swish.tiff) SWISH 1.1.1

*Note: SWISH has changed considerably since version 1.0! **The format is different and configuration variables have changed.** Current SWISH users should reindex their files using version 1.1. Please read the €list of changes€ since version 1.0.*

Contents

€What is SWISH?€

SWISH stands for **S**imple **W**eb **I**ndexing **S**ystem for **H**umans. With it, you can index directories of files and search the generated indexes.

For an example of swish can do, try €searching for the words "office and map" at EIT€. All of the search databases you see there were indexed by swish. When you do a search, it's the **swish** program that's doing the actual searching.

SWISH was created to fill the need of the growing number of Web administrators on the Internet - many current indexing systems are not well documented, are hard to use and install, and are too complex for their own good. Here are some pros and cons regarding **SWISH**:

· **It's simple.**

I've tried to make SWISH as simple as possible while keeping some of the things that people look for in an indexer. The drawback is that you can't do many things that full-featured indexers and searching programs can do, such as stemming (searching for different versions of a word) or the use of synonyms.

· **It's made for Web sites.**

In indexing HTML files, SWISH can ignore data in tags and give higher relevance to information in header and title tags. Titles are extracted from HTML files and appear in the search results. SWISH can automatically search your whole Web site for you in one pass, if it's under one directory. You can also search for words that exist in HTML titles, comments, and emphasized tags, and 8-bit HTML characters can be indexed, converted, and searched.

· **It's fairly nice on disk space and is pretty fast.**

Index files consist of only one file, so they can be transported around and easily maintained. The SWISH source is not large and generated indexes average out to around half the size of comparable WAIS indexes, or 1 to 5% of the size of your original HTML data. Searching is as fast as or better than using a non-commercial WAIS-based solution.

· **You can fix the source.**

I encourage people to send in patches and suggestions on how to make SWISH better. Although it's not in the public domain, I am always more than happy to integrate contributed code into the distribution. Please note the €license concerning its use€. If you do have licensing or business-related questions **only**, please contact Elizabeth Batson at €*batson@eit.com*€.

€Great! How do I get started?€

First, you can download the package starting from `€http://www.eit.com/software/swish/`€.
Please read the license before you download the package.

1. Put the SWISH package (`swish.11.tar.Z`) on your computer
2. Uncompress it (perhaps type `uncompress swish*`)
3. Untar it (perhaps type `tar -xvf swish.11.tar`)
4. Go into the `swish.11` directory
5. Read the `README` file for compilation instructions

Everything was written in pretty vanilla C, so it should work just about anywhere. Initial testing was done on the following systems: **SunOS 4.1.3**, **Solaris 2.4**, **BSDI 1.1**, **IRIX 5.3/4**, and **OSF/1 2.0**. It has been reported to compile on **SunOS 4.1.1**, **FreeBSD 2.0**, **NetBSD 1.0**, **Linux 1.2.2/1.2.8**, **OSF/1 3.0B**, and **AIX 3.2.5**.

The swish program can go under `/usr/local/bin` - you may want to put other SWISH things somewhere such as `/usr/local/httpd/swish`, if you're using NCSA's httpd. You'll also want to create a directory to hold SWISH databases, somewhere like `/usr/local/httpd/swish/sources`. You can store the files anywhere you like, as long as you remember where they are!

After you've compiled (and installed) SWISH, make sure the **swish** program is somewhere in your executable path (somewhere such as `/usr/local/bin`).

€Searching with SWISH€

In the SWISH distribution, there's a sample SWISH index (called **sample.swish**), and you can do a simple search on it. Try typing this:

```
  swish -f sample.swish -w internet and resources and archie
```

This will search the file **sample.swish** for files consisting of the words **internet** and **resources** and **archie**. You should get something back like this:

```
# SWISH format 1.1
search words: internet and resources and archie
# Name: Index of EIT's Web
# Saved as: sample.swish
# Counts: 7316 words, 94 files
# Indexed on: 12/03/95 17:50:43 PST
# Description: This is a full index of EIT's web site.
# Pointer: http://www.eit.com/cgi-bin/wwwwais/
# Maintained by: Kevin Hughes (kevinh@eit.com)
1000 http://www.eit.com/web/www.guide/guide.15.html "Guide to Cyberspace 6.1:
Index/Glossary" 11566
360 http://www.eit.com/web/netservices.html "Internet Resources List" 48391
.
```

The results tell you:

1. The format the results are in (so future versions of swish or other searching programs know this),
2. The search words you used,
3. Administrative information about the index (so you know who maintains it and how to find the original),
4. A result line - this is made up of:
   - **The relevance rank.** This number is generated with each result and is the program's "best guess" as to how relevant it thinks the file is to your query. This rank number, which can range from 1 to 1000, depends on a number of factors, such as how many times your search word appears in the file, how many words are in the file, and if the word appears in a title or header tag (if it's an HTML file), among other factors.
   - **The path name to the file.** This may be an address, such as a URL, or a full path to the file.

