xutil

general purpose utility functions for the CDE Desktop KornShell

```
# <NOTICE>
#
# Copyright (C) 1995 Addison-Wesley Publishing Company
# Permission is granted to distribute this program code without
# fee as long as this notice appears in its entirety.
#
# This code is distributed in "as is" condition, and no warranty
# is expressed or implied, including merchantability or use for any
# particular purpose.  Addison-Wesley Publishing Company will not
# held liable for any damages, whether actual or consequential,
# arising from the use of this code.
#
# This code is explained in the book:
#
#     "Desktop KornShell Graphical Programming
#      in the Common Desktop Environment"
#
#      By J. Stephen Pendergrast, Jr.
#      Published by Addison-Wesley Publishing Company
#      ISBN 0-201-63375-X
#
# <END-OF-NOTICE>

# This file contains general purpose utility functions for
# the CDE Desktop KornShell.  It is designed to be sourced into
# a shell script using the dot command, as:
#
#  . $XUTILDIR/xutil.sh
#
# where XUTILDIR is a variable set to the directory where this
# file resides.

# Global context:  The variable _XU is used as a global base for all
# XU library global variables

_XU=                    # initialize parent variable
_XU.VERSION=1.0         # for future compatibility
```

```
# Section #1 Aliases.
#
# Many of the XU convenience functions are simple aliases for other
# X commands.  These aliases are easier to remember because they all have
# the same XU prefix and are easier to type because their names
# don't involve multiple case changes.

alias \
      XUbell=XBell \
      XUcleararea=XClearArea \
      XUclearwindow=XClearWindow \
      XUcopyarea=XCopyArea \
      XUdefinecursor=XDefineCursor \
      XUdrawarc=XDrawArc \
      XUdrawimagestring=XDrawImageString \
      XUdrawline=XDrawLine \
      XUdrawlines=XDrawLines \
      XUdrawpoint=XDrawPoint \
      XUdrawpoints=XDrawPoints \
      XUdrawrectangle=XDrawRectangle \
      XUdrawsegments=XDrawSegments \
      XUdrawstring=XDrawString \
      XUfillarc=XFillArc \
      XUfillpolygon=XFillPolygon \
      XUfillrectangle=XFillRectangle \
      XUflush=XFlush \
      XUheightofscreen=XHeightOfScreen \
      XUraisewindow=XRaiseWindow \
      XUrootwindowofscreen=XRootWindowOfScreen \
      XUsync=XSync \
      XUtextwidth=XTextWidth \
      XUundefinecursor=XUndefineCursor \
      XUwidthofscreen=XWidthOfScreen \
      XUaddwmprotocolcallback=XmAddWMProtocolCallback \
      XUaddwmprotocols=XmAddWMProtocols \
      XUcommandappendvalue=XmCommandAppendValue \
      XUcommanderror=XmCommandError \
      XUcommandgetchild=XmCommandGetChild \
      XUcommandsetvalue=XmCommandSetValue \
      XUfileselectionboxgetchild=XmFileSelectionBoxGetChild \
      XUfileselectiondosearch=XmFileSelectionDoSearch \
      XUgetatomname=XmGetAtomName \
      XUgetcolors=XmGetColors \
      XUgetfocuswidget=XmGetFocusWidget \
      XUgetpostedfromwidget=XmGetPostedFromWidget \
      XUgettabgroup=XmGetTabGroup \
      XUgettearoffcontrol=XmGetTearOffControl \
      XUgetvisibility=XmGetVisibility \
      XUinternatom=XmInternAtom \
      XUistraversable=XmIsTraversable \
      XUlistadditem=XmListAddItem \
      XUlistadditemunselected=XmListAddItemUnselected \
      XUlistadditems=XmListAddItems \
      XUlistadd=XmListAddItems \
      XUlistadditemsunselected=XmListAddItemsUnselected \
      XUlistdeleteallitems=XmListDeleteAllItems \
      XUlistdeleteall=XmListDeleteAllItems \
      XUlistdeleteitem=XmListDeleteItem \
      XUlistdeleteitems=XmListDeleteItems \
      XUlistdeleteitemspos=XmListDeleteItemsPos \
      XUlistdeletepos=XmListDeletePos \
      XUlistdeletepositions=XmListDeletePositions \
      XUlistdeselectallitems=XmListDeselectAllItems \
      XUlistdeselectall=XmListDeselectAllItems \
      XUlistdeselectitem=XmListDeselectItem \
      XUlistdeselectpos=XmListDeselectPos \
      XUlistgetkbditempos=XmListGetKbdItemPos \
      XUlistgetmatchpos=XmListGetMatchPos \
      XUlistgetselectedpos=XmListGetSelectedPos \
      XUlistitemexists=XmListItemExists \
      XUlistitempos=XmListItemPos \
      XUlistposselected=XmListPosSelected \
      XUlistpostobounds=XmListPosToBounds \
      XUlistreplaceitemspos=XmListReplaceItemsPos \
      XUlistreplace=XmListReplaceItemsPos \
      XUlistreplaceitemsposunselected=XmListReplaceItemsPosUnselected \
      XUlistselectitem=XmListSelectItem \
      XUlistselectpos=XmListSelectPos \
      XUlistsetaddmode=XmListSetAddMode \
      XUlistsetbottomitem=XmListSetBottomItem \
      XUlistsetbottompos=XmListSetBottomPos \
      XUlistsethorizpos=XmListSetHorizPos \
      XUlistsetitem=XmListSetItem \
      XUlistsetkbditempos=XmListSetKbdItemPos \
      XUlistsetpos=XmListSetPos \
      XUlistupdateselectedlist=XmListUpdateSelectedList \
      XUmainwindowsep1=XmMainWindowSep1 \
      XUmainwindowsep2=XmMainWindowSep2 \
      XUmainwindowsep3=XmMainWindowSep3 \
      XUmainwindowsetareas=XmMainWindowSetAreas \
      XUmenuposition=XmMenuPosition \
      XUmessageboxgetchild=XmMessageBoxGetChild \
      XUoptionbuttongadget=XmOptionButtonGadget \
      XUoptionlabelgadget=XmOptionLabelGadget \
      XUprocesstraversal=XmProcessTraversal \
      XUremovewmprotocolcallback=XmRemoveWMProtocolCallback \
      XUremovewmprotocols=XmRemoveWMProtocols \
      XUscalegetvalue=XmScaleGetValue \
      XUscalesetvalue=XmScaleSetValue \
      XUscrollbargetvalues=XmScrollBarGetValues \
      XUscrollbarsetvalues=XmScrollBarSetValues \
      XUscrollvisible=XmScrollVisible \
      XUselectionboxgetchild=XmSelectionBoxGetChild \
      XUtextclearselection=XmTextClearSelection \
      XUtextdisableredisplay=XmTextDisableRedisplay \
      XUtextdisable=XmTextDisableRedisplay \
      XUtextenableredisplay=XmTextEnableRedisplay \
      XUtextenable=XmTextEnableRedisplay \
      XUtextfieldclearselection=XmTextFieldClearSelection \
      XUtextfieldcopy=XmTextFieldCopy \
      XUtextfieldcut=XmTextFieldCut \
      XUtextfieldgetbaseline=XmTextFieldGetBaseline \
      XUtextfieldgeteditable=XmTextFieldGetEditable \
      XUtextfieldgetinsertionposition=XmTextFieldGetInsertionPosition \
      XUtextfieldgetlastposition=XmTextFieldGetLastPosition \
      XUtextfieldgetmaxlength=XmTextFieldGetMaxLength \
      XUtextfieldgetselection=XmTextFieldGetSelection \
      XUtextfieldgetselectionposition=XmTextFieldGetSelectionPosition \
      XUtextfieldgetstring=XmTextFieldGetString \
      XUtextfieldinsert=XmTextFieldInsert \
      XUtextfieldpaste=XmTextFieldPaste \
      XUtextfieldpostoxy=XmTextFieldPosToXY \
      XUtextfieldremove=XmTextFieldRemove \
      XUtextfieldreplace=XmTextFieldReplace \
      XUtextfieldsetaddmode=XmTextFieldSetAddMode \
      XUtextfieldseteditable=XmTextFieldSetEditable \
      XUtextfieldsethighlight=XmTextFieldSetHighlight \
      XUtextfieldsetinsertionposition=XmTextFieldSetInsertionPosition \
