1 The Swish-e README File

1.2.1 Key features

- Quickly index a large number of documents in different formats including

- Suppoo3 - by on-line user and developer groups.

1.3 Where

See INSTALL for information on creating the PDF and Postscript versions of the documentation.

Patches or updates to the documentation should be done against the POD files, located in the pod directory

.

2.1 OVERVIEW

This

2.2 SYSTEM REQUIREMENTS

Swish-e makes use of a number of libraries and tools that are not distributed with Swish-e. Some libraries need to be installed before building Swish-e from source; other tools can be installed at any time. See below for details.

2.2.1 Software Requirements

Swish-e is written in C. It has been tested on a number of platforms, including Sun/Solaris, Dec Alpha, BSD, Linux, Mac OS X, and Open VMS.

The GNU C compiler (gcc) and GNU make are strongly recommended. Repeat: you will find life easier if you use the GNU tools.

2.2.2 Optional but Recommended Packages

Most of the packages listed below are available as easily installable packages. Check with your operating system vendor or install them from source. Most are very common packages that may already be installed on your computer.

As noted below, some packages need to be installed before building Swish-e from source, while others may be added after Swish-e is installed.

- **Libxml2**

  libxml2 is very strongly recommended. It is used for parsing both HTML and XML files. Swish-e can be built and installed without *libxml2*, but the HTML parser that is built into Swish-e is not as accurate as *libxml2*.

  ```
      http://xmlsoft.org/
  ```

  libxml2 must be installed before Swish-e is built, or it will not be used.

  If *libxml2* is installed in a non-standard location (e.g., *libxml2* is built with

Zlib must be installed before building Swish-e.

- **Perl Modules**

- **Indexing MS Word Documents**

  Indexing MS Word documents requires the Catdocprogram.

  ```
  http://www.45.free.net/~vitus/ice/catdoc
  ```

  Catdocmay be added after Swish-e is installed.

- **Indexing MP3 ID3 Tags**

  Indexing MP3 ID3 Tags requires the `MP3::Tag` Perl module. See http://search.cpan.org. `MP3::Tag` may be installed after Swish-e is installed.

- **Indexing MS Excel Files**

  Indexing MS Excel files is supported by the following Perl modules, also available at http://search.cpan.org.

  ```
  Spreadsheet::ParseExcel
  HTML::Entities
  ```

  These Perl modules may be installed after Swish-e is installed.

2.3 INSTALLATION

Here are brief

```
$ ./configure
$ make
$ make check
...
==================
All 3 tests passed
==================
```

```
$ su root  (or use sudo)
(enter password)
```

```
# make install
# exit
$ swish-e -V
SWISH-E 2.4.0
```

IMPORTANT: Once Swish-e is installed, do not run it as the superuser (root) -- root is only required during the installation step, when installing into system directories. Please do not break this rule.

NOTE: If you are upgrading from an older version of Swish-e, be sure and review the CHANGES file. Old index files may not be compatible wc1 (e ou:0 $ man 120.615 fntern
Swish-e can be run from thebuiloe directory

Note that the current directory is not where Swish-e was unpacked.

Swish-e uses a *configure* script. *configure* has many options, but it uses reasonable and standard defaults. Running

```
      $ ../swish-e-2.4.0/configure --help
```

will display the options.

Two options are of common interest: `--prefix` sets the top-level installation directory; `--disable-shared` will link Swish-e statically, whichopty be needed on some platforms (Solaris 2.6, perhaps).

Platforms pty require varying link instructions when libraries are installed in non-standard locations. Swish-e uses the GNU *autoconf* tools for building the package. *autoconf* is good at building and testing, but still requires you to provide information appropriate for your platform. This pty mean reading the manual page for your compiler and linker to see how to specify non-standard file locations.

For most Unix-type platforms, you can use `LDFLAGS` and `CPPFLAGS` environment variables to specify paths to "include" (header) files and to libraries that are not in standard locations.

libxml2 [illegible]

```
$ make install
$ $HOME/local/bin/swish-e -V
SWISH-E 2.4.0
```

Note the use of double quotes in the `LDFLAGS` line above. This allows `$HOME`

Note that the Perl modules are *not* installed in the system Perl library. Swish-e and the Perl scripts that

The list is typically *very low traffic*, so it won’t overload your inbox. Please take the time to subscribe. See http://Swish-e.org.

- The exact version of Swish-e that you are using. Running Swish-e with the `-V` switch will print the version number. Also, supply the output from `uname -a` or similar command that identifies the operating system you are running on. If you are running an old version of swish, be prepared for a response of "upgrade" to your question.

- A summary of the problem. This should include the commands issued (e.g. for indexing or

2.5.2 Creating PDF and Postscript documentation

The HTML version of the Swish-e documentation was created with `Pod::HtmlPsPdf`, a package of Perl modules written and/or modified by Stas Bekman to automate the conversion of documents

This will spider the web server running on the local host. The `-S` option defines the input source method to be "http", `-i` specfies the URL to spider, and `-v`

A special form of the `-S prog` input source method is:

```
./myprog --option | swish-e -S prog -i stdin -c config
```

This allows running Swish-e from a program (instead of running the external program from Swish-e). So, this also can be done as:

```
./myprog --option > outfile
swish-e -S prog -i stdin -c config < outfile
```

or

```
./myprog --option > outfile
cat outfile | swish-e -S prog -i stdin -c config
```

One final note about the `-S prog` input source method. The program specified

```
    swish-e -w swishtitle=(foo or bar) or swishdefault=(baz)
```

The Metaname "swishdefault" is the name that is used by Swish-e if no other name is specified. The following two searches are thus equivalent:

```
    swish-e -w foo
    swish-e -w swishdefault=foo
```

When indexing

This example will work with the documents in the *tests* directory. You may wish to review the *tests/test.config* configuration file used for the `make test` tests.

For example, a simple configuration file (*swish-e.conf*):

```
# Use spider.pl’s default configuration and specify the URL to spider
SwishProgParameters default http://localhost/apache_docs/index.html
```

```
# Allow extra searching by title, path
Metanames swishtitle swishdocpath
```

```
# Set StoreDescription for each parser
#  to display context with search results
StoreDescription TXT* 10000
StoreDescription HTML* <body> 10000
```

3. **Generate the Index**

Now, run Swish-e to create the index:

```
~/web_index$ swish-e -S prog -c swish.conf
```

Also note that Swish-e knows to find `spider.pl` at `/usr/local/lib/swish-e/spider.pl`. The script installation directory (called

You may wish to read the Swish-e FAQ question on filtering, before continuing here. How do I filter documents?

2.7.1 Filtering Overview

There are two ways to filter documents with Swish-e. Both are described in the SWISH-CONFIG man page. They use the `FileFilter` directive and the `SWISH::Filter` Perl module.

