# -\*- tcl -\*- # # fmt.html # # Copyright (c) 2001-2008 Andreas Kupries # # Definitions to convert a tcl based manpage definition into # a manpage based upon HTML markup. # ################################################################ ################################################################ dt\_source \_common.tcl ; # Shared code dt\_source \_html.tcl ; # HTML basic formatting proc c\_copyrightsymbol {} {return "[markup "&"]copy;"} proc bgcolor {} {return ""} proc border {} {return 0} proc Year {} {clock format [clock seconds] -format %Y} c\_holdBuffers require synopsis ################################################################ ## Backend for HTML markup # -------------------------------------------------------------- # Handling of lists. Simplified, the global check of nesting and # legality of list commands allows us to throw away most of the # existing checks. global liststack ; # stack of list tags to use in list\_end set liststack {} proc lpush {t class} { global liststack lappend liststack [list [tag/ $t] [litc\_getandclear]] return [taga $t [list class $class]] } proc lpop {} { global liststack set t [lindex $liststack end] set liststack [lreplace $liststack end end] foreach {t l} $t break litc\_set $l return $t } proc fmt\_plain\_text {text} { return $text } ################################################################ # Formatting commands. c\_pass 1 fmt\_manpage\_begin {title section version} {c\_cinit ; c\_clrSections ; return} c\_pass 2 fmt\_manpage\_begin {title section version} { global sec\_is\_open ; set sec\_is\_open 0 global subsec\_is\_open ; set subsec\_is\_open 0 global prev\_litem\_close ; set prev\_litem\_close {} global para\_is\_open ; set para\_is\_open 0 global liststack ; set liststack {} global defaultstyle XrefInit c\_cinit set module [dt\_module] set shortdesc [c\_get\_module] set description [c\_get\_title] set copyright [c\_get\_copyright] set hdr "" append hdr [tag html] [tag head] \n append hdr [tag\_ title "$title - $shortdesc"] \n if {![Extend hdr ByParameter meta]} { # Insert standard CSS definitions. append hdr [tag\_ style \ "[markup <]!--${defaultstyle}--[markup >]" \ type text/css] \n } append hdr [tag/ head] \n append hdr [ht\_comment [c\_provenance]]\n if {$copyright != {}} { append hdr [ht\_comment $copyright]\n } append hdr [ht\_comment "CVS: \$Id\$ $title.$section"] append hdr \n\n append hdr [tag body] [tag\* div class doctools] \n Extend hdr ByParameter header set thetitle "[string trimleft $title :]($section) $version $module \"$shortdesc\"" append hdr [tag\_ h1 $thetitle class title] \n append hdr [fmt\_section Name name] \n append hdr "[para\_open] $title - $description" return $hdr } 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 {} {c\_creset ; return} c\_pass 2 fmt\_manpage\_end {} { c\_creset set res "" set sa [c\_xref\_seealso] set kw [c\_xref\_keywords] set ca [c\_xref\_category] set ct [c\_get\_copyright] if {[llength $sa] > 0} { append res [fmt\_section {See Also} see-also] \n append res [join [XrefList [lsort $sa] sa] ", "] \n } if {[llength $kw] > 0} { append res [fmt\_section Keywords keywords] \n append res [join [XrefList [lsort $kw] kw] ", "] \n } if {$ca ne ""} { append res [fmt\_section Category category] \n append res $ca \n } if {$ct != {}} { append res [fmt\_section Copyright copyright] \n append res [join [split $ct \n] [tag br]\n] [tag br]\n } # Close last paragraph, subsection, and section. append res [para\_close][subsec\_close][sec\_close] Extend res ByParameter footer append res [tag/ div] [tag/ body] [tag/ html] return $res } c\_pass 1 fmt\_section {name id} {c\_newSection $name 1 end $id} c\_pass 2 fmt\_section {name id} { return "[sec\_open $id][tag\_ h2 [anchor $id $name]]\n[para\_open]" } c\_pass 1 fmt\_subsection {name id} {c\_newSection $name 2 end $id} c\_pass 2 fmt\_subsection {name id} { return "[subsec\_open $id][tag\_ h3 [anchor $id $name]]\n[para\_open]" } # Para breaks