      XUtextfieldsetmaxlength=XmTextFieldSetMaxLength \
      XUtextfieldsetselection=XmTextFieldSetSelection \
      XUtextfieldsetstring=XmTextFieldSetString \
      XUtextfieldshowposition=XmTextFieldShowPosition \
      XUtextfieldxytopos=XmTextFieldXYToPos \
      XUtextfindstring=XmTextFindString \
      XUtextgetbaseline=XmTextGetBaseline \
      XUtextgeteditable=XmTextGetEditable \
      XUtextgetinsertionposition=XmTextGetInsertionPosition \
      XUtextgetlastposition=XmTextGetLastPosition \
      XUtextgetlast=XmTextGetLastPosition \
      XUtextgetmaxlength=XmTextGetMaxLength \
      XUtextgetselection=XmTextGetSelection \
      XUtextgetselectionposition=XmTextGetSelectionPosition \
      XUtextgetstring=XmTextGetString \
      XUtextgettopcharacter=XmTextGetTopCharacter \
      XUtextinsert=XmTextInsert \
      XUtextpostoxy=XmTextPosToXY \
      XUtextremove=XmTextRemove \
      XUtextreplace=XmTextReplace \
      XUtextscroll=XmTextScroll \
      XUtextsetaddmode=XmTextSetAddMode \
      XUtextseteditable=XmTextSetEditable \
      XUtextsetinsertionposition=XmTextSetInsertionPosition \
      XUtextsetmaxlength=XmTextSetMaxLength \
      XUtextsetselection=XmTextSetSelection \
      XUtextsetstring=XmTextSetString \
      XUtextsettopcharacter=XmTextSetTopCharacter \
      XUtextshowposition=XmTextShowPosition \
      XUtextshow=XmTextShowPosition \
      XUtextxytopos=XmTextXYToPos \
      XUtogglebuttongadgetgetstate=XmToggleButtonGadgetGetState \
      XUtogglebuttongadgetsetstate=XmToggleButtonGadgetSetState \
      XUtogglebuttongetstate=XmToggleButtonGetState \
      XUupdatedisplay=XmUpdateDisplay \
      XUaddcallback=XtAddCallback \
      XUaddeventhandler=XtAddEventHandler \
      XUaddinput=XtAddInput \
      XUaddtimeout=XtAddTimeOut \
      XUaddworkproc=XtAddWorkProc \
      XUaugmenttranslations=XtAugmentTranslations \
      XUcallcallbacks=XtCallCallbacks \
      XUclass=XtClass \
      XUcreateapplicationshell=XtCreateApplicationShell \
      XUcreatemanagedwidget=XtCreateManagedWidget \
      XUcreatepopupshell=XtCreatePopupShell \
      XUcreatewidget=XtCreateWidget \
      XUdestroywidget=XtDestroyWidget \
      XUdisplay=XtDisplay \
      XUdisplayofobject=XtDisplayOfObject \
      XUgetvalues=XtGetValues \
      XUhascallbacks=XtHasCallbacks \
      XUismanaged=XtIsManaged \
      XUisrealized=XtIsRealized \
      XUissensitive=XtIsSensitive \
      XUisshell=XtIsShell \
      XUissubclass=XtIsSubclass \
      XUlasttimestampprocessed=XtLastTimestampProcessed \
      XUmainloop=XtMainLoop \
      XUmanagechildren=XtManageChildren \
      XUmapwidget=XtMapWidget \
      XUnametowidget=XtNameToWidget \
      XUoverridetranslations=XtOverrideTranslations \
      XUparent=XtParent \
      XUpopdown=XtPopdown \
      XUrealizewidget=XtRealizeWidget \
      XUremoveallcallbacks=XtRemoveAllCallbacks \
      XUremovecallback=XtRemoveCallback \
      XUremoveeventhandler=XtRemoveEventHandler \
      XUremoveinput=XtRemoveInput \
      XUremovetimeout=XtRemoveTimeOut \
      XUremoveworkproc=XtRemoveWorkProc \
      XUscreen=XtScreen \
      XUsetsensitive=XtSetSensitive \
      XUsetvalues=XtSetValues \
      XUuninstalltranslations=XtUninstallTranslations \
      XUunmanagechildren=XtUnmanageChildren \
      XUunmapwidget=XtUnmapWidget \
      XUunrealizewidget=XtUnrealizeWidget \
      XUwindow=XtWindow

# Some of the aliases are shorter versions of the corresponding
# X command; if the last "word" of the X command is redundant or adds no
# clarification of the command's usage, it is left out.  For example:
#
#          XtManageWidget    becomes     XUmanage
#
# because the only thing you *can* manage is a widget, the last word is
# not necessary.  Similarly, XtSetValues becomes XUset, XtGetValues becomes
# XUget, etc.

alias \
      XUset=XtSetValues \
      XUget=XtGetValues \
      XUmanage=XtManageChildren \
      XUunmanage=XtUnmanageChildren \
      XUmap=XtMapWidget \
      XUunmap=XtUnmapWidget \
      XUrealize=XtRealizeWidget \
      XUunrealize=XtUnrealizeWidget \
      XUdestroy=XtDestroyWidget \
      XUlistdeleteall=XmListDeleteAllItems
```

```
# Section 2:  Minor Conveniences
#
# These functions mimic some X commands, but add some minor value
# such as defaulting certain parameters that are not provided.

# XUpopup - exactly like XtPopup but defaults the mode to
# GrabNone if not given and if the widget is not a shell-
# class widget looks up the hierarchy to find a shell.
#
# Usage: XUpopup $widget [grab-type]
#
# If grab-type is not specified, it defaults to GrabNone

function XUpopup {
      typeset widget=$1
      typeset mode=$2

      while ! XtIsShell $widget && [[ $widget != "(null)" ]]
      do
            XtParent widget $widget
      done
      XtPopup $widget ${mode:-GrabNone}
}

# XUinitialize - like XtInitialize but assumes app-name is
# same as app-class lowercased.
#
# Usage: XUinitialize variable AppClass [other-args]

function XUinitialize {
      typeset -l lower="$2"
      XtInitialize "$1" "$lower" "$2" "$2" "${@:3}"
      eval "_XU.TOPLEVEL=\${$1}"
      if (( ${#ARGV[@]} ))
      then
            # If ARGV has stuff in it, then this user is running
            # the BETA version of CDE.  In the final product,
            # the name of ARGV was changed
            # to DTKSH_ARGV, and the name of TOPLEVEL was changed to
            # DTKSH_TOPLEVEL, and APPNAME was changed to DTKSH_APPNAME.
            # For convenience, we'll set things up
            # the way they are in the final CDE 1.0 product.

            set -A DTKSH_ARGV "${ARGV[@]}"
            DTKSH_TOPLEVEL="$TOPLEVEL"
            DTKSH_APPNAME="$APPNAME"
      fi
}

# XUsensitive - convenient way to make multiple widgets
# sensitive
#
# Usage: XUsensitive $widget ...

function XUsensitive {
      typeset w
      for w in "$@"
      do
            XtSetSensitive $w true
      done
}
```

```
# XUinsensitive - convenient way to make multiple widgets
# insensitive
#
# Usage: XUinsensitive $widget ...

function XUinsensitive {
      typeset w

      for w in "$@"
      do
            XtSetSensitive $w false
      done
}

# XUcolumn, XUrow:  make a RowColumn widget for stacking vertically
# or horizontally.
#
# Usage: XUcolumn variable $parent [resource:value ...]
# Usage: XUrow variable $parent [resource:value ...]

alias XUcolumn=XUrowcolumn

function XUrow {
      _XUcreateobject XmCreateRowColumn "$@" orientation:horizontal
}

# XUtogglebuttonsetstate:  like XmToggleButtonSetState but defaults state to
# true.
#
# Usage: XUtogglebuttonsetstate $widget [true|false]

function XUtogglebuttonsetstate {
      typeset w=$1 state=$2 notify=$3
      XmToggleButtonSetState $w $state ${notify:-true}
}

# XUtoggleselect: programmatically simulate a toggle button press
#
# Usage: XUtoggleselect $widget

function XUtoggleselect {
      XmToggleButtonSetState $1 true true
}
```