The `FileFilter` directive is a general-purpose method of filtering. It allows running of an external program for each document processed (based on file extension), and requires one or more external programs. These programs open an input file, convert as needed, and write their output to standard output.

Previous versions of Swish-e (before 2.4.0) used a collection of filter programs for converting will cause all `.pdf` files to be filtered through the *pdftotext* program (part of the *Xpdf* package) and to parse the resulting output (from *pdftotext*) with the text ("TXT") parser.

The other way to filter documents is to use a `-S prog` prograam and convert the documents before passing them onto Swish-e.

For example, `spider.pl` makes use of the `SWISH::Filter"` Perl module, included with the Swish-e distribution. `SWISH::Filter` is passed a document and the document's content type; it looks for modules and utilities to convert the document into one of the types that Swish-e can index.

3 CHANGES - List of revisions

Michael Levy found an error in the table used to translate 8859-1 to ascii7. Luckily, it was an upper case translation and the table is only used on lower case characters.

MetaNamesRank documentation

Changed the ’not yet implemented’ caveat to ’implemented but experimental’.

Added Continuation option to config processing

You can now use continuation lines in the config file:

```
         IgnoreWords \
             the \
             am \
             is \
             are \
             was
```

There may not be any characters following the backslash.

Fixed Buzzwords (and other word lists entered in the config)

Words entered in config were not converted to lower case before storing in the index.

Fixed metaname mapping problem in Merge

Peter Karman found an error when merging indexes where the source indexes had the same metanames, but listed in a different order in their config files. Words would then be indexed under the wrong metaID number in the output index.

SWISH::Filters [illegible] inde.

Change in way symbolic links are followed

John-Marc Chandonia pointed out that if a symlink is skipped by FileRules, then the actual file/directory is marked as "already seen" and cannot be indexed by other links or directly.

Now, files and directories are not marked "already seen" until after passing FileRules (i.e after a file is actually indexed or a directory is processed).

Could not set SwishSetSort() more than once

Fixed compile problem with building incremental indexing mode. This is an experimental option

- **Fixed segfault when generating warnings while parsing**

  Parser.c was incorrectly calling `warning()` incorrectly. And `-`Wall was not catching this!

3.1.8 Version 2.4.0 (Release Candidate 2) September 10, 2003

- **Indexing HTML**

Swish replaces all series of low ASCII chars with a single space character

- **Fixed code when removing files**

  Was not correctly removing words from index when parser aborted [jmruiz]

- **Merge segfault**

Fixed bug in xml.c reported by Rodney Barnett when very long words were indexed. [moseley]

You can now use -S prog to use an external include whitespace) as a single parameter. The backslash can also be used to escape the following

Currenly, IgnoreFirstChar and IgnoreLastChar will be stripped before processing Buzzwords.

In the future we may use separate IgnoreFirst/Last settings for buzzwords since, for example, you may wish to index all `+` within Swish-e words, but strip `+` from the start/end of Swish-e words, but not from the buzzword `C++`.

- **New directives: PropertyNamesNumeric PropertyNamesDate**

  Before Swish-e 2.2 all user-defined document properties were stored in the index as strings. PropertyNamesNumeric and PropertyNamesDate tell it that a property should be stored in binary format. This allows for correct sorting of numeric properties.

  Currenly, only integers can be stored, such as a unix timestamp. (Swish-e uses `strtoul` to convert the number to an unsigned long internally.)

PropertyNamesDate only indicates to Swish-e that a number is a unix timestamp, and to display the property as a formatted time when printing results. Swish does not currently parse date strings; you must provide a unix timestamp.

- **New directive: MetaNameAlias**

  You may now create alias names for MetaNames. This allow you to map or group multiple names to the same MetaName.

- **New directive: PropertyNameAlias**

  Creates aliases for a PropertyName.

- **New directive: PropertyNamesMaxLength**

  Sets the max length of a text property.

- **New directive: HTMLLinksMetaName**

  Defines a metaname to use for indexing href links in HTML documents. Available

XMLClassAttributes

4.1 Swish-e CONFIGURATION FILE

What files Swish-e indexes and how they are indexed, and where the index is written can be controlled by a configuration file.

configura

The configuration file is a text file composed of comments, blank lines, and **configuration**

```
FileFilterMatch pdftotext "'%p' -" "/\\.pdf$/"
```

Swish-e will convert tb653ouble backslash into a single backslash before passing tb65parameter to tb6 regular5expression compiler.5

Commented example configuration files are included in tb65*conf* directory of tb65Swish-edistribution.5

Some command line arguments can override directives spec

[illegible]

edXMLAttributes [DISABLE|error|ignore|index|auto]

ming [yes|NO]

x [yes|NO]

list of words|File: path]

rs *`string` of characters*

*

tributes `f via the  ( )Ti /R306 11 1Ti` `BT /R 792. 183 g 93The defaulethe  (.5 0`

Index file specifies the location of the generated index file. If not specified, Swish-e will create the file *index.swish-e* in the current directory.

```
IndexFile /usr/local/swish/site.index
```

obeyRobotsNoIndex [yes|NO]

When enabled, Swish-e will not index any HTML file that contains:

```
<meta name="robots"Tj ETent="noindex">
```

The default is to ignore these meta tags and index the document. This tag is described at http://www.robotstxt.org/wc/exclusion.html.

Note: This featTre is only available with the libxml2 HTML parser.

Also, if you are using the libxml2 parser (HTML2 and XML2) then you can use the following comments in your documents to prevent indexing:

```
   <!-- SwishCommand noindex -->
   <!-- SwishCommand index -->
```

and/or these may be used also:

For example, these are very helpful to prevent indexing of common headers, footers, and menus.

NOTE: This following

SwishSearchOperators <and-word> <or-word> <not-word>

NOTE: This following item is currently not available.

Using this config directive you can change the boolean search operators of Swish-e, e.g. to adapt these to you7 language. The default is: AND OR NOT

Example (german):

4.1.3 *Administrative Headers Directives*

Swish-e stores configuration information in the header of the index file. This information can be retrieved while searching or by functions in the Swish-e C library. There are a number of fields available for your own use. None of these fields are required:

IndexName *text*

IndexDescription *text*

IndexPointer *text*

IndexAdmin *text*

These vari information that goes into index files to help users and admin Index-
Name should be the name of your index, like a book title. IndexDescription is a short description of
n. IndexPointer should be a pointer to the origi-
uld be the name of the index maintainer and can
0 Td (ma)T 3.04430 T3Td (e thgeTj 25.041 - 9d (infortj -39626 0 Td (for drtjthe x files to hito i D/M/Y-396.106.62.2 Td 0 Tw (own .m
otet goesinforutoar.337 0 Td (min)Tat16.a

For the **fs** method of access **IndexDir**

```
    IndexDir ./myprogram.pl
```

See prog for details.

