SGML2TEX

mailto:pflynn@curia.ucc.ie (Peter Flynn)

Converted from sgml2tex.html on 31/03/1994 at 16:04:46 by SGML2TEX

Contents

SGML2TEX

SGML2TEX is a prototype program to convert the SGML tags in a document to control sequences using the conventions of TEX. This document reflects v0.95β of the program.

Environment

The program is currently written in PCL, a language developed explicitly for the Intel 80*86 chips, because of the speed of execution possible and the availability of a royalty-free run-time execute module (included with the distribution of this program). The program is therefore restricted in its current state to execution on MS-DOS or in an MS-DOS task window of DesqView, DesqView/X or MS-Windows. A future version will be written in a more portable language system, probably CWEB.

SGML is the Standard Generalised Markup Language (ISO 8879), the international standard for text markup. An 'SGML document' in the terms of the SGML2TEX program means an SGML 'document instance': that is, the user text of an SGML document proper, without any DTD (Document Type Description) or other markup declarations. SGML2TEX does not perform any parsing or validating of the SGML document, as public-domain parsers are freely available for this purpose. It is therefore your responsibility to ensure that only a valid conformant orthogonal SGML document instance without minimisation is processed by this program, with the syntax inherent in the SGML Reference Concrete Syntax. No responsibility can be taken for the results of processing any other form of SGML text, but suggestions for improvement are always welcomed. (In plain language this means the program will process a normalised SGML file with no DTD or SGML declaration attached, but nothing else.)

TEX is the typesetting system devised by Donald Knuth, and its add-on variants such as LATEX. The code output by the SGML2TEX program includes empty definitions for all elements, attributes and entities encountered, in a dummy style file. It is your responsibility to implement the style file in order to achieve the desired result. A configuration file option is available for the predefinition of known tags. The TEX system is available for almost all platforms in a choice of public-domain/shareware/freeware or commercial implementations: contact the TEX Users Group, Box 21041, Santa Barbara, CA 93121-1041, USA for further details (phone: [+1] (805) 963 1338; fax: [+1] (805) 963 8358; email tug@tug.org)

This is beta-release software. The program appears to perform as indicated below, but you are asked to report any bugs encountered to the author. The program is copyright of the author but unrestricted redistribution is permitted provided no modifications are made.

Installation

The program is distributed as a `.zip` file, so it must be copied to the root directory of your disk and unwrapped with the command `pkunzip -d -o sgml2tex` This creates a directory called `\pcl`, containing the program and the PCL runtime module, and a batch file `sgml2tex.bat` in the root directory of the disk where the unzipping took place. This batch file can be moved to wherever in the path you keep your batch utilities.

The `sgml2tex.htm` file (this documentation) and its conversions are unwrapped into the `\sgml` directory. It is suggested that this directory is used for testing. Details of the HTML format (an application of SGML) are available online[1].

The TEX `eplain` macros are used in the default translation, and a copy of `eplain.tex` is included in the `.zip` file: this is unwrapped into the `\emtex\texinput` directory, so if you are not using emTEX, you should move this to wherever you keep your TEX macro files.

The batch file performs all necessary path-setting for execution, and resets the path afterwards, so no modification to `config.sys` or `autoexec.bat` is necessary.

A preprocessed copy of this document is included as file `sgml2tex.ps` for printing on PostScript printers.

---

[1] See `http://www.ucc.ie/info/net/htmldoc.html`

Running the program

If the `sgml2tex.bat` file is used, the command to run the program is `sgml2tex [/option [filename] ... ] sgmlfile [texfile [stylefile]]`
where

- the commandline `/options` are one or more of:
  - `/l` [`filename`] to specify that logging is required. If no filename is given, the name of the SGML input file is used, with the filetype of `.log` (logging is copious and requires a lot of disk space);
  - `/d configname` to specify the name of a configuration file. A filetype of `.cfg` is assumed;
  - If `/l` is used (with or without a filename argument), it must precede `/d`.

- `sgmlfile` is the name of the SGML document to be processed. If no filetype is given, `.sgml` is tried, and failing that, the name specified by the `/d` option (if used) is tried. If no file by that name can be found, the user is prompted for another name;
- `texfile` is a name for the converted output file. If this is not supplied, the name of the SGML document is used, with a filetype of `.tex`;
- `stylefile` is a name for the output style file. If this is not supplied, the name of the SGML document is used, with a filetype of `.sty`;
- If either of the `.tex` or `.sty` files exist, the user is prompted to confirm before they are overwritten, with the option of giving a new name or quitting the program. The log file (if logging is requested) is always overwritten.
- If all three filename arguments are validly supplied on the commandline, the program does not prompt for confirmation if they exist, but goes ahead and overwrites the output files. This is to enable unattended batch operation.
- If no configuration file name is given in the `/d` option, the default configuration

file `sgml2tex.cfg` is used.

If the batch file is not used, the full command is `pcl run sgml2tex` with options and arguments as before. In this case it is your responsibility to ensure that the `\pcl` directory is accessible to the DOS path.

Processing

During processing, a percentage bar indicator shows how much of the file has been processed. Counters are displayed for lines, characters and words processed. Execution can be interrupted at any stage with Ctrl-Break, and the command `quit` can then be used to leave the program after doing so. After execution, control is returned to the DOS prompt.

