page*
DSEforLinux	*Total Files*	206	50	75.46
	Time(sec)	2"73	1"30	38.46
	Efficiency	75.46	38.46	982.73
	Average file/second	**365.55**		
Ubuntu 5.10	*Total Files*	178	8	987
	Time(sec)	1'42"60	1'16"75	2'10"59
	Efficiency	1.25	0.069	4.69
	Average file/second	**2.00**		
Namazu(Terminal)	*Total Files*	206	50	75.46
	Time(sec)	1"17	0"85	0"78
	Efficiency	176.07	58.82	1385.90
	Average file/second	**540.26**		

Table 4: Results of efficiency comparison between DSEforLinux, Ubuntu 5.10 and Namazu(Terminal)

From the results of **Table 4**, we can see that average file/second for DSEforLinux is 365.55. That means this search tools can search and display the

results in 365.55 files per second. The result for Ubuntu 5.10 is 2.00 files per second and the result for Namazu(Terminal) is 540.26 files per second.

DISCUSSION

Comparison of Accuracy between DSEL, Ubuntu File Search Tool and Namazu

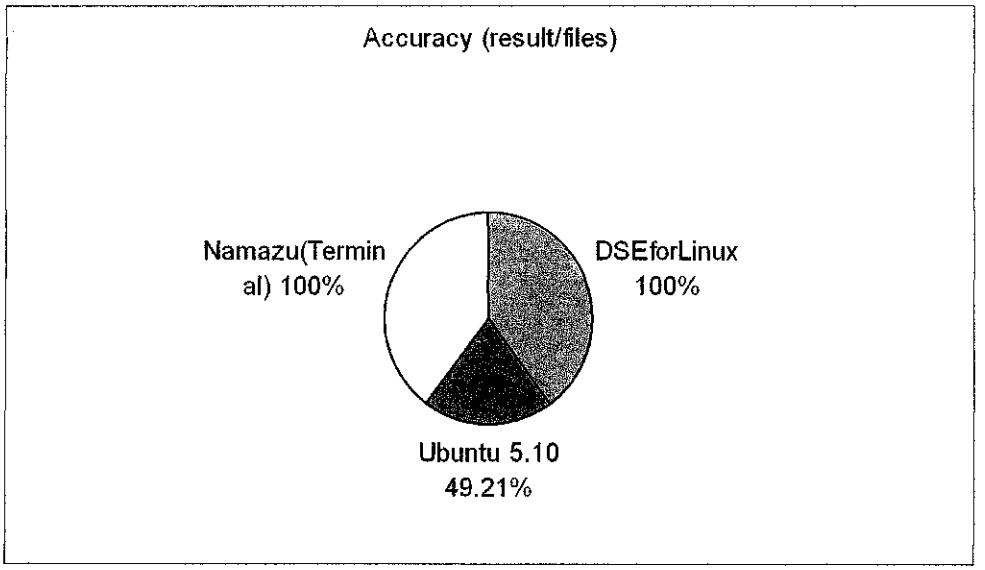

Figure 2: Accuracy graph of DSEforLinux, Ubuntu 5.10, and Namazu(Terminal)

Figure 2 shows graph accuracy between DSEforLinux, Ubuntu 5.10 and Namazu (Terminal). The data for this graph is taken from **Table 3**. From this graph we can say that DSEforLinux and Namazu(Terminal) have the same percentage of accuracy that is 100% compared to Ubuntu 5.10 that only 49.21%. That mean, DSEforLinux and Namazu(Terminal) are more accurate in searching the files compared to Ubuntu 5.10. The same value of DSEforLinux and Namazu(Terminal) because DSEforLinux is using the same indexer and index files that used by Namazu (Terminal). The engine of the DSEforLinux is taken from Namazu.

Comparison of Efficiency between DSEL, Ubuntu File Search Tool and Namazu

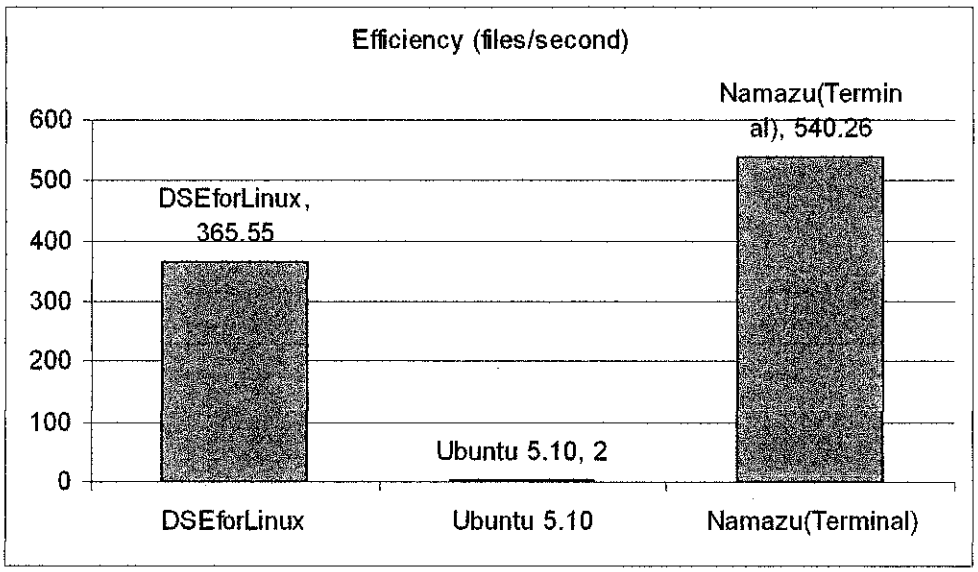

Figure 37 Efficiency graph of DSEforLinux, Ubuntu 5.10 and Namazu(Terminal)

Figure 3 shows graph of efficiency between DSEforLinux, Ubuntu 5.10 and Namazu(Terminal). The data for this graph is taken from **Table 4**. From this graph we can say that Namazu(Terminal) have the highest percentage of efficiency that are 540.26 files per second compared to DSEforLinux 365.55 files per second and Ubuntu 5.10 with 2 files per second. That mean, Namazu(Terminal) can process the query faster than DSEforLinux and Ubuntu 5.10. The factor that make Namazu (Terminal) can search faster than other two tools is Namazu not required an interface like DSEforLinux and it's process the query in the terminal (internal process) that are lighter than by using an external application. DSEforLinux required the web-base interface to give an instruction to internal system and get back the data from internal system to display the result. That takes a few second to process but both DSEforLinux and Namazu (Terminal) are using index files that can make the searching and matching process faster. Ubuntu 5.10 slower than others two tools because this tools need to search every single file in the folder and display the result on the screen. This Ubuntu 5.10 also not provided with the index files that slower the process of matching the keyword.

Below is the summary table of efficiency characteristic between these three tools:

Desktop Search Tools	Descriptions
DSEforLinux (Desktop Search Engine for Linux)	- Fast searching the files - Results displayed slower than Namazu because this tool must load a web-based page.
Ubuntu 5.10 Search File Tool	- Search files slower than DSEL and Namazu - Display results slower than another 2 tools and just display the file name and location without description or other details
Namazu (Terminal)	- Fast searching the files - Results displayed faster than DSEL because results are displayed directly in terminal.

Table 5: Comparison of Efficiency between DSEL, Ubuntu File Search Tool and Namazu