Perl Program Reference

tpt-perl-crf

Generated by Doxygen 1.2.11.1

Jan 13, 2003

Contents

Chapter 1

Voyant TechPubs Tools Main Page

Technical Publications (TechPubs) Tools are Perl programs that are used to create a comprehensive HTML system from mini-HTML systems created by Doxygen and Mif2Go.

These tools describes one possible solution to tackle two major tasks associated with application programming interface (API) technical publications:

- Multiple output formats (HTML and PDF) that are single-sourced from (FrameMaker) documentation in a modular way.
- API reference documentation that is single-sourced from the (C/C++) source code into multiple output formats.

Although the TechPubTools and techniques were designed specifically for C/C++ API documentation, they can be adopted and modified:

- to produce large online documentation systems from multiple FrameMaker books.
- to produce large online documentation systems from multiple Doxygen directories.
- to handle (imperfectly) other programming languages.

Warning:

The Perl Reference portion of this documentation has imperfections stemming from running Perl files through Doxygen, which was designed to handle C/C++.

Limitations and Caveats:

Note:

Documents produced by these tools are derivative works derived from the input used in their production; they are not affected by this license.

Chapter 2

Voyant TechPubs Tools Package List

2.1 Voyant TechPubs Tools Package List

Here are the packages with brief descriptions (if available):

Chapter 3

Voyant TechPubs Tools Module Index

3.1 Voyant TechPubs Tools Modules

Here is a list of all modules:

Chapter 4

Voyant TechPubs Tools File Index

4.1 Voyant TechPubs Tools File List

Here is a list of all files with brief descriptions:

Chapter 5

Voyant TechPubs Tools Page Index

5.1 Voyant TechPubs Tools Related Pages

Here is a list of all related documentation pages:

Chapter 6

Voyant TechPubs Tools Package Documentation

6.1 Package gen_nav

6.2 Package globe

6.3 Package scheck

6.4 Package xhelp

6.5 Package xscope

Chapter 7

Voyant TechPubs Tools Module Documentation

7.1 TechPubs Tools

Tools for technical publications departments.

Files

- file asn_bapitypes.pl

  Looks at asn files and stores information about enumerations so that it can be merged with with bapitypes.h.

- file csh_comment_chg.pl

  Changes the b shell script files so that it "looks" like a C file with respect to the comment format and function names.

- file dox_bug_filter.pl

  Input filter for Doxygen for most C/C++ code files.

- file dox_chg_not.pl

  Tool for Doxygen for most C/C++ code files.

- file dox_comment_chg.pl

  Used when Doxygenating C/C++ code files.

- file dox_ive_filter.pl

  Input filter for Doxygen for most IVE code files.

- file dox_vxworks_filter.pl

  Input filter for Doxygen for most VxWorks code files.

- file find_extract.pl

  Generates a file with command definitions and doxygen comments, intended as input to Doxgyen without as much noise.

- file globe.pm

  Defines global variables that can be used by other programs.

- file html_look_integrate.pl

  Looks into HTML files that have been provided by other sources and parses them for useful information.

- file html_look_integrate.pm

  Defines global variables that can be used by other programs.

- file ini_html_gen.pl

  Generates one or more HTML files based upon what is found in the ini file.

- file log_html_gen.pl

  Generates one or more HTML files based upon what is found in the messages.txt file that contains all of the log messages.

- file master_update.pl

  Updates the date and version in the supplied master file.

- file pl_comment_chg.pl

  Changes the perl files so that it "looks" like a C file with respect to the comment format and function names.

- file pl_comment_chg2.pl

  Changes the perl files so that it "looks" like a C file with respect to the comment format and function names.

- file pl_comment_chg3.pl

  Changes the perl files so that it "looks" like a C file with respect to the comment format and function names.

- file sapi_check.pl

  Checks the generated SAPI files for completeness.

- file strip_html.pl

  Strip out all HTML except for href's.

- file tree_js_2_script.pl

  Takes a tree js file from Doxygen and outputs a script file used by the TOC java applet.

- file voyant_indexer.pl

  Creates a comprehensive index from temporary index files that were generated from another program.

- file voyant_latex.pl

  Changes two Latex files in each zlx_ directory of the system so that they contain the appropriate information for the project.

- file voyant_mt_app.pl

  Creates table of contents navigation to be displayed in the navigation pane.

- file voyant_nav.pl

  Swaps out tagged areas in all HTML files in a given directory.

- file xhelp_all.pm

  Defines global variables that can be used by other programs.

- file xml_2_html_txt.pl

  Swaps out tagged areas in all HTML files in a given directory.

Modules

- Unix Shell Scripts

  Script files that control building and generating the system.

- Latex Tool

  Provides minor updating to LaTex files.

- Table of Contents Tool

  Generates master tree files from previously generated tree files to get master table of contents.

- Doxygen Filter Tools

    Provides input filter to "fake-out" Doxygen into thinking it has C code.

- Navigation Tools

    Swaps out navigation, creates temporary TOC files and index_ files.

- Indexer Tools

    Creates a comprehensive index from previously generated index_ files.

Functions

- int BEGIN ()

    Code to execute when first entered.

7.1.1 Detailed Description

Tools for technical publications departments.

7.1.2 Function Documentation

7.1.2.1 int BEGIN ()

Code to execute when first entered.

Parameters:
    None.

Returns:
    None.

Limitations and Caveats:
    None

Definition at line 68 of file asn_bapitypes.pl.

7.2 Indexer Tools

Creates a comprehensive index from previously generated index_ files.

Files

- file voyant_indexer.pl

  Creates a comprehensive index from temporary index files that were generated from another program.

Functions

- int ignore_item ()

  Compares the input term to a list of ignore terms.

- int add_to_index_struct ()

  Adds an element to the complicated hash table.

- int add_to_lev2_index_struct ()

  Adds an element to the second level of the complicated hash table.

- int word_chunking ()

  Performs word-chunking on the passed in entries that was extracted from the $globe::master_raw.

- int trash_special_characters ()

  Removes all special characters that we don't want in index as word chunks.

7.2.1 Detailed Description

Creates a comprehensive index from previously generated index_ files.

7.2.2 Function Documentation

7.2.2.1 add_to_index_struct ()

Adds an element to the complicated hash table.

Parameters:

in_entry the compacted entry into the hash

in_title the display text to show for the item

in_url the anchor to add to the URL array. If you send in

globe:

:word_c_boundary for the

in_url, then it won't add the URL to the list.

Return values:

Always returns 1.

$entry = compacted and clean display text for sorting. $subentry = compacted and clean display text for sorting. $idx_struct{$entry}{display} = display text $idx_struct{$entry}{url}[] = array of URL's for the $entry. $idx_struct{$entry}{sub}{$subentry}{display} = display text for the $entry's $subentry. $idx_struct{$entry}{sub}{$subentry}{url}[] = array of URL's for the $entry's $subentry.

Definition at line 736 of file voyant_indexer.pl.

7.2.2.2 add_to_lev2_index_struct ()

Adds an element to the second level of the complicated hash table.

Parameters:

in_entry the compacted entry into the hash.

in_sub the compacted subentry into the hash.

in_title the display text to show for the item.

in_url the anchor to add to the URL array.

Return values:

Always returns 1.

$entry = compacted and clean display text for sorting. $subentry = compacted and clean display text for sorting. $idx_struct{$entry}{display} = display text $idx_struct{$entry}{url}[] = array of URL's for the $entry. $idx_struct{$entry}{sub}{$subentry}{display} = display text for the $entry's $subentry. $idx_struct{$entry}{sub}{$subentry}{url}[] = array of URL's for the $entry's $subentry.

Definition at line 792 of file voyant_indexer.pl.

7.2.2.3 ignore_item ()

Compares the input term to a list of ignore terms.

Parameters:

term_to_test Term to test.

Returns:

Returns 1 if the term has matched an ignore term; Otherwise it returns 0.

It tests for fragments twice, so that ”get” doesn’t match on ”together”.

Limitations and Caveats:

If the term has a perl special character in it, it is sent back immediately and left alone.

Definition at line 648 of file voyant_indexer.pl.

Referenced by unless().

7.2.2.4 trash_special_characters ()

Removes all special characters that we don’t want in index as word chunks.

Parameters:

in_word The word that might have special characters.

Returns:

The word without special characters.

Limitations and Caveats:

Debug statements are left in.

Definition at line 1048 of file voyant_indexer.pl.

7.2.2.5 word_chunking ()

Performs word-chunking on the passed in entries that was extracted from the $globe::master_raw.

Parameters:

unproc_title the unprocessed title passed in $entry_chunk[0].

assoc_t_data the associated title and data given by $entry_chunk[1].

Returns:

Updated entries in the hash $globe::master_index. If a word-chunk already is available as a key into the hash, then this appends its information to the contents of the key using $globe::division_mult_entry.

Word-chunking is performed on the $unproc_title. Natural boundaries (spaces, dashes, underscores, changes in case in the middle of a word) are used to create additional two-level index entries that contain the word-chunk followed by where it came from.

The $globe::ignore_terms_file is used to eliminate unuseful word-chunked entries (such as ”the”, ”a”, ”to”, etc.)

The additional useful entries are appended to the contents of the hash

globe:

:master_raw using the

globe::division_mult_entry separator only if the new entry is not a duplicate.

Word-chunking is particular useful for API documentation so that the reader does not have to remember the exact name of a code item in order to find it. An initial index token of ”api_GetMovie-list” could be found not just under its name in the ”A’s”, but under ”get”, ”movie”, and ”list”.

$entry = compacted and clean display text for sorting. $subentry = compacted and clean display text for sorting. $idx_struct{$entry}{display} = display text $idx_struct{$entry}{url}[] = array of URL’s for the $entry. $idx_struct{$entry}{sub}{$subentry}{display} = display text for the $entry’s $subentry. $idx_struct{$entry}{sub}{$subentry}{url}[] = array of URL’s for the $entry’s $subentry.

Limitations and Caveats:

None.

Definition at line 868 of file voyant_indexer.pl.

7.3 Navigation Tools

Swaps out navigation, creates temporary TOC files and index_ files.

Files

- file html_look_integrate.pl

  Looks into HTML files that have been provided by other sources and parses them for useful information.

- file master_update.pl

  Updates the date and version in the supplied master file.

- file tree_js_2_script.pl

  Takes a tree js file from Doxygen and outputs a script file used by the TOC java applet.

- file voyant_nav.pl

  Swaps out tagged areas in all HTML files in a given directory.

- file xml_2_html_txt.pl

  Swaps out tagged areas in all HTML files in a given directory.

Functions

- int using_voy_nav ()

  What to do when no arguments are given.

- int get_input_file ()

  Grabs the information from the input file and puts into an array of hash elements.

- int get_hyperlinks ()

  Parses through the information in the globe::entire_file buffer for anchors.

- int verify_link ()

  Tests the potential link against various criteria to validate whether the link is of value.

- int spider_trace ()

  Traces the child hyperlinks of a starting file and places them into the data structure.

- int testing_structure ()

  Traces through the created data structure.

- int output_structure_script ()

  The starting point for outputting the TOC script files.

- int purge_full_path ()

  Changes the file name by purging a portion of its path.

- int starting_point_script ()

  Creates the top level TOC script that uses subscript references.

- int script_structure ()

  Traces through the data structure following the children and creates entries into the script file for the TOC.

- int handle_index_tokens ()

  Traces through the data structure and outputs appropriate entries for the index.

- int END ()

  Code to execute when first entered.

- int generate_and_step_through_list ()

  Generates list of files to look at.

- int do_dox_prev_next ()

  Does a cheap previous/next for Doxygen files.

- int change_nav ()

  Opens up an HTML file and replaces information between known tags.

7.3.1 Detailed Description

Swaps out navigation, creates temporary TOC files and index_ files.

7.3.2 Function Documentation

7.3.2.1 int END ()

Code to execute when first entered.

Parameters:
None.

Returns:
None.

Limitations and Caveats:
None

Definition at line 689 of file asn_bapitypes.pl.

7.3.2.2 int change_nav ()

Opens up an HTML file and replaces information between known tags.

Parameters:
in_file is the HTML file to edit.

Returns:
Effectively writes back out to the same HTML file.

This function performs all tasks associated with updating the individual HTML file. It reads the entire HTML file into memory.

