NAME

index − SWISH++ indexer

SYNOPSIS

index [*options*] *directory... file...*

DESCRIPTION

index is the SWISH++ file indexer. It indexes the specified files and files in the specified directories; files in subdiretories of specified directories are also indexed by default (unless either the **−r** or **−−no-recurse** option or the **RecurseSubdirs** variable is given). Files are indexed either only if their filename matches one of the patterns in the set specified with either the **−e** or **−−pattern** option or the **IncludeFile** variable (unless standard input is used; see next paragraph) or is not in the set specified with either the **−E** or **−−no-pattern** option or the **ExcludeFile** variable.

If there is a single filename of '`-`', the list of directories and files to index is instead taken from standard input (one per line). In this case, filename patterns of files to index need not be specified explicitly: all files, regardless of whether they match a pattern (unless they are in the set not to index specified with either the **−E** or **−−no-pattern** option or the **ExcludeFile** variable), are indexed, i.e., **index** assumes you know what you're doing when specifying filenames in this manner.

In any case, care must be taken not to specify files or subdirectories in directories that are also specified: since directories are recursively indexed by default (unless either the **−r** or **−−no-recurse** option or the **RecurseSubdirs** variable is given), explicitly specifying a subdirectory or file in a directory that is also specified will result in those files being indexed more than once.

Character Mapping

Characters in the ISO 8859-1 (Latin 1) character set are mapped to their closest ASCII equivalent before further examination and indexing. (Individual indexing modules may also do their own character mapping.)

Word Determination

Stop words, words that occur too frequently or have no information content, are not indexed. (There is a default built-in set of a few hundred such words.) Additionally, several heuristics are used to determine which words should not be indexed.

First, a word is checked to see if it looks like an acronym. A word is considered an acronym only if it starts with a capital letter and is composed exclusively of capital letters, digits, and punctuation symbols, e.g., ''AT&T.'' If a word looks like an acronym, it is indexed and no further checks are done.

Second, there are several other checks that are applied. A word is not indexed if it:

1. Is less than `Word_Min_Size` letters. (Default is 4.)
2. Contains less than `Word_Min_Vowels` vowels. (Default is 1.)
3. Contains more than `Word_Max_Consec_Same` of the same character consecutively (not including digits). (Default is 2.)
4. Contains more than `Word_Max_Consec_Consonants` consecutive consonants. (Default is 5.)
5. Contains more than `Word_Max_Consec_Vowels` consecutive vowels. (Default is 4.)
6. Contains more than `Word_Max_Consec_Puncts` consecutive punctuation characters. (Default is 1.)

Filters

Via the **FilterFile** configuration file variable, files matching particular patterns can be filtered prior to indexing. Via the **FilterAttachment** configuration file variable, e-mail attachments whose MIME types match particular patterns can be filtered prior to indexing. (See FILTERS in **swish++.conf**(4).)

Incremental Indexing

In order to add words from new documents to an existing index, either the entire set of documents can be reindexed or the new documents alone can be incrementally indexed. In many cases, reindexing everything is sufficient since **index** is really fast. For a very large document set, however, this may use too many

resources.

However, there is a pitfall for incremental indexing: if any of the **−f**, **−−word-files**, **−p**, or **−−word-percent** options or **WordFilesMax** or **WordPercentMax** variables are used, then words that are too frequent are discarded. If new documents are added containing very few of those words, then they could no longer be too frequent. However, there is no way to get them back since they were discarded.

The way around this problem is not to discard any words by specifying 101%. However, because no words are discarded, the size of the index file will be larger, perhaps significantly so.

It is possible that, in practice, the loss of words may not be that important especially if new documents are very similar to old documents and that words that were too frequent in the old set would also be too frequent in new set.

Another way around this problem is to do periodic full indexing.

INDEXING MODULES

index is written in a modular fashion where different types of files have different indexing modules. Currently, there are 7 modules: Text (plain text), HTML (HTML and XHTML), ID3 (ID3 tags found in MP3 files), LaTeX, Mail (RFC 822 and Usenet News), Manual (Unix manual pages in **nroff**(1) with **man**(7) macros), and RTF (Rich Text Format).

