NAME

ASPseek - Advanced Internet Search Engine

DESCRIPTION

ASPseek is an Internet search engine with all the bells and whistles you expect from such a product. This page introduces **ASPseek** for the new users, explains its basic concepts and provides references to more information.

ASPseek is written in C++, using STL library and mix of SQL server tables and binary files as a data storage. **ASPseek** consists of indexing program **index**(1), search daemon **searchd**(1), and CGI search front-end **s.cgi**(1).

index(1) walks across the sites and stores found pages in a special data structures (some data is stored in SQL, some other data that affects search speed are stored in binary files in `/usr/local/aspseek/var` directory). **searchd**(1) listens to and performs search queries, issued by search front-end **s.cgi**(1). Front-end then formats the results into nice-looking HTML page.

ASPseek is optimized for multiple sites and medium loads, and can be used for search within several million pages (URLs). User can search for words and phrases, use wildcards, and do a Boolean search.

Search scope can be limited to time period given, site, subdirectory of site, Web space (set of sites), or particular parts of HTML documents (notably title, body, description and keywords). Search results can be sorted by relevance or by date.

Below is a list of **ASPseek** features.

Ability to index and search through several millions of documents

Using **ASPseek**, you can build a database and search through many sites, and results for each query will be returned fast even if you have a few millions of documents indexed. Of course, this depends on hardware, so don't expect "good old" i486 machine to handle every site in .com domain. Everything depends on CPU(s), memory, disk speed etc. So do your own tests before you buy dedicated hardware.

The fact that **ASPseek** is optimized for high volumes should not stop you from using it to search your own site that contains few hundred of documents - it works there as well.

Very good relevancy of results

The purpose of search engine is to find what user wants. There can be thousands of URLs found as a result of search query, but it can all be irrelevant, so user will be unsatisfied.

Output results in **ASPseek** are sorted by relevancy (or rank), but rank calculation is not an easy task. Developers tried their best to incorporate greatest and latest techniques into **ASPseek** engine while maintaining good search speed.

Advanced search capabilities

User can ask search engine to search not only for the set of words, but also for the whole phrase. To search for phrase, just surround it with quotation marks, like this:

```
"many years ago"
```

Phrase searching considerably improves results, and this feature is rated to be the most useful by people.

If you do know the exact phrase, but forgot a singe word in the middle of the phrase, you can use askerisk mark (*****) instead of that word. So, query

```
"many * ago"
```

will return results with phrases like "*many years ago*", "*many days ago*" etc.

Boolean search is a search with logical expression. Expression can be composed using AND and OR operators, subexpressions can be grouped using parenthesis. Example:

```
(some OR any) AND (days OR months OR years)
```

Subexpression in parenthesis can be another boolean expression or just word, pattern, or phrase.

You can exclude the word from search by putting a minus sign before it. So, pages with that word will be excluded from search results. Example:

```
search engine -proprietary
```

Search by pattern allows to search documents containing words that match specified pattern. Character

’?’ means any character, character ’*’ means sequence of any characters. For example, to find all documents containing words beginning with provider, type:

```
provider*
```

ASPseek allows to narrow search up to one or few sites. For example, to find all documents containing word *bubble* on site *www.mysite.org*, type:

```
bubble site: www.mysite.org
```

The same way, if you want results from all sites parked at *mysite.com* (like *www.mysite.com*, *lib.mysite.com*, *smth.other.mysite.com*), you can just type:

```
bubble site: mysite.org
```

You can even use **site: org** to get results from all *.org* domains that are indexed.

Excluding the results from given site(s) are done in the same way:

```
bubble -site: mysite.org
```

Several **site:** limits can be used together.

You can also narrow result to pages modified (or created) within specified time period, which can be set in few ways: some time back from now, before/after given date, or between two dates. For example, you can narrow results to pages those were modified not earlier than one week ago.

And finally, you can find all pages those link to specified page, like this:

```
link: www.aspseek.org
```

It will show you all pages in the database that have links to *www.aspseek.org*.

You can combine any of the above facilities, as far as it makes sense.

Ispell support

When **ASPseek** is used with ispell support, **searchd**(1) can optionally find all forms for all specified words (example: *create* --> *create* OR *created* OR *creates*). So, it allows you to find the word in all of different forms.

Unicode storage mode

ASPseek can store information about documents in Unicode, thus making possible to implement a multi-language search engine. So, you can index and search the documents in English, Russian and even Chinese, all in one database.

HTTP, HTTPS, HTTP proxy, FTP (via proxy) protocols

As **ASPseek** is a Web search engine, it uses HTTP protocol to index sites. **ASPseek** also supports secure https:// protocol. FTP protocol is not supported directly, but you can use proxy (like squid) and index FTP sites via proxy.

ASPseek supports "basic authorization" feature of HTTP so you can index password-protected areas (for example private information in your intranet).

