# -\*- tcl -\*-

#

# fmt.text -- Engine to convert a doctools document into plain text.

#

# Copyright (c) 2003 Andreas Kupries <andreas\_kupries@sourceforge.net>

#

################################################################

################################################################

# Load shared code and modify it to our needs.

dt\_source \_common.tcl

dt\_source \_text.tcl

proc c\_copyrightsymbol {} {return "(c)"}

rename fmt\_initialize BaseInitialize

proc fmt\_initialize {} {BaseInitialize ; TextInitialize ; return}

################################################################

# Special manpage environments

proc NewExample {} {

global currentEnv

return [NewEnv Example {

set currentEnv(verbatim) 1

append currentEnv(prefix) "| "

set currentEnv(example) .

}] ; # {}

}

proc Example {} {

global currentEnv

if {![info exists currentEnv(exenv)]} {

SaveContext

set verb [NewExample]

RestoreContext

# Remember verbatim mode in the base environment

set currentEnv(exenv) $verb

SaveEnv

}

return $currentEnv(exenv)

}

proc NewList {what} {

# List environments

# Per list several environments are required.

switch -exact -- $what {

enumerated {NewOrderedList}

itemized {NewUnorderedList}

arguments -

commands -

options -

tkoptions -

definitions {NewDefinitionList}

}

}

proc NewUnorderedList {} {

global currentEnv lmarginIncrement

# Itemized list - unordered list - bullet

# 1. Base environment provides indentation.

# 2. First paragraph in a list item.

# 3. All other paragraphs.

set base [NewEnv Itemized {

incr currentEnv(lmargin) $lmarginIncrement

set bullet [Bullet currentEnv(bulleting)]

}] ; # {}

set first [NewEnv First {

set currentEnv(wspfx) [::textutil::repeat::blank $lmarginIncrement]

set currentEnv(listtype) bullet

set currentEnv(bullet) $bullet

}] ; SetContext $base ; # {}

set next [NewEnv Next {

incr currentEnv(lmargin) $lmarginIncrement

}] ; SetContext $base ; # {}

set currentEnv(\_first) $first

set currentEnv(\_next) $next

set currentEnv(pcount) 0

SaveEnv

return

}

proc NewOrderedList {} {

global currentEnv lmarginIncrement

# Ordered list - enumeration - enum

# 1. Base environment provides indentation.

# 2. First paragraph in a list item.

# 3. All other paragraphs.

set base [NewEnv Enumerated {

incr currentEnv(lmargin) $lmarginIncrement

set bullet [EnumBullet currentEnv(enumeration)]

}] ; # {}

set first [NewEnv First {

set currentEnv(wspfx) [::textutil::repeat::blank $lmarginIncrement]

set currentEnv(listtype) enum

set currentEnv(bullet) $bullet

}] ; SetContext $base ; # {}

set next [NewEnv Next {

incr currentEnv(lmargin) $lmarginIncrement

}] ; SetContext $base ; # {}

set currentEnv(\_first) $first

set currentEnv(\_next) $next

set currentEnv(pcount) 0

SaveEnv

return

}

proc NewDefinitionList {} {

global currentEnv lmarginIncrement

# Definition list - terms & definitions

# 1. Base environment provides indentation.

# 2. Term environment

# 3. Definition environment

set base [NewEnv DefL {

incr currentEnv(lmargin) $lmarginIncrement

}] ; # {}

set term [NewEnv Term {

set currentEnv(verbatim) 1

}] ; SetContext $base ; # {}

set def [NewEnv Def {

incr currentEnv(lmargin) $lmarginIncrement

}] ; SetContext $base ; # {}

set currentEnv(\_term) $term

set currentEnv(\_definition) $def

SaveEnv

return

}

################################################################

# Final layouting.

c\_holdBuffers require

proc fmt\_postprocess {text} {text\_postprocess $text}

################################################################

# Implementations of the formatting commands.

c\_pass 1 fmt\_plain\_text {text} NOP

c\_pass 2 fmt\_plain\_text {text} {text\_plain\_text $text}

c\_pass 1 fmt\_manpage\_begin {title section version} NOP

c\_pass 2 fmt\_manpage\_begin {title section version} {

Off

set module [dt\_module]

set shortdesc [c\_get\_module]

set description [c\_get\_title]

set copyright [c\_get\_copyright]

set hdr [list]

lappend hdr "$title - $shortdesc"

lappend hdr [c\_provenance]

lappend hdr "[string trimleft $title :]($section) $version $module \"$shortdesc\""

set hdr [join $hdr \n]

Text $hdr

CloseParagraph [Verbatim]

Section NAME

Text "$title - $description"