Note: Not all directives work with all methods.

NoContents *list of file suffixes*

Files with these suffixes will **not** have their contents indexed, but will have their path name (file name) indexed instead.

If the file’s tyla37s HTML or HTML2 (as set byIndexContents or DefaultContents) then the file will be parsed for a HTML title and that title will be indexed. Note that you must set the file’s tyla3with IndexContentsDefaultContents

For example, you may index your files locally (with the File system indexing method), yet return a (URs ona)Tj 4014.872 -13.2 Td 7 Tw yourwebT1orver. x

FileFilter .pdag 1pda2html
IndexContent HTML .pda

If ConvertHTMLEntities

Then you can limit your search to just META **meta1** like this:

```
swish-e -w ’meta1=(apples or oranges)’
```

You may nest the XML and the start/end tag versions:

```
<keywords>
    <tag1>
        some content
    </tag1>
    <tag2>
        some other content
    </tag2>
<keywords>
```

Then you can search in both tag2 and tag2 with:

```
swish-e -w ’keywords=(query words)’
```

Swish-e indexes all text as some metaname. The default is `swishdefault`, so these two queries are the same:

HTMLLinksMetaName *metaname*

Allows indexing of HTML links. Normally, HTML links (href tags) are not indexed by Swish-e. This directive

For example, by

[redacted]

```
        <person class="first">
            John
        </person>
        <person class="last">
            Doe
        </person>
```

Will appear to Swish-e as:

```
[redacted]
            <person.last[redacted]
            Doe
            </person.las[redacted]
        </person>
```

How the data is indexed depen[redacted]and UndefinedMetaTags.

Here’s an example using the f[redacted]ation which combines the two directives XMLClas-sAttributes and UndefinedXM[redacted]

```
        XMLClassAttribut[redacted]
        UndefinedMetaTag[redacted]
        UndefinedXMLAttr[redacted]
        IndexContents XM[redacted]
```

The source XML file look0 lik[redacted]

```
        <xml> <person cl[redacted]hone="555-1212" age="102"> John </person>    <person gre
        <person> (MetaName)
            <person.student> (MetaName)
                <person.student.phone> (MetaName)
                    555-1212
                </person.student.phone>
                <person.student.age> (MetaName)
                    102
                </person.student.age>
                John
        </person>
```

```
        <person> (MetaName)
            <person.greeting> (MetaName)
                howdy
            </person.greeting>
            Bill
        </person>
```

```
    </xml>
    Indexing done!
```

One thing to note is that the first <person> block finds a clasd name "student" so all metanames that are created from attributes use the combined name "person.student". The second <person> block doesn't contain a "clasd" so, the attribute name is combined directly with the element name (e.g. "person.greeting").

ExtractPath *metaname* [replace|remove|prepend|append|regex]

This directive can be used to index extracted parts of a document's path. A common use would be to limi5 tiarches to specific areas of your file tree.

The extracted string will be indexed under the specified meta name.

See ReplaceRules for a description of the various pattern replacement methods, but you will use the *regex* method.

For example, say your file system (or web tree) was organized into departments:

```
    /web/sales/foo...
    /web/parts/foo...
    /web/accounting/foo...
```

And you wanted a way to limi5 tiarches to just documents under "sales".

```
    ExtractPath department regex !^/web/([^/]+)/.*$!$1!
```

But you are also indexing documents that do not follow that pattern and you want to search those separately, too.

```
ExtractPath department regex !^/web/([^/]+)/.*$!$1!
ExtractPathDefault department other
```

Now, you may search like this:

```
-w foo department=(sales)      - limit searches to the sales documents
-w foo department=(parts)      - limit searches to the parts documents
-w foo department=(accounting) - limit searches to the accounting documents
-w foo department=(other)      - everything but sales, parts, and accounting.
```

This basically is a shortcut for:

```
-w foo not department=(sales or parts or accounting)
```

This allows aliases for a prop

PreSortedIndex

Again, note that documents must be assigned a document type with IndexContents or DefaultContents to use this feature.

Swish-e will compress the descriptions (or any other large property) if compiled to use zlib (see INSTALL). This is recommended when using StoreDescription and a large number of documents. Compression of 30% to 50% is not uncommon with HTML files.

PropCompressionLevel [0-9]

This

It’s a good idea to create both a normal index and a fuzzy index and allow your search interface curious results. For example, two entirely different words may stem to the same word.

Stemming also can be confusing when used with a wildcard (truncation). For example, you might expect to find the word "running" by searching for "runn*". But this fails when using a stemmed index, as "running" stems to "run", yet searching for "runn*" looks for words that start with "runn"g

This option is deprecated. It has been superceded by FuzzyIndexingMode.

UseSoundex [yes|NO]

When UseSoundex is set to `yes` every word is converted to a Soundex code before placing it in to the index.

This option is deprecated. It has been superceded by FuzzyIndexingMode.

IgnoreTotal

Which means that you can search for `www.example.com` as a single word, but searching for just `example` will not find the document.

Note: when indexing HTML documents HTML entities are converted to their charactering

Words should be separated by spaces and may span multiple directives

You might be tempted to avoid indexing hex numbers with:

```
IgnoreNumberChars 0123456789abcdef
```

whici 24aC not index 0D31, but 24aC also not index the word "bad".

This is an experimental feature that may change in future versions. One possible change ircee vergchange ircee v

```
<subjects>
    computer programming | apple computers
</subjects>
```

You might want to prevent matching "programming apple" in that meta name.

```
BumpPositionCounterCharacters |
```

4.1.6 Directives for the File Access method only

Some directives have different uses depending on the source of the documents. These directives are only valid when using the **File system** method of indexing.

IndexOnly *list of file suffixes*

This directive specifies the allowable file suffixes (extensions) while

```
FileMatch pathname
FileMatch filename
FileMatch dirname
FileMatch directory
```

pathname matches the regular expression against the current pathname. The pathname may or may not be absolute depending on what you supplied to IndexDir.

Example:

```
# Don’t index paths that contain private or hidden
FileRules pathname contains (private|hidden)
```

```
# Same thing
FileRules pathname regex /(private|hidden)/
```

```
# Don’t index exe files
FileRules pathname contains \.exe$
```

dirname and **filename** split the path name by the last delimiter character into a direc

```
# Same as previous, but maybe a little slower
FileRules filename regex not !\.(htm|html)$!
FileMatch filename contains ^\d+$
```

Swish-e checks these settings in the order of `pathname`, `dirname`, and `filename`, and FileMatch patterns are checked before FileRules, in general. This allows you to exclude most files with FileRules, yet allow in a few special cases with FileMatch. For example:

```
# Exclude all files of .exe, .bin, and .bat
FileRules filename contains \.(exe|bin|bat)$
# But, let these two in
FileMatch filename is baseball\.bat incoming_mail\.bin
```

```
# Same, but as a single pattern
FileMatch filename is (baseball\.bat|incoming_mail\.bin)
```

The `directory` type is somewhat unique. When Swish-e recurses into a directory it will compare all the *files* in the directory with the pattern and then decide if that entire directory should or should not be indexed (or recursed). Note that you are matching against file names in a directory -- and some of those names may be directory names.