In general, this routine looks for paired HTML comments containing a tag name and ”begin” or ”end”. Whatever is between these paired comments is replaced with the updated information extracted from the master. The sequence of replacements is:

- If the HTML file has $globe::m_define{variable}[0], it means that it came from a FrameMaker book. As such, these variables contain information regarding the previous topic, next topic, whether or not it is a first topic or last topic for the chapter, etc. This information is extracted from the HTML file and placed into variables to be used by this topic.
- Info in the file header is swapped out. If the tags aren't found, an appropriate spot for the new tags is created before the terminating head tag.
- Info in the file footer is swapped out. If the tags aren't found, an appropriate spot for the new tags is created before the terminating body tag.
- If the HTML file came from a FM book, replace the navigation bar associated with the book (the second navigation bar.) Although this navigation bar appears below the common navigation bar, it is swapped out first, because if the location tags aren't found, this routine finds an appropriate spot for the new tags immediately following the body tag. This is easy to spot.

- If the HTML file came from Doxygen, replace the navigation bar associated with Doxygen topics (the second navigation bar.) Although this navigation bar appears below the common navigation bar, it is swapped out first, because if the location tags aren't found, this routine finds an appropriate spot for the new tags immediately following the body tag. This is easy to spot.
- Replace the navigation bar common to all topics in the system. If the location tags aren't found, this routine finds an appropriate spot for the new tags immediately following the body tag. This is why this needs to happen after inserting either the FM navigation or the Doxygen navigation.
- Sometimes HTML is generated by Doxygen or Mif2Go that hard-codes fonts or styles that cannot be controlled by a CSS. One of the lists that can be defined in the master file is a ”zap” list. This is read into @globe::voy_html_zap hash. Each item in the zap list has three elements:

  1. start of the offending tag but incomplete.
  2. end of first offending tag.
  3. end of tag pairs.

  The HTML tags defined in the zap list get taken out of the HTML document.
- Items in the header, footer, or navigation have items coming from the master that were not fully defined. Part of the voy_order provides the complete description. The HTML document gets the complete definition of these items. This includes:

  – PDF file name.
  – The name of the manual or group.
  – The document number.
  – Document title is changed to match the first heading.

The title is extracted into the index token list.

The headings are added to the index token list as well as the topic order list.

Object tags are how FM index tokens are passed to MS-HTML using Mif2Go. These are converted into special HTML comments. They are also extracted into the index token list.

If the file came from Doxygen, this parses out interesting items to add to the index token list.

While processing the file, hash tables are created:

- @globe::book_topics: key is html file name, <Title> is stored
- @globe::book_topic_order: key is html file name, next html file is stored
- @globe::book_topic_level: key is html file name, <Headn> is stored

This is important in order to handle subsequent creation of the mini-TOC file (tree.html) in another routine.

Limitations and Caveats:

This is a big routine that accomplishes many different tasks while it has the HTML file in memory. Some of the items have to be handled in the order given for them to be swapped and positioned properly.

Definition at line 720 of file voyant_nav.pl.

7.3.2.3 int do_dox_prev_next ()

Does a cheap previous/next for Doxygen files.

Parameters:

None

Returns:

None

Limitations and Caveats:

None

Definition at line 565 of file voyant_nav.pl.

7.3.2.4 int generate_and_step_through_list ()

Generates list of files to look at.

Parameters:

None

Returns:

None

Limitations and Caveats:

None

Definition at line 508 of file voyant_nav.pl.

7.3.2.5 int get_hyperlinks ()

Parses through the information in the globe::entire_file buffer for anchors.

Parameters:

_file_4_links contains the fully qualified name of the file.

_level contains the level, or distance from, the root for use in assigning to the hyperlinks discovered.

All qualified anchors represent children of the _file_4_links. This routine creates the children's url, display text, and level.

This routine calls the routine verify_link, which has criteria to test against. The criteria is intended to make the qualification of the link fail. In other words, when it returns from verify_link with a failed (0) criteria, this get_hyperlinks routine knows that it should not add a child for that hyperlink.

Limitations and Caveats:
Assumes that $globe::entire_file has an HTML file in it.

Definition at line 699 of file html_look_integrate.pl.

7.3.2.6 int get_input_file ()

Grabs the information from the input file and puts into an array of hash elements.

Parameters:
file_incoming This is the file to open.

Reads in the file into memory for later processing to find anchors. While it's here, it tries to find the title for the file for later display and file naming purposes.

Definition at line 590 of file html_look_integrate.pl.

7.3.2.7 int handle_index_tokens ()

Traces through the data structure and outputs appropriate entries for the index.

Parameters:
begin_file The starting point in the structure.

Note:
This is a RECURSIVE routine. It creates appropriate entries for each of the children into the globe::index_info data structure, which later gets output to a file.

Limitations and Caveats:
Does a convoluted thing to get the path to be truly integrated into the system. Also, it creates a helluva lot of entries for BSP (4022) before any word-chunking might be considered.

Definition at line 1733 of file html_look_integrate.pl.

Referenced by globe_file_cnt().

7.3.2.8 int output_structure_script ()

The starting point for outputting the TOC script files.

Parameters:

_in_file / htree_file the prefix of the name of the script file to output.

trace_start_file starting point in the data structure that has children to trace through.

globe_path_purge the key into the $globe::nav_path_purge hash, which contains paths to purge from the output hyperlinks in the TOC, etc.

Note:

This calls script_structure which is a recursive routine that traces through the data structure.

Definition at line 1454 of file html_look_integrate.pl.

7.3.2.9 int purge_full_path ()

Changes the file name by purging a portion of its path.

Parameters:

_filename original filename.

part2rm portion of the path in the filename to remove.

Returns:

1 and the updated filename if successful; 0 and the unmodified filename if unsuccessful.

Definition at line 1529 of file html_look_integrate.pl.

7.3.2.10 int script_structure ()

Traces through the data structure following the children and creates entries into the script file for the TOC.

Parameters:

begin_file is the point to start in the data structure.

For every begin_file, this outputs entries into the already open OUTSCRIPT file for each of its children.

Whenever a child is encountered, it is tested (using globe_found_files) to see if it has already been processed. This prevents it looping forever.

If a given child has not been traced, this calls itself with that child.

It was deemed desireable to still print out children entries even if we weren't going to trace them (because they already had been traced.)

Note:

This is a RECURSIVE routine. It stops when all children at that level have had appropriate entries made and none have been flagged to be traced.

Limitations and Caveats:

$globe_found_files{$begin_file} is used in a global way. $globe::nav_purge_path is also a global hash. The key into the hash ($globe_path_purge) was defined a level higher and used globally.

Definition at line 1658 of file html_look_integrate.pl.

7.3.2.11 int spider_trace ()

Traces the child hyperlinks of a starting file and places them into the data structure.

Parameters:

start_file The starting HTML file to trace.

This calls routines to open and read the children HTML files and to locate the appropriate hyperlinks within those files.

Limitations and Caveats:

This assumes that the data structure has already been started by having a root file, by having already opened it and acquired its children.

Definition at line 1216 of file html_look_integrate.pl.

7.3.2.12 int starting_point_script ()

Creates the top level TOC script that uses subscript references.

Parameters:

htree_file The prefix of the script file to be output.

which_script The key into the hash for whether we're creating a "script_::define" or "script_master" script. This had the appropriate file names.

which_purge_path The key into the hash for the appropriate purge path depending upon whether or not this is a "define" or "master" version.

This knows about the top level entries and what they should be named. The names have been added to the appropriate hash. It can generate the master script that calls subscripts depending upon which version we're doing.

The difference between a define and master version are the paths for links. Normally, the other tools would handle this for me. However, because this is starting out already as a nested version, it does not integrate well. It is better to have this take care of all entries appropriately so that the other tools only has to integrate the top level script (through a master_nav file).

Definition at line 1567 of file html_look_integrate.pl.

7.3.2.13 int testing_structure ()

Traces through the created data structure.

Parameters:

begin_file is the starting point to begin tracing.

This assumes that the begin_file is present in the data structure and has children to trace through.

Note:

This is a RECURSIVE algorithm and can be used as a template for other recursive things involving the same data structure.

Definition at line 1379 of file html_look_integrate.pl.

7.3.2.14 int using_voy_nav ()

What to do when no arguments are given.

Parameters:

None

Returns:

None

Limitations and Caveats:

None

Definition at line 563 of file html_look_integrate.pl.

Referenced by _arg_inc().

7.3.2.15 int verify_link ()

Tests the potential link against various criteria to validate whether the link is of value.

Parameters:

potential_link Contains a fully qualified hyperlink.

contains the owning file.

Returns:

If a criteria is matched, this routine returns 0 (meaning uninteresting link. If not of the criteria sets off a flag, this then returns the potential link as validated.

Criteria that makes the link uninteresting are things like: linking to itself, up linking to a top-level entity, linking to an html topic that we want to exclude as being a child, linking to other children at the same level, etc.

Definition at line 1014 of file html_look_integrate.pl.

7.4 Doxygen Filter Tools

Provides input filter to ”fake-out” Doxygen into thinking it has C code.

Files

- file dox_bug_filter.pl

  Input filter for Doxygen for most C/C++ code files.

- file dox_chg_not.pl

  Tool for Doxygen for most C/C++ code files.

- file dox_comment_chg.pl

  Used when Doxygenating C/C++ code files.

- file dox_ive_filter.pl

  Input filter for Doxygen for most IVE code files.

- file dox_vxworks_filter.pl

  Input filter for Doxygen for most VxWorks code files.

- file strip_html.pl

  Strip out all HTML except for href's.

7.4.1 Detailed Description

Provides input filter to ”fake-out” Doxygen into thinking it has C code.

7.5 Table of Contents Tool

Generates master tree files from previously generated tree files to get master table of contents.

Files

- file voyant_mt_app.pl

  Creates table of contents navigation to be displayed in the navigation pane.

Functions

- int get_tag_chunk ()

  Gets the information between two tags.

- int replace_tag_chunk ()

  Replaces the information in between the tags with the supplied information.

- int creating_book_toc ()

  Reads in the tree file associated with the master order of a book and creates the table of contents section for it.

- & script_output ()

  Breaks up the book tree files into even smaller chapter tree files.

- int using_voy_mt_app ()

  What to do when no arguments are given.

7.5.1 Detailed Description

Generates master tree files from previously generated tree files to get master table of contents.

7.5.2 Function Documentation

7.5.2.1 int creating_book_toc ()

Reads in the tree file associated with the master order of a book and creates the table of contents section for it.

Parameters:

Varables defines aboved.

IN_HTREE files that were generated from the voyant_nav.pl program. Moreover, these are expected to begin with "tree_" and then contain the path of where they came from.

Return values:

section is a fully formatted and tagged section that can be placed directly into a template.

This is the old way called by create_master_tree_script, which itself is no longer called.

This was originally part of the TIORDER loop above and still uses variables as if still above. It was broken out to provide more clarity particularly when adding collapsing/expanding chapters for a book.

Bug:

This uses not only globe variables, but also define variables defined in create_master_tree_script. This expects the tree_...html files contain the path as part of the name.

Definition at line 572 of file voyant_mt_app.pl.

7.5.2.2 int get_tag_chunk ()

Gets the information between two tags.

Parameters:

input_chunk Text to parse for tags.

start_flag starting tag to look for.

end_flag ending flag to look for.

critical specifies whether or not to record error messages.

Return values:

before the information before the starting tag, 0 if one of tags is missing.

info the information between the tags, 0 if one of tags is missing.

after the information after the ending tag, 0 if one of tags is missing.

Limitations and Caveats:
Does not return the tags.

Definition at line 738 of file globe.pm.

Referenced by _arg_inc().

7.5.2.3 int replace_tag_chunk ()

Replaces the information in between the tags with the supplied information.

Parameters:

input_chunk Text to parse for tags.

start_flag starting tag to look for.

end_flag ending flag to look for.

replacement_text Text to put between tags.

reinsert_tags specifies whether or not tags should be reinserted

critical specifies whether or not to record error messages.

Return values:

success 1 if found tags, 0 if one of tags is missing.

output with replacement text and tags if found tags, the original input text if one of tags is missing.

Limitations and Caveats:
None.

Definition at line 820 of file globe.pm.

7.5.2.4 int script_output ()

Breaks up the book tree files into even smaller chapter tree files.

Parameters:

global_variables

Returns:
Returns a series of HTML files that represent the expanding/collapsing of chapters within a book.

This is the old way called by create_master_tree_script, which itself is no longer called.

It handles the naming and the writing of all tree navigation files. It changes icons where appropriate.

The entire mini-TOC for the book was already available. This accomplishes the task of creating chapter chunks that expand/collapse by stripping all the topics from other chapters except the top level. The top level is left, but changes its icon to be a closed book and changes the hyperlink to point to the appropriate master tree file associated with that chapter.

Limitations and Caveats:
Upon entering, it is known that this is a book file. In fact, the entire mini-TOC for that book is already available.

Definition at line 630 of file voyant_mt_app.pl.