inside and outside of lists are identical proc fmt\_nl {} {para\_open} proc fmt\_para {} {para\_open} c\_pass 2 fmt\_require {pkg {version {}}} NOP c\_pass 1 fmt\_require {pkg {version {}}} { if {$version != {}} { append pkg " " $version } c\_hold require [tag\_ li "package require [bold $pkg pkgname]"] return } c\_pass 2 fmt\_usage {cmd args} NOP c\_pass 1 fmt\_usage {cmd args} { if {[llength $args]} { set text "$cmd [join $args " "]" } else { set text $cmd } c\_hold synopsis [tag\_ li $text] return } c\_pass 1 fmt\_call {cmd args} { if {[llength $args]} { set text "$cmd [join $args " "]" } else { set text $cmd } c\_hold synopsis [tag\_ li [link $text "#[c\_cnext]"]] return } c\_pass 2 fmt\_call {cmd args} { if {[llength $args]} { set text "$cmd [join $args " "]" } else { set text $cmd } return [fmt\_lst\_item [anchor [c\_cnext] $text]] } c\_pass 1 fmt\_description {did} NOP c\_pass 2 fmt\_description {did} { set result "" set syn [c\_held synopsis] set req [c\_held require] # Create the TOC. # Pass 1: We have a number of special sections which were not # listed explicitly in the document sources. Add them # now. Note the inverse order for the sections added # at the beginning. c\_newSection Description 1 0 $did if {$syn != {} || $req != {}} {c\_newSection Synopsis 1 0 synopsis} c\_newSection {Table Of Contents} 1 0 toc if {[llength [c\_xref\_seealso]] > 0} {c\_newSection {See Also} 1 end see-also} if {[llength [c\_xref\_keywords]] > 0} {c\_newSection Keywords 1 end keywords} if {[c\_xref\_category] ne ""} {c\_newSection Category 1 end category} if {[c\_get\_copyright] != {}} {c\_newSection Copyright 1 end copyright} set sections $::SectionList # Pass 2: Generate the markup for the TOC, indenting the # links according to the level of each section. append result [fmt\_section {Table Of Contents} toc] [para\_close] \n append result [taga ul {class toc}] \n set lastlevel 1 set close 0 foreach {name id level} $sections { # level in {1,2}, 1 = sectio n, 2 = subsection if {$level == $lastlevel} { # Close previous item. if {$close} { append result [tag/ li] \n } } elseif {$level > $lastlevel} { # Start list of subsections append result \n [tag ul] \n } else { # level < lastlevel # End list of subsections, and of previous item (two # actually, the subsection, and the section item). append result [tag/ li] \n [tag/ ul] \n [tag/ li] \n } # Start entry if {$level == 1} { append result [taga li {class section}] [link $name "#$id"] } else { append result [taga li {class subsection}] [link $name "#$id"] } set close 1 set lastlevel $level } if {$lastlevel > 1 } { append result [tag/ ul] \n } if {$close} { append result [tag/ li] \n } append result [tag/ ul] \n # Implicit sections coming after the TOC (Synopsis, then the # description which starts the actual document). The other # implict sections are added at the end of the document and # are generated by 'fmt\_manpage\_end' in the second pass. if {$syn != {} || $req != {}} { append result [fmt\_section Synopsis synopsis] [para\_close] [taga div {class synopsis}] \n if {$req != {}} { append result [tag\_ ul \n$req\n class requirements] \n } if {$syn != {}} { append result [tag\_ ul \n$syn\n class syntax] \n } append result [tag/ div] \n } append result [fmt\_section Description $did] \n return $result } ################################################################ proc fmt\_list\_begin {what {hint {}}} { # NOTE: The hint is ignored. Use stylesheet definitions to modify # item and general list spacing. switch -exact -- $what { enumerated {set tag ol} itemized {set tag ul} arguments - commands - options - tkoptions - definitions {set tag dl} } return [para\_close][lpush $tag $what] } proc fmt\_list\_end {} { set res [para\_close][litc\_getandclear]\n[lpop][para\_open] return $res } proc fmt\_lst\_item {text} { set res [para\_close][litc\_getandclear]\n[tag\_ dt $text]\n[tag dd][para\_open] litc\_set [tag/ dd] return $res } proc fmt\_bullet {} { set res [para\_close][litc\_getandclear]\n[tag li][para\_open] litc\_set [tag/ li] return $res } proc fmt\_enum {} { set res [para\_close][litc\_getandclear]\n[tag li][para\_open] litc\_set [tag/ li] return $res } proc fmt\_cmd\_def {command} { fmt\_lst\_item [fmt\_cmd $command] } proc fmt\_arg\_def {type name {mode {}}} { set text "" append text $type " " [fmt\_arg $name] if {$mode != {}} { append text " (" $mode ")" } fmt\_lst\_item $text } proc fmt\_opt\_def {name {arg {}}} { set text [fmt\_option $name] if {$arg != {}} {append text " " $arg} fmt\_lst\_item $text } proc fmt\_tkoption\_def {name dbname dbclass} { set text "" append text "Command-Line Switch:\t[fmt\_option $name][tag br]\n" append text "Database Name:\t[bold $dbname optdbname][tag br]\n" append text "Database Class:\t[bold $dbclass optdbclass][tag br]\n" fmt\_lst\_item $text } ################################################################ proc fmt\_example\_begin {} { return [para\_close]\n[tag\* pre class example] } proc fmt\_example\_end {} { return [tag/ pre]\n[para\_open] } proc fmt\_example {code} { return "[fmt\_example\_begin][fmt\_plain\_text $code][fmt\_example\_end]" } ################################################################ proc fmt\_arg {text} { italic $text arg } proc fmt\_cmd {text} { bold [XrefMatch $text sa] cmd } proc fmt\_emph {text} { em $text } proc fmt\_opt {text} { span "?$text?" opt } proc fmt\_comment {text} {ht\_comment $text} proc fmt\_sectref {title {id {}}} { global SectionNames if {$id == {}} { set id [c\_sectionId $title] } if {[info exists SectionNames($id)]} { return [span [link $title "#$id"] sectref] } else { return [bold $title sectref] } } proc fmt\_syscmd {text} {bold [XrefMatch $text sa] syscmd} proc fmt\_method {text} {bold $text method} proc fmt\_option {text} {bold $text option} proc fmt\_widget {text} {bold $text widget} proc fmt\_fun {text} {bold $text function} proc fmt\_type {text} {bold $text type} proc fmt\_package {text} {bold [XrefMatch $text sa kw] package} proc fmt\_class {text} {bold $text class} proc fmt\_var {text} {bold $text variable} proc fmt\_file {text} {return "\"[bold $text file]\""} proc fmt\_namespace {text} {bold $text namespace} proc fmt\_uri {text {label {}}} { if {$label == {}} {set label $text} return [link $label $text] } proc fmt\_term {text} {italic [XrefMatch $text kw sa] term} proc fmt\_const {text} {bold $text const} ################################################################ global sec\_is\_open set sec\_is\_open 0 proc sec\_open {id} { global sec\_is\_open set res [para\_close][subsec\_close][sec\_close][tag\* div id $id class section] set sec\_is\_open 1 return $res } proc sec\_close {} { global sec\_is\_open if {!$sec\_is\_open} {return ""} set sec\_is\_open 0 return [tag/ div]\n } ################################################################ global subsec\_is\_open set subsec\_is\_open 0 proc subsec\_open {id} { global subsec\_is\_open set res [para\_close][subsec\_close][tag\* div id $id class subsection] set subsec\_is\_open 1 return $res } proc subsec\_close {} { global subsec\_is\_open if {!$subsec\_is\_open} {return ""} set subsec\_is\_open 0 return [tag/ div]\n } ################################################################ # Piece of html to close the previous list element, if any. # Saved on the list stack global prev\_litem\_close set prev\_litem\_close {} proc litc\_getandclear {} { global prev\_litem\_close set res $prev\_litem\_close set prev\_litem\_close {} return $res } proc litc\_set {value} { global prev\_litem\_close set prev\_litem\_close $value return } ################################################################ global para\_is\_open set para\_is\_open 0 proc para\_open {} { global para\_is\_open set res [para\_close][tag p] set para\_is\_open 1 return $res } proc para\_close {} { global para\_is\_open if {!