    - **The title of the file.** If this is an HTML file, this is the title. This may also be the name of the file (if there is no title).
    - **The size of the file.** This size is always in bytes.
5. A period. This signifies the end of the results. A line with a period always signifies the end of swish output.

If there are errors, instead of the results list, you may get one of the following error lines. These lines will always be prefixed with `err:`.

- `err: no results`
  There were no results of the search.
- `err: could not open index file`
  Either the index file could not be found or it couldn't be opened.
- `err: no search words specified`
  No words were specified for searching.
- `err: a word is too common`
  A search word was used that was too common to give any meaningful feedback.
- `err: the index file is empty`
  No words are in the index file.

- `err: the index file format is unknown`
  SWISH can't read the particular format of the file.

€Indexing with SWISH€

SWISH has the capability to use configuration files in which you can specify all sorts of options for indexing. To use a configuration file, call it something such as **swish.conf**, and place it somewhere such as `/usr/local/httpd/swish/`. The configuration file below is an example of a typical SWISH configuration file:

```
# SWISH configuration file
# Lines beginning with hash marks (#) and
# blank lines are ignored.

IndexDir /usr/local/www
# This is the root directory of the Web tree you want to index.
# It can be a space-separated list of files and directories
# you want indexed. You can specify more than one of these directives.
```

```
IndexFile /usr/local/httpd/swish/sources/index.swish
# This is the name your SWISH index will be saved as.

IndexOnly .html .txt .c .ps .gif .au .hqx .xbm .mpg .pict .tiff
# Only files with these suffixes will be indexed.

IndexReport 3
# This is how detailed you want reporting. You can specify numbers
# 0 to 3 - 0 is totally silent, 3 is the most verbose.

FollowSymLinks yes
# Put "yes" to follow symbolic links in indexing, else "no".

NoContents .ps .gif .au .hqx .xbm .mpg .pict .tiff
# Files with these suffixes won't have their contents indexed,
# only their file names.

ReplaceRules replace "/usr/local/www" "http://www.eit.com"
# ReplaceRules append ""
# ReplaceRules prepend ""
# ReplaceRules allow you to make changes to file pathnames
# before they're indexed.

FileRules pathname contains admin testing demo trash construction confidential
FileRules filename is index.html
FileRules filename contains # % ~ .bak .orig .old old.
FileRules title contains construction example pointers
FileRules directory contains .htaccess
# Files matching the above criteria will *not* be indexed.

IgnoreWords SwishDefault
# The IgnoreWords option allows you to specify words to ignore (stopwords).
# Comment out for no stopwords; the word "SwishDefault" will include
# a list of default stopwords. Words should be separated by spaces
# and may span multiple directives.
```

To index a site using the options in a configuration file, type:

```
swish -c /usr/local/httpd/swish/swish.conf
```

To run swish and index your site.

Taking as an example the above configuration in the script, you'd have the directory `/usr/local/httpd/swish/sources` and one file called `index.swish` in the directory. The name of the database you've just created is `index.swish`.

€Configuration file options€

You can specify variables and values in the configuration file by typing the variable name (it's not case sensitive), a space (tabs are OK), and the value you want for the variable. If the value has spaces, you can enclose it in quotes to keep the space. If you want to specify multiple values, separate the values with a single space. In the configuration file, lines beginning with a hash mark (#) and blank lines are ignored.

€Basic index variables€

- **IndexDir** `directory`

  The **IndexDir** variable tells swish what directories and files to index. Each specified directory will be indexed recursively. You can use more than one of these directives - here are some examples:

```
IndexDir /usr/local/www /src/code.html
IndexDir /users/tony/public_html/home.html /web
```

· **IndexFile** `indexfile`

The **IndexFile** variable tell swish what to save the indexed results as. Indexes generated by swish should have a suffix of `.swish`.

· **IndexOnly** `.suffix1 .suffix2 .suffix3 ...`

Only files with these suffixes will be indexed. If you omit this variable, swish will index every file it comes across. Suffix checking is not case sensitive.

· **IndexReport** `3`

This variable can have the values `0` to `3`. If you specify `3`, swish will tell you what's going on while it's indexing, printing out directory and file names, number of words indexed, and so on, as well as give information about other operations. The value `0` will make swish completely silent.