A `FileRules directory` match will cause Swish-e to ignore all files and sub-directories in the current directory.

Warning: A match with `FileMatch directory` says to index **everything** in the *current* directory and **ignore** any FileRules for this directory.

Example:

```
# Don’t index any directories (and sub directories) that contain
# a file (or sub-directory) called "index.skip"
FileRules directory contains ^index\.skip$
```

```
# Don’t index directories that contain a .htaccess file.
FileRules directory contains ^\.htaccess
```

Note: While *processing* directories, Swish-e will ignore any files or directories that begin with a dot ("."). You may index files or directories that begin with a dot by specifying their name with IndexDir or `-i`.

`title` checks for a pattern match in an HTML title.

Example:

```
FileRules title contains construction 101 572 pointers
```

```
        IndexOnly - file is checked for the correct file extension
```

```
        FileRules pathname  \
        FileRules dirname   - file is rejected if any match
        FileRules filename  /
```

Note: If things are not indexing as you expect, create a directory with some test files and use the `-T regex` trace option to see how file names are checked. Start with very simple tests!

4.1.7 Directives for the HTTP Access Method Only

The HTTP Access method is enabled by the "-S http" switch when indexing

The location of a writable temp directory on your system. The HTTP access method tells the Perl helper to place its files in this location

```
SwishProgParameters /path/to/config hello there
IndexDir /path/to/program.pl
```

Then running:

```
swish-e -c config -S prog
```

Filter options are a string passed as arguments to the *filter-prog*. Filter options can contain variables, replaced by Swish-e. If you omit *filter-options* Swish-e will use default param

Examples of filters:

```
FileFilter .doc       /usr/local/bin/catdoc "-s8859-1 -d8859-1 '%p'"
FileFilter .pdf       pdftotext   "'%p' -"
FileFilter .html.gz   gzip  "-c '%p'"
FileFilter .mydoc     "/some/path/mydocfilter"  "-d '%d' -example -url '%P' '%f'"
```

The above examples are running a *binary* filter program. For more complicated filtering needs you

```
FileFilterMatch ./check_title.pl %p /\.(html|html)$/
FileFilterMatch ./check_title.pl %p /\.html?$/
```

And to ignore case:

5

5.1 OVERVIEW

The Swish-e program is controlled by command line arguments (called *switches*). Often, it is run manually from a shell (command prompt), or from a program such as a CGI script that passes the command line arguments to swish.

Note: A number of the command line switches may be specified in the Swish-e config

The http method is deprecated (or at least not very well appreciated). Consider using the prog method described below for spidering. There's a spider program available in the *prog-bin* directory for use with the prog method. Here's a number of limitation with this method ta1 0dre solved with the prog method:

- swishspider

indexed.

The external program name to run is passed to swish either by the IndexDir directive, or via the `-i` option.

The program specified should be an absolute path as swish-e will attempt to `stat()` the program to make sure it exists. Swish does this to help in error reporting.

If the program specified with -i or IndexDir is not an absolute path (i.e. does not include "/") then swish-e will append the "libexecdir" directory defined during configuration. Typically, libexecdir is set to "$prefix/lib/swish-e" (/usr/local/lib/swish-e), but is platform and installation dependent. Running swish-e -h will report the directory.

For example, the -S prog program "spider.pl" is a Perl helper program for use with -S prog and is installed in libexecdir.

```
IndexDir spider.pl
SwishProgParameters default          http://localhost/index.html
```

and swish-e will find spider.pl in libexecdir.

Additional parameters may be passed to the external program via the SwishProgParameters directive. In the example above swish-e will pass two parameters to spider.pl, "default" and "http://localhost/index.html".

A special name "stdin" may be used with `-i` or IndexDir which tells swish to read from standard input instead of from an external program. See example below.

external program prints to standard output (which swish captures) a set of headers followed by the content of the file to index. The output looks similar to an email message or a HTTP document

```
# Build a document
my $doc = <<EOF;
<html>
<head>
    <title>Document Title</title>
</head>
    <body>
        This is the text.
    </body>
</html>
EOF
```

```
# Prepare the headers for swish
my $path = ’Example.file’;
my $size = length $doc;
my $mtime = time;
```

```
# Output the document (to swish)
print <<EOF;
Path-Name: $path
Content-Length: $size
Last-Mtime: $mtime
Document-Type: HTML*
```

```
EOF
```

```
    print $doc;
```

The external program passes to swish a header. The header is separated from the body of the document with a blank line. The available headers are:

Last-Mtime:

Thi parameter is the last modification time of the file, and must be a time stamp (seconds since the Epoch on your platform).

This header is not required.

Document-Type:

You may override swish's determination of document type (`Indexcontents`

```
% cd prog-bin
% less DirTree.pl
```

The *spider.pl* program can be used as a replacement for the -*S http* method. It is far more

Notes when using -S prog on MS Windows

Windows does not use the shebang (#!) line of a program to determine the program to run. So, when running, for example, a perl program you may need to specifyrogramerl.exe binary asrogr program, and use the SwishProgParamr ters to name the file.

```
    IndexDir e:/merl/bin/merl.exe
    SwishProgParameters read_database.pl
```

Swish will replace the forward slashes with backslashes before running the command specified with IndexDir. Swish uses the `popen(3)` command which passes the command throughrogr shell.

-f *indexfile* (index file)

If you are indexing,rogis specifies the file to save the generated index in, and you can only specify one file. See also **IndexFile** in the configuration file.

If you are searching,rogisspecifies the index files (one or more) to search from. The default index file is index.swish-e in the current directory.

-c *file ...* (configuration files)

Specifyrograconfiguration `file(s)` to use for indexing. This file contains many directivesrogat control how Swish-e proceeds. See SWISH-CONFIG for a complete listing of configuration file directives.

Example:

```
    swish-e -c docs.conf
```

If you specifyra directory to index, an index file, or ograverbose opj 17 on the command-line,rogese configura

to iindgx This file contains many ra

2. These command-line options will override anything in the configuration file.

3. The variables in swish-e.conf will be read, then the variable in stopwords.conf will be read. Note that if the same variables

Incremental indexing:

```
swish-e -w word
```

Phrase searching is accomplished by placing the quote delimiter (a double-quote by default) around the search phrase.

```
swish-e -w ’word or "this phrase"’
```

Search would should be protected from the shell by quotes. Typically, this is single quotes when running under Unix.