```
# Section #3: Value Added Commands
#
# These commands have no equivalent in the standard X functions
# and add value by making commonly needed procedures available as
# simple functions.

# XUtextappend:  append text to the end of a Text or TextField widget
#
# Usage: XUtextappend $widget text-string

function XUtextappend {
      typeset w=$1 text=$2
      integer pos
      typeset c=$(XtClass - $w)

      ${c}GetLastPosition pos $w
      ${c}Insert $w $pos "$text"
}

# XUtextfind:  search for a string in a Text or TextField widget
#
# Usage: XUtextfind [-b] variable $widget start-position pattern
#
#       -b option searches backward instead of the default of forward

function XUtextfind {
      if [[ $1 == -b ]]
      then
            _direction=TEXT_BACKWARD
            shift
      fi
      XmTextFindString "$2" "$3" "$4" "${_direction:-TEXT_FORWARD}" "$1"
}

# XUbusy - change the cursor to a busy cursor for $widget.
# Schedule a WorkProc to change it back automatically when the subroutine
# returns.
#
# Usage: XUbusy [$widget]
#
#      if the widget argument is not provided, the root widget is used

function XUbusy {
      typeset w=$1 tmp
      w=${w:-$DTKSH_TOPLEVEL}
      typeset dpy=$(XtDisplay - $w)
      typeset win=$(XtWindow - $w)

      if [[ ! "${_XU.HourGlassCursor}" ]]
      then _DtGetHourGlassCursor _XU.HourGlassCursor $(XtDisplay - $w)
      fi
      XDefineCursor $dpy $win ${_XU.HourGlassCursor}
      XFlush $dpy
      XUdefer "XUundefinecursor $dpy $win"
}

# XUdefer:  defer a command until the next time through the event loop
#
# Usage: XUdefer dtksh-command

function XUdefer {
      typeset tmp
      XtAddWorkProc tmp "_XUdefer '$@'"
}

# _XUdefer: work routine used in XUdefer which evals args and unschedules
# the worproc by returning 1.

function _XUdefer { # dtksh-command
      eval "$@"
      return 1
}

# XUlistselectall: select all items in a List widget
#
# Usage: XUlistselectall $widget

function XUlistselectall {
      integer i n

      XtGetValues $1 itemCount:n
      for (( i = 1; i <= n; i++ ))
      do
            XmListPosSelected $1 $i || XmListSelectPos $1 $i false
      done
}

# XUlistappend: append items to a List
#
# Usage: XUlistappend $widget item ...

function XUlistappend {
      XmListAddItems "$1" 0 "${@:2}"
}

# XUlistdelete: delete items from a list
#
# Usage (to delete by position):  XUlistdelete $widget start-item count
# Usage (to delete by string match): XUlistdelete -s $widget string ...

function XUlistdelete {
      if [[ $1 == '-s' ]]
      then
            shift
            typeset list=$1
            XmListDeleteItems "$@"
      else
            typeset list=$1 item=$2 count=$3
            XmListDeleteItemsPos "$list" "${count:-1}" "$item"
      fi
}

# XUlistselect:  select items in a List
#
# Usage:  XUlistselect [-n] [-s] $widget item ...
#
#     -s option means items are strings to match,
#        otherwise they are indexes
#
#     -n option causes notification by calling selection callbacks

function XUlistselect {
      typeset stringflag=Pos notifyflag=false

      while [[ $1 == -* ]]
      do
            case "$1" in
            -s)   stringflag=Item; shift ;;
            -n) notifyflag=true; shift ;;
            esac
      done
      typeset list=$1
      shift
      while (( $# ))
      do
            XmListSelect$stringflag $list $1 $notifyflag
            shift
      done
}

# XUlistdeselect:  deselect items in a List
#
# Usage:  XUlistdeselect [-n] [-s] $widget item ...
#
#     -s option means items are strings to match,
#        otherwise they are indexes
#
#     -n option causes notification by calling selection callbacks

function XUlistdeselect {
      typeset stringflag=Pos notifyflag=false

      while [[ $1 == -* ]]
      do
            case "$1" in
            -s)   stringflag=Item; shift ;;
            -n) notifyflag=true; shift ;;
            esac
      done
      typeset list=$1
      shift
      while (( $# ))
      do
            XmListDeselect$stringflag $list $1
            shift
      done
}

# XUlisttop: set the topmost item in a (presumably scrolled) List
#
# Usage: XUlisttop [-s] $list item
#
#     -s option means items are strings to match,
#        otherwise they are indexes

function XUlisttop {
      if [[ $1 == '-s' ]]
      then
            shift
            XmListSetItem "$@"
      else
            XmListSetPos "$@"
      fi
}
```

```
# XUlistbottom: set the bottommost item in a (presumably scrolled) List
#
# Usage: XUlistbottom [-s] $list item
#
#     -s option means items are strings to match,
#        otherwise they are indexes

function XUlistbottom {
      if [[ $1 == '-s' ]]
      then
            shift
            XmListSetBottomItem "$@"
      else
            XmListSetBottomPos "$@"
      fi
}

# XUlistfind: find items in a list, returning indexes of matching items
#
# Usage: XUlistfind variable $widget search-string

function XUlistfind {
      if [[ $1 != - ]]
      then nameref var=$1
      fi
      typeset list=$2 string=$3
      typeset matches

      XmListGetMatchPos matches "$list" "$string"
      matches=${matches//,/ }
      if [[ $1 == - ]]
      then print $matches
      else var="$matches"
      fi
}

# XUlistgetselected: find selected items in a list,
# printing indexes to standard out.
#
# Usage: XUlistgetselected variable $list-widget

function XUlistgetselected {
      typeset var=$1 list=$2
      typeset matches

      XmListGetSelectedPos matches "$list"
      matches=${matches//,/ }       # substitute spaces for commas
      if [[ $var == - ]]
      then print $matches
      else eval "$var='$matches'"
      fi
}

# XUlistget: get the text of a particular list item at an index, storing
# it in a variable.
#
# Usage: XUlistget variable $list-widget index

function XUlistget {
      typeset var=$1 list=$2 index=$3
      typeset items item array

      XtGetValues $list items:items
```

```
	n=0
	while (( ++n != index )) && [[ "$items" ]]
	do
		items=${items#*[!\\],}
	done
	item=${items%%[!\\],*}
	item="$item${items:${#item}:1}"
	item=${item//\\,/,}
	if [[ $var == - ]]
	then print "$item"
	else eval "$var='${item}'"
	fi
}

# XUlistparse: parse a comma-separated list of items into individual
# elements of an array.
#
# Usage: XUlistparse array item-list

function XUlistparse {
	nameref array=$1
	typeset items=$2 item rest
	integer n

	n=0
	while :
	do
		rest=${items#*[!\\],}

		if [[ ${#rest} == ${#items} ]]
		then item=${items}
		else item=${items:0:${#items}-${#rest}-1}
		fi

		if [[ $array != - ]]
		then array[$((n++))]=$item
		else print "$item"
		fi

		    if [[ ${#rest} == ${#items} ]]
		    then break
		    else items=${rest}
		    fi
	done
}

# XUtextdelete: delete a string from a Text widget
#
# Usage: XUtextdelete $text-widget string

function XUtextdelete {
	XmTextReplace "$@" ''
}

# XUtextselect: set the selected portion of a Text widget
#
# Usage: XUtextselect $text-widget start-pos end-pos

function XUtextselect {
	typeset t=$(XtLastTimestampProcessed - $(XtDisplay - $1))
	XmTextSetSelection "$@" $t
}
```

```
# XUtextdeselect: clear the selected portion of a Text widget
#
# Usage: XUtextdeselect $text-widget

function XUtextdeselect { # $text-widget
      typeset t=$(XtLastTimestampProcessed - $(XtDisplay - $1))
      XmTextClearSelection "$@" $t
}