CloseParagraph

return

}

c\_pass 1 fmt\_moddesc {desc} {c\_set\_module $desc}

c\_pass 2 fmt\_moddesc {desc} NOP

c\_pass 1 fmt\_titledesc {desc} {c\_set\_title $desc}

c\_pass 2 fmt\_titledesc {desc} NOP

c\_pass 1 fmt\_copyright {desc} {c\_set\_copyright $desc}

c\_pass 2 fmt\_copyright {desc} NOP

c\_pass 1 fmt\_manpage\_end {} NOP

c\_pass 2 fmt\_manpage\_end {} {

set sa [c\_xref\_seealso]

set kw [c\_xref\_keywords]

set ca [c\_xref\_category]

set ct [c\_get\_copyright]

CloseParagraph

if {[llength $sa] > 0} {Section {SEE ALSO} ; Text [join [lsort $sa] ", "] ; CloseParagraph}

if {[llength $kw] > 0} {Section KEYWORDS ; Text [join [lsort $kw] ", "] ; CloseParagraph}

if {$ca ne ""} {Section CATEGORY ; Text $ca ; CloseParagraph}

if {$ct != {}} {Section COPYRIGHT ; Text $ct ; CloseParagraph [Verbatim]}

return

}

c\_pass 1 fmt\_section {name {id {}}} NOP

c\_pass 2 fmt\_section {name {id {}}} {CloseParagraph ; Section $name ; return}

c\_pass 1 fmt\_subsection {name {id {}}} NOP

c\_pass 2 fmt\_subsection {name {id {}}} {CloseParagraph ; Subsection $name ; return}

c\_pass 1 fmt\_para {} NOP

c\_pass 2 fmt\_para {} {CloseParagraph ; return}

c\_pass 2 fmt\_require {pkg {version {}}} NOP

c\_pass 1 fmt\_require {pkg {version {}}} {

set result "package require $pkg"

if {$version != {}} {append result " $version"}

c\_hold require $result

return

}

c\_pass 1 fmt\_usage {cmd args} {c\_hold synopsis "$cmd [join $args " "]"}

c\_pass 2 fmt\_usage {cmd args} NOP

c\_pass 1 fmt\_call {cmd args} {c\_hold synopsis "$cmd [join $args " "]"}

c\_pass 2 fmt\_call {cmd args} {fmt\_lst\_item "$cmd [join $args " "]"}

c\_pass 1 fmt\_description {id} NOP

c\_pass 2 fmt\_description {id} {

On

set syn [c\_held synopsis]

set req [c\_held require]

if {$syn != {} || $req != {}} {

Section SYNOPSIS

if {($req != {}) && ($syn != {})} {

Text $req\n\n$syn

} else {

if {$req != {}} {Text $req}

if {$syn != {}} {Text $syn}

}

CloseParagraph [Verbatim]

}

Section DESCRIPTION

return

}

################################################################

c\_pass 1 fmt\_list\_begin {what {hint {}}} NOP

c\_pass 2 fmt\_list\_begin {what {hint {}}} {

#puts\_stderr "<<fmt\_list\_begin $what>>"

global currentEnv

if {[info exists currentEnv(\_definition)]} {

CloseParagraph $currentEnv(\_definition)

} elseif {[info exists currentEnv(pcount)]} {

if {$currentEnv(pcount) == 0} {CloseParagraph $currentEnv(\_first)}

if {$currentEnv(pcount) > 0} {CloseParagraph $currentEnv(\_next)}

incr currentEnv(pcount)

} else {

CloseParagraph

}

SaveContext

NewList $what

Off

#puts\_stderr "<<fmt\_list\_begin \_\_\_\_\_>>"

return

}

c\_pass 1 fmt\_list\_end {} NOP

c\_pass 2 fmt\_list\_end {} {

#puts\_stderr "<<fmt\_list\_end>>"

global currentEnv

if {[info exists currentEnv(\_definition)]} {

CloseParagraph $currentEnv(\_definition)

} else {

if {$currentEnv(pcount) == 0} {CloseParagraph $currentEnv(\_first)}

if {$currentEnv(pcount) > 0} {CloseParagraph $currentEnv(\_next)}

}

RestoreContext

#puts\_stderr "<<fmt\_list\_end \_\_\_\_>>"

return

}

c\_pass 1 fmt\_lst\_item {text} NOP

c\_pass 2 fmt\_lst\_item {text} {

global currentEnv

#puts\_stderr "<<fmt\_lst\_item \{$text\}>>"

if {[IsOff]} {

On

} else {

CloseParagraph $currentEnv(\_definition)

}

Text $text

CloseParagraph $currentEnv(\_term)