SGML elements in the document are converted to a TEX-compatible form. The program represents the SGML of the source document by

- prefixing start-tags with `\start` and end-tags with `\finish`, leaving the SGML tag-names intact but capitalised, and suffixed with curly braces ({}). A null entry (dummy `\def`) is made in the style file for the user to implement;
- converting character entities in the form `&name;` to the form `\name{}` and similarly identifying them in the style file;
- converting attributes in the same way but giving their values in curly braces as TEX arguments. The style file entry the dummy definition with one `#1` argument to mark this.

All multiple spaces, tabs and linends are considered equivalent to a single space unless they are the content of an element defined in the configuration file as ‘protected’ (see details of the `special` keyword [p. 9] in the section on configuration files below).

Example

For an example, the SGML fragment

```
<p>Goethe's use of storm imagery can be summarised in the last
lines of <booktitle ref="43">Torquato Tasso</booktitle>:
here we find the phrase <quote>berstend rei&szlig;t / Der Boden
unter meinen F&uuml;&szlig;en auf.</quote></p>
```

is converted to

```
\startP{}Goethe's use of storm imagery can be summarised in the last
lines of \startBOOKTITLE{}\REF{43}Torquato Tasso\finishBOOKTITLE{}:
here we find the phrase \startQUOTE{}berstend rei\szlig{}t / Der Boden
unter meinen F\uuml{};\szlig{};en auf.\finishQUOTE{}\finishP{}
```

The style file output with the example above would contain the following entries:

```
\def\startP{}
\def\finishP{}
\def\startBOOKTITLE{}
\def\finishBOOKTITLE{}
\def\startQUOTE{}
\def\finishQUOTE{}
\def\REF1{}
\def\szlig{}
\def\uuml{}
```

It is the user's responsibility to define these adequately, for example:

```
\def\startP{}
\def\finishP{\par}
\def\startBOOKTITLE{\it}
\def\finishBOOKTITLE{\footnote*{\the\fntext{}}\rm}
\def\startQUOTE{``}
\def\finishQUOTE{''}
\newtoks\fntext
\def\REF1{\fntext={1}}
\def\szlig{\ss}
\def\uuml{\"u}
```

in a way which is meaningful for the visual appearance required.

The configuration file

The default configuration file for a given SGML document is taken from the file type of the SGML document file, but with its own file type of `.cfg` unless the `/d` option specifies otherwise (if this is not used, the default file `sgml2tex.cfg` is used, if present).

The configuration file can establish predefined equivalences for element names, attribute names and character entities, so avoiding the need to hand-edit a style file, and allowing further files using the same configuration file to be processed with reference to an existing style file.

The following statements can be put in the configuration file (a worked example is at the end of this document). The delimiter between tokens is one or more spaces. For this reason, space characters are not currently permitted within the TeX-strings.

`element` *name TeX-start-string TeX-end-string*

- The keyword '`element`' is required, followed by
  1. the name of the element in the SGML document (without STAGO (>) or TAGC (<) delimiters);
  2. some TeX code to use when the start-tag is encountered;
  3. some TeX code to use when the end-tag is encountered.

- If either TeX-string is a hyphen (-), then replacement will not be used, and the default action will be performed (translation using `\start` or `\finish` respectively, as described above). If either TeX-string is a caret (^), then that start-tag or end-tag respectively will be omitted from the output entirely.
- The two TeX-strings can include the following substitution parameters:
  - `%s` – the current SGML document file name
  - `%e` – the current SGML document file type

- `%n` – a forced newline
- `%d` – the current date
- `%t` – the current time
- `%%` – a TEX comment character

`attribute name TEX-pre-string TEX-post-string`

- The keyword '`attribute`' is required, followed by

  1. the name of the attribute in the SGML document;
  2. some TEX code to use when the attribute is encountered (to precede the attribute value);
  3. some TEX code to use to follow the attribute value.

`entity` *name TEX-string*

- The keyword '`entity`' is required, followed by the entity name (without ERO (&) or REFC (;) delimiters), and some TEX code to substitute when it is encountered.

`special` *name keyword*

- The ⇒keyword '`special`' is required, followed by an SGML element name and one of the following settings:

  - `formatted` – the element content will be processed as normal (*ie* embedded elements, entities and attributes will be converted), but spaces, tabs and linends will be respected;
  - `uninterpreted` – the element content will not be converted: sample SGML code contained in it will be left untouched, but spaces, tabs and linends will be collapsed as in normal processing;
  - `unprocessed` – element content will not be converted and spaces, tabs and linends will be respected.

`style` *name*

- The keyword '`style`' is required, followed by a filename of a style file to use in place of the default. This name is included in the preamble to the output `.tex` file using an `\input` statement.

`map` *char string [string]*

- The keyword 'map' is required, followed by a single character to remap, and a string to remap it to. An optional second string provides an alternate mapping for use in attribute values, where normal string mapping may be undesirable.

Here is an example, suitable for the Goethe text quoted earlier:

```
element   p          ^       \par{}
element   booktitle  \it{}    \footnote*{\the\fntext{}}\rm{}
element   quote      ``      ''
attribute  ref       \fntext={  }}
entity    szlig      \ss{}
entity    uuml       \"u
style     german.sty
map       \          \\
```

The file german.sty is the user's responsibility but would be presumed to contain any further TEX code required.

A sample configuration file and style file, html.cfg and html.sty, are provided. The program can be tested with this present file, sgml2tex.htm, in order to convert, process and print the documentation.