Text Module

This module simply indexes plain text files performing character mapping and word determination as has already been described.

HTML and XHTML Module

Additional processing is done for HTML and XHTML files. The additional processing is:

1. Character and numeric (decimal and hexadecimal) entity references are converted to their ASCII character equivalents before further examination and indexing. For example, ‘‘résumé’’ becomes ‘‘resume’’ before indexing.

2. If a matched set of `<TITLE>` ... `</TITLE>` tags is found within the first `TitleLines` lines of the file (default is 12), then the text between the tags is stored in the generated index file as the file's title rather than the file's name. (Every non-space whitespace character in the title is converted to a space; leading and trailing spaces are removed.)

3. If an HTML or XHTML element contains a `CLASS` attribute whose value is among the set of class names specified as those not to index (via one or more of either the **−C** or **−−no-class** option or the **ExcludeClass** variable), then all the text up to the tag that ends the element will not be indexed.

   For an element that has an optional end tag, ‘‘the tag that ends the element’’ is either the element's end tag or a tag of another element that implicitly ends it; for an element that does not have an end tag, ‘‘the tag that ends the element’’ is the element's start tag. (See the EXAMPLES.)

   All elements from the HTML 4.0 specification (including deprecated elements), Ruby elements, plus common, browser-specific elements are recognized; unrecognized elements are ignored. (See the **−H** or **−−dump-html** option.)

4. If an HTML or XHTML element contains a `TITLE` attribute, then the words specified as the value of the `TITLE` attribute are indexed.

5. If an `AREA`, `IMG`, or `INPUT` element contains an `ALT` attribute, then the words specified as the value of the `ALT` attribute are indexed.

6. If a `META` element contains both a `NAME` and `CONTENT` attribute, then the words specified as the value of the `CONTENT` attribute are indexed associated with the meta name specified as the value of the `NAME` attribute.

   (However, if either the **−A** or **−−no-assoc-meta** options or the **AssociateMeta** variable is specified, then the words specified as the value of the `CONTENT` attribute are still indexed, but not associated with the meta name.)

(See also the **−m**, **−−meta**, **−M**, and **−−no-meta** options or the **IncludeMeta** or **ExcludeMeta** variables.) Meta names can later be queried against specifically using **search**(1).

7. If a `TABLE` element contains a `SUMMARY` attribute, then the words specified as the value of the `SUMMARY` attribute are indexed.

8. If an `OBJECT` element contains a `STANDBY` attribute, then the words specified as the value of the `STANDBY` attribute are indexed.

9. All other HTML or XHTML tags and comments (anything between < and > characters) are discarded.

In compliance with the HTML specification, any one of no quotes, single quotes, or double quotes may be used to contain attribute values and attributes can appear in any order. Values containing whitespace, however, must be quoted. The specification is vague as to whether whitespace surrounding the = is legal, but **index** allows it.

ID3 Module

ID3 tags are used to store audio meta information for MP3 files (generally). Since audio files contain mostly binary information, only the ID3 tag text fields are indexed. ID3 tag versions 1.x and 2.x (through 2.4) are supported (except for encrypted frames). If a file contains both 1.x and 2.x tags, only the 2.x tag is indexed. The processing done for files containing an ID3 tag is:

1. If a title field is found, then the value of the title is stored in the generated index file as the file's title rather than the file's name. (Every non-space whitespace character in the title is converted to a space; leading and trailing spaces are removed.)

2. Words that are the value of fields are indexed associated with the field name as a meta name. (However, if either the **−A** or **−−no-assoc-meta** options or the **AssociateMeta** variable is specified, then the words specified as the value of the header are still indexed, but not associated with the header.)

   (See also the **−m**, **−−meta**, **−M**, and **−−no-meta** options or the **IncludeMeta** or **ExcludeMeta** variables.) Meta names can later be queried against specifically using **search**(1).

   For ID3v1.x, the recommended fields to be indexed are: **album**, **artist**, **comments**, **genre**, and **title**.