· **FollowSymLinks** `value`

Normally swish ignores symbolic links to files whe indexing. If you want it to follow such links, define this value as `yes`, else define it as `no`.

- **NoContents** `.suffix1 .suffix2 .suffix3 ...`

  This variable lets you control what files will have their contents indexed. If a file with a suffix in this list is indexed, only its file name (and not any words in the file) will be indexed. This is useful because normally SWISH will try to index the contents of every file, even files without words (such as images or movies). Suffix checking is case-insensitive.

- **IgnoreWords** `word1 word2 ...`

  Here you can specify words to ignore when searching. Usually these words (called **stopwords**) are words that occur too many times in your data to make indexing them worthwhile. If you specify a word as `SwishDefault`, it will be replaced with swish's default list - a few hundred very common English words.

- **IgnoreLimit** `number1 number2`

  After indexing, swish can automatically tell which words are the most common and omit them from the index according to these parameters. Here are some examples:

```
  1. IgnoreLimit 80 256
  2. IgnoreLimit 50 50
```

1. Swish will ignore all words that occur in over 80% of the files and that also occur in over 256 different files.
2. Swish will ignore all words that occur in over 50% of the files and that also occur in over 50 different files.

Using **IgnoreLimit** and **IgnoreWords** can help trim the size of your index files considerably - experiment with parameters to see what works best at your site. You can also use **IgnoreLimit** to limit the CPU resources that searches take.

· **IndexName** `"value"`
· **IndexDescription** `"value"`
· **IndexPointer** `"value"`
· **IndexAdmin** `"value"`

These variables specify information that goes into index files to help users and administrators. **IndexName** should be the name of your index, like a book title. **IndexDescription** is a short description of the index or a URL pointing to a more full description. **IndexPointer** should be a pointer to the original information, most likely a URL. **IndexAdmin** should be the name of the index maintainer and can include name and email information. These values should not be more than 70 or so characters and should be contained in quotes. Note that the automatically generated date in index files is in `D/M/Y` and 24-hour format.

€Using ReplaceRules€

When results are returned from swish searches, you may get a bunch of funny pathnames to files that you can't access. Using **ResultRules**, you can specify a series of operations to perform on the pathname result to change it into a URL and other things if you desire.

There are three operations you can specify: **replace**, **append**, and **prepend**. They will parse the pathname in the order you've typed these commands. More than one command and its arguments can appear on the same line, but it's easier to read when commands are broken up over a few lines. You can't put a command and its argument(s) on different lines, however.

Here's the syntax:

```
replace "the string you want replaced" "what to change it to"
   This replaces all occurrences of the old string
   with the new one.
prepend "a string to add before the result"
append "a string to add after the result"
```

Study the above sample configuration file and try things out. You'll find that by having swish return URLs instead of pathnames, you can create interfaces to swish that can allow users to get to the search results over the

World-Wide Web.

€Using FileRules€

You can specify certain file directives in the configuration file - any files or directories matching these criteria will be **ignored** and will not be indexed. Prepend all of these operations with the **FileRules** directive:

- **pathname contains** `string1 string2 string3 ...`

  Any path names containing exactly these strings, whether they be paths to directories or paths to files, will be ignored. Using this you can avoid indexing temporary directories or private material.

- **filename is** `filename`

  Any file name exactly matching the specified file name will be ignored (this is case-sensitive). This cannot be a path.

- **filename contains** `string1 string2 string3 ...`

  Any file name containing these strings will be ignored (this is **not** case-sensitive). This cannot be a path.

- **title contains** `string1 string2 string3 ...`

  Any HTML file with a title that contains these strings will be ignored (this is case-insensitive).

- **directory contains** `string1 string2 string3 ...`

Any directory that contains any of these specified file names will be ignored (this is case-insensitive).

€Usage€

```
  usage: swish [-i dir file ... ] [-c file] [-f file] [-l] [-v (num)]
         swish -w word1 word2 ... [-f file1 file2 ...] [-m num] [-t str]
         swish -M index1 index2 ... outputfile
         swish -D file
         swish -V

options: defaults are in brackets
         -i : create an index from the specified files
         -w : search for words "word1 word2 ..."
         -t : tags to search in - specify as a string
              "HBthec" - in head, body, title, header,
              emphasized, or comments
         -f : index file to create or search from [index.swish]
         -c : configuration file to use for indexing
         -v : verbosity level (0 to 3) [0]
```

```
          -l : follow symbolic links when indexing
          -m : the maximum number of results to return [40]
          -M : merges index files
          -D : decodes an index file
          -V : prints the current version

version: 1.1
   docs: http://www.eit.com/software/swish/
```

To see the usage, run swish with a **-z** or **-?** option.

