nuweb.tcl: A TCL/Tk Front End for Nuweb

Mark Wroth

March 11, 2001

1 Identification

This TCL/Tk script acts as a graphical front end to `Nuweb`. This is Revision 1.3, dated March 11, 2001.

2 Background

`Nuweb`, by Preston Briggs, is a *literate programming* processor that produces output code files and TEX documentation files from a single source file, called a *web* file.

In its original form, `Nuweb` is a command line utility; the user interacts with the program by typing the command name, various command options, and the name of the target file. This TCL/Tk script provides a Graphical User Interface (GUI) easing the selection of the target web file, the choice of options, and the repetitive running of `Nuweb`.

3 Implementation

3.1 Organization

The basic organization of this script is a series of sections that set up the various pieces of the GUI, a setup of the GUI itself, and a set of supporting procedures.

```
"nuweb.tcl" 1 ≡
    # $Id: nuwebtcl.w,v 1.3 2001/03/11 20:33:30 Mark Exp $
    ⟨Initialization 2a⟩
    ⟨Set up configuration buttons 2c, ... ⟩
    ⟨Set up the menus 3c, ... ⟩
    ⟨Define the action buttons 5b, ... ⟩
    ⟨Display the GUI 7⟩
    ⟨Supporting procedures 6c, ... ⟩
```

3.2 Initial Configuration

Set up the initial configuration; first we get the name of the intended file, and then set the initial values of the various command line options.

⟨Initialization 2a⟩ ≡

```
⟨Set target path and name 2b⟩
set noTeX   0
set noCode  0
set verbose 1
set noTest  0
set numberScraps 0
```

Macro referenced in scrap 1.

Because of the way we use it, the requester that we use to get the target file name involves several related actions; we get the full name of the path and file, and then split it up into the path and file name components.

There are several places where we want to execute the same set of actions. While a procedure would do this, it's easier to use a defined scrap; the `.tcl` script will be slightly larger, but this has little effect.

⟨Set target path and name 2b⟩ ≡

```
    set target [tk_getOpenFile -filetypes {
    {{Nuweb files} {.w}}
    {{All files} {""}}
    }]
set target_path [file dirname $target]
set target_name [file tail $target]
eval cd \"$target_path\"
```

Macro referenced in scraps 2a, 3a, 4a.

The next scrap sets up the name of the executable file. It is stored in a variable to facilitate changing it via a configuration menu pick.

⟨Set up configuration buttons 2c⟩ ≡

```
set nuweb_name "nuweb"
```

Macro defined by scraps 2c, 3ab.
Macro referenced in scrap 1.

3.3 The Configuration Buttons

The primary configuration is the name of the target file. This button displays the name of the currently selected file, and allows the user to change it if desired by clicking on the button and selecting the new file from the file requester that will appear.

This is a very compact user interface. However, some users might not find it intuitive (since Windows tends to provide a separate "browse" button next to a label widget). I don't consider this a problem, as I am probably the only user for this particular script. If this script does get wider dissemination, however, consider reimplementing this with a label and a browse button.

⟨Set up configuration buttons 3a⟩ ≡

```
button .browse -textvariable target_name -command {
  ⟨Set target path and name 2b⟩
}
label .pathname -textvariable target_path
```

Macro defined by scraps 2c, 3ab.
Macro referenced in scrap 1.

We also allow the user to set the various options of `nuweb`; this section sets the options up in a two-column pane. The option defaults are set above (Section 3.2).

⟨Set up configuration buttons 3b⟩ ≡

```
frame .config
frame .config.left
frame .config.right
checkbutton .noTeX -text "Suppress TeX Output" \
   -variable noTeX -anchor w
checkbutton .noCode -text "Suppress Code Output" \
   -variable noCode -anchor w
checkbutton .verbose -text "Verbose Output" \
   -variable verbose -anchor w
checkbutton .noTest -text "Do Not Test For File Changed" \
   -variable noTest -anchor w
checkbutton .numberScraps -text "Number Scraps Consecutively" \
   -variable numberScraps -anchor w
```

Macro defined by scraps 2c, 3ab.
Macro referenced in scrap 1.