7.5.2.5 int using_voy_mt_app ()

What to do when no arguments are given.

Parameters:
None

Returns:
None

Limitations and Caveats:
None

Definition at line 656 of file voyant_mt_app.pl.

Referenced by _arg_inc().

7.6 Latex Tool

Provides minor updating to LaTex files.

Files

- file voyant_latex.pl

  Changes two Latex files in each zlx_ directory of the system so that they contain the appropriate information for the project.

7.6.1 Detailed Description

Provides minor updating to LaTex files.

7.7 Unix Shell Scripts

Script files that control building and generating the system.

Chapter 8

Voyant TechPubs Tools File Documentation

8.1 asn_bapitypes.pl File Reference

Looks at asn files and stores information about enumerations so that it can be merged with with bapitypes.h.

Functions

- int BEGIN ()

  Code to execute when first entered.

- & read_manipulate_master ()

  Reads in the file generated by development tools and appends any dox comments that those tools may have stripped from the source files.

- & write_output_file ($out_file)
- if (@file_errors)
- int create_file_list ()

  Creates a temporary file that lists all of the files which should be processed.

- int process_file_list ()

  Gathers information from all files given in the list.

- if (0)
- & fix_enum_list ()

  Resolves enumeration items that were built from other components.

- return (0)
- int write_output_file ()

  Uses the second argument of the input (master file) and appends ".gen" to the name and outputs the enhanced data.

- int using_asn_bapi ()

  What to do when no arguments are given.

- int END ()

  Code to execute when first entered.

Variables

- out_file = "$bapitypes_file.gen"

8.1.1 Detailed Description

Looks at asn files and stores information about enumerations so that it can be merged with with bapitypes.h.

A platform-independent definition of enumerations et al are defined in asn files. Comments have been added to these files but the normal asn tools do not pass them through to the h files.

This tool parses the asn files and stores information. Then it parses the generated bapitypes.h file. It merges information stored with information from bapitypes.h and outputs a file with the merged information.

Author:
Glenn C. Maxey

Definition in file asn_bapitypes.pl.

8.1.2 Function Documentation

8.1.2.1 int create_file_list ()

Creates a temporary file that lists all of the files which should be processed.

Returns:
The file is created in the directory given by the first input parameter to the program.

Limitations and Caveats:

Definition at line 192 of file asn_bapitypes.pl.

8.1.2.2 int fix_enum_list ()

Resolves enumeration items that were built from other components.

Duplicates entries in the hash table for the equivalent of the component. Removes place-holder entries in the hash that say component.

Limitations and Caveats:
This is called several times so that it can resolve things as new things are learned.

Definition at line 442 of file asn_bapitypes.pl.

8.1.2.3 if (0)

Definition at line 397 of file asn_bapitypes.pl.

8.1.2.4 if (@ *file_errors*)

Definition at line 165 of file asn_bapitypes.pl.

8.1.2.5 int process_file_list ()

Gathers information from all files given in the list.

Opens the file that contains the list of files that are of interest. Then it opens each of those files one at a time.

Returns:
This creates two hash tables. One is for the code top item and the other is for enumeration values (of a code top item). A ”code top item” is some code element definition. Top means that it is not part of an enumeration.

In both cases, what the hash stores is the doxygen comment associated with it.

The key into ct_item is the name of the item fixed up to what will be seen later in bapitypes.

The key into enum_item is the name of the enumeration plus the enumeration item fixed up to what will be seen later in bapitypes.

Limitations and Caveats:
None.

Definition at line 232 of file asn_bapitypes.pl.

8.1.2.6 int read_manipulate_master ()

Reads in the file generated by development tools and appends any dox comments that those tools may have stripped from the source files.

Returns:
$bapitypes contains the original information along with comments.

Definition at line 494 of file asn_bapitypes.pl.

8.1.2.7 return (0)

8.1.2.8 int using_asn_bapi ()

What to do when no arguments are given.

Parameters:
None

Returns:
None

Limitations and Caveats:
None

Definition at line 665 of file asn_bapitypes.pl.

8.1.2.9 int write_output_file ()

Uses the second argument of the input (master file) and appends ".gen" to the name and outputs the enhanced data.

Returns:
A new file named similar to the input file but with a ".gen" extension.

Definition at line 626 of file asn_bapitypes.pl.

8.1.2.10 & write_output_file ($ *out_file*)

8.1.3 Variable Documentation

8.1.3.1 out_file = "$bapitypes_file.gen"

Definition at line 159 of file asn_bapitypes.pl.

8.2 csh_comment_chg.pl File Reference

Changes the b shell script files so that it ”looks” like a C file with respect to the comment format and function names.

Defines

- #define comment_count 0

8.2.1 Detailed Description

Changes the b shell script files so that it ”looks” like a C file with respect to the comment format and function names.

This is intended as an input filter to Doxygen.

Parameters:

Input file that uses shell (hash-hash-bang) comments.

Returns:

Output is a file that uses C-style comments required by Doxygen.

If a line begins with hash-hash-bang, it is considered part of a Doxygen comment block. Lines that only have hash are turned into // comments and are later ignored by Doxygen.

Author:

Glenn C. Maxey

Definition in file csh_comment_chg.pl.

8.2.2 Define Documentation

8.2.2.1 #define comment_count 0

8.3 dox_bug_filter.pl File Reference

Input filter for Doxygen for most C/C++ code files.

Defines

- #define comment_count 0

8.3.1 Detailed Description

Input filter for Doxygen for most C/C++ code files.

This changes code comments and other items on-the-fly so that it can be more effectively processed by Doxygen. In particular, our coding standard calls for slash-slash-bang comments. However, these comments are not interpretted as blocks even when placed together. Sometimes comments from a file header were being placed on code items. By changing the comment style to be C-style, a true comment block could be created with no misinterpretation.

Other changes that this file does include:

- formatting some auto-generated classes to be correct for extraction.
- replacing @_bug with @_lim – defined in the project file – so that the word ”bug” never appears in our output.

Parameters:

Input source file.

Returns:

Output is file with changes.

Author:

Glenn C. Maxey

Definition in file dox_bug_filter.pl.

8.3.2 Define Documentation

8.3.2.1 #define comment_count 0

8.4 dox_chg_not.pl File Reference

Tool for Doxygen for most C/C++ code files.

Functions

- if (/\\param/)
- if (/\\brief/)
- if (/\\retval/)
- if (/\\return/)
- if (/\\ingroup/)
- if (/\@lim/)
- if (/\\note/)

Variables

- NEW_LINE __pad0__

8.4.1 Detailed Description

Tool for Doxygen for most C/C++ code files.

This changes Doxgyen commands to a style that is more readable. Forward and backward slashes make my eyes dizzy and aren't in the JavaDoc convention.

This file isn't used any more. Back when we started using Doxygen, we did not have a good handle on how to structure comments, how to format commands, etc. This tool fixes a mistaken direction taken in the early efforts.

Parameters:
: ***Input*** source file.

Returns:
: Output is file with changes.

Author:
: Glenn C. Maxey

Definition in file dox_chg_not.pl.

8.4.2 Function Documentation

8.4.2.1 if (/\\note/)

Definition at line 92 of file dox_chg_not.pl.

8.4.2.2 if (/\@lim/)

Definition at line 88 of file dox_chg_not.pl.

8.4.2.3 if (/\\ingroup/)

Definition at line 84 of file dox_chg_not.pl.

8.4.2.4 if (/\\return/)

Definition at line 80 of file dox_chg_not.pl.

8.4.2.5 if (/\\retval/)

Definition at line 76 of file dox_chg_not.pl.

8.4.2.6 if (/\\brief/)

Definition at line 72 of file dox_chg_not.pl.

8.4.2.7 if (/\\param/)

Definition at line 68 of file dox_chg_not.pl.

8.4.3 Variable Documentation

8.4.3.1 NEW_LINE __pad0__

Definition at line 65 of file dox_chg_not.pl.

8.5 dox_comment_chg.pl File Reference

Used when Doxygenating C/C++ code files.

Defines

- #define comment_count 0

8.5.1 Detailed Description

Used when Doxygenating C/C++ code files.

This is a tool to change the Doxygen comment style from C-style comments to comments with slash-slash-bang. The reason is that influential members of the coding convention group decided that they liked more C++ style comments and did not want to have mixed comment styles. Hence, our coding conventions specify this style.

This tool changes existing C style comments to this slash-slash-bang comments.

Note:
Input filters need to be used to change those slash-slash-bang comments into the other style, because the other style works better in Doxygen. If the engineers want to see one style comments in the code that they own, more power to them. The filters will get it back to what Doxygen likes better.

I suspect that this tool will not actually be used much despite the coding conventions. Existing comment styles are grandfathered in. Also, new development tools like TogetherSoft produce more JavaDoc like files and prefer those comments. Hence, another suspicion is that our coding conventions will flop back to the C-style JavaDoc comments making this tool never used.

Parameters:
Input source file.

Returns:
Output is file with changes.

Author:
Glenn C. Maxey

Definition in file dox_comment_chg.pl.

8.5.2 Define Documentation

8.5.2.1 #define comment_count 0

8.6 dox_ive_filter.pl File Reference

Input filter for Doxygen for most IVE code files.

Defines

- #define comment_count 0

8.6.1 Detailed Description

Input filter for Doxygen for most IVE code files.

This "fakes-out" Doxygen into thinking that IVE (Pascal-like) code files are really C files.

Parameters:
 Input source file.

Returns:
 Output is file with changes.

Author:
 Glenn C. Maxey

Definition in file dox_ive_filter.pl.

8.6.2 Define Documentation

8.6.2.1 #define comment_count 0

8.7 dox_vxworks_filter.pl File Reference

Input filter for Doxygen for most VxWorks code files.

Defines

- #define comment_tot_count 0
- #define comment_count 0
- #define period_count 0

8.7.1 Detailed Description

Input filter for Doxygen for most VxWorks code files.

This changes code comments and other items on-the-fly so that it can be more effectively processed by Doxygen.

Parameters:
Input source file.

Returns:
Output is file with changes.

Author:
Glenn C. Maxey

Definition in file dox_vxworks_filter.pl.

8.7.2 Define Documentation

8.7.2.1 #define comment_count 0

8.7.2.2 #define comment_tot_count 0

8.7.2.3 #define period_count 0

8.8 doxygenate.pl File Reference

Functions

- sysopen (INFILE,$infile, 0)||die"ERROR!Can't open\"$infile\"

Variables

- infile = $ARGV[0]

8.8.1 Function Documentation

8.8.1.1 sysopen (INFILE, $ *infile*, 0)

Definition at line 19 of file doxygenate.pl.

8.8.2 Variable Documentation

8.8.2.1 infile = $ARGV[0]

Definition at line 18 of file doxygenate.pl.

8.9 find_extract.pl File Reference

Generates a file with command definitions and doxygen comments, intended as input to Doxgyen without as much noise.

Defines

- #define in_file_list $_[0]
- #define in_file_list $_[0]
- #define src_content ""
- #define in_file_list $_[0]
- #define in_file_list $_[0]
- #define src_content ""
- #define in_text $_[0]
- #define nest_cnt 0
- #define have_it 0
- #define piece_to_search $_[0]
- #define dox_flag_b "\/**"
- #define dox_flag_e "*\/"
- #define out ""
- #define all_doc $_[0]
- #define all_line

Functions

- int BEGIN ()

8.9.1 Detailed Description

Generates a file with command definitions and doxygen comments, intended as input to Doxgyen without as much noise.

xhelp_gen.pl input_scope_package root_path_to_files output_file

Parameters:

input_scope_package A file path and name to a perl package that has the 2-b-included and 2-b-excluded xhelp commands.

root_path_to_files must be terminated with a forward (\/) slash. If output file does not have a forward slash (\/) – an indication of a path –, then the <root path to files> is assumed.

output_file name of the first output file. This plus another file with a leading underscore are generated. The file without the underscore is intended as the input to Doxygen.

The input scope package is required to have:

- The xscope package definition.
- Routine:
  - declare_variable.
  - Routine: memory_clean_up.
- Array:
  - x_names, which contains all prefixes that could be of interest.
  - needed_in, which contains a list of xhelp commands that are absolutely needed-to-be-in.
  - needed_out, which contains a list of commands that we don't want to expose at all if they happen to come through.
  - include_f_type, which are file types that should be viewed.

The definitions of ”clues” in the xscope file is the key to the success of this tool. It needs a hint on what to look for.

Note:

The when standards in Doxygen comments were followed.

The intent is that different projects can define their own files that have the xscope definitions of what to look for.

This program does divide and conquor.