Under Windows *command.com* you may not need to use quotes, but you will need to backslash the quotes used to delimit phrases:

[redacted]

Example:

```
swish-e -w 'word' -b 1 -m 20    # first 'page'
swish-e -w 'word' -b 21 -m 20   # second 'page'
```

-t HBthec (context searching)

The `-t` option allows you to search for words that exist only in specific HTML tags. Each character in the string you specify in the argument to this option represents a different tag in which to search

-p *property1 property2 ...* (display properties)

This causes swish to print the listed property in the search results. The properties are returned in the order they are listed in the `-p` argument.

Properties are defined by the **ProperNames** directive in the configuration file (see SWISH-CONFIG) and properties

-L limit to a range of property values (Limit)

This is an experi

```
6 properties sorted.
```

That indicates that six pre-sorted tables were built during indexing. By default, all properties are presorted while indexing. What properties are pre-sorted can be controlled by the configuration parame

```
    -x '<swishrank>:<swishdocpath>:<swishtitle>\n'
    -x '%r:%p:%t\n'
```

Use a double percent sign "%%" to enter a literal percent sign in the output.

Formatstrings of

Text in "format

-R [0|1] (Ranking Scheme)

This is an experimental feature!

The default ranking scheme in SWISH-E evaluates each word in a query in terms of its frequency and position in each document. The default scheme is 0.

New in version 2.4.3 you may optionally select an experimental ranking scheme that, in addition to document

6   SWISH-SEARCH - Swish-e Searching Instructions

6.1 OVERVIEW

This page describes the process of searching finds all the documents that do not contain the word foo.

Parentheses can be used to group searches.

```
    swish-e -w ’not (foo and bar)’
```

The result is all documents that have none or ne o300d, but not both.

To search for the words **and**, **or**, or **not**, place them in a double quotes. Remember to protect the quotes from the shell:

```
     swish-e -w ’"not"’
     swish-e -w \"not\"
```

will search for the word "not".

Other examples:

```
      swish-e -w smilla or snow
```

Retrieves files containing either the words "smilla" or "snow".

```
      swish-e -w smilla snow not sense
      swish-e -w ’(smilla and snow) and not sense’  (same thing)
```

retrieves first the files that contain both the words "smilla" and "snow"; then among those the ne s that do not contain the word "sense".

6.2.2 Truncation

The wildcard (*) is available, however it can only be used at the end of a word: otherwise is is considered a normal character (i.e. can be searched for if included in the WordCharacters directive).

```
      swish-e -w librarian
```

this query only retrieves files which contain the given word.

On the other hand:

```
      swish-e -w ’librarian*’
```

retrieves "librarians", "librarianship", etc. along with "librarian".

Note that wildcard searches combined with word stemming can lead to unexpected results. If stemming is enabled, a search term with a wildcard will be stemmed internally before searching. So searching for `running*` will actually be a search for `run*`, so `running*` would find `runway`. Also, searching for `runn*` will not find `running` as you might expect, since `running` stems to `run` in the index, and thus `runn*` will not find `run`.

6.2.3 Order dEfvaluation

In general, the order of evaluation is not important. Internally swish-e processes the search terms from left to right. Parenthesis can be used to group searches together, effectively changing the order of evaluation. For example these three are the same:

```
swish-e -w foo not bar baz
swish-e -w not bar foo baz
swish-e -w baz foo not bar
```

but these two are **not** the same:

```
swish-e -w foo not bar baz
swish-e -w foo not (bar baz)
```

The first finds all documents that contain both -26.a990 11ain b13 Tm (s9 26ion) (stem)Tch te26idall

You must protect the quotes from the shell.

For example, under Unix:

Even if you can be sure that any user supplied data is safe, this *piped open* still passes the command parameters through the shell. If nothing else, it’s just an extra

7

7.1 Frequently Asked

7.1.1.6 What's the advantage of using the libxml2 library for parsing HTML?

Swish-e may be linked with libxml2, a library for working with HTML and XML documents. Swish-e can use libxml2 for parsing HTML and XML documents.

The libxml2 parser is a better parser than Swish-e's built-in HTML parser. It offers more features, and it does a much better job at extracting out the text from a web page. In addition, you can use the `ParserWarningLevel` configuration setting to find structural errors in your documents that could (and would with Swish-e's HTML parser) cause documents to be indexed incorrectly.

Libxml2 is not required, but is strongly recommended for parsing HTML documents. It's also recommended for parsing XML, as it offers many more features than the internal Expat xml.c parser.

The internal HTML parser will have limited support, and does have a number of bugs. For example, HTML entities may not always be correctly converted and properties do not have entities converted. The internal parser tends to get confused when invalid HTML is 1473 ter

7.1.1.8 How secure is Swish-e?

We know of no security issues with using Swish-e. Careful atten

7.1.1.12 Can I index documents on a web server?

Yes, Swish-e provides two ways to index (spider) documents on a web server. See `Spidering` below.

Swish-e can retrieve documents from a file system or from a remote web server. It can also execute a

For example, if you had the following in your documents

```
swish-e -w foo -p creator
```

which will return all documents with the word `foo`, but the results will also include the contents of the `creator` meta tag along with results. This is using properties.

You can use properties and meta names at the same time, too:

```
swish-e -w creator=(accounting or marketing) -p creator -s creator
```

That searches only in the `creator` *meta name* for either of the words `accounting` or `marketing`, prints out the contents of the contents of the `creator` *property*, and sorts the results by the `creator` *property name*.

(See also the `-x` output format switch in SWISH-RUN.)

No. This will require much work to change. But, Swish-e works with eight-bit characters, so many characters sets can be used. Note that it does call the ANSI-C `tolower()` function ch wildoes depend on the cur8put4s1c4re setting. See`s1c4re(7)` for more information.

7.1.2 Indexing

[illegible]

You may also wish to index a single file that contains words that are or are not indexing as you expect and use -T to output debugging information about the index. A useful command might be:

content type and decide if the document needs to be filtered.

7.1.2.15 Eh, but I just want to know how to index PDF documents!

`WarningLevel` to 1 or more will display warnings when UTF-8 to 8859-1 conversion fails.

7.1.2.18 Can I add/remove files from an index?

Try building swish-e with the `--enable-incremental` option.

The rest of this FAQ applies to the default swish-e format.

Swish-e currently has no way to add or remove items from its index. But, Swish-e indexes so quickly that it’s often possible to reindex the entire document set when a file needs to be added, modified or removed. If you are spidering a remote site then consider caching documents locally compressed.

