NAME

clipboard − Manipulate Tk clipboard

SYNOPSIS

clipboard *option* ?*arg arg ...*?

DESCRIPTION

This command provides a Tcl interface to the Tk clipboard, which stores data for later retrieval using the selection mechanism (via the **−selection CLIPBOARD** option). In order to copy data into the clipboard, **clipboard clear** must be called, followed by a sequence of one or more calls to **clipboard append**. To ensure that the clipboard is updated atomically, all appends should be completed before returning to the event loop.

The first argument to **clipboard** determines the format of the rest of the arguments and the behavior of the command. The following forms are currently supported:

clipboard append ?**−displayof** *window*? ?**−format** *format*? ?**−type** *type*? ?**− −**? *data*

Appends *data* to the clipboard on *window*'s display in the form given by *type* with the representation given by *format* and claims ownership of the clipboard on *window*'s display.

Type specifies the form in which the selection is to be returned (the desired "target" for conversion, in ICCCM terminology), and should be an atom name such as **STRING** or **FILE_NAME**; see the Inter-Client Communication Conventions Manual for complete details. *Type* defaults to **STRING**.

The *format* argument specifies the representation that should be used to transmit the selection to the requester (the second column of Table 2 of the ICCCM), and defaults to **STRING**. If*format* is **STRING**, the selection is transmitted as 8-bit ASCII characters. If *format* is **ATOM**, then the *data* is divided into fields separated by white space; each field is converted to its atom value, and the 32-bit atom value is transmitted instead of the atom name. For any other *format*, *data* is divided into fields separated by white space and each field is converted to a 32-bit integer; an array of integers is transmitted to the selection requester. Note that strings passed to**clipboard append** are concatenated before conversion, so the caller must take care to ensure appropriate spacing across string boundaries. All items appended to the clipboard with the same *type* must have the same *format*.

The *format* argument is needed only for compatibility with clipboard requesters that do not use Tk. If the Tk toolkit is being used to retrieve the **CLIPBOARD** selection then the value is converted back to a string at the requesting end, so *format* is irrelevant.

A **− −** argument may be specified to mark the end of options: the next argument will always be used as *data*. This feature may be convenient if, for example, *data* starts with a **−**.

clipboard clear ?**−displayof** *window*?

Claims ownership of the clipboard on *window*'s display and removes any previous contents. *Window* defaults to ".". Returns an empty string.

clipboard get ?**−displayof** *window*? ?**−type** *type*?

Retrieve data from the clipboard on *window*'s display. *Window* defaults to ".". *Type* specifies the form in which the data is to be returned and should be an atom name such as **STRING** or **FILE_NAME**. *Type* defaults to **STRING**. This command is equivalent to "**selection get −selection CLIPBOARD**".

Note that on modern X11 systems, the most useful type to retrieve for transferred strings is not **STRING**, but rather **UTF8_STRING**.

EXAMPLES

Get the current contents of the clipboard.

```
if {[catch {clipboard get} contents]} {
   # There were no clipboard contents at all
```

```
}
```

Set the clipboard to contain a fixed string.

```
clipboard clear
clipboard append "some fixed string"
```

You can put custom data into the clipboard by using a custom **−type** option. This is not necessarily portable, but can be very useful. The method of passing Tcl scripts this way is effective, but should be mixed with safe interpreters in production code.

```
# This is a very simple canvas serializer;
# it produces a script that recreates the item(s) when executed
proc getItemConfig {canvas tag} {
  set script {}
  foreach item [$canvas find withtag $tag] {
    append script {$canvas create } [$canvas type $item]
    append script { } [$canvas coords $item] { }
    foreach config [$canvas itemconf $item] {
      lassign $config name − − − value
      append script [list $name $value] { }
    }
    append script \n
  }
  return [string trim $script]
}

# Set up a binding on a canvas to cut and paste an item
set c [canvas .c]
pack $c
$c create text 150 30 −text "cut and paste me"
bind $c <<Cut>> {
  clipboard clear
  clipboard append −type TkCanvasItem \
      [getItemConfig %W current]
  # Delete because this is cut, not copy.
  %W delete current
}
bind $c <<Paste>> {
  catch {
    set canvas %W
    eval [clipboard get −type TkCanvasItem]
  }
}
```

SEE ALSO

interp(n), selection(n)

KEYWORDS

clear, format, clipboard, append, selection, type