€Command-line options€

-w *word1 word2 ...* (search words)

This performs a case-insensitive search using a number of keywords. If no index file to search is specified, swish will try to search a file called `index.swish` in the current directory. You don't need to put quotes around search words.

You can use the booleans **and**, **or**, or **not** in searching. Without these booleans, swish will assume you're

anding the words together. Evaluation takes place from left to right only, although you can use parentheses to force the order of evaluation.

You can also use wildcards (asterisks) to search for matches to the beginnings of words only - you can't put asterisks at the front or in the middle of words.

```
  example 1: swish -w john and doe or jane
  example 2: swish -w john and (doe or not jane)
  example 3: swish -w not (john or jane) and doe
  example 4: swish -w j* and doe
```

1. This search evaluates the expression from left to right.
2. This search will also be evaluated from left to right, although the operation in parentheses will be evaluated as a whole first.
3. `john or jane` will be evaluated first, a `not` operation will be performed on that, then everything will be `and`ed with `doe`.
4. This will search for all files that contain words starting with the letter `j` and that also contain `doe`.

-t *"HBthec"* (context criteria)

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 to search for the word in. **H** means all `&ltHEAD>` tags, **B** stands for `<BODY>` tags, **t** is all `<TITLE>` tags, **h** is `<H1>` to `<H6>` (header) tags, **e** is emphasized tags (this may be `<B>`, `<I>`, `<EM>`, or `<STRONG>`), and **c** is HTML comment tags (`<!-- ... -->`).

```
  example 1: swish -w apples oranges -t t
  example 2: swish -w keywords draft release -t c
  example 3: swish -w world wide web -t the
```

1. This search will look for files with these two words in their titles only.
2. This search will look for files with these words in comments only.
3. This search will look for words in titles, headers, and emphasized tags.

-m *(number)* (number of results)

While searching, this specifies the maximum number of results to return. The default is 40. If no numerical value is given, the default is assumed. If the value is 0 or the string `all`, there will be no limit to the number of results. The configuration file value overrides this value.

-i *directory file ...* (files to index)

This specifies the directories and/or files to index. Directories will be indexed recursively.

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

This specifies the configuration file to use for searching. You can use this as an only option to swish to do automatic indexing, if all the necessary variables are set in the configuration file.

If you specify a directory to index, an index file, or the verbose option on the command-line, these values will override any specified in the configuration file.

You can specify multiple configuration files in order to split up common preferences. For instance, you might store a file with the stopwords in it and have multiple other files that have different index file information.

```
  example 1: swish -c swish.conf
  example 2: swish -i /usr/local/www -f index.swish -v -c swish.conf
  example 3: swish -c swish.conf stopwords.conf
```

1. The settings in the configuration file will be used to index a site.
2. These command-line options will override anything in the configuration file.
3. The variables in `swish.conf` will be read, then the variable in `stopwords.conf` will be read. Note that if the same variables occur in both files, older values may be written over.

-f *indexfile1 indexfile2 ...* (index file)

If you are indexing, this specifies the file to save the generated index in, and you can only specify one file. If you are searching, this specifies the index files (one or more) to search from. The default index file is `index.swish` in the current directory.

-l (symbolic links)

Specifying this option tells swish to follow symbolic links when indexing. The configuration file value will currently override the command-line value.

-M *indexfile1 indexfile2 indexfile3...* (index merging)

This allows you to merge two or more index files - the last file you specify on the list will be the output file. Merging removes all redundant file and word data. To estimate how much memory the operation will need, sum up the sizes of the files to be merged and divide by two. That's about the maximum amount of memory that will be used. You can use the **-v** option to produce feedback while merging and the **-c** option with a configuration file to include new administrative information in the new index file.

-D *indexfile* (decode)

This option is provided so you can check the word, file, and maintenance information in index files. You can specify multiple files to decode.

-v *(number)*, -V (verbose and version options)

The **-v** option can take a numerical value from `0` to `3`. Specify `0` for completely silent operation and `3` for detailed reports. If no value is given then `3` is assumed.

The **-V** option makes swish spit out its version number.

€Questions and answers€

Swish crashes and burns on a certain file. What can I do?

You can use a **FileRules** operation to exclude the particular file name, or pathname, or its title. If there are serious problems in indexing certain types of files, they may not have valid text in them (they may be binary files, for instance). You can use **NoContents** to exclude that type of file.