   For ID3v2.2, the recommended text fields (with reassignments) to be indexed are: **com=comments**, **tal=album**, **tcm=composer**, **tco=genre**, **tcr=copyright**, **ten=encoder**, **txt=lyricist**, **tt1=content**, **tt2=title**, **tt3=subtitle**, **ipl=musicians**, **tot=original-title**, **tol=original-lyricist**, **toa=original-artist**, **tp1=artist**, **tp2=performers**, **tp3=conductor**, **tpb=publisher**, **txx=user**, **slt=lyrics**, and **ult=lyrics**.

   For ID3v2.4, the recommended text fields (with reassignments) to be indexed are: **comm=comments**, **talb=album**, **tcom=composer**, **tcon=genre**, **tcop=copyright**, **tenc=encoder**, **text=lyricist**, **tipl=people**, **tit1=content**, **tit2=title**, **tit3=subtitle**, **tmcl=musicians**, **tmoo=mood**, **toal=original-title**, **toly=original-lyricist**, **tope=original-artist**, **town=owner**, **tpe1=artist**, **tpe2=performers**, **tpe3=conductor**, **tpub=publisher**, **tsst=set-subtitle**, **txxx=user**, **user=terms**, **sylt=lyrics**, and **uslt=lyrics**.

   ID3v2.3 is the same as 2.4 except replace **tmcl=musicians** with **ipls=musicians**.

   All text fields (with reassignments) for all versions of ID3 can (and should) be specified concurrently so it need not be known in advance which version(s) of ID3 MP3 files are encoded with.

3. For ID3v2.x, text fields that are compressed are uncompressed prior to indexing.

4. For ID3v2.x, Unicode text that is encoded in either UTF-8 or UTF-16 (either big- or little-endian) is decoded prior to indexing.

LaTeX Module

Additional processing is done for LaTeX files. If a **\title** command is found within the first `TitleLines` lines of the file (default is 12), then the value of the title is stored in the generated index file as the file's title rather than the file's name. (Every non-space whitespace character in the title is converted to a space; leading and trailing spaces are removed.)

Mail Module

Additional processing is done for mail and news files. The additional processing is:

1. If a **Subject** header is found within the first `TitleLines` lines of the file (default is 12), then the value of the subject is stored in the generated index file as the file's title rather than the file's name. (Every non-space whitespace character in the title is converted to a space; leading and trailing spaces are removed.)
2. Words that are the value of a header are indexed associated with the header name as a meta name. (However, if either the **−A** or **−−no-assoc-meta** options or the **AssociateMeta** variable is specified, then the words specified as the value of the header are still indexed, but not associated with the header.)

   (See also the **−m**, **−−meta**, **−M**, and **−−no-meta** options or the **IncludeMeta** or **ExcludeMeta** variables.) Meta names can later be queried against specifically using **search**(1).

   The recommended headers to be indexed are: **Bcc**, **Cc**, **Comments**, **Content-Description**, **From**, **Keywords**, **Newsgroups**, **Resent-To**, **Subject**, and **To**.
3. MIME attachments are indexed.
4. Text that is in the text/enriched content type is converted to plain text prior to indexing.
5. Text that is encoded as either quoted-printable or base-64 is decoded prior to indexing.
6. Unicode text that is encoded in either the UTF-7 or UTF-8 character set is decoded prior to indexing.
7. Text in vCards is indexed such that the values of types (fields) are associated with the types as meta names.

   The recommended vCard types (with reassignments) to be indexed are: **adr=address**, **categories**, **class**, **label=address**, **fn=name**, **nickname**, **note**, **org**, **role**, and **title**.

Indexing mail and news files is most effective only when there is exactly one message per file. While Usenet news files are usually this way, mail files are not. Mail files, e.g., mailboxes, are usually comprised of multiple messages. Such files would need to be split up into files of individual messages prior to indexing since there's no point in indexing a single mailbox: every search result would return a rank of 100 for the same file. Therefore, the **splitmail**(1) utility is included in the SWISH++ distribution.

Manual Module

Additional processing is done for Unix manual page files. The additional processing is:

1. If a **NAME** section heading macro (`.SH`) is found within the first `TitleLines` lines of the file (default is 12), then the contents of the next line are stored in the generated index file as the file's title rather than the file's name. (Every non-space whitespace character in the title is converted to a space; leading and trailing spaces as well as backslash sequences, such as `\f2`, are removed.)
2. Words that are in a section are indexed associated with the name of the section as a meta name. (However, if either the **−A** or **−−no-assoc-meta** options or the **AssociateMeta** variable is specified, then the words in a section are still indexed, but not associated with the section heading.)