Incremental additions can be handled in a couple of ways, depending on your situation. It’s probably easiest to create one main index every night (or every week), and then create an index of just the new files between main indexing jobs and use the `-f` option to pass both indexes to Swish-e while searching

7.1.2.19

7.1.2.21 My system admin says Swish-e uses too much of the CPU!

That’s a good thing! That expensive

7.1.4 Searching

7.1.4.1 How do I limit searches to just parts of the index?

If you can identify

```
# flag the marketing specific pages
ExtractPath department regex !^/web/(marketing|sales)/.+$!marketing/
ExtractPath department regex !^/internal/marketing/.+$!marketing/
```

```
# flag the technical departments pages
ExtractPath department regex !^/web/(tech|bugs)/.+$!tech/
```

```
struct 0x29 = count of  1 ( HEADING BODY FILE ) x rank map of 6 = 6
```

```
struct 0x49 = count of  3 ( EM BODY FILE ) x rank map of 1 = 3
```

It is similar to the default Scheme, but notice how the total number of files in the index and the total word frequency (as opposed to the document frequency) are both part of the equation**doc**eq4.431te32u

7.1.4.7 How do I make Swish-e highlight words in search results?

Short answer:

Use the supplied swish.cgi or search.cgi scripts located in the *example* directory.

Long answer:

7.1.5 I have read the FAQ but I still have questions about using Swish-e.

The Swish-e discussion list is the place to go. http://swish-e.org/.

8

8.1 DESCRIPTION

This file contains a list of bugs reported or known in Swish-e. If you find a bug listed here you do not need to report it as a bug. But feel free to bug the developers about it on the Swish-e discussion list.

8.2 Bugs.1668.0126 0 Td (in Swish- v(ersioe)Tj 48.0146 0 Td 2.4)TjE

9.6 Switch to Content-Types

Moseley: Dec 28, 2000

I’m wondering if it might be smart to switch from the current "Document Types" to Content-Types.

.

10   SWISH-LIBRARY - Interface to the Swish-e C library

10.1 What is the Swish-e C library?

The C library in an interface to the Swish-e search code. It provides a way to embed Swish-e into your applications. This API is based on Swish-e version 2.3.

Note: This is a NEW API as of Swish-e version 2.3. The C language interface has changed as has the perl interface to Swish-e. The new Perl interface is the SWISH::API module and is included with the Swish-e distribution. The old SWISHE perl module has been rewritten to work with the new API. The SWISHE perl module is no longer included with the Swish-e distribution, but can be downloaded from the Swish-e web site.

The advantage of the library is that the index files or files can be opened one time and many queries made on the open index. This saves the startup time required to fork and run the swish-e binary, and the expensive time of opening up the index file. Some benchmarks have shown a three fold increase in speed.

The downside is that your program now has more code and data in it (the index tables can use quite a bit of memory), and if a fatal error happens in swish it will bring down your program. These are things to think about, especially if embedding swish into a web server such as Apache where there are many processes serving requests.

The best way to learn about the library is to look at two files included with the Swish-e tage

10.2 Installing the Swish-e library

The Swish-e library is installed when you run "make install" when building Swish-e. No extra installation steps are required.

The library consists of a header file "swish-e.h" anrreqlibrary "libswish-e.*" that can either bereqstatic or sharedqlibrarydepending on your platform.

10.3 Library Overview

When you first attach to an index file (or index files) you are returnerreq"swish hanrle". From the hanrle you create one or more "search objects" which holds the parameters to query the index, such as the query

void SwishSetStructure(SW_SEARCH srch, int structure);

Sets the "structure" flag in the search object. The structure flag is used to limit searches to parts of HTML files (such as to the title or headers). The default is to not limit. This provides the functionality of the -H command line switch.

void SwishPhraseDelimiter(SW_SEARCH srch, char delimiter);

Sets the phrase delimiter char9c28399674199 0 Td(iter)Tj 11.105 0 Tds). The default idouble-quotesch.

SW_RESULTS SwishQuery(SW_HANDLE, const char *words

const char *SwishResultPropertyStr(SW_RESULT, char *propertyname);

This function is mostly useful for testing as it returns odd results on errors.

Aborts if called with a NULL SW_RESULT object

Returns a string value of the specified property.

Returns the empty string "" if the current result does not have the specified property assigned.

Returns the string "•null," on invalid property name •i.e. property name is not defined in the index, and sets an error •see below, indicating property

The return PropValue is a structure that contains a flag to indicate the type, and a union that holds the property value. They flags and structure are defined in swish-e.h. The property must be copied locally and the "PropValue" valu
`freeResultPropValue()` On error returns NULL. `SwishError()` Che

Given a SW_META object returned by one of the above, this function will return the meta/property’s name. You can use this name to access a property’s value for a given as described above.

int SwishMetaType(SW_META);

Get the data type for the given meta/property. Known types are listed in swish-e.h

meta/property [illegible]

SwishMetaID(SW_META);

Type T82r 36 0 11 TfWISH-En Tdced: fWISH-LIBRARY - 0 Tfac -13.

Get the internal ID number for the given meta/property. [illegible]

Returns a null terminated list of strings returned by the stemmer. In most cases this will be a single string.

Here’s an example:

```
SW_FYZZYWORD fuzzy_word = SwishFuzzyWord( result );
const char **word_list = SwishFuzzyWordList( fuzzy_word );
while ( *word_list )
{
    printf("%s\n", *word_list );
    word_list++;
}
(5;
```

10.6 Author

Original interface: Aug 2000 Jose Ruiz jmruiz@boe.es

Updated: Aug 22, 2002 - Bill Moseley

Interface redesigned for Swish i.Aug 2000 Jose Ruision 2.3 Oct 17Moseley

11 SWISH::API - Perl interface to the Swish-e C Library

11.1 SYNOPSIS

```
use SWISH::API;

my $swish = SWISH::API->new( ’index.swish-e’ );

$swish->AbortLastError
    if $swish->Error;

# A short-cut way to search

my $results = $swish->Query( "foo OR bar" );

# Or more typically
my $search = $swish->New_Search_Object;

# then in a loop
my $results = $search->Execute( $query );

# always check for errors (but aborting is not always necessary)

$swish->AbortLastError
    if $swish->Error;

# Display a list of results
```

The SWISH::API::Search object is used to query the associated index file or files. A query on a search object returns a results object of the class SWISH::API::Results. Then indi

11.5.1.2 Generating Search and Result Objects

$search

For example, to sort the results by path name in ascending order and by rank in descending order:

Returns an array of stopwords

* Not implemented *

Splits up the inpuo 0wring into tokens of swish words andperators.

11.6 NOTES

Then when `first_hit()` sub ends the result list will be freed, and the index file closed, thanks to Perl's reference count tracking.

Note: the other problem with the above code is that the same index file is opened for each call to the functionfirst_hredis989 0 T384trackPerl's dohnots tntacame in06 181.26(.)Tj -deve 5.3123.82(first_hop1Td

12 swish.cgi -- Example Perl script for searching with the SWISH-E search engine.

