NAME

swish++.conf − SWISH++ configuration file format

DESCRIPTION

The configuration file format used by SWISH++ consists of three types of lines: blank lines, comments, and variable definitions.

Blank lines

Blank lines, or lines consisting entirely of whitespace, are ignored.

Comments

Comments start with the # character and continue up to and including the end of the line. While leading whitespace is permitted, **comments are treated as such only if they are on lines by themselves**.

Variable definitions

Variable definition lines are of the form:

variable_name argument(s)

where *variable_name* is a member of one of the types described in the remaining sections, and *argument(s)* are specific to every variable name. For *variable_name*, case is irrelevant.

Boolean variables

Variables of this type take one argument that must be one of: `f`, `false`, `n`, `no`, `off`, `on`, `t`, `true`, `y`, or `yes`. Case is irrelevant. Variables of this type are: **AssociateMeta**, **ExtractFilter**, **FollowLinks**, **Incremental**, **RecurseSubdirs**, **SearchBackground**, **StemWords**, and **StoreWordPositions**.

Enumeration variables

Variables of this type are just like string variables (see below) except that the argument *must* be one of a set of pre-determined values. Case is irrelevant. Variables of this type are: **ResultsFormat** and **SearchDaemon**. **ResultsFormat** must be either: `classic` or `XML`. **SearchDaemon** must be one of: `none`, `tcp`, `unix`, or `both`.

Filter variables

Variables of this type are of the form:

pattern command

where *pattern* is a shell pattern (regular expression) and *command* is the command-line to execute the filter.

Within a command, there are a few `%` substitutions that are done at run-time:

- **b** Basename of filename.
- **B** Basename minus last extension.
- **e** Extension of filename.
- **E** Second-to-last extension of filename.
- **f** Entire filename.
- **F** Filename minus last extension.

That is: the `%` and one character immediately after it are substituted as described in the above table. Substituted filenames are skipped past and not rescanned for more substitutions, but the remainder of the command is. To use a literal `%` or `@`, simply double it. (For more on filter variables, see FILTERS below.)

Variables of this type are: **FilterAttachment** and **FilterFile**.

Integer variables

Variables of this type take one numeric argument. A special string of `infinity` is taken to mean ‘‘the largest possible integer value.’’ Case is irrelevant. Variables of this type are: **FilesReserve**, **ResultsMax**, **SocketQueueSize**, **SocketTimeout**, **ThreadsMax**, **ThreadsMin**, **ThreadTimeout**, **TitleLines**, **Verbosity**, **WordFilesMax**, **WordPercentMax**, **WordsNear**, and **WordThreshold**.

For **WordThreshold**, only the super-user can specify a value larger than the compiled-in default.

Percentage variables
Variables of this type are like integer variables except that if an optional trailing percent sign (`%`) is present, the value is taken to be a percentage rather than an absolute number. Variables of this type are: **FilesGrow**.

String variables
Variables of this type take one argument that is the remainder of the line minus leading and trailing whitespace. To preserve whitespace, surround the argument in either single or double quotes. The quotes themselves are stripped from the argument, but only if they match. Variables of this type are: **ExtractExtension**, **Group**, **IndexFile**, **PidFile**, **ResultSeparator**, **SocketFile**, **StopWordFile**, **TempDirectory**, and **User**.

Set variables
Variables of this type take one or more arguments separated by whitespace. Variables of this type are: **ExcludeClass**, **ExcludeFile**, **ExtractFile**, and **ExcludeMeta**.

Other variables
Variables of this type are: **IncludeFile**, **IncludeMeta**, and **SocketAddress**.

An **IncludeFile** configuration file line is of the form:

module_name pattern ...

where *module_name* is the name of the module (case is irrelevant) to handle the indexing of the filename *pattern*s that follow. Module names are: `text` (plain text), `HTML` (HTML and XHTML), `ID3` (ID3 tags), `LaTeX` (LaTeX source), `Mail` (mail and news messages), `Man` (Unix manual pages), and `RTF` (Rich Text Format).

An **IncludeMeta** configuration file line is of the form:

`name`**[**=*new_name*]**** ...