# XUsettraversalorder: allow Enter key to traverse widgets instead of
# executing default action
#
# Usage: XUsettraversalorder $form-widget $default-widget $widget ...

function XUsettraversalorder {
   typeset form=$1 default=$2
   shift 2

   while (( $# > 1 ))
   do
         XtAddCallback $1 focusCallback \
            "XtSetValues $form defaultButton:NULL"
         XtAddCallback $1 losingFocusCallback \
            "XtSetValues $form defaultButton:$default"

         XtOverrideTranslations $1 \
             "Ctrl<Key>Return:ksh_eval(\"XmProcessTraversal $1
TRAVERSE_NEXT_TAB_GROUP; XtCallCallbacks $1 activateCallback\")
            <Key>Return:ksh_eval(\"XmProcessTraversal $1 TRAVERSE_NEXT_TAB_GROUP;
XtCallCallbacks $1 activateCallback\")"
      shift
   done
}

# XUregisterpopup: add an event handler to a widget that
# allows button 2 to pop up a menu.
#
# Usage: XUregisterpopup $widget $popupmenu-widget

function XUregisterpopup {
      XtAddEventHandler $1 ButtonPressMask false "_registerpop $2"
}

# _registerpopup: position and popup a menu

function _registerpop { # $popupmenu-widget
      if [[ ${EH_EVENT.XBUTTON.BUTTON} == Button[23] ]]
      then
            XmMenuPosition $1 ${EH_EVENT}
            _XU.EH_WIDGET=${EH_WIDGET}
            XtManageChild $1
      fi
}

# XUregisterwindowclose: register a command to be called when the window
# manager tries to close the application, and prevent the window manager
# from forcibly killing the application.
#
# Usage: XUregisterwindowclose command

function XUregisterwindowclose {
      XmInternAtom DELETE_ATOM \
            $(XtDisplay - ${_XU.TOPLEVEL}) WM_DELETE_WINDOW false
      XmAddWMProtocolCallback ${_XU.TOPLEVEL} $DELETE_ATOM "$*"
      XtSetValues ${_XU.TOPLEVEL} deleteResponse:DO_NOTHING
}

# XUtextautocursor:  arrange for the cursor to be invisible when a
# Text or TextField widget does not have focus.
#
# Usage: XUtextautocursor $text-widget ...

function XUtextautocursor {
      while (( $# ))
      do
            XtAddCallback $1 focusCallback \
                  "XtSetValues $1 cursorPositionVisible:true"
            XtAddCallback $1 losingFocusCallback \
                  "XtSetValues $1 cursorPositionVisible:false"
            shift
      done
}

# _XUtextforcecase: used to force case from a callback

function _XUtextforcecase {   # [-u|-l]
      typeset $1 v=${CB_CALL_DATA.TEXT.PTR}

      CB_CALL_DATA.TEXT.PTR=$v
}

# Force the case of a Text or TextField widget
#
# Usage XUtextforcecase [-u|-l] $text-widget ...

function XUtextforcecase {    # [-u|-l] text-widget ...
      typeset casearg=$1
      shift

      while (( $# ))
      do
            XtAddCallback $1 modifyVerifyCallback "_XUtextforcecase '$casearg'"
            shift
      done
}

# Used to disallow characters during a modifyVerifyCallback

function _XUtextallowchars {  # pattern
      typeset pattern=$1
      typeset v=${CB_CALL_DATA.TEXT.PTR}
      typeset orig_v=$v
      integer i

      for (( i = 0; i < ${#v}; i++ ))     # check each character
      do
            if [[ "${v:i:1}" != $pattern ]]
            then
                  v="${v:0:i}${v:i+1}"
            fi
      done
      if [[ "$orig_v" != "$v" ]]
      then
            CB_CALL_DATA.TEXT.PTR="$v"
            CB_CALL_DATA.TEXT.LENGTH="${#v}"
            XBell $(XtDisplay - ${_XU.TOPLEVEL}) 0
      fi
}

# XUtextallowchars:  set up Text or TextField widgets so only a specific
# set of characters are allowed.
#
# Usage: XUtextallowchars [$widget pattern] ...
#
#    any number of pairs of widgets and patterns may be specified.

function XUtextallowchars {
      while (( $# ))
      do
            XtAddCallback $1 modifyVerifyCallback "_XUtextallowchars '$2'"
            shift 2
      done
}

# XUwidgethelp: turn the cursor to a question mark, when user selects a widget,
# call its associated helpCallback.  If it has no helpCallback, look back up
# the widget hierarchy until one is found.
#
# Usage XUwidgethelp $parent-widget
#
#     the argument is usually a top-level Dialog or Shell widget handle.

function XUwidgethelp { # $1 = widget id
      typeset _parent=$1 widget result

      DtHelpReturnSelectedWidgetId result $_parent widget || return

      while [[ $widget ]] && [[ $widget != "(null)" ]]
      do
            XtHasCallbacks result $widget helpCallback
            if [[ $result == CallbackHasSome ]]
            then
                  print $result
                  XtCallCallbacks $widget helpCallback
                  return
            fi
            XtParent widget $widget || return
      done
}

# XUsethelp: conveniently set help callbacks for multiple widgets
#
# Usage: XUsethelp [$widget command] ...
#
#     arguments are pairs of widgets and help commands.
#

function XUsethelp {
      while (( $# ))
      do
            XtSetValues "$1" helpCallback:"$2"
            shift 2
      done
}
```

```
# XUsetaccelerators:  conveniently set accelerators and accelerator text
# for multiple widgets.
#
# Usage: XUsetaccelerators [$widget accelerator accelerator-text] ...
#
#     any number of triples of widget handles, accelerators, and
#     accelerator-text may be specified

function XUsetaccelerators {
      while (( $# ))
      do
            XtSetValues "$1" accelerator:"$2" acceleratorText:"$3"
            shift 3
      done
}

# XUsetfocus:  set the focus to a particular widget
#
# Usage: XUsetfocus $widget [direction]
#
#     if direction is not specified, the widget itself gets focus

function XUsetfocus {
      direction=$2

      XmProcessTraversal $1 ${direction:-TRAVERSE_CURRENT}
}

#
# XmCreateSimpleDialog: a fake routine used by XUcreatesimpledialog
#
# Creates a DialogShell with a Form child containing a RowColumn
# and action area.
#
# Same arguments as the XmCreate... commands

function XmCreateSimpleDialog {
      nameref _w=$1
      typeset _d

      XmCreateDialogShell _d "$2" dialogshell "${@:4}"
      XtCreateWidget $1 "$1" XmForm $_d
      XtCreateManagedWidget $1.WORK work_area XmRowColumn "${_w}" \
            orientation:horizontal \
            packing:PACK_COLUMN \
            entryAlignment:ALIGNMENT_END \
            $(XUattach top 0 left 0 right 0)
      XUaddactionarea $1.ACTION $_w ${_w.WORK}
}

# XUaddactionarea: create a manager suitable for action area buttons.
#
# Usage: XUaddactionarea variable $form-widget $under-widget
#
# creates a Separator below the $under-widget, then creates a Form
# widget underneath the separator.  Anchors the bottom, left, and
# right of the Form, and returns its handle in the variable

function XUaddactionarea {
      typeset name=$1 _parent=$2 under=$3
      eval $name=
      XtCreateManagedWidget ${name}.SEPARATOR \
            separator XmSeparator "${_parent}" \
            $(XUattach under $under 0 left 0 right 0)
      nameref sep=${name}.SEPARATOR
      XtCreateWidget $name action XmForm ${_parent} \
            $(XUattach bottom 4 left 4 right 4 under ${sep} 4)
}