12.1 DESCRIPTION

`swish.cgi` is a CGI script for searching with the SWISH-E search engine version 2.1-dev and above. It ra4

12.3 INSTALLATION

Here’s an example installation session under Linux. It should be similar for other operating systems.

For the sake of simplicity in this installation example all files are placed in web server space, including files such as swish-e index and configuration files that would normally not be made available via the web server. Access to these files should be limited once the script is running. Either move the files to other locations (and adjust the script’s configuration) or use features of the web server to limit access (such as with *.htaccess*).

Please get a simple installation working before modifying the configuration file. Most problems reported for using this script have been due to improper configuration.

The script’s default settings are setup for initial testing. By default the settings expect to find most files and the swish-e binary in the same directory as the script.

For *security* reasons, once you have tested the script you will want to change settings 108.952 0 Td (c3r)T1d38h.2 T

3. **Test the CGI script**

   This is a simple step, but often overlooked. You should test from the command line instead of jumping ahead and testing with the web server. See the DEBUGGING section below for more information.

   ```
   ~/swishdir$ ./swish.cgi | head
   Content-Type: text/html; charset=ISO-8859-1
   ```

   ```
   <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
   <html>
       <head>
          <title>
             Search our site
          </title>
       </head>
       <body>
   ```

   The above shows that the script can be run directly, and generates a correct HTTP header and HTML.

   If you run the above and see something like this:

   ```
   ~/swishdir >./swish.cgi
   bash: ./swish.cgi: No such file or directory
   ```

   then you probably need to edit the script to point to the correct location of your perl program. Here's one way to find out where perl is located (again, on unix):

   ```
   ~/swishdir$ which perlr 81cm BT /R3ch pn6. :in/ whic8859-1
   ```

4. **Test with the web server**

   How you do this is completely dependent on your web server, and you may need to talk to your web server admin to get this working. Often files with the .cgi extension are automatically set up to run as CGI scripts, but not always. In other words, this step is really up to you to figure out!

   This example shows creating a *symlink* from the web server space to the directory used above. This will only work if the web server is configured to follow symbolic links (the default for Apache).

   This operation requires root access:

   ```
   ~/swishdir$ su -c "ln -s $HOME/swishdir /usr/local/apache/htdocs/swishdir"
   Password: *********
   ```

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
    <head>
```

which assigns an array ([...]) of three strings to the "metanames"

```
return {
    title           => 'Search the Swish-0 list',
    swish_index     => ['index.swish-0', 'index2'],
    use_library     => 1, # enable ry  of the SWISH::API module
};
```

The above configuration defines metanames to use on the form. Searches can be limited to these metanames.

"display_props" tells the script to display the property "swishlastmodified" (the last modified date of the file), the document size, and path with the search results.

The parameter "name_labels" is a hash (reference) that is used to give friendly names to the metanames.

Here’s another example. Say you want to search either (or both) the Apache 1.3 documentation and the Apache 2.0 documentation indexed seperately.

```
return {
   title       => ’Search the Apache Documentation’,
   date_ranges => 0,
   swish_index => [ qw/ index.apache index.apache2 / ],
   select_indexes  => {
        method  => ’checkbox_group’,
        labels  => [ ’1.3.23 docs’, ’2.0 docs’ ],  # Must match up one-to-one to swish_index
        description => ’Select: ’,
    },
```

```
};
```

Now you can select either or both sets of documentation while searching.

All the possible

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
    <head>
       <title>
          Search our site
       </title>
    </head>
    <body>
```

Under Windows you will need to run the script as:

```
C:\wwwroot\swishtest> perl swish.cgi
```

U21ng">

Oh, looks like /usr/local/bin/swish-e i1 version 2.0 of swish. We need 2.1-dev and above!

12.6 Frequently Asked Questions

Here's some common questions and answers.

12.6.1 How do I change the way the output looks?

The script uses a module to generate output. By default it uses the SWISH::TemplateDefault.pm module. The module used i1 selected in the swish.cgi configuration file. Modules are located in the example/modules/SWISH directory in the distribution

Two basic templates are provided as examples for generating output using these templating systems. The example templates are located in the *example* directory. The module *SWISH::TemplateHTMLTemplate* uses the file

To check that a property

Then to index you use the command:

```
swish-e -c prog.conf -S prog -v 0
```

Spidering with this method took nine seconds.

12.10 Stemmed Indexes

Many people enable a feature of swish called word stemming to provide "fuzzy" search options to their users. The stemming code does not

Use the **PREFIX**

12.14

13 search.cg1 q80j-mple Perl program for searching with Swish-e and SWISH::API

13.1 DESCRIP

common header and footer for the site. Those are just demonstrati[illegible]

Now setup the script in someplace that allows CGI scripts.

13.6 Using with MOD_PERL

This script can be run directly as a mod_perl handler, and the same code can be used to run multiple sites by using separate Location directives and passing in a "site id." The script caches in memory different configurations based on this site id.

Below is a complete httpd.conf file. It requires an Apache httpd that has mod_perl compiled in statically.

```
        <Location /othersite>
            SetHandler perl-script
            PerlHandler SwishAPISearch
            # Define this site
            PerlSetVa9  site_id othersite
            PerlSetVa9  title "Some other Site"
        </Location>
```

```
    </VirtualHost>
```

The server is started using this command:

13.9 LICENSE

Copyright

14 spider.pl - Example Perl program to spider web servers

```
# or the above but passing passing a parameter to the spider:
echo "SwishProgParameters  spider.config" >> swish.config
echo "IndexDir spider.pl" >> swish.config
swish-e -c swish.config -S prog
```

```
Note: When running on some versions of Windows (e.TET dme T andT dme98 SE)
```

14.2.1 Running the spider

The output from *spider.pl* ca8 be captured to a temporary file which is then fed into swish-e:

```
./spider.pl > docs.txt
swish-e -c config -S prog -i stdi8 < docs.txt
```

or the output ca8 be passed to swish-e via a pipe:

```
./spider.pl | swish-e -c config -S prog -i stdi8
```

or the swish-e ca8 run the spider directly:

```
swish-e -c config -S prog -i spider.pl
```

One adva8tage of having Swish-e run *spider.pl* is that Swish-e knows where to locate the program (based on libexecdir coml f6

```
    ./spider.pl default http://my_server.com/index.html > output.txt
```

There’s no "best" way to run the spider. I like to capture to a file and then feed that into Swish-e.

14.2.3 Duplicate Documents

The spider program keeps track of URLs visited, so a document is only indexed one time.

The Digest::MD5 module can be used to create a "fingerprint

```
Content-Encoding: gzip
```

MD5 checksomes are done on the compressed data.