1. Using the prefix list, it greps through the code looking at the file types of interest.
2. Using the command list, it greps through the code looking at the file types of interest.
3. The temporary file with grep results is stripped of interesting entries.
4. A file hash is created that is a hash of hashes.

   ```
   $file_hash {$src_file} {$code_item} {r} {$return_type} = unim-
   portant number
   ```

   ```
   $file_hash {$src_file} {$code_item} {p} {$prototype} = doxy-
   gen
   ```

   It contains:

- {$src_file} the source code files where items of interest were found.
- {$code_item} code items of interest, each associated with a source file.
- {r} {$return_type} a hash of return types for each code item; supports overloading.
- {p} {$prototype} a hash of code definitions for each code item; supports overloading. It contains to the doxygen comment block.

5. The source files from the hash are opened and searched for their respective code item definitions.

6. The source files are searched for comment blocks associated with the code item definitions.

7. The file hash is fleshed out for the prototype definitions and their respective comment blocks.

8. The file hash is stepped through and output to generated files. The generated files are intended for input to doxygen.

   (a) _list_<name>.gen file is for accountability and lists all code items and whether or not they were accompanied by a doxygen template. This is sorted by source file name. Intended to communicate what is in and out.

   (b) <name>.gen contains a fake function definition with the associated doxygen template. This list is sorted by code item. Intended for xhelp.

   (c) class_<name>.gen file contains a fake class definition with the associated doxygen template. This list is sorted by class name. Intended for APIs implemented through classes (e.g., DSP drivers). Generated when $xscope::sapi==0.

   (d) dox_template_<name> contains just the associated doxygen template. The list is sorted based on the @fn code item name. Intented for SAPI in conjunction with the vproto.gen that has the function prototypes. Generated when $xscope::sapi==1; this is set in the sapi.pm file.

Author:
Glenn C. Maxey

Definition in file find_extract.pl.

8.9.2 Define Documentation

8.9.2.1 #define all_doc $_[0]

8.9.2.2 #define all_line

8.9.2.3 #define dox_flag_b "\/**"

8.9.2.4 #define dox_flag_e "*\/"

8.9.2.5 #define have_it 0

8.9.2.6 #define in_file_list $_[0]

8.9.2.7 #define in_file_list $_[0]

8.9.2.8 #define in_file_list $_[0]

8.9.2.9 #define in_file_list $_[0]

8.9.2.10 #define in_text $_[0]

8.9.2.11 #define nest_cnt 0

8.9.2.12 #define out ""

8.9.2.13 #define piece_to_search $_[0]

8.9.2.14 #define src_content ""

8.9.2.15 #define src_content ""

8.9.3 Function Documentation

8.9.3.1 int BEGIN ()

Definition at line 176 of file find_extract.pl.

8.10 globe.pm File Reference

Defines global variables that can be used by other programs.

Defines

- #define src_file $_[0]
- #define separate_comments $_[1]
- #define rcnc_debug 0
- #define src_file $_[0]
- #define tag_name $_[1]
- #define debug_pcff 0
- #define src_file $_[0]
- #define tag_xml $_[1]
- #define debug_xims 0
- #define chunk_to_parse $_[0]
- #define src_file_name $_[1]
- #define tag_name $_[2]
- #define name 0
- #define targ_f_name 0
- #define data 0
- #define before ""
- #define after ""
- #define debug_xtt 0
- #define tag_param "target"
- #define chunk_to_parse $_[0]
- #define tag_param $_[1]
- #define file_name $_[2]
- #define tag_name $_[3]
- #define before ""
- #define after ""
- #define lm_label ""
- #define data ""
- #define debug_pbt 0
- #define f_name $_[0]
- #define tag_xml $_[1]
- #define targ_xml_tag $_[2]
- #define debug_v_n_f 0
- #define temp_return ""

Functions

- int BEGIN ()
- int get_tag_chunk ()

  Gets the information between two tags.

- int replace_tag_chunk ()

  Replaces the information in between the tags with the supplied information.

- int read_code_n_comment ()

  Reads in the entire file but separates code from comments in two different structures.

- int process_comments_from_file ()

  Processes the comment lines from the file that may have tags in them.

- if ($globe::comment_file=~/$globe::x_define{$tag_name}[0]/)
- int xtag_into_mess_structure ()

  Creates a data structure from each line in the message file. This builds a data structure:.

- int xml_tag_target ()

  Gets the data associated with a target in the chunk and passes the chunk back.

8.10.1 Detailed Description

Defines global variables that can be used by other programs.

Most of the variables refer to tags that we expect to find in the HTML files. Others define variables that we want to use in a global way.

Tag names that are used in the master template files, all HTML files of the system, Mif2Go mif2htm.ini files, and Doxgyen header/footer files are defined here for all programs. They are in one place to facilitate their editing and maintenance.

Author:

Glenn C. Maxey

Definition in file globe.pm.

8.10.2 Define Documentation

8.10.2.1 #define after ””

8.10.2.2 #define after ””

8.10.2.3 #define before ””

8.10.2.4 #define before ””

8.10.2.5 #define chunk_to_parse $_[0]

8.10.2.6 #define chunk_to_parse $_[0]

8.10.2.7 #define data ””

8.10.2.8 #define data 0

8.10.2.9 #define debug_pbt 0

8.10.2.10 #define debug_pcff 0

8.10.2.11 #define debug_v_n_f 0

8.10.2.12 #define debug_xims 0

8.10.2.13 #define debug_xtt 0

8.10.2.14 #define f_name $_[0]

8.10.2.15 #define file_name $_[2]

8.10.2.16 #define lm_label ””

8.10.2.17 #define name 0

8.10.2.18 #define rcnc_debug 0

8.10.2.19 #define separate_comments $_[1]

8.10.2.20 #define src_file $_[0]

8.10.2.21 #define src_file $_[0]

8.10.2.22 #define src_file $_[0]

8.10.2.23 #define src_file_name $_[1]

8.10.2.24 #define tag_name $_[3]

8.10.2.25 #define tag_name $_[2]

8.10.2.26 #define tag_name $_[1]

8.10.2.27 #define tag_param $_[1]

8.10.2.28 #define tag_param ”target”

8.10.2.29 #define tag_xml $_[1]

8.10.2.30 #define tag_xml $_[1]

8.10.2.31 #define targ_f_name 0

8.10.2.32 #define targ_xml_tag $_[2]

8.10.2.33 #define temp_return ””

8.10.3 Function Documentation

8.10.3.1 int BEGIN ()

Definition at line 109 of file globe.pm.

8.10.3.2 if ()

Definition at line 990 of file globe.pm.

8.10.3.3 int process_comments_from_file ()

Processes the comment lines from the file that may have tags in them.

Parameters:

src_file is the file associated with the comments.

tag_name defines the tag identifier to look for.

This calls another routine that puts $tag_name name tags into a data structure. When this routine is done, we no longer need globe::comment_file.

Limitations and Caveats:

Does not test the tag_name is valid or has been defined.

Definition at line 972 of file globe.pm.

8.10.3.4 int read_code_n_comment ()

Reads in the entire file but separates code from comments in two different structures.

Parameters:

src_file The file to open and read in.

separate_comments 0 puts it all into $entire_file; 1 puts all of the comments into $comment_file and everything else into $entire_file.

Limitations and Caveats:

Definition at line 896 of file globe.pm.

8.10.3.5 int xml_tag_target ()

Gets the data associated with a target in the chunk and passes the chunk back.

Parameters:

chunk_to_parse A chunk with a target inside of it.

src_file_name The file associated with the target.

Returns:

The chunk with the target expanded into what it is after hunting it down. The $tag_name tag with the target in it is removed whether or not any associated data could be hunted down.

This calls the validate_and_fetch routine, which in turn calls the globe::read_code_n_comment and process_...._file that were called earlier at a higher level. They are reused in hunting down targets and the targets' data.

Limitations and Caveats:

This does no error checking if the the chunk has the the target in it or if the $tag_name (xml identifier) is defined in the globe.pm file under

globe:

:m_define{

tag_name}.

Definition at line 1117 of file globe.pm.

8.10.3.6 int xtag_into_mess_structure ()

Creates a data structure from each line in the message file. This builds a data structure:.

globe:
:tag_mess{file}{

file_name}{$name} = $data;

globe:
:tag_mess{name}{

name}[$i] = $file_name;

$hname = created name for hash made from actual names of parents and children

globe:
:xtag_struct{

hname}{name} = actual item name

globe:
:xtag_struct{

hname}{parent} = owning section

globe:
:xtag_struct{

hname}{child}[] = array of associated children

globe:
:xtag_struct{

hname}{comment} = comment flag

globe:
:xtag_struct{

hname}{desc_html} = description in html

globe:
:xtag_struct{

hname}{desc_man} = description in man

$name = name associated with number $msg_struct{$name}{severity} = severity associated with number $msg_struct{$name}{message} = message associated with number $msg_struct{$name}{comment} = comment associated with number

Limitations and Caveats:

Definition at line 1068 of file globe.pm.

8.11 html_look_integrate.pl File Reference

Looks into HTML files that have been provided by other sources and parses them for useful information.

Defines

- #define globe ::_last_level 1
- #define file_incoming $_[0]
- #define _file_4_links $_[0]
- #define _level $_[1]
- #define before ""
- #define piece 0
- #define after ""
- #define b_href ""
- #define href 0
- #define keep_href ""
- #define a_href ""
- #define d
- #define path $globe::org{$_file_4_links}{path}
- #define potential_link $_[0]
- #define owning_file $_[1]
- #define _href ""
- #define _rel "\.\/"
- #define path $globe::org{$owning_file}{path}
- #define _href $potential_link
- #define _rel "../"
- #define start_file $_[0]
- #define base_level $_[1]
- #define new_level 0
- #define _f_cnt 1
- #define new_file_entry
- #define to_do
- #define begin_file $_[0]
- #define child
- #define loc_title
- #define f
- #define _in_file $_[0]
- #define htree_file $_[0]

- #define trace_start_file $_[1]
- #define globe_path_purge $_[2]
- #define s_path ""
- #define _filename $_[0]
- #define part2rm $_[1]
- #define htree_file $_[0]
- #define which_script $_[1]
- #define which_purge_path $_[2]
- #define begin_file $_[0]
- #define child
- #define strip_p ""
- #define trace 0
- #define f
- #define _lev
- #define begin_file $_[0]
- #define child
- #define f
- #define begin_file $_[0]
- #define child
- #define f

Functions

- int BEGIN ()
- if (0)
- if (1)
- int using_voy_nav ()

  What to do when no arguments are given.

- int get_input_file ()

  Grabs the information from the input file and puts into an array of hash elements.

- int get_hyperlinks ()

  Parses through the information in the globe::entire_file buffer for anchors.

- if ($piece)
- while ($b_anc_end=∼/</)
- if ($b_anc_end!∼/[\S+]/)
- return (1)
- int verify_link ()

  Tests the potential link against various criteria to validate whether the link is of value.

- int spider_trace ()

  Traces the child hyperlinks of a starting file and places them into the data structure.

- foreach new_file_entry (keys%to_do)
- int testing_structure ()

  Traces through the created data structure.

- if ($globe::org{$begin_file}{child}[$f]{level} >$globe::_last_level+1)
- if (!(exists($globe_all_files{$child})))
- int output_structure_script ()

  The starting point for outputting the TOC script files.

- int purge_full_path ()

  Changes the file name by purging a portion of its path.

- int starting_point_script ()

  Creates the top level TOC script that uses subscript references.

- unless (open(OUT_SCRIPT,">$htree_file"))
- foreach _in_file (@xscope::top_files)
- int script_structure ()

  Traces through the data structure following the children and creates entries into the script file for the TOC.

- if (!(exists($globe_found_files{$child})))
- int handle_index_tokens ()

  Traces through the data structure and outputs appropriate entries for the index.

Variables

- globe_file_cnt = 0
- POTENTIAL_L __pad1__
- not_critical
- case_in
- b_anc_end = ∼ s/\ \
- g
- _l_cnt

8.11.1 Detailed Description

Looks into HTML files that have been provided by other sources and parses them for useful information.

It extracts appropriate information needed for a entries in an index and table of contents.

It starts at the files listed in @xscope::top_files array. For every HTML file there, it finds their hyperlinks. It successively traces through the hyperlinks creating a data structure. This is the data structure used in the TOC. The hyperlinks that it traces are also used directly in the table of contents.

Certain files can be excluded from creating children or tracing further. All top files in particular are added to this list so that it does not loop continuously through things it already knows, or more realistically, so that it doesn't build a top-levels data structure under some other top-level data.

This has several routines (spider_trace, index_token_generation, script_generation etc.) that call themselves recursively. The key to stop the recursion is when the owning files for the children (hyperlinks) have already been visited.