# XUattach: simplified way to specify form constraints
#
# Usage: XUattach [left|right|top|bottom] [offset] ...
# Usage: XUattach [leftpos|rightpos|toppos|bottompos] [offset] ...
# Usage: XUattach [under|over|leftof|rightof] widget [offset] ...
#
#     multiple options may be mixed: XUattach left 5 under $w 3

function XUattach {
      typeset edge

      while (( $# ))
      do
            case "$1" in
            left|right|top|bottom)
                  print "${1}Attachment:ATTACH_FORM"

                  if [[ $2 == [0-9]* ]]
                  then
                        print "${1}Offset:${2}"
                        shift 2
                  else
                        shift
                  fi
                  ;;
            leftpos|rightpos|toppos|bottompos)
                  edge=${1%%pos}
                  print "${edge}Attachment:ATTACH_POSITION"
                  shift
                  if [[ $1 == [0-9]* ]]
                  then
                        print "${edge}Position:${1}"
                        shift
                  fi
                  ;;
            under)
                  print "topWidget:$2 topAttachment:ATTACH_WIDGET"
                  shift 2
                  if [[ $1 == [0-9]* ]]
                  then
                        print "topOffset:${1}"
                        shift
                  fi
                  ;;
            over)
                  print "bottomWidget:$2 topAttachment:ATTACH_WIDGET"
                  shift 2
                  if [[ $1 == [0-9]* ]]
                  then
                        print "bottomOffset:${1}"
                        shift
                  fi
                  ;;
            leftof)
                  print "rightWidget:$2 rightAttachment:ATTACH_WIDGET"
                  shift 2
                  if [[ $1 == [0-9]* ]]
```

```
                then
                      print "rightOffset:${1}"
                      shift
                fi
                ;;
          rightof)
                print "leftWidget:$2 leftAttachment:ATTACH_WIDGET"
                shift 2
                if [[ $1 == [0-9]* ]]
                then
                      print "leftOffset:${1}"
                      shift
                fi
                ;;
          *)
                print "$0: unknown argument skipped: $1" 1>&2
                shift
                ;;
          esac
      done
}

# XUalign2col: align a RowColumn widget into two columns
#
# Usage: XUalign2col $widget

function XUalign2col {
      typeset n

      XtGetValues $1 numChildren:n
      XtSetValues $1 \
            packing:PACK_COLUMN \
            numColumns:$((n/2)) \
            orientation:HORIZONTAL
}

# XUalignlabels: make all label widget arguments the same width
#
# Usage: XUalignlabels $widget ...

function XUalignlabels {
      integer max=0 width
      typeset w
      for w in "$@"
      do
            XtGetValues $w width:width
            if (( width > max ))
            then max=width
            fi
      done
      for w in "$@"
      do
            XtSetValues $w width:$max
      done
}

# XUaddtogglefields: add multiple labeled ToggleButtons to a parent
#
# Usage: XUaddtogglefields $parent [variable caption label] ...

function XUaddtogglefields {
      typeset _parent=$1
      shift
      while (( $# ))
      do
            eval $1=
            XtCreateManagedWidget $1.LABEL label XmLabel $_parent \
                  labelString:"$2"
            XtCreateManagedWidget $1 toggle XmToggleButton $_parent \
                  labelString:"$3" fillOnSelect:true
            shift 3
      done
}

# XUaddtextfields: add multiple captioned TextField widgets to a parent
#
# Usage: XUaddtextfields $parent [variable caption verify-command columns ...]

function XUaddtextfields {
      typeset _parent=$1

      shift
      while (( $# ))
      do
            eval $1=
            XtCreateManagedWidget $1.LABEL label XmLabel $_parent \
                  labelString:"$2"
            XtCreateManagedWidget $1 textfield XmTextField $_parent \
                  activateCallback:"$3" columns:$4 \
                  cursorPositionVisible:false \
                  focusCallback:"XtSetValues \${$1} cursorPositionVisible:true" \
                  losingFocusCallback:"XtSetValues \${$1}
cursorPositionVisible:false"
            shift 4
      done
}

# XUaddoptionmenu: create a simple captioned option menu
#
# Usage: XUaddoptionmenu variable $parent caption [ label ... ]
#
# Instead of a label defining a button, a dash "-" will put a separator
# in the menu.

function XUaddoptionmenu {
      typeset var=$1 _parent=$2 label=$3
      typeset tmp pull

      eval $var=
      XtCreateManagedWidget $var.LABEL label XmLabel $_parent \
            labelString:"$label"
      XmCreatePulldownMenu pull $_parent pull
      XmCreateOptionMenu "$var" "$_parent" $var \
            labelString:"" \
            subMenuId:"$pull"
      eval "typeset -A $var.CHILD"
      shift 3
      while (( $# ))
      do
            case "$1" in
            -)
                  XtCreateManagedWidget tmp tmp XmSeparatorGadget $pull
                  shift
                  ;;
            *)
```

```
                        XtCreateManagedWidget "${var}.CHILD[$1]" \
                                push XmPushButtonGadget \
                                "$pull" \
                                labelString:"$1" \
                                activateCallback:"$var.VALUE='$1'"
                        shift
                        ;;
                esac
        done
        eval tmp="\${$var}"
        XtManageChild $tmp
}

# XUactionbuttons: add buttons suitable for an action area
# to a parent widget.  A parent created by XUaddactionarea is
# a good choice.  Arranges for the labels to be nicely laid out.
#
# usage: XUactionbuttons $form [ variable label command ] ...

function XUactionbuttons {
        typeset _parent=$1
        integer i=1 numbuttons=0
        typeset defbut canbut

        shift

        while (( $# ))
        do
                case "$1" in
                -d) defbut=$2; shift ;;
                -c) canbut=$2; shift ;;
                esac
                XtCreateManagedWidget "$1" actionbutton XmPushButton $_parent \
                        labelString:"$2" \
                        activateCallback:"$3" \
                        $(XUattach leftpos $i rightpos $((i+2)) )
                (( numbuttons++ ))
                shift 3
                i=i+3
        done
        XtSetValues $_parent fractionBase:$(( numbuttons*3+1 ))
        typeset grandparent
        XtParent grandparent $_parent
        if [[ "$defbut" ]]
        then
                nameref _def=$defbut
                XtSetValues $grandparent defaultButton:$_def
        fi
        if [[ "$canbut" ]]
        then
                nameref _can=$canbut
                XtSetValues $grandparent cancelButton:$_can
        fi
        XtManageChild $_parent
}

# XUaddintegerfields: add captioned fields suitable for integer display
#
# Usage: XUaddintegerfields $parent [variable label columns ...]

function XUaddintegerfields {
        typeset _parent=$1 _r
        shift
```

```
	while (( $# ))
	do
		XUaddtextfields $_parent "$1" "$2" XUverifyinteger "$3"
		eval _r=\${$1}
		XUtextallowchars ${_r} '[0-9]'
		shift 3
	done
}

# XUverifymoney: verify that the CB_WIDGET is of the form suitable
# for currancy (2 decimal digit number). This function is suitable as
# a verification callback for a Text or TextField widget.
#
# Usage: XUverifymoney

function XUverifymoney {
	typeset v=

	XtGetValues $CB_WIDGET value:v
	case "$v" in
	+([0-9]))
		XtSetValues $CB_WIDGET value:"$v.00"
		;;
	+([0-9]).[0-9])
		XtSetValues $CB_WIDGET value:"${v}0"
		;;
	+([0-9]).[0-9][0-9])
		;;
	*)
		XUverifyerror "Bad monetary value" "$v"
		XUdefer "XmProcessTraversal $CB_WIDGET TRAVERSE_CURRENT"
		;;
	esac
}

# XUaddmoneyfields: add captioned fields suitable for money
#
# Usage: XUaddmoneyfields $parent [variable label columns ...]

function XUaddmoneyfields {
	typeset _parent=$1
	shift
	while (( $# ))
	do
		XUaddtextfields "$_parent" "$1" "$2" XUverifymoney "$3"
		eval _r=\${$1}
		XUtextallowchars ${_r} '[0-9.]'
		shift 3
	done
}

function XUverifyerror {
	XUerror "Error" "$1: $2" :
	XBell $(XtDisplay - ${_XU.TOPLEVEL}) 50
}

function XUverifyfloat {
	typeset v

	XtGetValues $CB_WIDGET value:v
	if [[ $v != *([0-9])?(.*([0-9])) ]]
	then
		XUverifyerror "Bad Floating Point Value" "$v"
```

```
      fi
}

# XUverifyinteger: verify that CB_WIDGET holds an integer
#
# Usage: XUverifyinteger

function XUverifyinteger {
      typeset v

      XtGetValues $CB_WIDGET value:v
      if [[ $v != *([0-9]) ]]
      then
            XUverifyerror "Bad Integer Value" "$v"
      fi
}


# XUaddfloatfields: add captioned TextFields suitable for floating
# point values.
#
# Usage: XUaddfloatfields $parent [variable label columns ...]

function XUaddfloatfields {
      typeset _parent=$1
      shift
      while (( $# ))
      do
            XUaddtextfields "$_parent" "$1" "$2" XUverifyfloat "$3"
            eval _r=\${$1}
            XUtextallowchars ${_r} '[0-9.]'