   Spaces in multi-word section headings are converted to dashes, e.g., ''see also'' becomes ''see-also'' as a meta name. (See also the **−m**, **−−meta**, **−M**, and **−−no-meta** options or the **IncludeMeta** or **ExcludeMeta** variables.) Meta names can later be queried against specifically using **search**(1).

   The recommended sections to be indexed are: **AUTHOR**, **BUGS**, **CAVEATS**, **DESCRIPTION**, **DIAGNOSTICS**, **ENVIRONMENT**, **ERRORS**, **EXAMPLES**, **EXIT-STATUS**, **FILES**, **HISTORY**, **NAME**, **NOTES**, **OPTIONS**, **RETURN-VALUE**, **SEE-ALSO**, **SYNOPSIS**, and **WARNINGS**.

RTF Module

This module simply indexes rich text format files without all formatting commands.

OPTIONS

Options begin with either a '−' for short options or a ''−−'' for long options. Either a '−' or ''−−'' by itself explicitly ends the options; either short or long options may be used. Long option names may be abbreviated so long as the abbreviation is unambiguous.

For a short option that takes an argument, the argument is either taken to be the remaining characters of the same option, if any, or, if not, is taken from the next option unless said option begins with a '`-`'.

Short options that take no arguments can be grouped (but the last option in the group can take an argument), e.g., `-lrv4` is equivalent to `-l -r -v4`.

For a long option that takes an argument, the argument is either taken to be the characters after a '=', if any, or, if not, is taken from the next option unless said option begins with a '`-`'.

–?
––help Print the usage ("help") message and exit.

–A
––no-assoc-meta Do not associate words with meta names.

–c*f*
––config-file=*f* The name of the configuration file, *f*, to use. (Default is `swish++.conf` in the current directory.) A configuration file is not required: if none is specified and the default does not exist, none is used; however, if one is specified and it does not exist, then this is an error.

–C*c*
––no-class=*c* For HTML or XHTML files only, a class name, *c*, of an HTML or XHTML element whose text is not to be indexed. Multiple **–C** or **––no-class** options may be specified.

–e*m***:***p***[,***p***...]**
––pattern=*m***:***p***[,***p***...]** A module name, *m*, and a filename pattern (or set of patterns separated by commas), *p*, of files to index. Case is irrelevant for the module name, but significant for the patterns. Multiple **–e** or **––pattern** options may be specified.

–E*p***[,***p***...]**
––no-pattern=*p***[,***p***...]** A filename pattern (or set of patterns separated by commas), *p*, of files *not* to index. Case is significant. Multiple **–E** or **––no-pattern** options may be specified.

–f*n*
––word-files=*n* The maximum number of files, *n*, a word may occur in before it is discarded as being too frequent. (Default is infinity.)

–F*n*
––files-reserve=*n* Reserve space for this number of files, *n*, to start. More space will be allocated as necessary, but with a slight performance penalty. (Default is 1000.)

–g*n*
––files-grow=*n* Grow the space for the reserved number of files, *n*, when incrementally indexing. The number can either be an absolute number of files or a percentage (when followed by a percent sign `%`). Just as with the **–F** option, more space will be allocated as necessary, but with a slight performance penalty. (Default is 100.)

–H
––dump-html Dump the built-in set of recognized HTML and XHTML elements to standard output and exit.

–i*f*
––index-file=*f* The name of the generated index file, *f* (for new indexes; default is `swish++.index` in the current directory) or the old index file when doing incremental indexing.

–I
––incremental Incrementally add the indexed files and words to an existing index. The existing index is not touched; instead, a new index is created having the same pathname of the existing index with "`.new`" appended.

−l
−−follow-links Follow symbolic links during indexing. (Default is not to follow them.) This option is not available under Microsoft Windows since it doesn't support symbolic links.