Author:
Glenn C. Maxey

Definition in file html_look_integrate.pl.

8.11.2 Define Documentation

8.11.2.1 #define _f_cnt 1

8.11.2.2 #define _file_4_links $_[0]

8.11.2.3 #define _filename $_[0]

8.11.2.4 #define _href $potential_link

8.11.2.5 #define _href ””

8.11.2.6 #define _in_file $_[0]

8.11.2.7 #define _lev

8.11.2.8 #define _level $_[1]

8.11.2.9 #define _rel ”../”

8.11.2.10 #define _rel ”\.\/”

8.11.2.11 #define a_href ””

8.11.2.12 #define after ””

8.11.2.13 #define b_href ””

8.11.2.14 #define base_level $_[1]

8.11.2.15 #define before ””

8.11.2.16 #define begin_file $_[0]

8.11.2.17 #define begin_file $_[0]

8.11.2.18 #define begin_file $_[0]

8.11.2.19 #define begin_file $_[0]

8.11.2.20 #define child

8.11.2.21 #define child

8.11.2.22 #define child

8.11.2.23 #define child

8.11.2.24 #define d

8.11.2.25 #define f

8.11.2.26 #define f

8.11.2.27 #define f

8.11.2.28 #define f

8.11.2.29 #define file_incoming $_[0]

8.11.2.30 #define globe ::_last_level 1

8.11.2.31 #define globe_path_purge $_[2]

8.11.2.32 #define href 0

8.11.2.33 #define htree_file $_[0]

8.11.2.34 #define htree_file $_[0]

8.11.2.35 #define keep_href ""

8.11.2.36 #define loc_title

8.11.2.37 #define new_file_entry

8.11.2.38 #define new_level 0

8.11.2.39 #define owning_file $_[1]

8.11.2.40 #define part2rm $_[1]

8.11.2.41 #define path $globe::org{$owning_file}{path}

8.11.2.42 #define path $globe::org{$_file_4_links}{path}

8.11.2.43 #define piece 0

8.11.2.44 #define potential_link $_[0]

8.11.2.45 #define s_path ""

8.11.2.46 #define start_file $_[0]

8.11.2.47 #define strip_p ""

8.11.2.48 #define to_do

8.11.2.49 #define trace 0

8.11.2.50 #define trace_start_file $_[1]

8.11.2.51 #define which_purge_path $_[2]

8.11.2.52 #define which_script $_[1]

8.11.3 Function Documentation

8.11.3.1 int BEGIN ()

Definition at line 107 of file html_look_integrate.pl.

8.11.3.2 foreach _in_file (@xscope::top_files)

Definition at line 1604 of file html_look_integrate.pl.

8.11.3.3 if (!(exists($globe_found_files{$child})))

Definition at line 1697 of file html_look_integrate.pl.

8.11.3.4 if (!(exists($globe_all_files{$child})))

Definition at line 1423 of file html_look_integrate.pl.

8.11.3.5 if ()

Definition at line 1400 of file html_look_integrate.pl.

8.11.3.6 if ($b_anc_end!∼//[\S+])

Definition at line 932 of file html_look_integrate.pl.

8.11.3.7 if ($ *debug_btxs* = ∼ /master/)

Definition at line 730 of file html_look_integrate.pl.

8.11.3.8 if (1)

Definition at line 410 of file html_look_integrate.pl.

8.11.3.9 if (0)

Definition at line 363 of file html_look_integrate.pl.

8.11.3.10 foreach new_file_entry (keys% *to_do*)

Definition at line 1321 of file html_look_integrate.pl.

8.11.3.11 return (1)

8.11.3.12 unless (open(OUT_SCRIPT,">$htree_file"))

Definition at line 1586 of file html_look_integrate.pl.

8.11.3.13 while ()

Definition at line 787 of file html_look_integrate.pl.

8.11.4 Variable Documentation

8.11.4.1 POTENTIAL_L __pad1__

Definition at line 721 of file html_look_integrate.pl.

8.11.4.2 _l_cnt

Definition at line 977 of file html_look_integrate.pl.

8.11.4.3 b_anc_end = ∼ s/\ \

Definition at line 950 of file html_look_integrate.pl.

8.11.4.4 case_in

Definition at line 728 of file html_look_integrate.pl.

8.11.4.5 g

Definition at line 949 of file html_look_integrate.pl.

8.11.4.6 globe_file_cnt = 0

Definition at line 460 of file html_look_integrate.pl.

8.11.4.7 not_critical

Definition at line 728 of file html_look_integrate.pl.

8.12 html_look_integrate.pm File Reference

Defines global variables that can be used by other programs.

Functions

- int BEGIN ()

8.12.1 Detailed Description

Defines global variables that can be used by other programs.

Most of the variables refer to tags that we expect to find in the HTML files. Others define variables that we want to use in a global way.

Author:
Glenn C. Maxey

Definition in file html_look_integrate.pm.

8.12.2 Function Documentation

8.12.2.1 int BEGIN ()

Definition at line 66 of file html_look_integrate.pm.

8.13 ini_html_gen.pl File Reference

Generates one or more HTML files based upon what is found in the ini file.

Defines

- #define before ""
- #define after ""
- #define piece ""
- #define debug_master 0
- #define debug_cds 0
- #define debug_btxs 0
- #define plus 0
- #define seed $_[0]
- #define level $_[1]
- #define pseed $_[0]
- #define html $_[1]
- #define org_seed $pseed
- #define outpath "$globe::xtag_struct{$pseed}{name}"
- #define before ""
- #define after ""
- #define piece ""
- #define seed $_[0]
- #define level $_[1]
- #define seed $_[0]
- #define seed $_[0]
- #define level $_[1]
- #define html_name $seed
- #define zap_it_string $_[0]
- #define debug_all 0
- #define ttt_error_out "_TBD_: "
- #define prev "_TBD_"
- #define next "_TBD_"
- #define html_name "_TBD_"
- #define file_out
- #define file_out2
- #define file_out_tag
- #define file_out_doc
- #define out_string ""

- #define out_head ””
- #define output_html 0
- #define outtxt ””
- #define outhtm ””
- #define debug_ws 0
- #define seed $_[0]
- #define level $_[1]
- #define html_out $_[2]
- #define debug_rs 0
- #define seed $_[0]
- #define debug_rwm 0
- #define hname $_[0]
- #define trace_parent
- #define file_out
- #define debug_wom 0
- #define temp $globe::xtag_struct{$hname}{desc_man}
- #define debug_gdm 0
- #define b1
- #define a1
- #define p1
- #define nada ”_n_a_d_a_”
- #define start_tag (”<br”, ”<p”, ”<font” , ”<i”, ”<b”, ”<ul”, ”<ol”, ”<li”, ”<”)
- #define end_tag ($nada, ”<\/p”, ”<\/font”, ”</i”, ”</b”, ”</ul”, ”</ol”, ”</li”, $nada)
- #define replace_start (”\.LP\n”, ”\.LP\n”, ”\\fI”, ”\\fI”, ”\\fB”, ””, ””, ”\.HP\n\- ”, ””)
- #define replace_end (””, ””, ”\\fR”, ”\\fR”, ”\\fR”, ””, ””, ”\n\.RE\n”, ””)
- #define close_tag ”>”
- #define seed $_[0]
- #define level $_[1]

Functions

- int BEGIN ()
- int create_data_structure ()

  Creates a data structure from each line in the message file.
- if (0)
- return (1)
- int build_trace_xtag_struct ()

  Traces through the input and builds an xtag_struct for the INI file.
- if ($name=∼/\[/)

Variables

- default

8.13.1 Detailed Description

Generates one or more HTML files based upon what is found in the ini file.

../perl/ini_html_gen.pl

Parameters:

root_dir something like doc_publish/cref_sysxini/

src_ini source file to parse ./z_include/sysx.ini

html_tmpl template for generated files voyant_master_nav.html

org_title optional title for the example file.

a_header optional additional source file with tag information.

Author:

Glenn C. Maxey

Definition in file ini_html_gen.pl.

8.13.2 Define Documentation

8.13.2.1 #define a1

8.13.2.2 #define after ""

8.13.2.3 #define after ""

8.13.2.4 #define b1

8.13.2.5 #define before ””

8.13.2.6 #define before ””

8.13.2.7 #define close_tag ”>”

8.13.2.8 #define debug_all 0

8.13.2.9 #define debug_btxs 0

8.13.2.10 #define debug_cds 0

8.13.2.11 #define debug_gdm 0

8.13.2.12 #define debug_master 0

8.13.2.13 #define debug_rs 0

8.13.2.14 #define debug_rwm 0

8.13.2.15 #define debug_wom 0

8.13.2.16 #define debug_ws 0

8.13.2.17 #define end_tag ($nada, ”<\/p”, ”<\/font”, ”</i”, ”</b”, ”</ul”, ”</ol”, ”</li”, $nada)

8.13.2.18 #define file_out

8.13.2.19 #define file_out

8.13.2.20 #define file_out2

8.13.2.21 #define file_out_doc

8.13.2.22 #define file_out_tag

8.13.2.23 #define hname $_[0]

8.13.2.24 #define html $_[1]

8.13.2.25 #define html_name "_TBD_"

8.13.2.26 #define html_name $seed

8.13.2.27 #define html_out $_[2]

8.13.2.28 #define level $_[1]

8.13.2.29 #define level $_[1]

8.13.2.30 #define level $_[1]

8.13.2.31 #define level $_[1]

8.13.2.32 #define level $_[1]

8.13.2.33 #define nada "_n_a_d_a_"

8.13.2.34 #define next "_TBD_"

8.13.2.35 #define org_seed $pseed

8.13.2.36 #define out_head ””

8.13.2.37 #define out_string ””

8.13.2.38 #define outhtm ””

8.13.2.39 #define outpath ”$globe::xtag_struct{$pseed}{name}”

8.13.2.40 #define output_html 0

8.13.2.41 #define outtxt ””

8.13.2.42 #define p1

8.13.2.43 #define piece ””

8.13.2.44 #define piece ””

8.13.2.45 #define plus 0

8.13.2.46 #define prev ”_TBD_”

8.13.2.47 #define pseed $_[0]

8.13.2.48 #define replace_end (””, ””, ”\\fR”, ”\\fR”, ”\\fR”, ””, ””, ”\n\.RE\n”, ””)

8.13.2.49 #define replace_start (”\.LP\n”, ”\.LP\n”, ”\\fI”, ”\\fI”, ”\\fB”, ””, ””, ”\.HP\n\- ”, ””)

8.13.2.50 #define seed $_[0]

8.13.2.51 #define seed $_[0]

8.13.2.52 #define seed $_[0]

8.13.2.53 #define seed $_[0]

8.13.2.54 #define seed $_[0]

8.13.2.55 #define seed $_[0]

8.13.2.56 #define seed $_[0]

8.13.2.57 #define start_tag (”<br”, ”<p”, ”<font” , ”<i”, ”<b”, ”<ul”, ”<ol”, ”<li”, ”<”)

8.13.2.58 #define temp $globe::xtag_struct{$hname}{desc_man}

8.13.2.59 #define trace_parent

8.13.2.60 #define ttt_error_out ”_TBD_: ”

8.13.2.61 #define zap_it_string $_[0]

8.13.3 Function Documentation

8.13.3.1 int BEGIN ()

Definition at line 96 of file ini_html_gen.pl.

8.13.3.2 int build_trace_xtag_struct ()

Traces through the input and builds an xtag_struct for the INI file.

Parameters:

h_ind The name of the parent section as the hash index.

file_line_index The index to the file array.

Returns:

The hash index for the parent; that is either the parent already passed in, or itself if it is determined to have children based on whether or not it is enclosed in braces. In addition, the xtag_struct is built up for the element.

$name is the key into the hash made up of all previous levels and this level.

globe:

:xtag_struct{

name}{name} = Display name for item.

Definition at line 774 of file ini_html_gen.pl.

8.13.3.3 int create_data_structure ()

Creates a data structure from each line in the message file.

$hname = created name for hash made from actual names of parents and children

globe:

:xtag_struct{

hname}{name} = actual item name

globe:

:xtag_struct{

hname}{parent} = owning section

globe:

:xtag_struct{

hname}{child}[] = array of associated children

globe:

:xtag_struct{

hname}{comment} = comment flag

globe:
:xtag_struct{

hname}{comment_full} = complete comments with no tags

globe:
:xtag_struct{

hname}{desc_html} = description in html

globe:
:xtag_struct{

hname}{desc_man} = description in man

Limitations and Caveats:

Definition at line 727 of file ini_html_gen.pl.

8.13.3.4 if ($ *nested_scripts* = ∼ /\[/)

Definition at line 841 of file ini_html_gen.pl.