How do I allow users on the Web to search my indexes?

Good question. You will need a gateway €CGI€ program that presents users with a search form and options, calls swish with these options, and returns the data to them in a nice HTML format. Swish is not meant to do this. One swish-compatible gateway you can currently use is WWWWAIS, available at €`http://www.eit.com/software/wwwwais/`€.

I want to make my own gateway program.

Great! Good gateways can be made that take advantage of swish's features. If you do make one, even a simple one, please let me know and I can include it in the distribution.

Swish isn't indexing a certain word or phrase.

By default, swish tries to make it best guesses as to what it thinks are reasonable words and filters out "garbage" words according to a set of rules, for instance, if swish encounters a word that has no vowels, it doesn't index it. You can change these rules by editing the `conf.h` file in the `src` directory of the swish distribution package. By editing the rules, you may be able to index quite a few more words, or less, depending on your preference.

How can I index all my compressed files?

Swish doesn't currently have the capability to do on-the-fly filtering of files. In the meantime, first index the uncompressed data, compress it, and using a **ReplaceRules** operation, change the suffix of indexed files to **.Z** or whatever is appropriate. That way users can retrieve the compressed information.

Can I index 8-bit text?

Yes, if the text uses the HTML equivalents for the ISO-Latin-1 (ISO8859-1) character set. Upon indexing swish will convert all numbered entities it finds (such as `&#169;`) to named entities (such as `&copy;`). To search for words including these codes, type the named entity (if it exists) in place of the 8-bit character.

Swish will also convert entities to ASCII equivalents, so words that might look like this in HTML: `resum&eacute;` can be searched as this: `resume`. Please read the README file included with the distribution for information on changing these options.

How can I index phrases?

Currently the only way to do this is to use the HTML entity `&#32;` (non-breaking space) to represent a space in your HTML. It will then be indexed with a space. To search for the phrase, you'd have to enter `&#32;` to represent a space also.

How can I implement keywords in my documents?

In your HTML files you can put keywords in comments, such as:

```
<!-- keywords computer camera -->
```

...then when you search, swish should be called with the **-t c** option, such as:

```
swish -t c -w keywords computer
```

All documents that contains the words **keywords** and **computer** in their comments will then be returned.

Swish has an option in the source code that you can define to give more relevance to the words inside comments; if you're doing keywords in this fashion, you may want to use that option.

I want to generate a list of files to be indexed and pass it to swish.

One thing you can do is make a simple script to generate a configuration file full of **IndexDir** directives. For instance, make a separate file called `files.conf` and put something like this in it:

```
IndexDir /this_is_file_1/file.html
IndexDir /usr/local/www
IndexDir file2.html /some/directory/
...
```

Then call swish like this (assuming you're using a main `swish.conf` file):

```
swish -c swish.conf files.conf
```

I run out of memory trying to index my files.

It's true that indexing can take up a lot of memory! One thing you can do is make many indices of smaller content instead of trying to do everything at once. You can then merge all the smaller pieces together.

What other features are planned?

These are things that are highly dependent on how busy I get. For one, the parser could stand improvements. I have also been thinking about incorporating proximity, so that you can search for words that are close together or far apart. I would like to possibly incorporate stemming and soundex matches, but I need to do some research in this area. The ability to filter files is also on the list, as is the ability to run as a distributed server.

I know how to implement custom fields, using comments - this would require some changes to the parser and this feature hasn't been implemented in this version due to time constraints.

I would very much like to make swish aware of the other meta-indexing programs out there (such as Harvest), so it can be used as a drop-in search engine. If you have information about doing this, please let me know.

€Other pointers€

Here are some other search engines and related things out there:

· €Harvest€ is the big meta-indexer, as is €GLOSS€.
· €Glimpse€ is a grep-like engine...
· €FFW (Freetext search for Web)€ is a swish-like engine written in C++.
· €CNIDR Isite€ is a text retrieval package appears to be very `Z39.50` specific.

· Of course, to get other related information, €go to Yahoo€.

€That's it!€

I'd like to say thanks to all those who contributed input into swish - it's mostly that which has driven the development of this version.

- As always, patches, improvements, suggestions, and corrections are gratefully accepted. Send 'em all to **Kevin Hughes** at €kevinh@eit.com€.
- Due to the inordinate amount of email Kevin gets, he makes no promises that he will have time to respond to your message. He will eventually read everything you send him, however.

· **If you have questions about licensing**, please send email to Elizabeth Batson, €batson@eit.com€. She should only be contacted if you have licensing or business related questions.

Last update: 8/25/95