−m*m*[=*n*]
−−meta=*m*[=*n*] The value of a meta name, *m*, for which words are to be associated when indexed. Case is irrelevant. Multiple **−m** or **−−meta** options may be specified.

A meta name can be reassigned when followed by a new name, *n*, meaning that the name *n* and not *m* is stored in the generated index file so that queries would use the new name rather than the original.

By default, words associated with all meta names are indexed. Specifying at least one meta name via this option changes that so that only the words associated with a member of the set of meta names explicitly specified via one or more **−m** or **−−meta** options are indexed.

−M*m*
−−no-meta=*m* The value of a meta name, *m*, for which words are not to be indexed. Case is irrelevant. Multiple **−M** or **−−no-meta** options may be specified.

−p*n*
−−word-percent=*n* The maximum percentage, *n*, of files a word may occur in before it is discarded as being too frequent. (Default is 100.) If you want to keep all words regardless, specify 101.

−r
−−no-recurse Do not recursively index the files in subdirectories, that is: when a directory is encountered, all the files in that directory are indexed (modulo the filename patterns specified via either the **−e**, **−−pattern**, **−E**, or **−−no-pattern** options or the **IncludeFile** or **ExcludeFile** variables) but subdirectories encountered are ignored and therefore the files contained in them are not indexed. This option is most useful when specifying the directories and files to index via standard input. (Default is to index the files in subdirectories recursively.)

−s*f*
−−stop-file=*f* The name of a file, *f*, containing the set of stop-words to use instead of the built-in set. Whitespace, including blank lines, and characters starting with # and continuing to the end of the line (comments) are ignored.

−S
−−dump-stop Dump the built-in set of stop-words to standard output and exit.

−t*n*
−−title-lines=*n* The maximum number of lines, *n*, into a file to look at for a file's title. (Default is 12.) Larger numbers slow indexing.

−T*d*
−−temp-dir=*d* The path of the directory, *d*, to use for temporary files. The directory must exist. (Default is `/tmp` for Unix or `/temp` for Windows.)

If your OS mounts swap space on `/tmp`, as indexing progresses and more files get created in `/tmp`, you will have less swap space, indexing will get slower, and you may run out of memory. If this is the case, you should specify a directory on a real filesystem, i.e., one on a physical disk.

−v*n*
−−verbosity=*n* The verbosity level, *n*, for printing additional information to standard output during indexing. The verbosity levels, 0-4, are:

0 No output is generated except for errors. (This is the default.)
1 Only run statistics (elapsed time, number of files, word count) are printed.
2 Directories are printed as indexing progresses.
3 Directories and files are printed with a word-count for each file.
4 Same as 3 but also prints all files that are not indexed and why.

–V
––version Print the version number of **SWISH++** to standard output and exit.

–W*n*
––word-threshold=*n* The word count past which partial indicies are generated and merged since all the words are too big to fit into memory at the same time. If you index and your machine begins to swap like mad, lower this value. Only the super-user can specify a value larger than the compiled-in default.

CONFIGURATION FILE

The following variables can be set in a configuration file. Variables and command-line options can be mixed, the latter taking priority.

AssociateMeta	Same as **–A** or **––no-assoc-meta**
ExcludeClass	Same as **–C** or **––no-class**
ExcludeFile	Same as **–E** or **––no-pattern**
ExcludeMeta	Same as **–M** or **––no-meta**
FilesGrow	Same as **–g** or **––files-grow**
FilesReserve	Same as **–F** or **––files-reserve**
FilterAttachment	(See FILTERS in **swish++.conf**(4).)
FilterFile	(See FILTERS in **swish++.conf**(4).)
FollowLinks	Same as **–l** or **––follow-links**
IncludeFile	Same as **–e** or **––pattern**
IncludeMeta	Same as **–m** or **––meta**
Incremental	Same as **–I** or **––incremental**
IndexFile	Same as **–i** or **––index-file**
RecurseSubdirs	Same as **–r** or **––no-recurse**
StopWordFile	Same as **–s** or **––stop-file**
TempDirectory	Same as **–T** or **––temp-dir**
TitleLines	Same as **–t** or **––title-lines**
Verbosity	Same as **–v** or **––verbosity**
WordFilesMax	Same as **–f** or **––word-files**
WordPercentMax	Same as **–p** or **––word-percent**
WordThreshold	Same as **–W** or **––word-threshold**

EXAMPLES

Unix Command-Lines

All these example assume you change your working directory to your web server's document root prior to indexing.