8.13.3.5 if (0)

Definition at line 748 of file ini_html_gen.pl.

8.13.3.6 return (1)

8.13.4 Variable Documentation

8.13.4.1 default

Initial value:

```
~ s/^\s*
   $default =~ s/\s*$
   if ($debug_btxs) {
```

```
}
$new_h_ind = "$h_ind\-$name"
```

Definition at line 849 of file ini_html_gen.pl.

8.14 log_html_gen.pl File Reference

Generates one or more HTML files based upon what is found in the messages.txt file that contains all of the log messages.

Defines

- #define before ""
- #define after ""
- #define piece ""
- #define by_name 1

Functions

- int BEGIN ()
- int create_data_structure ()

  Creates a data structure from each line in the message file.

- if (0)
- return (1)
- int write_output_html ()

  Sorts the data structure by name and creates the html files.

- int generate_index ()

  $name = name associated with number
  globe:
  :xtag_struct{
  name}{prev} = previous html file
  globe:
  :xtag_struct{
  name}{next} = next html file
  globe:
  :xtag_struct{
  name}{html} = associated html file
  globe:
  :xtag_struct{
  name}{severity} = severity associated with number
  globe:
  :xtag_struct{
  name}{message} = message associated with number

globe:
:xtag_struct{
name}{comment} = comment associated with number.

- int generate_script ()

  $name = name associated with number
  globe:
  :xtag_struct{
  name}{prev} = previous html file
  globe:
  :xtag_struct{
  name}{next} = next html file
  globe:
  :xtag_struct{
  name}{html} = associated html file
  globe:
  :xtag_struct{
  name}{severity} = severity associated with number
  globe:
  :xtag_struct{
  name}{message} = message associated with number
  globe:
  :xtag_struct{
  name}{comment} = comment associated with number.

- int using_scheck ()

  What to do when no arguments are given.

- int END ()

8.14.1 Detailed Description

Generates one or more HTML files based upon what is found in the messages.txt file that contains all of the log messages.

log_html_gen.pl ../cms_middleware/doc_publish/book_cms_logging/ ../../zz_-stage/bridgegroup/middleware/supptsys/messages.txt ../cms_middleware/voyant_-master_nav.html "Middleware "

Author:
Glenn C. Maxey

Definition in file log_html_gen.pl.

8.14.2 Define Documentation

8.14.2.1 #define after ””

8.14.2.2 #define before ””

8.14.2.3 #define by_name 1

8.14.2.4 #define piece ””

8.14.3 Function Documentation

8.14.3.1 int BEGIN ()

Definition at line 95 of file log_html_gen.pl.

8.14.3.2 int END ()

Definition at line 820 of file log_html_gen.pl.

8.14.3.3 int create_data_structure ()

Creates a data structure from each line in the message file.

$name = name associated with number

globe:
 :xtag_struct{

name}{severity} = severity associated with number

globe:
:xtag_struct{

name}{message} = message associated with number

globe:
:xtag_struct{

name}{comment} = comment associated with number

Limitations and Caveats:

Definition at line 424 of file log_html_gen.pl.

8.14.3.4 int generate_index ()

$name = name associated with number

globe:
:xtag_struct{

name}{prev} = previous html file

globe:
:xtag_struct{

name}{next} = next html file

globe:
:xtag_struct{

name}{html} = associated html file

globe:
:xtag_struct{

name}{severity} = severity associated with number

globe:
:xtag_struct{

name}{message} = message associated with number

globe:
:xtag_struct{

name}{comment} = comment associated with number.

Definition at line 675 of file log_html_gen.pl.

8.14.3.5 int generate_script ()

$name = name associated with number

globe:
:xtag_struct{

name}{prev} = previous html file

globe:
:xtag_struct{

name}{next} = next html file

globe:
:xtag_struct{

name}{html} = associated html file

globe:
:xtag_struct{

name}{severity} = severity associated with number

globe:
:xtag_struct{

name}{message} = message associated with number

globe:
:xtag_struct{

name}{comment} = comment associated with number.

Definition at line 721 of file log_html_gen.pl.

8.14.3.6 if (0)

Definition at line 467 of file log_html_gen.pl.

8.14.3.7 return (1)

8.14.3.8 int using_scheck ()

What to do when no arguments are given.

Parameters:
: ***None***

Returns:
: None log_html_gen.pl ../cms_middleware/doc_publish/book_cms_logging/ ../../zz_-stage/bridgegroup/middleware/supptsys/messages.txt ../cms_middleware/voyant_-master_nav.html ”Middleware ”

Limitations and Caveats:
: None

Definition at line 785 of file log_html_gen.pl.

8.14.3.9 int write_output_html ()

Sorts the data structure by name and creates the html files.

Parameters:
: ***file_out*** The name of the file to output. $name = name associated with number

globe:
: :xtag_struct{

name}{prev} = previous html file

globe:
: :xtag_struct{

name}{next} = next html file

globe:
: :xtag_struct{

name}{html} = associated html file

globe:
: :xtag_struct{

name}{severity} = severity associated with number

globe:
:xtag_struct{

name}{message} = message associated with number

globe:
:xtag_struct{

name}{comment} = comment associated with number

Definition at line 506 of file log_html_gen.pl.

8.15 master_update.pl File Reference

Updates the date and version in the supplied master file.

Functions

- int BEGIN ()

8.15.1 Detailed Description

Updates the date and version in the supplied master file.

Parameters:

globe::master_nav_file [optional path and] filename for the HTML file to use as the master for information. This file has several specially flagged HTML comment sections that are required. Information from the tagged sections is updated based on the input.

Author:

Glenn C. Maxey

Definition in file master_update.pl.

8.15.2 Function Documentation

8.15.2.1 int BEGIN ()

Definition at line 74 of file master_update.pl.

8.16 pl_comment_chg.pl File Reference

Changes the perl files so that it ”looks” like a C file with respect to the comment format and function names.

Defines

- #define comment_count 0

8.16.1 Detailed Description

Changes the perl files so that it ”looks” like a C file with respect to the comment format and function names.

This is intended as an input filter to Doxygen.

Parameters:
Input file that uses Perl (hash-hash-bang) comments.

Returns:
Output is a file that uses C-style comments required by Doxygen.

If a line begins with hash-hash-bang, it is considered part of a Doxygen comment block. Lines that only have hash are turned into // comments and are later ignored by Doxygen.

Author:
Glenn C. Maxey

Definition in file pl_comment_chg.pl.

8.16.2 Define Documentation

8.16.2.1 #define comment_count 0

8.17 pl_comment_chg2.pl File Reference

Changes the perl files so that it ”looks” like a C file with respect to the comment format and function names.

Defines

- #define comment_count 0

8.17.1 Detailed Description

Changes the perl files so that it ”looks” like a C file with respect to the comment format and function names.

This is intended as an input filter to Doxygen.

Parameters:

Input file that uses Perl (hash-hash-bang) comments.

Returns:

Output is a file that uses C-style comments required by Doxygen.

If a line begins with hash-hash-bang, it is considered part of a Doxygen comment block. Lines that only have hash are turned into // comments and are later ignored by Doxygen.

Author:

Glenn C. Maxey

Definition in file pl_comment_chg2.pl.

8.17.2 Define Documentation

8.17.2.1 #define comment_count 0

8.18 pl_comment_chg3.pl File Reference

Changes the perl files so that it ”looks” like a C file with respect to the comment format and function names.

Defines

- #define comment_count 0

8.18.1 Detailed Description

Changes the perl files so that it ”looks” like a C file with respect to the comment format and function names.

This is intended as an input filter to Doxygen.

Parameters:

Input file that uses Perl (hash-hash-bang) comments.

Returns:

Output is a file that uses C-style comments required by Doxygen.

If a line begins with hash-hash-bang, it is considered part of a Doxygen comment block. Lines that only have hash are turned into // comments and are later ignored by Doxygen.

Author:

Glenn C. Maxey

Definition in file pl_comment_chg3.pl.

8.18.2 Define Documentation

8.18.2.1 #define comment_count 0

8.19 sapi_check.pl File Reference

Checks the generated SAPI files for completeness.

Defines

- #define src_file $_[0]
- #define which_test $_[1]
- #define in_file $_[0]
- #define in_file $_[0]

Functions

- int BEGIN ()
- int check_prototype ()
- int check_class_dox ()
- int write_output_list ()

  Creates a file lists all of the potential errors discovered in the generated files.
- int using_scheck ()
- int END ()

8.19.1 Detailed Description

Checks the generated SAPI files for completeness.

This uses the "sapi.pm" file that is generated by the "vproto.pl" tool. It can be named something else, but is the $scope_pm input variable to this program.

The "sapi.pm" file was intended for use with the xhelp tool for reducing noise. It has several data structures defined and then // undefined. They were defined because they could have been useful in locating the proper methods and doxygen comments in the code. (It turns out that the ingroup is more effective at that.)

Those data structures were // undefined (with "if (1)" enclosing an "// undef") so that they wouldn't result in unnecessary grep operations.

Those data structures, however, are useful for testing the integrity of the system in terms of missing commands.

This tool expects you to turn off ("if (0)") these // undefinition within the the sapi.pm file. Run this tool and look at the output.

Author:
Glenn C. Maxey

Definition in file sapi_check.pl.

8.19.2 Define Documentation

8.19.2.1 #define in_file $_[0]

8.19.2.2 #define in_file $_[0]

Referenced by globe_file_cnt().

8.19.2.3 #define src_file $_[0]

8.19.2.4 #define which_test $_[1]

8.19.3 Function Documentation

8.19.3.1 int BEGIN ()

Definition at line 76 of file sapi_check.pl.

8.19.3.2 int END ()

Definition at line 465 of file sapi_check.pl.

8.19.3.3 int check_class_dox ()

Definition at line 348 of file sapi_check.pl.

8.19.3.4 int check_prototype ()

Definition at line 308 of file sapi_check.pl.

8.19.3.5 int using_scheck ()

Definition at line 432 of file sapi_check.pl.

8.19.3.6 int write_output_list ()

Creates a file lists all of the potential errors discovered in the generated files.

Parameters:

file_out The name of the file to output.

Definition at line 396 of file sapi_check.pl.

8.20 strip_html.pl File Reference

Strip out all HTML except for href's.

8.20.1 Detailed Description

Strip out all HTML except for href's.

Parameters:
: ***Input*** source file.

Returns:
: Output is file with changes.

Author:
: Glenn C. Maxey

Definition in file strip_html.pl.

8.21 tree_js_2_script.pl File Reference

Takes a tree js file from Doxygen and outputs a script file used by the TOC java applet.

Functions

- int BEGIN ()
- if (@ARGV<=1)
- if (@ARGV > 1)

8.21.1 Detailed Description

Takes a tree js file from Doxygen and outputs a script file used by the TOC java applet.

Author:
Glenn C. Maxey

Definition in file tree_js_2_script.pl.

8.21.2 Function Documentation

8.21.2.1 int BEGIN ()

Definition at line 87 of file tree_js_2_script.pl.

8.21.2.2 if (@ *ARGV*, 1)

Definition at line 152 of file tree_js_2_script.pl.

8.21.2.3 if ()

Definition at line 148 of file tree_js_2_script.pl.

8.22 voyant_indexer.pl File Reference

Creates a comprehensive index from temporary index files that were generated from another program.

Defines

- #define in_file
- #define ind_file $_[0]
- #define unproc_title $_[0]
- #define assoc_t_data $_[1]
- #define capital 1
- #define w_cnt 0
- #define c_word
- #define term
- #define proc_title &trash_special_characters($unproc_title)
- #define remember_letter "0"
- #define remember_level ""
- #define out_file $globe::path . "m_idx"
- #define very_critical 0

Functions

- int BEGIN ()
- if (@ARGV >$_arg_inc)
- if (0)
- int ignore_item ()

  Compares the input term to a list of ignore terms.

- int add_to_index_struct ()

  Adds an element to the complicated hash table.

- int add_to_lev2_index_struct ()

  Adds an element to the second level of the complicated hash table.

- int word_chunking ()

  Performs word-chunking on the passed in entries that was extracted from the $globe::master_raw.

- int trash_special_characters ()

  Removes all special characters that we don't want in index as word chunks.

- unless (open(OUT_INDEX,">out_fileremember_letter.html"))
- if ($first_letter!~/[a-zA-Z0-9]/)
- if ($first_letter=~/$remember_letter/i)

Variables

- _arg_inc
- _cnt = 0
- PURGE_ENTRY __pad2__
- first_letter = substr($entry, 0, 1)

8.22.1 Detailed Description

Creates a comprehensive index from temporary index files that were generated from another program.