#puts\_stderr "<<fmt\_lst\_item \_\_\_\_\_>>"

return

}

c\_pass 1 fmt\_bullet {} NOP

c\_pass 2 fmt\_bullet {} {

global currentEnv

if {[IsOff]} {On ; return}

if {$currentEnv(pcount) == 0} {CloseParagraph $currentEnv(\_first)}

if {$currentEnv(pcount) > 0} {CloseParagraph $currentEnv(\_next)}

set currentEnv(pcount) 0

return

}

c\_pass 1 fmt\_enum {} NOP

c\_pass 2 fmt\_enum {} {

global currentEnv

if {[IsOff]} {On ; return}

if {$currentEnv(pcount) == 0} {CloseParagraph $currentEnv(\_first)}

if {$currentEnv(pcount) > 0} {CloseParagraph $currentEnv(\_next)}

set currentEnv(pcount) 0

return

}

c\_pass 1 fmt\_cmd\_def {command} NOP

c\_pass 2 fmt\_cmd\_def {command} {fmt\_lst\_item [fmt\_cmd $command]}

c\_pass 1 fmt\_arg\_def {type name {mode {}}} NOP

c\_pass 2 fmt\_arg\_def {type name {mode {}}} {

set text "$type [fmt\_arg $name]"

if {$mode != {}} {append text " ($mode)"}

fmt\_lst\_item $text

return

}

c\_pass 1 fmt\_opt\_def {name {arg {}}} NOP

c\_pass 2 fmt\_opt\_def {name {arg {}}} {

set text [fmt\_option $name]

if {$arg != {}} {append text " $arg"}

fmt\_lst\_item $text

return

}

c\_pass 1 fmt\_tkoption\_def {name dbname dbclass} NOP

c\_pass 2 fmt\_tkoption\_def {name dbname dbclass} {

set text ""

append text "Command-Line Switch:\t[fmt\_option $name]\n"

append text "Database Name:\t[strong $dbname]\n"

append text "Database Class:\t[strong $dbclass]\n"

fmt\_lst\_item $text

}

################################################################

c\_pass 1 fmt\_example\_begin {} NOP

c\_pass 2 fmt\_example\_begin {} {

global currentEnv para

if {[info exists currentEnv(\_definition)]} {

CloseParagraph $currentEnv(\_definition)

} elseif {[info exists currentEnv(pcount)]} {

if {$para != {}} {

if {$currentEnv(pcount) == 0} {CloseParagraph $currentEnv(\_first)}

if {$currentEnv(pcount) > 0} {CloseParagraph $currentEnv(\_next)}

incr currentEnv(pcount)

}

} else {

CloseParagraph

}

return

}

c\_pass 1 fmt\_example\_end {} NOP

c\_pass 2 fmt\_example\_end {} {

global currentEnv para

set penv {}

if {[info exists currentEnv(\_definition)]} {

set penv $currentEnv(\_definition)

} elseif {[info exists currentEnv(pcount)]} {

if {$currentEnv(pcount) == 0} {set penv $currentEnv(\_first)}

if {$currentEnv(pcount) > 0} {set penv $currentEnv(\_next)}

incr currentEnv(pcount)

}

if {$penv != {}} {

# Save current list context, get chosen paragraph context and

# then create an example context form this. After closing the

# paragraph we get back our main list context.

SaveContext

SetContext $penv

CloseParagraph [Example]

RestoreContext

} else {

CloseParagraph [Example]

}

return

}

c\_pass 1 fmt\_example {code} NOP

c\_pass 2 fmt\_example {code} {

fmt\_example\_begin

fmt\_plain\_text $code

fmt\_example\_end

return

}

c\_pass 1 fmt\_nl {} NOP

c\_pass 2 fmt\_nl {} {

global currentEnv

if {[info exists currentEnv(\_definition)]} {

CloseParagraph $currentEnv(\_definition)

} else {

if {$currentEnv(pcount) == 0} {CloseParagraph $currentEnv(\_first)}

if {$currentEnv(pcount) > 0} {CloseParagraph $currentEnv(\_next)}

incr currentEnv(pcount)

}

return

}

################################################################

# Visual markup of words and phrases.

proc fmt\_arg {text} {return $text}

proc fmt\_cmd {text} {return $text}

proc fmt\_emph {text} {em $text }

proc fmt\_opt {text} {return "?$text?" }

proc fmt\_comment {text} {return}

proc fmt\_sectref {text {label {}}} {

if {![string length $label]} {set label $text}

return "-> $text"

}

proc fmt\_syscmd {text} {strong $text}

proc fmt\_method {text} {return $text}

proc fmt\_option {text} {return $text}

proc fmt\_widget {text} {strong $text}

proc fmt\_fun {text} {strong $text}

proc fmt\_type {text} {strong $text}

proc fmt\_package {text} {strong $text}

proc fmt\_class {text} {strong $text}

proc fmt\_var {text} {strong $text}

proc fmt\_file {text} {return "\"$text\""}

proc fmt\_namespace {text} {strong $text}

proc fmt\_uri {text {label {}}} {

if {$label == {}} {

# Without label we use the link directly as part of the text.

return "<URL:$text>"

} else {

return "[em $label] <URL:$text>"

}

}

proc fmt\_term {text} {em $text}

proc fmt\_const {text} {strong $text}

################################################################