3.4 Menus

As a convenience—and as a programming exercise—we set up a series of menus that allow the user to perform various actions. Some of these capabilities duplicate the capabilities provided by other elements of the GUI, and some are accessible only from the menu structure.

The menus themselves will be contained in a frame.

⟨Set up the menus 3c⟩ ≡

```
frame .menubar -relief groove -borderwidth 4
```

Macro defined by scraps 3c, 4abc, 5a.
Macro referenced in scrap 1.

The “File” menu

⟨Set up the menus 4a⟩ ≡

```
menubutton .menubar.file   -text "File" \
  -direction below -menu .menubar.file.menu
menu .menubar.file.menu\
  -tearoff 0
.menubar.file.menu add command -label "Set target" -command {
  ⟨Set target path and name 2b⟩}
.menubar.file.menu add command -label "Exit" -command {
  exit}
```

Macro defined by scraps 3c, 4abc, 5a.
Macro referenced in scrap 1.

The “configuration” menu.

⟨Set up the menus 4b⟩ ≡

```
menubutton .menubar.config -text "Config" \
  -direction below -menu .menubar.config.menu
menu .menubar.config.menu \
  -tearoff 0
.menubar.config.menu add command -label "Nuweb" -command {
  set nuweb [tk_getOpenFile -filetypes {
      {{Executables} {.exe}}
      {{All files} {""}}
  }]
  }
```

Macro defined by scraps 3c, 4abc, 5a.
Macro referenced in scrap 1.

The “Help” menu.

⟨Set up the menus 4c⟩ ≡

```
menubutton .menubar.help   -text "Help" \
  -direction below -menu .menubar.help.menu
menu .menubar.help.menu \
  -tearoff 0
.menubar.help.menu add command -label "Help" \
        -command {showHelp}
.menubar.help.menu add command -label "About" -command {
  tk_messageBox -type ok -message \
```

```
"This is nuweb.tcl, a TCL/Tk application providing a
 graphical front end for Nuweb.\n\n
 Copyright (c) Mark Wroth <mark@astrid.upland.ca.us> 2001\n\n
 Free distribution under the terms of the Gnu Public License
 is authorized."
}
```

Macro defined by scraps 3c, 4abc, 5a.
Macro referenced in scrap 1.

Finally, the menus are packed in the top frame.

⟨Set up the menus 5a⟩ ≡

```
pack .menubar.file .menubar.config  -side left -fill x
pack .menubar.help -side right -fill x
```

Macro defined by scraps 3c, 4abc, 5a.
Macro referenced in scrap 1.

3.5 Actions

The primary action from this script is to run the `nuweb` processor with the configuration options and target selected by the user.

We are going to write the normal and error output of the script to log files (via output redirection in the `exec` command). Because of this, As part of the initialization of this command script, we will delete the log file so it contains only the output from this run when they are written by redirecting the output of `nuweb`.

It is an oddity of `nuweb` that *all* of its output is written to standard error; for this reason only the standard error is redirected into the log file.

⟨Define the action buttons 5b⟩ ≡

```
button .go -text "Tangle/Weave" -command {
    file delete nuwebtcl.log
    file delete nuwebtcl.error
    if $noTeX {set t " -t "}   else {set t ""}
    if $noCode {set o " -o "}  else {set o ""}
    if $verbose {set v " -v "} else {set v ""}
    if $noTest {set c " -c "}  else {set c ""}
    if $numberScraps {set n " -n "} else {set n ""}

    catch {
      eval exec $nuweb_name \
        $t $o $v $c $n \
        \"$target\" \
        2> nuwebtcl.log
    }
  if [expr [file size nuwebtcl.log] > 0] {
```

```
        showFile nuwebtcl.log
        } else {
        file delete nuwebtcl.log
        }
    }
```

Macro defined by scraps 5b, 6ab.
Macro referenced in scrap 1.

Note the redirection of `nuweb`'s output to log files. This enables us to automatically display the error log if `nuweb` exited abnormally.

Once `nuweb` has returned, we examine the size of the redirected files. If the error file has non-zero length, we display it, using procedures stolen from the "Widget Tour". If it, or the log file, has zero length, we will delete it.

⟨Define the action buttons 6a⟩ ≡

```
button .showlog -text "Show Log" -command {
  showFile nuwebtcl.log
}
```

Macro defined by scraps 5b, 6ab.
Macro referenced in scrap 1.