Parameters:

globe::path location to find the index files. The name should be terminated with a slash (\). Although the directory_name is the first command line parameter, the true input are the index files contained within that directory. Index files must begin with "index_".

globe::master_tree_file [optional path and] filename for the HTML file to use as a template for all index files to be generated. Ideally, this should contain navigation tools to get between the generated index files [a-z] and other parts of the system, such as the table of contents. This file has several specially flagged HTML comment sections that are required.

globe::ignore_terms_file [optional path and] filename for a text file that contains words to ignore in the word-chunking process.

Returns:

This creates a series of HTML files that begin with "m_idx_". Generally, what follows in the name is the first character of the first word within the file. All index entries beginning with that character are in that file. These files are created in $globe::path.

1. The $globe::master_tree_file is viewed first to make sure that it has all required information.

2. This program issues a system call to create a list of candidate input index_ files in the $globe::path directory.

3. Then it steps through each of those files and concatentates their input into $globe::master_raw.

   globe:

4. :master_raw is turned into a hash table

   idx_struct. The key into the hash table is the index entry. What gets stored is both the display text and an array of URLs.

   - $entry = compacted, clean, lower-case key from display text for sorting.
   - $subentry = compacted, clean, lower-case key from display text for sorting.
   - $idx_struct{$entry}{display} = display text.
   - $idx_struct{$entry}{url}[] = array of URL's for the $entry.
   - $idx_struct{$entry}{sub}{$subentry}{display} = display text for the $entry's $subentry.
   - $idx_struct{$entry}{sub}{$subentry}{url}[] = array of URL's for the $entry's $subentry.

5. Word-chunking is performed on each element in the $globe::master_index. Natural boundaries (spaces, dashes, underscores, changes in case in the middle of a word) are used to create additional two-level index entries that contain the word-chunk followed by where it came from. The $globe::ignore_terms_file is used to eliminate unuseful word-chunked entries (such as ”the”, ”a”, ”to”, etc.) The additional useful entries are appended to the contents of the hash

   globe:
   :master_raw using the

   globe::division_mult_entry separator only if the new entry is not a duplicate. Word-chunking is particular useful for API documentation so that the reader does not have to remember the exact name of a code item in order to find it. An initial index token of ”api_GetMovie-list” could be found not just under its name in the ”A's”, but under ”get”, ”movie”, and ”list”.

6. The expanded list is sorted.

7. Exact duplicates in terms of index entry and URL are removed from the $globe::master_raw hash.

8. The sorted list is output to a series of m_idx_ files. New m_idx_ files are created whenever a new character starts a word in the list.

9. When an index entry is referenced by multiple URLs, the additional references appear in the output as small document icons next to the first reference in plain text.

More information about the true input ”index_” files. These files are an unsorted running list of index tokens that were extracted from the HTML files in a directory. The tokens have two parts: the index entry and its URL. The separator is :,: (($globe::word_url_boundary). Additionally, the index entry can have two levels. In such cases, the seperator is :;: ($globe::word_c_boundary). Finally, a given index entry can represent multiple references or URLs. In such cases, the multiple entries are separated by :;;;: ($globe::division_mult_entry)

If the separators are changed in the generator program (voyant_nav.pl), they need to be changed here, too. The variable names in both programs are the same. The separators themselves were chosen because they were deemed never to occur in an index entry or URL and aren't Perl special characters.

More information about the master_file. Aside from serving as a template for all generated index files, this file chunks information using specially tagged HTML comments in order to simplify locating where generated information is to be placed. In addition, some tags contain information critical to the proper operation of the indexer.

The critical tags given by Perl variable and HTML syntax are:

- ($globe::m_define{order}[0]) = "< !– begin voy_order –"
- ($globe::m_define{order}[1]) = "!– end voy_order –>"
- ($globe::m_define{structure}[0]) = "< !– begin voy_structure –>"
- ($globe::m_define{structure}[1]) = "< !– end voy_structure –>"
- ($globe::m_define{topic_list}[0]) = "< !– begin voy_topic_list –>"
- ($globe::m_define{topic_list}[1]) = "< !– end voy_topic_list –>"

The HTML syntax can be changed. However, the voy order sections are identical for various programs and their master_files which facilitates propogating information.

Limitations and Caveats:

This does not support index entries that might be or have Perl special characters. These are often eliminated early in the process.

The input index_ files cannot have ".htm" as part of the name. This assumes that input information was of the proper format with an index entry, $globe::word_url_boundary, and URL. If any of the input index_ files did not have this, this can cause problems.

Rather than passing in variables which can create copies in memory, many items use global variables that are defined in globe.pm. When a variable is known to be global, its name begins with "$globe::". The intent is to facilitate maintenance by having all user-defined tags in one place outside of the program.

Many debug statements are left in the code, although commented out or programmed out with if (0){...}. On occassion, a statement is copied, commented out, and then the copy modified in order to keep old techniques around while verifying new techniques. Old techniques were not always purged once the new one worked.

Author:

Glenn C. Maxey

Definition in file voyant_indexer.pl.

8.22.2 Define Documentation

8.22.2.1 #define assoc_t_data $_[1]

8.22.2.2 #define c_word

8.22.2.3 #define capital 1

8.22.2.4 #define in_file

8.22.2.5 #define ind_file $_[0]

8.22.2.6 #define out_file $globe::path . ”m_idx”

8.22.2.7 #define proc_title &trash_special_characters($unproc_title)

8.22.2.8 #define remember_letter ”0”

8.22.2.9 #define remember_level ””

8.22.2.10 #define term

8.22.2.11 #define unproc_title $_[0]

8.22.2.12 #define very_critical 0

8.22.2.13 #define w_cnt 0

8.22.3 Function Documentation

8.22.3.1 int BEGIN ()

Definition at line 209 of file voyant_indexer.pl.

8.22.3.2 if ($ *nested_scripts* = ∼ /$remember_letter/i)

Definition at line 1173 of file voyant_indexer.pl.

8.22.3.3 if ($first_letter!∼//[a-zA-Z0-9])

Definition at line 1167 of file voyant_indexer.pl.

8.22.3.4 if (0)

Definition at line 602 of file voyant_indexer.pl.

8.22.3.5 if (@ *ARGV*, $ *_arg_inc*)

Definition at line 274 of file voyant_indexer.pl.

8.22.3.6 unless (open(OUT_INDEX,">out_fileremember_letter.html"))

Definition at line 1152 of file voyant_indexer.pl.

8.22.4 Variable Documentation

8.22.4.1 PURGE_ENTRY __pad2__

Definition at line 1160 of file voyant_indexer.pl.

8.22.4.2 _arg_inc

Definition at line 302 of file voyant_indexer.pl.

8.22.4.3 _cnt = 0

Definition at line 1157 of file voyant_indexer.pl.

8.22.4.4 first_letter = substr($entry, 0, 1)

Definition at line 1166 of file voyant_indexer.pl.

8.23 voyant_latex.pl File Reference

Changes two Latex files in each zlx_ directory of the system so that they contain the appropriate information for the project.

Defines

- #define refman ""

Functions

- int BEGIN ()
- if (@ARGV >$_arg_inc)
- if (exists($globe::m_info{$def_type}))

Variables

- _arg_inc

8.23.1 Detailed Description

Changes two Latex files in each zlx_ directory of the system so that they contain the appropriate information for the project.

Parameters:

globe::path default location to find the Latex temporary directories. The name should be terminated with a slash (\). This should be the top directory where various master-doxygen.sty and master-header.tex can be found and more importantly, the zlx_ directories where generated LaTex files from Doxygen reside.

globe::master_tree_file [optional path and] filename for the file that stores the information needed for the project and the latex files. It does need additional comment tags containing the voy_latex variables and the latex_header variables.

Returns:

Generates appropriate header.tex and doxygen.sty files for each zlx_ directory in the system. These are then used in Latex's PDF generation for each subproject.

This tool reads in the voyant_master_nav.html file (or equivalent). This file must specify:

- The voy_order for the system that lists all directories, associated names, and associated PDF files.
- The voy_latex which contains variables used in LaTex build such as master file names, company name, etc.
- The latex_head which contains additional information for the refman.tex file.

This tool steps through all cref_ directory names in the voy_order and generates an equivalent zlx_ directory name which should be found under $globe::path. The voy_latex variables declare where additional master.tex and master.sty files reside. These contain call-outs for variables that voy_latex resolves for the given project. All variables in the master files are resolved with information appropriate for the project.

Upon conclusion, appropriate refman.tex and doxygen.sty files are written to each zlx_ directory.

Author:
Glenn C. Maxey

Definition in file voyant_latex.pl.

8.23.2 Define Documentation

8.23.2.1 #define refman ""

8.23.3 Function Documentation

8.23.3.1 int BEGIN ()

Definition at line 117 of file voyant_latex.pl.

8.23.3.2 if (exists($globe::m_info{$def_type}))

Definition at line 638 of file voyant_latex.pl.

8.23.3.3 if (@ *ARGV*, $ *_arg_inc*)

Definition at line 176 of file voyant_latex.pl.

8.23.4 Variable Documentation

8.23.4.1 _arg_inc

Definition at line 195 of file voyant_latex.pl.

8.24 voyant_mt_app.pl File Reference

Creates table of contents navigation to be displayed in the navigation pane.

Defines

- #define key ""
- #define out_file ""
- #define section
- #define holder
- #define master_script ""
- #define temp_file_buf ""
- #define key ""
- #define out_file ""
- #define section
- #define master_script ""
- #define temp_file_buf ""

Functions

- int BEGIN ()
- if (@ARGV >$_arg_inc)
- if ($nested_scripts)
- if (open(OUT_MNAV,">$globe::path$out_file"))
- if (0)
- & script_output ()

  Breaks up the book tree files into even smaller chapter tree files.
- unless (open(OUT_MNAV,">$globe::path$out_file"))
- int creating_book_toc ()

  Reads in the tree file associated with the master order of a book and creates the table of contents section for it.
- int using_voy_mt_app ()

  What to do when no arguments are given.
- int END ()

Variables

- _arg_inc

8.24.1 Detailed Description

Creates table of contents navigation to be displayed in the navigation pane.

Parameters:

globe::path is the directory_name or location to find the tree script files. The name should be terminated with a slash (\). Although the directory_name is the first command line parameter, the true input are the tree script files contained within that directory. Tree script files must begin with "tree_" and end with ".script".

globe::master_filename [optional path and] filename for the project file that contains the directory ordering and official titles.

Returns:

m_tree_ files for each directory of the system. The files are named such that they are associated with their owning directory.

Tree script files are another way of saying "Table of Contents" files. Tree files themselves are mini-TOC files that were generated by voyant_nav.pl or tree_js_2_script.pl.

This program reads in the project file (master_filename). Based on the directory names, it makes some assumptions about the names of the tree script files that it expects to see. It steps through all tree script files, reads them in, creates associated master tree script files, and then creates a master tree script file that nests the other tree script files in order.

If an expected tree script file is missing, an error message is output but a working master tree is still generated that ignores the missing mini-TOC.

Note:

When the original tree script files were created, they had no path information in the hyperlinks. This was to make them more general and more easy to use in other directory structures. The tree script files needed to stand on their own and be independent of any resulting directory structure where they might be shared.

Hence, this program must create master script files that include the expect relative paths to the destination topics. The relative path information is again the directory names from the project file.

Limitations and Caveats:

The tree files (mini-TOC) that are used as input have unique naming requirements.

- The mini-TOC files must begin with ”tree_”
- The mini-TOC files must end with ”.script”.
- The mini-TOC files must contain the name of the directory they are associated with.

Rather than passing in variables which can create copies in memory, many items use global variables that are defined in globe.pm. When a variable is known to be global, its name begins with ”$globe::”. The intent is to facilitate maintenance by having all user-defined tags in one place outside of the program.

Author:

Glenn C. Maxey

Definition in file voyant_mt_app.pl.

8.24.2 Define Documentation

8.24.2.1 #define holder

8.24.2.2 #define key ””

8.24.2.3 #define key ””

8.24.2.4 #define master_script ””

8.24.2.5 master_script ””

Definition at line 520 of file voyant_mt_app.pl.

8.24.2.6 #define out_file ””

8.24.2.7 out_file ""

Definition at line 530 of file voyant_mt_app.pl.

8.24.2.8 #define section

8.24.2.9 #define section

8.24.2.10 #define temp_file_buf ""

8.24.2.11 #define temp_file_buf ""

8.24.3 Function Documentation

8.24.3.1 int BEGIN ()

Definition at line 142 of file voyant_mt_app.pl.

8.24.3.2 int END ()

Definition at line 689 of file voyant_mt_app.pl.