            shift 3
      done
}

# subroutine used by XUtextautoselect to select all of a text
# widget when it gains focus

function _XUtextautoselect {  # no args
      typeset widget=$CB_WIDGET
      typeset v

      XtGetValues $widget value:v
      XmTextFieldSetSelection $widget \
            0 ${#v} \
            $(XtLastTimestampProcessed - $(XtDisplay - $widget))
}

# XUtextautoselect: set up TextFields so when they gain focus
# the text is selected (allowing type-over)
#
# Usage: XUtextautoselect $widget ...

function XUtextautoselect {
      while (( $# ))
      do
            XtAddCallback $1 focusCallback "_XUtextautoselect"
            XtSetValues $1 cursorPositionVisible:false
            shift
      done
}

# Subroutine used by XUtextautotraverse to traverse to the next
# widget when a maximum character position is exceeded.

function _XUtextautotraverse {      # max-width
      typeset width=$1 widget=$CB_WIDGET
      typeset p
      XtGetValues $widget cursorPosition:p
      if (( p >= width ))
      then XmProcessTraversal $widget TRAVERSE_NEXT_TAB_GROUP
      fi
}

# XUtextautotraverse: set up TextField widgets to traverse
# automatically when a maximum character position is exceeded.
#
# Usage: XUtextautotraverse [$widget numchars ...]

function XUtextautotraverse {
      while (( $# ))
      do
            XtAddCallback $1 valueChangedCallback "_XUtextautotraverse $2"
            shift 2
      done
}

# Subroutine used to validate a date.

function _XUdatevalidate {    # [MM|DD|YY] month day year
      typeset widget=$CB_WIDGET
      typeset valtype=$1 mm=$2 dd=$3 yy=$4
      typeset p v m d y

      XtGetValues $widget cursorPosition:p value:v
      case "$1" in
      DD)
            if ((v > 31 || v < 1))
            then
                  XUverifyerror "Bad day, must be between 1 and 31" "$v"
                  XtSetValues $CB_WIDGET cursorPosition:0
                  XmProcessTraversal $CB_WIDGET TRAVERSE_CURRENT
                  return
            fi
            ;;
      MM)
            if ((v > 12 || v < 1))
            then
                  XUverifyerror "Bad month, must be between 1 and 12" "$v"
                  XtSetValues $CB_WIDGET cursorPosition:0
                  XmProcessTraversal $CB_WIDGET TRAVERSE_CURRENT
                  return
            fi
            ;;
      YY)
            XtGetValues $mm value:m
            XtGetValues $dd value:d
            XtGetValues $yy value:y
            if ! XUisdate $m $d $y
            then
                  XUverifyerror "Bad date" "$m/$d/$y"
                  XtSetValues $mm cursorPosition:0
                  XmProcessTraversal $CB_WIDGET TRAVERSE_CURRENT
                  return
            fi
            ;;
      esac
      if (( p >= 2 ))
      then XmProcessTraversal $widget TRAVERSE_NEXT_TAB_GROUP
      fi
}

# XUadddatefields: add TextField widgets capable of holding
# a date.  All validation is set up for you.
#
# Usage: XUadddatefields $parent [variable label ...]
#
#     subvariables hold individual components: variable.MM, variable.DD,
#     and variable.YY

function XUadddatefields {
      typeset _parent=$1

      shift
      while (( $# ))
      do
            eval $1=
            nameref tmp=$1 _m=$1.MM _d=$1.DD _y=$1.YY
            XtCreateManagedWidget $1.LABEL label XmLabel $_parent \
                  labelString:"$2"

            XtCreateManagedWidget $1 rc XmRowColumn $_parent \
                  orientation:horizontal
            XtCreateManagedWidget $1.MM textfield XmTextField $tmp \
                  valueChangedCallback:"_XUdatevalidate MM \
                        \${$1.MM} \${$1.DD} \${$1.YY}" \
                  columns:2
            XtCreateManagedWidget lab lab XmLabelGadget $tmp labelString:"/"
            XtCreateManagedWidget $1.DD textfield XmTextField $tmp \
                  valueChangedCallback:"_XUdatevalidate \
                        DD \${$1.MM} \${$1.DD} \${$1.YY}" \
                  columns:2
            XtCreateManagedWidget lab lab XmLabelGadget $tmp labelString:"/"
            XtCreateManagedWidget $1.YY textfield XmTextField $tmp \
                  losingFocusCallback:"_XUdatevalidate \
                        YY \${$1.MM} \${$1.DD} \${$1.YY}" \
                  columns:2
            XUtextallowchars $_m '[0-9]' $_d '[0-9]' $_y '[0-9]'
            XUtextautoselect $_m $_d $_y
            XUtextautocursor $_m $_d $_y
            XUtextautotraverse $_y 2
            shift 2
      done
}

# XUtextcut: cut selected text from a Text or TextField widget
#
# Usage: XUtextcut $widget

function XUtextcut {
      typeset t=$(XtLastTimestampProcessed - $(XtDisplay - $1))
      typeset c=$(XtClass - $1)     # XmText or XmTextField

      ${c}Cut "$@" $t
}
```

```
# XUtextpaste: paste text to a Text or TextField widget
#
# Usage: XUtextpaste $widget

function XUtextpaste {
      typeset c=$(XtClass - $1)
      ${c}Paste "$@"
}

# XUtextcopy: copy selected text from a Text or TextField widget
#
# Usage: XUtextcopy $widget

function XUtextcopy {
      typeset t=$(XtLastTimestampProcessed - $(XtDisplay - $1))
      typeset c=$(XtClass - $1)

      ${c}Copy "$@" $t
}


# XUtextsethighlight: set the highlight mode for text in a
# Text or TextField widget
#
# Usage: XUtextsethighlight $widget start-pos end-pos [type]
#

function XUtextsethighlight {
      typeset widget=$1 left=$2 right=$3 type=$4
      typeset c=$(XtClass - $1)

      type=${type:-HIGHLIGHT_SECONDARY_SELECTED}
      ${c}SetHighlight $widget "$left" "$right" "$type"
}

# XUselectionchildren: get the children of a SelectionDialog
#
# Usage: XUselectionchildren variable
#
#     the variable should hold a SelectionDialog handle, and
#     subvariables are created for each child

function XUselectionchildren {
      nameref _w=$1
      typeset child

      for child in CANCEL_BUTTON DEFAULT_BUTTON HELP_BUTTON \
            APPLY_BUTTON LIST LIST_LABEL OK_BUTTON SELECTION_LABEL \
            SEPARATOR TEXT WORK_AREA
      do
            XmSelectionBoxGetChild ${!_w}.$child $_w DIALOG_$child
      done
}
```