To index all HTML and text files on a web server:

```
index -v3 -e 'html:*.*htm*' -e 'text:*.txt' .
```

To index all files not under directories named `CVS`:

```
find . -name CVS -prune -o -type f -a -print | index -e 'html:*.*htm*' -
```

Windows Command-Lines

When using the Windows command interpreter, single quotes around filename patterns don't work; you *must* use double quotes:

```
index -v3 -e "html:*.*htm*" -e "text:*.txt" .
```

This is a problem with Windows, not SWISH++. (Double quotes will also work under Unix.)

Using `CLASS` **Attributes to Index HTML Selectively**

In an HTML or XHTML document, there may be sections that should not be indexed. For example, if every page of a web site contains a navigation menu such as:

```
<SELECT NAME="menu">
  <OPTION>Home
  <OPTION>Automotive
  <OPTION>Clothing
  <OPTION>Hardware
</SELECT>
```

or a common header and footer, then, ordinarily, those words would be indexed for every page and therefore be discarded because they would be too frequent. However, via either the **-C** or **--no-class** option or the **ExcludeClass** variable, one or more class names can be specified and then HTML or XHTML elements belonging to one of those classes will not have the text up to the tag that ends them indexed. Given a class name of, say, `no_index`, the above menu can be changed to:

```
<SELECT NAME="menu" CLASS="no_index">
```

and then everything up to the `</SELECT>` tag will not be indexed.

For an HTML element that has an optional end tag (such as the `<P>` element), the text up to the tag that ends it will not be indexed, which is either the element's own end tag or a tag of some other element that implicitly ends it. For example, in:

```
<P CLASS="no_index">
This was the poem that Alice read:
<BLOCKQUOTE>
  <B>Jabberwocky</B><BR>
  'Twas brillig, and the slithy toves<BR>
  Did gyre and gimble in the wabe;<BR>
  All mimsy were the borogoves,<BR>
  And the mome raths outgrabe.
</BLOCKQUOTE>
```

the `<BLOCKQUOTE>` tag implicitly ends the `<P>` element (as do all block-level elements) so the only text that is not indexed above is: "This was the poem that Alice read."

For an HTML or XHTML element that does not have an end tag, only the text within the start tag will not be indexed. For example, in:

```
<IMG SRC="home.gif" ALT="Home" CLASS="no_index">
```

the word "Home" will not be indexed even though it ordinarily would have been if the `CLASS` attribute were not there.

Filters

(See Filters under EXAMPLES in **swish++.conf**(4).)

EXIT STATUS

Exits with one of the values given below:

0 Success.

1	Error in configuration file.
2	Error in command-line options.
10	Unable to open temporary file.
11	Unable to write index file.
12	Unable to write temporary file.
13	Root-only operation attempted.
30	Unable to read stop-word file.
40	Unable to read index file.
127	Internal error.

CAVEATS

1. Generated index files are machine-dependent (size of data types and byte order).
2. The word-determination heuristics employed are heavily geared for English. Using SWISH++ as-is to index and search files in non-English languages is not recommended.
3. Unless otherwise noted above, the character encoding always used is ISO 8859-1 (Latin 1). Character encodings that are specified in HTML or XHTML files are ignored.
4. An e-mail message can have both an encoding and a non-ASCII or non-ISO-8859-1 charset simultaneously, e.g., base64-encoded UTF-8. (In practice, this particular case should never happen since UTF-7 should be used instead; but you get the idea.)

   However, handling both an encoding and such a charset simultaneously is problematic; hence, an e-mail message or attachment can have either an encoding or a non-ASCII or a non-ISO-8859-1 character set, but not both. If it does, the encoding takes precedence.

FILES

`swish++.conf`	default configuration file name
`swish++.index`	default index file name

ENVIRONMENT

`TMPDIR` If set, the default path of the directory to use for temporary files. The directory must exist. This is superseded by either the **−T** or **−−temp-dir** option or the **TempDirectory** variable.