Text/html and text/plain document types support

ASPseek can understand documents written in HTML, and plain text documents. These are the most popular formats in Internet.

Other formats, such as PDF, RTF, etc, can be supported with the help of any external program/script which is able to convert that formats to HTML or plain text.

Multithreaded design, async DNS resolver etc

ASPseek uses POSIX threads, that means that one process have many threads running in parallel. So index downloads documents from many sites, and search daemon processes many search queries simultaneously. This not only helps **ASPseek** to scale well on SMP (multiprocessor) systems, but also improves indexing speed, because in case of one thread most time will be spent on waiting for data from network.

One thing that slow indexing process down a lot is DNS lookup (a process of determining IP address using server name). To avoid delays, asynchronous lookups (lookup is done by separate dedicated processes) and IP address cache are implemented.

Stopwords

Stopwords are a words that have no meaning by itself. Examples: *is*, *are*, *at*, *this*. Searching for *at* is useless, so such words are excluded from search query. Stopwords are also excluded from database during indexing, so database becomes smaller and faster.

There is no "built-in" stopwords in **ASPseek**, they are loaded during start-up from files. Many stopword files for different languages comes with **ASPseek**.

Charset guesser

Some broken or misconfigured servers don't tell clients the charset in which they provide content. If you are indexing such servers, or using **ASPseek** to index ftp servers (FTP protocol does know nothing about charsets), charset guesser can be used to deal with it. Charset guesser uses word frequency tables (called langmaps) to determine correct charset.

Robot exclusion standard (robots.txt) support

ASPseek fully supports this standard. It is intended for web site authors for telling the robot (for example, **ASPseek**'s **index**(1)) to skip indexing some directories of their sites.

For more information see `http://www.robotstxt.org/wc/robots.html`

Settings to control network bandwidth usage and Web servers load

You can precisely control network bandwidth that **index**(1) uses. Exactly, you can limit the bandwidth (expressed in bytes per second) used by **index**(1) for given time-of-day. For example, you can limit the bandwidth during business hours so people at your office will not experience slow Internet.

You can also set the minimum time between two queries to the same Web server, so it will not be overloaded and got down to its knees while you run **index**(1).

Real-time asynchronous indexing

Some search engines requires that search should be stopped for the time of database update. **ASPseek** does not need it, so you can search non-stop.

More to say, there is a special mode of indexing called "real-time" indexing. You can use it for small number of documents, and as far as such document is downloaded and processed, changes are immediately visible in search interface. This feature is a great help if you are building search engine for pages with rapidly-changing content such as online news etc.

Note that number of documents in "real-time" database is limited. It's about 1000 on our hardware (your mileage may vary), and the more documents you have in "real-time" database, the slower will be speed of indexing into that (and only that) database. This will not affect search speed though.

Documents from "real-time" database are moved to normal database after running **index**(1) in a normal way.

Sorting results by relevance or by date

Search engines usually returns most relevant results first. But if you are looking for latest pages, you can tell **ASPseek** to sort results by last modification date, so recently modified (or created) pages will be displayed first.

Excerpts, query words highlighting

Excerpt is a piece of found document with words searched for highlighted, just to give an idea of what the document is about. You can customize the number of excerpts displaying and their length. If you will disable excerpts, the beginning of document will be displayed.

Every found document is accompanied with the "Cached" link. **ASPseek** keeps a local compressed copy of every document processed, so user can see the the whole document with (optional) highlighted words that were searched for, even if it has been removed from original site (that happens sometimes).

Grouping results by site

Results from one site can be grouped together. If grouping by sites is on, only two results are displayed from the same site by default, and user can see other pages from the same site by following a "More results from ..." link.

Clones

Clones are identical documents at different locations. They are detected and grouped together, so user will not be presented with a page full of URLs to the identical documents.

Clone detection is usually limited by one site (so identical documents from different sites are not counted as clones), but you can change this by recompiling **ASPseek** with `--disable-clones-by-site` option.

Spaces and subsets

Space is the set of sites. So, if you want to provide the search narrowed to some area, you can create a space and search within that space. Only whole sites (e.g. *http://www.mysite.com/*) are allowed to be included in space.

Subsets can also be used to restrict the search. You can create subset and put URL mask (like *http://www.mysite.com/mydir/%*) into that, and then limit search scope to only given subset.

You can restrict search scope to not only one but several subsets or spaces.

HTML templates for easy-to-customize search results

You can customize your search pages, so they will look like and be seamlessly integrated with the rest of your site. This is done by simple editing of search template file.

SEE ALSO

index(1), **searchd**(1), **s.cgi**(1), **aspseek.conf**(5), **searchd.conf**(5), **s.htm**(5), **aspseek-sql**(5), **http://www.aspseek.org/**.

AUTHORS

Copyright (C) 2000, 2001, 2002 by SWsoft.
This man page by Kir Kolyshkin <kir@asplinux.ru>