```
# XUmenusystem: create a menu system
#
# Usage: XUmenusystem $parent [variable label mnemonic action ...]
#     ...
#
# action can be either a ksh command string or an open curly brace,
# in which case a submenu is created up to the matching close curly brace.
# Menus may be nested in this manner to any depth.

function XUmenusystem {
      typeset _parent="$1" tmp menu buttontype exclusivevar
      integer level

      shift

      while (( $# != 0 ))
      do
            if [[ $1 == "{" ]] || [[ $1 == "}" ]]
            then return
            fi
            buttontype=XmPushButtonGadget
            exclusivevar=""
            if [[ $1 == -t ]]
            then buttontype=XmToggleButtonGadget
                 shift
            elif [[ $1 == -e ]]
            then buttontype=XmToggleButtonGadget
                 exclusivevar=$2
                 shift 2
            fi
            if [[ $1 == "-" ]]
            then
                  XtCreateManagedWidget tmp tmp XmSeparator $_parent
                  shift 4
            elif [[ $4 = "{" ]]
            then
                  XmCreatePulldownMenu menu "$_parent" menu
                  XtCreateManagedWidget "$1" "$1" XmCascadeButton \
                        "$_parent" \
                        labelString:"$2" \
                        subMenuId:"$menu" \
                        mnemonic:"$3"
                  eval "$1.PULLDOWN=$menu"
                  shift 4
                  XUmenusystem $menu "$@"
                  level=1
                  while (( level > 0 && $# > 0 ))
                  do
                        if [[ $1 == "{" ]]
                        then let level++
                        elif [[ $1 == "}" ]]
                        then let level--
                        fi
                        shift
                  done
            else
                  if [[ $buttontype == XmPushButtonGadget ]]
                  then
                        XtCreateManagedWidget "$1" "$1" $buttontype \
                              "$_parent" \
                              labelString:"$2" \
                              mnemonic:"$3" \
                              activateCallback:"$4"
                  else
                        XtCreateManagedWidget "$1" "$1" $buttontype \
                              "$_parent" \
                              labelString:"$2" \
                              mnemonic:"$3" \
                              valueChangedCallback:"$4"
                        if [[ "$exclusivevar" ]]
                        then
                              eval tmp="\${$1}"
                              eval "$exclusivevar=\"\${$exclusivevar} $tmp\""
                              XtAddCallback $tmp valueChangedCallback \
                                    "_XUmenuexclusive $tmp \"\${$exclusivevar}\""
                              XtSetValues $tmp indicatorType:ONE_OF_MANY
                        fi
                  fi
                  shift 4
            fi
      done
}

# Subroutine used to implement exclusive ToggleButtons in a
# menu

function _XUmenuexclusive {   # $toggle-widget other-widgets
      typeset s

      XtGetValues $1 set:s
      if [[ $s == true ]]
      then
            for i in $2
            do
                  if [[ $i != $1 ]]
                  then XtSetValues $i set:false
                  fi
            done
      fi
}
```

```
#
# Widget creation commands.
#
# All use the  _XUcreateobject routine to create different widgets.
# The general usage is:
#
# Usage: XU<widgetname> [-u] variable $parent [resource:value ...]
#
# Where <widgetname> is the lower case of the widget name, and the -u option
# can be used to create the widget unmanaged.  For example:
#
# XUlabel mylabel $rowcol labelString:"hello"
#

#
# Names for widgets created by _XUcreateobj are composed of the
# last element of the variable that holds them.  This function strips
# out any hierarchical parents of the variable, and strips off any subscript
# so it forms a legal widget name.  For example, the variable:
#
# MB.FILE[1]  would become a widget name:  FILE.
#

# Subroutine: strip a variable down to its last component, with no
# index

function _XUstripvar {  # variable-name
      typeset wname=$1
      wname=${wname##*.}
      wname=${wname%%\[*]}
      print "$wname"
}

# Subroutine used to create an object.

function _XUcreateobject {    # command [-u] args...
      integer nomanage=0

      if [[ $2 == "-u" ]]
      then
            nomanage=1
            set "$1" "${@:3}"
      fi
      typeset _c=$1 _n=$2 _p=$3 _r
      shift 3
      typeset _wname=${_n##*.}
      _wname=${_wname%%\[*]}
      $_c "$_n" "$_p" "$_wname" "${@}"
      eval _r=\"\${$_n}\"
      XtAddCallback $_r destroyCallback "unset $_n"
      (( nomanage )) || XtManageChild $_r
}
```

```
# Aliases are used to map every widget creation command to a call to
# _XUcreateobject

alias XUhelpdialog='_XUcreateobject DtCreateHelpDialog'
alias XUquickhelpdialog='_XUcreateobject DtCreateHelpQuickDialog'
alias XUarrowbutton='_XUcreateobject XmCreateArrowButton'
alias XUarrowbuttongadget='_XUcreateobject XmCreateArrowButtonGadget'
alias XUbulletinboard='_XUcreateobject XmCreateBulletinBoard'
alias XUbulletinboarddialog='_XUcreateobject XmCreateBulletinBoardDialog'
alias XUcascadebutton='_XUcreateobject XmCreateCascadeButton'
alias XUcascadebuttongadget='_XUcreateobject XmCreateCascadeButtonGadget'
alias XUcommand='_XUcreateobject XmCreateCommand'
alias XUdialogshell='_XUcreateobject XmCreateDialogShell'
alias XUdrawingarea='_XUcreateobject XmCreateDrawingArea'
alias XUdrawnbutton='_XUcreateobject XmCreateDrawnButton'
alias XUerrordialog='_XUcreateobject XmCreateErrorDialog'
alias XUfileselectionbox='_XUcreateobject XmCreateFileSelectionBox'
alias XUfileselectiondialog='_XUcreateobject XmCreateFileSelectionDialog'
alias XUform='_XUcreateobject XmCreateForm'
alias XUformdialog='_XUcreateobject XmCreateFormDialog'
alias XUframe='_XUcreateobject XmCreateFrame'
alias XUinformationdialog='_XUcreateobject XmCreateInformationDialog'
alias XUlabel='_XUcreateobject XmCreateLabel'
alias XUlabelgadget='_XUcreateobject XmCreateLabelGadget'
alias XUlist='_XUcreateobject XmCreateList'
alias XUmainwindow='_XUcreateobject XmCreateMainWindow'
alias XUmenubar='_XUcreateobject XmCreateMenuBar'
alias XUmenushell='_XUcreateobject XmCreateMenuShell'
alias XUmessagebox='_XUcreateobject XmCreateMessageBox'
alias XUmessagedialog='_XUcreateobject XmCreateMessageDialog'
alias XUoptionmenu='_XUcreateobject XmCreateOptionMenu'
alias XUpanedwindow='_XUcreateobject XmCreatePanedWindow'
alias XUpopupmenu='_XUcreateobject XmCreatePopupMenu'
alias XUpromptdialog='_XUcreateobject XmCreatePromptDialog'
alias XUpulldownmenu='_XUcreateobject XmCreatePulldownMenu'
alias XUpushbutton='_XUcreateobject XmCreatePushButton'
alias XUpushbuttongadget='_XUcreateobject XmCreatePushButtonGadget'
alias XUquestiondialog='_XUcreateobject XmCreateQuestionDialog'
alias XUradiobox='_XUcreateobject XmCreateRadioBox'
alias XUrowcolumn='_XUcreateobject XmCreateRowColumn'
alias XUscale='_XUcreateobject XmCreateScale'
alias XUscrollbar='_XUcreateobject XmCreateScrollBar'
alias XUscrolledlist='_XUcreateobject XmCreateScrolledList'
alias XUscrolledtext='_XUcreateobject XmCreateScrolledText'
alias XUscrolledwindow='_XUcreateobject XmCreateScrolledWindow'
alias XUselectionbox='_XUcreateobject XmCreateSelectionBox'
alias XUselectiondialog='_XUcreateobject XmCreateSelectionDialog'
alias XUseparator='_XUcreateobject XmCreateSeparator'
alias XUseparatorgadget='_XUcreateobject XmCreateSeparatorGadget'
alias XUtext='_XUcreateobject XmCreateText'
alias XUtextfield='_XUcreateobject XmCreateTextField'
alias XUtogglebutton='_XUcreateobject XmCreateToggleButton'
alias XUtogglebuttongadget='_XUcreateobject XmCreateToggleButtonGadget'
alias XUwarningdialog='_XUcreateobject XmCreateWarningDialog'
alias XUworkarea='_XUcreateobject XmCreateWorkArea'
alias XUworkingdialog='_XUcreateobject XmCreateWorkingDialog'
alias XUsimpledialog='_XUcreateobject XmCreateSimpleDialog -u'

function XUtoplevelshell {
      XtCreatePopupShell $1 $1 TopLevelShell "${@:2}"
}
```

```
# XUaddtogglebuttons: add Toggle buttons to a parent, with mnemonics
#
# Usage XUaddtogglebuttons $parent [variable label mnemonic command ...]

XUaddtogglebuttons()
{
   typeset _parent=$1 callback
   shift

   while (( $# > 2 ))
   do
            typeset v=$(_XUstripvar $1)
            XtCreateManagedWidget "$1" "$v" XmToggleButton "$_parent" \
                  labelString:"$2" mnemonic:"$3" valueChangedCallback:"$4" \
                  fillOnSelect:true selectColor:black
                  shift 4
   done
}