$para\_is\_open} {return ""} set para\_is\_open 0 return [tag/ p] } ################################################################ global xref ; array set xref {} global \_\_var array set \_\_var { meta {} header {} footer {} xref {} } proc Get {varname} {global \_\_var ; return $\_\_var($varname)} proc fmt\_listvariables {} {global \_\_var ; return [array names \_\_var]} proc fmt\_varset {varname text} { global \_\_var if {![info exists \_\_var($varname)]} {return -code error "Unknown engine variable \"$varname\""} set \_\_var($varname) $text return } # Engine parameter handling proc Extend {v \_ by} { set html [Get $by] if {$html == {}} { return 0 } upvar 1 $v text append text [markup $html] \n return 1 } global defaultstyle set defaultstyle { HTML { background: #FFFFFF; color: black; } BODY { background: #FFFFFF; color: black; } DIV.doctools { margin-left: 10%; margin-right: 10%; } DIV.doctools H1,DIV.doctools H2 { margin-left: -5%; } H1, H2, H3, H4 { margin-top: 1em; font-family: sans-serif; font-size: large; color: #005A9C; background: transparent; text-align: left; } H1.title { text-align: center; } UL,OL { margin-right: 0em; margin-top: 3pt; margin-bottom: 3pt; } UL LI { list-style: disc; } OL LI { list-style: decimal; } DT { padding-top: 1ex; } UL.toc,UL.toc UL, UL.toc UL UL { font: normal 12pt/14pt sans-serif; list-style: none; } LI.section, LI.subsection { list-style: none; margin-left: 0em; text-indent: 0em; padding: 0em; } PRE { display: block; font-family: monospace; white-space: pre; margin: 0%; padding-top: 0.5ex; padding-bottom: 0.5ex; padding-left: 1ex; padding-right: 1ex; width: 100%; } PRE.example { color: black; background: #f5dcb3; border: 1px solid black; } UL.requirements LI, UL.syntax LI { list-style: none; margin-left: 0em; text-indent: 0em; padding: 0em; } DIV.synopsis { color: black; background: #80ffff; border: 1px solid black; font-family: serif; margin-top: 1em; margin-bottom: 1em; } UL.syntax { margin-top: 1em; border-top: 1px solid black; } UL.requirements { margin-bottom: 1em; border-bottom: 1px solid black; } } ################################################################ proc XrefInit {} { global xref \_\_var foreach item $\_\_var(xref) { foreach {pattern fname fragment} $item break set fname\_ref [dt\_fmap $fname] if {$fragment != {}} {append fname\_ref #$fragment} set xref($pattern) $fname\_ref } proc XrefInit {} {} return } proc XrefMatch {word args} { global xref foreach ext $args { if {$ext != {}} { if {[info exists xref($ext,$word)]} { return [XrefLink $xref($ext,$word) $word] } } } if {[info exists xref($word)]} { return [XrefLink $xref($word) $word] } # Convert the word to all-lower case and then try again. set lword [string tolower $word] foreach ext $args { if {$ext != {}} { if {[info exists xref($ext,$lword)]} { return [XrefLink $xref($ext,$lword) $word] } } } if {[info exists xref($lword)]} { return [XrefLink $xref($lword) $word] } return $word } proc XrefList {list {ext {}}} { set res [list] foreach w $list {lappend res [XrefMatch $w $ext]} return $res } proc XrefLink {dest label} { # Ensure that the link is properly done relative to this file! set save $dest #puts\_stderr "XrefLink $dest $label" set here [file split [dt\_fmap [dt\_file]]] set dest [file split $dest] #puts\_stderr "XrefLink < $here" #puts\_stderr "XrefLink > $dest" while {[string equal [lindex $dest 0] [lindex $here 0]]} { set dest [lrange $dest 1 end] set here [lrange $here 1 end] if {[llength $dest] == 0} {break} } set ul [llength $dest] set hl [llength $here] if {$ul == 0} { set dest [lindex [file split $save] end] } else { while {$hl > 1} { set dest [linsert $dest 0 ..] incr hl -1 } set dest [eval file join $dest] } #puts\_stderr "XrefLink --> $dest" if {[string equal $dest [lindex [file split [dt\_fmap [dt\_file]]] end]]} { # Suppress self-referential links, i.e. links made from the # current file to itself. Note that links to specific parts of # the current file are not suppressed, only exact links. return $label } return [link $label $dest] }