It is like a set variable except arguments may optionally be followed by reassignments. For example, a value of:

```
adr=address
```

says to include and index the words associated with the meta name `adr`, but to store the name as `address` in the generated index file so that queries would use `address` rather than `adr`.

A **SocketAddress** configuration file line is of the form:

[*host* **:** **]** *port*

that is: an optional host and colon followed by a port number. The *host* may be one of a host name, an IPv4 address (in dot-decimal notation), an IPv6 address (in colon notation) if supported by the operating system, or the `*` character meaning "any IP address." Omitting the *host* and colon also means "any IP address."

FILTERS

Filtering files
Via the **FilterFile** configuration file variable, files matching patterns can be filtered prior to indexing or extraction. For example, to uncompress `bzip2`'d, `gzip`'d, and `compress`'d files prior to indexing or extraction, the **FilterFile** variable lines in a configuration file would be:

```
FilterFile *.bz2  bunzip2 -c %f > @%F
FilterFile *.gz   gunzip -c %f > @%F
FilterFile *.Z    uncompress -c %f > @%F
```

Given that, a filename such as `foo.txt.gz` would become `foo.txt`. If files having `txt` extensions should be indexed, then it will be. Note that the command on the **FilterFile** line must *not* simply be:

```
gunzip @%f                          # WRONG!
```

because `gunzip` will *replace* the compressed file with the uncompressed one.

Here's an example to convert PDF to plain text for indexing using the **xpdf**(1) package's `pdftotext` command:

```
FilterFile *.pdf  pdftotext %f @%F.txt
```

A file can be filtered more than once prior to indexing or extraction, i.e., filters can be ''chained'' together. For example, if the uncompression and PDF examples shown above are used together, compressed PDF files will also be indexed or extracted, i.e., filenames ending with one of double extensions.

Note, however, that just because a filename has an extension for which a filter has been specified does *not* mean that a file will be filtered and subsequently indexed or extracted. When **index** or **extract** encounters a file having an extension for which a filter has been specified, it performs the filename substitution(s) on it first to determine what the target filename would be. If the extension of *that* filename should be indexed or extracted (because it is among the set of extensions specified with either the **−e** or **−−pattern** options or the **IncludeFile** variable or is not among the set specified with either the **−E** or **−−no-pattern** options or the **ExcludeFile** variable), *then* the filter(s) are executed to create it.

Filtering attachments

Via the **FilterAttachment** configuration file variable, e-mail attachments whose MIME types match particular patterns can be filtered and thus indexed. An attachment is written to a temporary file by itself (after having been base-64 decoded, if necessary) and a filter command is called on that file.

For example, to convert a PDF attachment to plain text so it can be indexed, the **FilterAttachment** variable line in a configuration file would be:

```
FilterAttachment application/pdf    pdftotext %f @%F.txt
```

MIME types *must* be specified entirely in lower case. Patterns can be useful for MIME types. For example:

```
FilterAttachment application/*word  extract -f %f > @%F.txt
```

can be used regardless of whether the MIME type is `application/msword` (the official MIME type for Microsoft Word documents) or `application/vnd.ms-word` (an older version).

The MIME types that are built into **index**(1) are: `text/plain`, `text/enriched` (but only if the RTF module is compiled in), `text/html` (but only if the HTML module is compiled in), `text/*vcard`, `message/rfc822`, `multipart/`*something* (where *something* is one of: `alternative`, `mixed`, or `parallel`). **FilterAttachment** variable lines can override the handling of the built-in MIME types.

Unlike file filters, attachment filters *must* convert directly to plain text and can not be ''chained'' together. (This restriction exists because there is no way to know what any intermediate MIME types would be to apply more filters.)

SEE ALSO

bzip(1), **compress**(1), **extract**(1), **gunzip**(1), **gzip**(1), **index**(1), **pdftotext**(1), **search**(1), **uncompress**(1), **glob**(7)

Nathaniel S. Borenstein. ''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.

International Standards Organization. ‘‘ISO/IEC 9945-2: Information Technology -- Portable Operating System Interface (POSIX) -- Part 2: Shell and Utilities,’’ 1993.

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

AUTHOR

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