# XUaddbuttons: add PushButtons to a parent
#
# Usage: XUaddbuttons $parent [variable label command ...]

XUaddbuttons()
{
   typeset _parent=$1 callback
   shift

   while (( $# > 2 ))
   do
            XtCreateManagedWidget "$1" "$1" XmPushButton "$_parent" \
                  labelString:"$2" activateCallback:"$3"
                  shift 3
   done
}

# XUaddpixbuttons: add PushButtons that display pictures to a parent
#
# Usage: XUaddpixbuttons $parent [variable pixmap command ...]

XUaddpixbuttons()
{
   typeset _parent=$1 callback
   shift

   while (( $# > 2 ))
   do
            XtCreateManagedWidget "$1" "$1" XmPushButton "$_parent" \
                  labelType:PIXMAP labelPixmap:"$2" activateCallback:"$3"
                  shift 3
   done
}

. /usr/dt/lib/dtksh/DtFuncs.dtsh

# Usage: XUwarning [-m] title message ok-cb quit-cb help-cb

function XUwarning {
      typeset mode=DIALOG_MODELESS
      if [[ "$1" == -m ]]
      then mode=DIALOG_PRIMARY_APPLICATION_MODAL; shift
      fi
      DtkshDisplayWarningDialog "$1" "${2//     /}" "$3" "$4" "$5" $mode
}

# Usage: XUquestion [-m] title message ok-cb quit-cb help-cb

function XUquestion {  # [-m] title message ok-cb quit-cb help-cb
      typeset mode=DIALOG_MODELESS
      if [[ "$1" == -m ]]
      then mode=DIALOG_PRIMARY_APPLICATION_MODAL; shift
      fi
      DtkshDisplayQuestionDialog "$1" "${2//    /}" "$3" "$4" "$5" $mode
}

# Usage: XUinformation [-m] title message ok-cb quit-cb help-cb

function XUinformation {  # [-m] title message ok-cb quit-cb help-cb
      typeset mode=DIALOG_MODELESS
      if [[ "$1" == -m ]]
      then mode=DIALOG_PRIMARY_APPLICATION_MODAL; shift
      fi
      DtkshDisplayInformationDialog "$1" "${2// /}" "$3" "$4" "$5" $mode
}

# Usage: XUerror [-m] title message ok-cb quit-cb help-cb

function XUerror {  # [-m] title message ok-cb quit-cb help-cb
      typeset mode=DIALOG_MODELESS
      if [[ "$1" == -m ]]
      then mode=DIALOG_PRIMARY_APPLICATION_MODAL; shift
      fi
      DtkshDisplayErrorDialog "$1" "${2// /}" "$3" "$4" "$5" $mode
}

# Usage: XUworking [-m] title message ok-cb quit-cb help-cb

function XUworking {  # [-m] title message ok-cb quit-cb help-cb
      typeset mode=DIALOG_MODELESS
      if [[ "$1" == -m ]]
      then mode=DIALOG_PRIMARY_APPLICATION_MODAL; shift
      fi
      DtkshDisplayWorkingDialog "$1" "${2//    /}" "$3" "$4" "$5" $mode
      if [[ ! "$3" ]]
      then
            XtUnmanageChild $(XmMessageBoxGetChild - \
                        $_DT_WORKING_DIALOG_HANDLE \
                        DIALOG_OK_BUTTON)
      fi
      _XU.WorkingDialog=$_DT_WORKING_DIALOG_HANDLE
}
```

```
#
# Functions for simple flat-file database access and updates
#

# XUdbopen -   open a file, reading in records into an array.
# The user can specify the field delimiter, it defaults
# to a pipe "|" symbol.
#
# This must step through each line of the file, and thus the read time
# grows proportionately with the file size.  It was clocked at about 250
# lines per second on an Intel 486/66 (6 records per line, lines about 80
# characters long each).

# Usage: XUdbopen variable file [delimiter]

function XUdbopen {
      nameref handle=$1
      typeset file=$2 delimiter=$3

      delimiter=${delimiter:-|}
      unset handle
      handle=
      handle.file=${file:-NoName}
      typeset -L1 handle.delimiter=$delimiter
      handle.needsync=FALSE
      if [[ -r ${handle.file} ]]
      then XUdbread handle < ${handle.file}
      fi
}

# Usage: XUdbread db-variable

function XUdbread {     # dbhandle
      nameref handle=$1
      typeset line
      integer recnum=0

      unset handle.record
      while read line
      do
            handle.record[recnum++]=$line
      done
}

# Do a lookup based on an arbitrary pattern anywhere in the record.
# This requires a linear search.  On a 486/66 this can scan about 500
# records per second.

# Usage: XUdbfindpattern db-variable pattern

function XUdbfindpattern {    # DBhandle pattern
      nameref handle=$1
      typeset pattern=$2

      for ((i = 0; i < ${#handle.record[@]}; i++))
      do
            if [[ ${handle.record[i]} == $pattern ]]
            then
                  print ${handle.record[i]}
            fi
      done
}
```

```
# This function writes any pending DB changes back out to the file
#
# It was clocked at about 300 records per second on an Intel 486/66.

# Usage: XUdbsync db-variable

function XUdbsync {     #  DBhandle
      nameref handle=$1
      integer i

      if [[ ${handle.needsync} == FALSE ]]
      then return
      fi

      XUdbwrite handle > "${handle.file}"
      handle.needsync=FALSE
}

#
# Write all records, in order, to the standard output
#
# Usage: XUdbwrite db-variable

function XUdbwrite {    # DBhandle
      nameref handle=$1
      typeset outfile=$2
      integer i

      for ((i = 0; i < ${#handle.record[@]}; i++))
      do
            print ${handle.record[i]}
      done
}

#
# Delete a record.  Quite fast since it uses array functions.
#
# Usage XUdbdelete db-variable index

function XUdbdelete {   # DBhandle index
      nameref handle=$1
      typeset index=$2

      set -A handle.record \
            "${handle.record[@]:0:index-1}" "${handle.record[@]:index+1}"
}

#
# Append a new record.
#
# Usage: XUdbappend db-variable record

function XUdbappend {   # DBhandle record
      nameref handle=$1
      typeset record=$2

      handle.record[${#handle.record[@]}]=$record
      handle.needsync=TRUE
}

#
# Uses sort(1) to sort the database.  Since it requires several operations
# to perform this, it is fairly slow.  It can process about 100 records
# per second on a 486/66.
#
# Example:
#
#     XUdbsort $DB -f +3      # sort by field 3, case insensitive (-f)
#
# Usage: XUdbsort db-variable [sort-options]

function XUdbsort { #  DBhandle sort-options
      nameref handle=$1

      XUdbwrite handle | \
            sort -t"${handle.delimiter}" "${@:2}" | \
            XUdbread handle
}

# Usage: XUdbsplit db-variable index array-variable

function XUdbsplit {    # DBhandle index variable
      nameref handle=$1
      typeset IFS=${handle.delimiter}

      set -A $3 ${handle.record[$2]}
}

#
# Validation Functions
#

# Usage: XUisinteger string ...

function XUisinteger {  # strings ...
      typeset s

      for s
      do
            if [[ $s != +([0-9]) ]]
            then return 1
            fi
      done
      return 0
}

# Usage: XUisblank string ...

function XUisblank {  # strings
      typeset s

      for s in "$@"
      do
            if [[ $s ]] || [[ $s != +([   ]) ]]
            then return 1
            fi
      done
      return 0
}

# Usage: XUisfloat string ...

function XUisfloat {  # strings ...
      typeset s

      for s
      do
            if XUisblank "$s" || [[ $s != ?(+([0-9]))?(.+([0-9])) ]]
            then return 1
            fi
      done
      return 0
}

set -A _XU.MONTHDAYS 0 31 29 31 30 31 30 31 31 30 31 30 31

# Usage: XUisdate mm dd yy

function XUisdate {     # mm dd yy
      typeset mm=$1 dd=$2 yy=$3

      if ! XUisinteger "$mm" "$dd" "$yy"
      then return 1
      fi
      if (( mm < 1 || mm > 12 || dd < 1 || dd > _XU.MONTHDAYS[mm] || yy < 0 ))
      then return 1
      fi
      # hard code the test for leap years
      if (( mm == 2 && dd == 29 && ((1900+yy)%4) != 0 ))
      then return 1
      fi
      return 0
}
```