Finally, we provide a button to exit from the application. This is not strictly necessary, since the "close" widget automatically provided by the window manager has the same functionality, but it seems appropriate.

⟨Define the action buttons 6b⟩ ≡

```
button .exit -text "EXIT" -command {exit}
```

Macro defined by scraps 5b, 6ab.
Macro referenced in scrap 1.

3.6 Help Display

This procedure displays the text stored in the variable "helptext" which is defined in the `nuweb` scrap "help text for display").

⟨Supporting procedures 6c⟩ ≡

```
proc showHelp {} {
     set helptext "⟨help text for display 9⟩"
     set w .text
     catch {destroy $w}
     toplevel $w
     wm title $w "Nuweb TCL/Tk Help"
     wm iconname $w "text"

     frame $w.buttons
     pack $w.buttons -side bottom -fill x -pady 2m
```

```
button $w.buttons.dismiss -text Dismiss -command "destroy $w"
pack $w.buttons.dismiss  -side bottom -expand 1

text $w.text -relief sunken -bd 2 -yscrollcommand "$w.scroll set" \
   -setgrid 1 -height 30 \
   -tabs {1c 2c 3c} \
   -wrap word
scrollbar $w.scroll -command "$w.text yview"
pack $w.scroll -side right -fill y
pack $w.text -expand yes -fill both
$w.text insert end $helptext
$w.text mark set insert 0.0 }
```

Macro defined by scraps 6c, 8ab.
Macro referenced in scrap 1.

3.7 Display The Interface

The actual display of the GUI is defined by the series of pack commands. Since the interface is actually quite simple, the pack commands needed to implement it are likewise not complicated.

⟨Display the GUI 7⟩ ≡

```
pack .menubar -side top -fill x
pack .pathname .browse .config -side top -fill x
pack .config.left .config.right -in .config -side left
pack .noTeX .noCode -in .config.left -side top -anchor w
pack .verbose .noTest .numberScraps -in .config.right \
  -side top -anchor w
pack .go .showlog .exit -side left -expand 1 -fill x
```

Macro referenced in scrap 1.

4 Supporting Procedures

The procedures shown here are relatively generic; they provide the general function of displaying a text file.

4.1 Show Files

These procedures take a single argument, the name of a text file, and display it in a toplevel window.

4.1.1 The "showFile" Procedure

This procedure was adapted from `text.tcl`, a demonstration script that creates a text widget that describes the basic editing functions, found in the "Widget

Tour" that is part of the Tool Command Language (TCL) distribution. Specifically, it is "text.tcl,v 1.2 1998/09/14 18:23:30 stanton Exp".

The script was adapted to turn it into a procedure and to arrange to fill it with text read from a file rather than with a text hard coded into the procedure. Additionally, the "Show Code" option was removed, since it was not relevant to this application.

⟨Supporting procedures 8a⟩ ≡

```
proc showFile {thefile} {
     set w .text
     catch {destroy $w}
     toplevel $w
     wm title $w "$thefile"
     wm iconname $w "text"

     frame $w.buttons
     pack $w.buttons -side bottom -fill x -pady 2m
     button $w.buttons.dismiss -text Dismiss -command "destroy $w"
     pack $w.buttons.dismiss  -side bottom -expand 1

     text $w.text -relief sunken -bd 2 -yscrollcommand "$w.scroll set" \
        -setgrid 1 -height 30
     scrollbar $w.scroll -command "$w.text yview"
     pack $w.scroll -side right -fill y
     pack $w.text -expand yes -fill both
     textLoadFile $w.text $thefile
     $w.text mark set insert 0.0 }
```

Macro defined by scraps 6c, 8ab.
Macro referenced in scrap 1.

4.1.2 The "textLoadFile" Procedure

This procedure is extracted from `search.tcl`, a demonstration script found in the "Widget Tour" distributed with TCL. Specifically, it is "Id: search.tcl,v 1.2 1998/09/14 18:23:30 stanton Exp".