8.24.3.3 if (0)

Definition at line 521 of file voyant_mt_app.pl.

8.24.3.4 if (open(OUT_MNAV,">$globe::path$out_file"))

Definition at line 375 of file voyant_mt_app.pl.

8.24.3.5 if ($ *nested_scripts* = ∼ /master/)

Definition at line 286 of file voyant_mt_app.pl.

8.24.3.6 if (@ *ARGV*, $ *_arg_inc*)

Definition at line 205 of file voyant_mt_app.pl.

8.24.3.7 unless (open(OUT_MNAV,">$globe::path$out_file"))

Definition at line 532 of file voyant_mt_app.pl.

8.24.4 Variable Documentation

8.24.4.1 _arg_inc

Definition at line 224 of file voyant_mt_app.pl.

8.25 voyant_nav.pl File Reference

Swaps out tagged areas in all HTML files in a given directory.

Defines

- #define _file_list $_[0]
- #define t_prev ""
- #define t_next ""
- #define in_file $_[0]
- #define rel_to_file ""
- #define out_file "_temp"
- #define out_line ""
- #define globe ::entire_file ""
- #define globe ::have_header 0
- #define globe ::have_footer 0
- #define globe ::have_common_top 0
- #define globe ::have_dox 0
- #define target_cnt 0
- #define target_prefix "_ggg_"
- #define out_file "_temp"
- #define temp_content ""
- #define temp_html ""
- #define htree_file "tree.html"
- #define script_file "tree.script"
- #define nested_script 1

Functions

- int BEGIN ()
- if (@ARGV >$_arg_inc)
- if (!&generate_and_step_through_list($_file_list))
- if ($globe::path=~/cref/i)
- if (@globe::index_info)
- & organize_topic_order ()
- int using_voy_nav ()
- int generate_and_step_through_list ()

  Generates list of files to look at.

- int do_dox_prev_next ()

  Does a cheap previous/next for Doxygen files.

- int change_nav ()

  Opens up an HTML file and replaces information between known tags.

- if (0)
- unless (open(IN_HTML,$in_file))
- while (< IN_HTML >)
- unless (open(OUT_HTML,">$out_file"))

Variables

- _arg_inc

8.25.1 Detailed Description

Swaps out tagged areas in all HTML files in a given directory.

Parameters:

globe::path location to find the HTML files. The name should be terminated with a slash (\). Although the directory_name is the first command line parameter, the true input are the HTML files contained within that directory.

globe::master_nav_file [optional path and] filename for the HTML file to use as the master for information. This file has several specially flagged HTML comment sections that are required. Information from the tagged sections are lifted and placed into the tagged sections of the input HTML files.

Return values:

input_html_file is modified with new information in the tagged areas. This includes the information in the head, navigation at the top of the body, and copyright information at the bottom of the body.

_file_list is a temporary file with a list of all HTML files in the given directory.

_index_list is a temporary file with extracted index tokens.

tree.html is a temporary file with a mini-TOC (table of contents) for the given book.

This program does most of the work in creating a comprehensive HTML system that spans the mini-HTML systems generated by Doxygen and Mif2Go (from FrameMaker source). The reason it does most of the work is that it has to read the files anyway.

It started out as a tool just to swap out the header, navigation, and copyright areas. However, as long as it was visiting each and every HTML file in the input directory and ”knew” what was in those files and how the files related to one another, this tool was expanded to handle generating a mini-TOC file () and an index file. It:

- Reads in the master file to acquire information from tagged areas.
- Creates a list of HTML files in the input directory.
- Opens each HTML file and swaps out common navigation with that coming from the master file.

Before writing a _temp file (and then copying this back over the original input file) and as it processes the file:

- It creates several hash tables that ultimately determine topic order, topic level in tree, etc.
- It generates a mini-TOC file (tree.html) before finishing.
- It processes object tokens coming from the MS-HTML output of Mif2Go to turn them into index token comments. It also writes these tokens to a temporary file _index_file.
- From the topic hash tables, it generates more index tokens for the _index_file.

Limitations and Caveats:

In order for this to work correctly, the file names for the chapters have to be named correctly so that they sort into a natural order. This is achieved by having the generation place the same prefix on all HTML files for a given chapter. I chose a two-digit number followed by an underscore. In this manner, all files for a given chapter are grouped together by the prefix. Earlier processing of the files determined which of those topics was the first and last for the chapter.

Rather than passing in variables which can create copies in memory, many items use global variables that are defined in globe.pm. When a variable is known to be global, its name begins with ”$globe::”. The intent is to facilitate maintenance by having all user-defined tags in one place outside of the program.

This used information that came specifically from our implementation of styles in FrameMaker. In particular, we know what the format names are that are used for chapter headings. The number is picked off of that.

Author:

Glenn C. Maxey

Definition in file voyant_nav.pl.

8.25.2 Define Documentation

8.25.2.1 #define _file_list $_[0]

8.25.2.2 #define globe ::have_dox 0

8.25.2.3 #define globe ::have_common_top 0

8.25.2.4 #define globe ::have_footer 0

8.25.2.5 #define globe ::have_header 0

8.25.2.6 #define globe ::entire_file ""

8.25.2.7 #define htree_file "tree.html"

8.25.2.8 #define in_file $_[0]

8.25.2.9 #define nested_script 1

8.25.2.10 #define out_file "_temp"

8.25.2.11 out_file "_temp"

Definition at line 759 of file voyant_nav.pl.

8.25.2.12 #define out_line ""

8.25.2.13 #define rel_to_file ""

8.25.2.14 #define script_file "tree.script"

8.25.2.15 #define t_next ""

8.25.2.16 #define t_prev ""

8.25.2.17 #define target_cnt 0

8.25.2.18 #define target_prefix "_ggg_"

8.25.2.19 #define temp_content ""

8.25.2.20 #define temp_html ""

8.25.3 Function Documentation

8.25.3.1 int BEGIN ()

Definition at line 218 of file voyant_nav.pl.

8.25.3.2 if (0)

Definition at line 751 of file voyant_nav.pl.

8.25.3.3 if (@globe::index_info)

Definition at line 445 of file voyant_nav.pl.

8.25.3.4 if ($globe::path = ∼ /cref/i)

Definition at line 433 of file voyant_nav.pl.

8.25.3.5 if (!&generate_and_step_through_list($_file_list))

Definition at line 426 of file voyant_nav.pl.

8.25.3.6 if (@ *ARGV*, $ *_arg_inc*)

Definition at line 285 of file voyant_nav.pl.

8.25.3.7 & organize_topic_order ()

8.25.3.8 unless (open(OUT_HTML,">$out_file"))

Definition at line 773 of file voyant_nav.pl.

Referenced by globe_file_cnt(), and unless().

8.25.3.9 unless (open(IN_HTML,$in_file))

Definition at line 761 of file voyant_nav.pl.

8.25.3.10 int using_voy_nav ()

Definition at line 476 of file voyant_nav.pl.

8.25.3.11 while (< IN_HTML >)

Definition at line 768 of file voyant_nav.pl.

8.25.4 Variable Documentation

8.25.4.1 _arg_inc

Definition at line 311 of file voyant_nav.pl.

8.26 voyant_tp_tools.h File Reference

8.27 xhelp_all.pm File Reference

Defines global variables that can be used by other programs.

Functions

- int BEGIN ()

8.27.1 Detailed Description

Defines global variables that can be used by other programs.

Most of the variables refer to tags that we expect to find in the HTML files. Others define variables that we want to use in a global way.

Author:
Glenn C. Maxey

Definition in file xhelp_all.pm.

8.27.2 Function Documentation

8.27.2.1 int BEGIN ()

Definition at line 68 of file xhelp_all.pm.

8.28 xml_2_html_txt.h File Reference

Variables

- const char ∗ xmlTagDelimiter [4] = {"<", ">", "</", ">"}

  The xml tags look similar to: <diagmenu item="/IMOD" audience="manufacturing"> </diagmenu item="/IMOD" audience="manufacturing"> The internal modifiers (e.g., item, audience) can be in any order with any amount of white space in between. There can also be any amount of white space on either side of equals sign (=). The opening and closing tags (diagmenu) and modifiers (item, audience) _are_ case sensitive. The contents of the (item) modifier _is_ case sensitive. The contents of the (audience) modifier is _not_ case sensitive.// @brief Specifies the starting and ending characters for all xml tags. perl% @xmlTagDelimiter = ("<", ">", "</", ">") $xmlTagDelimiter{start} = ///.

- const char ∗ xmlTags [ ] = {"diagmenu", "xhelp"}

  @brief Specifies the valid xml tags. perl% @xmlTags = ("diagmenu", "xhelp");///.

- const char ∗ diagmenu [ ] = {"item", "audience"}

  @brief Specifies the allowable modifiers for a particular xml tag. perl% @{$xmlTags{diagmenu}{modifier}} = ("item", "audience"); $xmlTags{diagmenu}{modifier}[0] = "item" $xmlTags{diagmenu}{modifier}[1] = "audience"///.

- const char ∗ xhelp [ ] = {"item", "audience"}
- const char ∗ audience [ ] = {"customer", "manufacturing", "engineering"}

  @brief Specifies the allowable values for a particular xml tag modifier. perl% @{$xmlTags{diagmenu}{audience}} = ("customer", "manufacturing", "engineering"); $xmlTags{diagmenu}{audience}[0] = "customer"; $xmlTags{diagmenu}{audience}[1] = "manufacturing"; $xmlTags{diagmenu}{audience}[2] = "engineering";///.

8.28.1 Variable Documentation

8.28.1.1 const char∗ audience[ ] = {”customer”, ”manufacturing”, ”engineering”}

@brief Specifies the allowable values for a particular xml tag modifier. perl% @{$xmlTags{diagmenu}{audience}} = (”customer”, ”manufacturing”, ”engineering”); $xmlTags{diagmenu}{audience}[0] = ”customer”; $xmlTags{diagmenu}{audience}[1] = ”manufacturing”; $xmlTags{diagmenu}{audience}[2] = ”engineering”;//.

Definition at line 88 of file xml_2_html_txt.h.

8.28.1.2 const char∗ diagmenu[ ] = {”item”, ”audience”}

@brief Specifies the allowable modifiers for a particular xml tag. perl% @{$xmlTags{diagmenu}{modifier}} = (”item”, ”audience”); $xmlTags{diagmenu}{modifier}[0] = ”item” $xmlTags{diagmenu}{modifier}[1] = ”audience”//.

Definition at line 77 of file xml_2_html_txt.h.

8.28.1.3 const char∗ xhelp[ ] = {”item”, ”audience”}

Definition at line 78 of file xml_2_html_txt.h.

8.28.1.4 const char∗ xmlTagDelimiter[4] = {”<”, ”>”, ”</”, ”>”}

The xml tags look similar to: <diagmenu item=”/IMOD” audience=”manufacturing”> </diagmenu item=”/IMOD” audience=”manufacturing”> The internal modifiers (e.g., item, audience) can be in any order with any amount of white space in between. There can also be any amount of white space on either side of equals sign (=). The opening and closing tags (diagmenu) and modifiers (item, audience) _are_ case sensitive. The contents of the (item) modifier _is_ case sensitive. The contents of the (audience) modifier is _not_ case sensitive.// @brief Specifies the starting and ending characters for all xml tags. perl% @xmlTagDelimiter = (”<”, ”>”, ”</”, ”>”) $xmlTagDelimiter{start} = //.

Definition at line 62 of file xml_2_html_txt.h.

8.28.1.5 const char∗ xmlTags[ ] = {”diagmenu”, ”xhelp”}

@brief Specifies the valid xml tags. perl% @xmlTags = (”diagmenu”, ”xhelp”);//.

Definition at line 68 of file xml_2_html_txt.h.

8.29 xml_2_html_txt.pl File Reference

Swaps out tagged areas in all HTML files in a given directory.

Functions

- int BEGIN ()
- if (@ARGV<=1)
- if (@ARGV > 1)

8.29.1 Detailed Description

Swaps out tagged areas in all HTML files in a given directory.

Author:
Glenn C. Maxey

Definition in file xml_2_html_txt.pl.

8.29.2 Function Documentation

8.29.2.1 int BEGIN ()

Definition at line 69 of file xml_2_html_txt.pl.

8.29.2.2 if (@ *ARGV*, 1)

Definition at line 134 of file xml_2_html_txt.pl.

8.29.2.3 if ()

Definition at line 130 of file xml_2_html_txt.pl.

Chapter 9

Voyant TechPubs Tools Page Documentation

9.1 Bug List

Member creating_book_toc() This uses not only globe variables, but also define variables defined in create_master_tree_script. This expects the tree_...html files contain the path as part of the name.

Index