SEE ALSO

extract(1), **find**(1), **nroff**(1), **search**(1), **splitmail**(1), **swish++.conf**(4), **glob**(7), **man**(7).

Tim Berners-Lee. ‘‘The text/enriched MIME Content-type,’’ *Request for Comments 1563*, Network Working Group of the Internet Engineering Task Force, January 1994.

David H. Crocker. ‘‘Standard for the Format of ARPA Internet Text Messages,’’ *Request for Comments 822*, Department of Electrical Engineering, University of Delaware, August 1982.

Frank Dawson and Tim Howes. ‘‘vCard MIME Directory Profile,’’ *Request for Comments 2426*, Network Working Group of the Internet Engineering Task Force, September 1998.

Ned Freed and Nathaniel S. Borenstein. ‘‘Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies,’’ *Request for Comments 2045*, RFC 822 Extensions Working Group of the Internet Engineering Task Force, November 1996.

David Goldsmith and Mark Davis. ‘‘UTF-7, a mail-safe transformation format of Unicode,’’ *Request for Comments 2152*, Network Working Group of the Internet Engineering Task Force, May 1997.

International Standards Organization. *ISO 8859-1: Information Processing -- 8-bit single-byte coded graphic character sets -- Part 1: Latin alphabet No. 1,* 1987.

−−. *ISO 8879: Information Processing -- Text and Office Systems -- Standard Generalized Markup Language (SGML),* 1986.

−−. *ISO/IEC 9945-2: Information Technology -- Portable Operating System Interface (POSIX) -- Part 2: Shell and Utilities,* 1993.

Leslie Lamport. *LaTeX: A Document Preparation System, 2nd ed.*, Addison-Wesley, Reading, MA, 1994.

Martin Nilsson. *ID3 tag version 2*, March 1998.

−−. *ID3 tag version 2.3.0*, February 1999.

−−. *ID3 tag version 2.4.0 - Main Structure*, November 2002.

−−. *ID3 tag version 2.4.0 - Native Frames*, November 2002.

Steven Pemberton, et al. *XHTML 1.0: The Extensible HyperText Markup Language*, World Wide Web Consortium, January 2000.

Dave Raggett, Arnaud Le Hors, and Ian Jacobs. ''On SGML and HTML: SGML constructs used in HTML: Entities,'' *HTML 4.0 Specification, §3.2.3,* World Wide Web Consortium, April 1998.

−−. ''The global structure of an HTML document: The document head: The `title` attribute,'' *HTML 4.0 Specification, §7.4.3,* World Wide Web Consortium, April 1998.

−−. ''The global structure of an HTML document: The document head: Meta data,'' *HTML 4.0 Specification, §7.4.4,* World Wide Web Consortium, April 1998.

−−. ''The global structure of an HTML document: The document body: Element identifiers: the `id` and `class` attributes,'' *HTML 4.0 Specification, §7.5.2,* World Wide Web Consortium, April 1998.

−−. ''Tables: Elements for constructing tables: The `TABLE` element,'' *HTML 4.0 Specification, §11.2.1,* World Wide Web Consortium, April 1998.

−−. ''Objects, Images, and Applets: Generic inclusion: the `OBJECT` element,'' *HTML 4.0 Specification, §13.3,* World Wide Web Consortium, April 1998.

−−. ''Objects, Images, and Applets: How to specify alternate text,'' *HTML 4.0 Specification, §13.8,* World Wide Web Consortium, April 1998.

−−. ''Index of Elements,'' *HTML 4.0 Specification,* World Wide Web Consortium, April 1998.

Marcin Sawicki, et al. *Ruby Annotation*, World Wide Web Consortium, April 2001.

The Unicode Consortium. ''Encoding Forms,'' *The Unicode Standard 3.0, §2.3,* Addison-Wesley, 2000.

Francois Yergeau. ''UTF-8, a transformation format of ISO 10646,'' *Request for Comments 2279*, Network Working Group of the Internet Engineering Task Force, January 1998.

AUTHOR

Paul J. Lucas <*pauljlucas@mac.com*>