⟨Supporting procedures 8b⟩ ≡

```
proc textLoadFile {w file} {
    if [file exists $file] {
      set f [open $file]
      $w delete 1.0 end
      while {![eof $f]} {
         $w insert end [read $f 10000]
      }
      close $f
  }
}
```

Macro defined by scraps 6c, 8ab.

Macro referenced in scrap 1.

No modifications to this procedure were needed for this application.

A User Help Text

This scrap defines the actual text to be displayed in the help window. Note that the entire scrap will be within quotes when it is used; the text defined here needs to conform to the Tcl syntax.

⟨help text for display 9⟩ ≡

```
\
OVERVIEW
This is a Tcl/Tk front end for the Nuweb literate programming \
processor. It takes a single target file and (depending on the \
command options supplied) produces a TeX source file and a set \
of code files defined in the target file.

TARGET FILE
The name of the target file is displayed in a text button near \
the top of the Nuweb.tcl window, with the path to that file \
displayed above it.  The nuweb process will execute in this \
directory, which means that all relative path names defined \
in the target file will be interpreted relative to this directory.
\tTo change the target file, click on the text button containing \
the target file name.  A file requester will appear; navigate to and \
select the new target file using the requester.  By default, the \
requester will show only files ending in \".w\", since this is the \
usual extension given to nuweb target files.
\tThe target file requester can also be summoned via the \"File\" \
menu. \n
USER OPTIONS\n
The user options for nuweb can be set by checking the option boxes \
displayed in the center of the GUI.  These options are:
\t - Suppress TeX Output: causes nuweb to omit the \"weave\" phase \
of execution.  This means that no TeX documentation file will be produced.
\t - Suppress Code Output: causes nuweb to omit the \"tangle\" phase \
of execution. This means that no code output files will be produced.
\t - Verbose Output: causes nuweb to write the names of the input and \
output files. This information (and any warning or error messages) are \
captured in a log file which is automatically displayed when nuweb \
finishes executing.  The log file can also be displayed by selecting the \
\"Show Log\" command button.
\t - Do Not Test For File Change: normally, nuweb tests to see that its \
output files have changed before actually writing them. This is intended \
to help with make file dependencies, where updating the file modification \
date can cause unnecessary compilation.  By checking this option, you can \
cause nuweb to overwrite the existing files whether they are changed or \
not.
```

```
\t - Number Scraps Consecutively: have nuweb number the scraps in the TeX \
documentation file with consecutive numbers, rather than by the page they \
are defined on.
TANGLE/WEAVE\n
To execute nuweb with the currently selected target file and options, click \
on the button labeled \"Tangle/Weave\".
CONFIGURATION\n
The Nuweb Tcl application assumes that the nuweb executable is found \
somewhere in the operating system path.  If this is not the case, the path \
to the nuweb executable can be set using the \"Config\" menu \"Nuweb\" \
option.
EXITING FROM THE APPLICATION\n
There are three different ways to exit from the Nuweb Tcl application:
\t - Click on the button marked \"Exit\"
\t - Select the \"Exit\" item on the \"File\" menu
\t - Click on the \"Close Window\" button provided by the operating \
system's window manager.
```

Macro referenced in scrap 6c.

Reference Material

Change History

Log : *nuwebtcl.w*, *v* Revision 1.3 2001/03/11 20:33:30 Mark Added user help. I'm not quite sure I understand how the text widget displays the resulting text—some of the formatting is not quite what I expected. But the help seems to be usable, which is enough for now.

Revision 1.2 2001/03/10 06:38:40 Mark Added a basic menu structure.

Corrected a potentially confusing behavior which caused the script to place the Nuweb outputs in the directory from which the script was invoked rather than the directory in which the target file resides.

Added protection against the impacts of directory hierarchies which contain spaces by adding quotation marks around the directory names.

Completed basic debugging and testing. This appears to be a functional version.

Acronyms

GUI Graphical User Interface

TCL Tool Command Language

References

[1] John K. Ousterhout. *Tcl and the Tk Toolkit*. Reading, MA: Addison-Wesley, 1994.

[2] Paul Raines and Jeff Tranter. *TCL/TK in a Nutshell*. Sebastopol CA: O'Reilly & Associates, 1999.

[3] Sun Microsystems, Inc. "Tk widget demonstration". Distributed with TCL 8.3, copyright 1996-1997.