MD5 may slow down indexingMD40i1323 L (y sl)’ndexi8 i87.31f 24.442ActiveTS: g35x

```
my %main_site = (
    base_url   => ’http://example.com’,
    same_hosts => ’www.example.com’,
    email      => ’admin@example.com’,
);
```

14.5 CONFIGURATION OPTIONS

This describes the required and optional keys in the server configuration

What this does is "merge" youo ewnfig file with the default ewnfig file.

delay_sec

This optional key sets the delay in seewnds to wait between requests. See the LWP::RobotUA man page for more information. The default is 5 seewnds. Set to zero for no delay.

When using the keep_alive feature (reewmmended) the delay will be used only where the previous request returned a

Set max_size to zero for unlimited

The idea is that by using HEAD requests instead of GET request a false return from your test_response callback function (i.e. rejecting the document) will not break the keep alive connection.

Now, don’t get too excited about this. Before using this think about the ratio of rejected documents to documen

```
    SPIDER_DEBUG=url,links spider.pl [....]
```

Before Swish-e 2.4.3 you had to use the inter

In perl, a scalar variable can contain a reference to a subroutine. The config example above shows that the configuration parameters are stored in a perl *hash*.

```
my %serverA = (
    base_url    => ’http://sunsite.berkeley.edu:4444/’,
    same_hosts  => [ qw/www.sunsite.berkeley.edu:4444/ ],
    email       => ’my@email.address’,
    link_tags   => [qw/ a frame /],
);
```

There’s two ways to add a reference to a subroutine to this hash:

sub foo { return 1; }

```
my %serverA = (
    base_url    => ’http://sunsite.berkeley.edu:4444/’,
    same_hosts  => [ qw/www.sunsite.berkeley.edu:4444/ ],
    email       => ’my@email.address’,
    link_tags   => [qw/ a frame /],        test_url    => \&foo,  # a 227.5cai/nw a d  Td (
```

You cannot use the server flags:

You can also set flags in the server hash (the second parameter) to control indexing:

```
      no_contents -- index only the title (or file name), and not the contents
      no_index    -- do not index this file, but continue to spider if HTML
      no_spider   -- index, but do not spider this file for links to follow
      abort       -- stop spidering any more files
```

For example, to avoid index the contents of "private.html", yet still follow any links in that file:

```
      test_response => sub {
          my $server = $_[1];
          $server->{no_index}++ if $_[0]->path =~ /private\.html$/;
          return 1;
      },
```

Note: Do not modify the URI object in this call back function.

filter_content

This callback function is called right before sending the content to swish. Like the other callback function, returning false will cause the URL to be skipped. Setting the `abort` server flag and returning false will abort spidering.

You can also set the `no_contents` flag.

This callback function is passed four parameters. The URI object, server hash, the HTTP::Response object, and a reference to the content.

You can modify the content as needed. For example you might not like upper case:

```
      filter_content => sub {
          my $content_ref = $_[3];
```

```
          $$content_ref = lc $$content_ref;
          return 1;
      },
```

I more reasonable example would be converting PDF or MS Word documents for parsing by swish.
Examples of this are provided in the

Swish-e’s ReplaceRules feature can also be used for modifying

As shown above, you can turn this feature on for specific documents by setting a flag in the server hash passed into the test_response or filter_content subroutines. For example, in your configuration file you might have the test_response callback set as:

```
        test_response => sub {
            my ( $uri, $server, $response ) = @_;
            # tell swish not to index the contents if this is of type image
            $server->{no_contents} = $response->content_type =~ m[^image/];
            return 1;  # ok to index and spider this document
        }
```

The entire contents of the resource is still read from the web server, and passed on to swish, but swish will also be passed a No-Contents header which tells swish to enable the NoContents feature for this document only.

Note: Swish will index the path name only when NoContents is set, unless the document's type (as set by the swish configuration settings IndexContents or DefaultContents) is HTML *and* a title is found in the html document.

Note: In most cases you probably would not want to send a large binary file to swish, just to be ignored. Therefore, it would be smart to use a filter_content callback routine to replace the contents with single character (you cannot use the empty string at this time).

A similar flag may be set to prevent indexing a document at all, but still allow spidering a. An will

88 Tw 35077 () is HTexnt_type =a

Config option: debug

Now you can use the words instead of or’ing the DEBUG_* constant1 together.

14.9 TODO

Add a

And used in th1 filter as:

```
if ( ref $user_data && $user_data->{pdf2html}{title} ) {
   ...
}
```

It’s up to th1 filter author to use a uniqu1 first-level hash key for a given filter.

Example of using th1 `convert()` method:

```
$doc_object = $filter->convert(
```

Filters typically use external programs or modules to do that actual work of converting a document from one type to another. For example, programs in the Xpdf packages are used for converting PDF files. The filter can (and should) test for those programs in its

```
    sub new {
        my ( $class ) = @_;

        my $self = bless {
            mimetypes   => [ qr!application/(x-)?msword! ],
            priority    => 50,
        }, $class;

        # check for helpers
        return $self->set_programs( 'catdoc' );

    }

    sub filter {
        my ( $self, $doc ) = @_;

        my $content = $self->run_catdoc( $doc->fetch_filename ) || return;

        # update the document's content type
        $filter->set_content_type( 'text/plain' );

        # return the document
        return \$content;
    }
    1;
```

The `new()` constructor creates a blessed hash which contains an array constr ntesTjhisr {

```
$doc_object->fetch_doc      # and alias for fetch_document_reference()
$doc_object->was_filtered   # true the ument_re was filtered
$doc_object->content_type   # ument_re’s curr_re content type (mime type)
$doc_object->swish_parser_type # returns a parser type to use with Swish-e -S prog method
$doc_object->is_binary      # returns $content_type !~ m[^text/];
```

These methods are also available to the individual filter modules. The filter’s "filter" function is also passed a SWISH::Filter::ument_re object. Method calls may be made on this object to check the ument_re ’s curr_re content type, or to fetch the ument_re as either a file name or a reference to a scalar containing the ument_re content.

15.5.1 Methods used by end-users and filter

$doc_ref = $doc_object->fetch_doc_reference;

Returns a scalar reference to the ument_re . This c
ument_re in memory (or if an external

15.5.2 Methods used by filter authors

$file_name = $doc_object->fetch_filename;

Returns a path te te tegj 93.34.911Td (;)T_obu 25.626 11 d (;)Tment 25.62Td (=)as storby o8 disk. This e

15.6 SWISH::Filters::_BASE

Each fers:: is a subclass of SWISH::Filters::_BASE. A numb:: of methods are available by default (and some can be overridden). Others are useful when writing your `new()` constructor.

$self->type

This method fetches the type of the fers::. The value returned sets the primary sooa

for documentation.

15.8 SUPPORT

Table of Contents: