

Re: Redirect file output into variables

Source: <http://coding.derkeiler.com/Archive/Tcl/comp.lang.tcl/2003-10/1145.html>

From: Michael Schlenker (*schlenk_at_uni-oldenburg.de*)

Date: 10/22/03

Date: Wed, 22 Oct 2003 20:29:41 +0200

Markus Elfring wrote:

>>Can you explain more? Opening files and redirecting their
>>contents to variables, lists, or arrays is done all the time
>>by Tcl programmers. Same is true for writing vars, etc. to
>>files. Or do you mean something else?
>
>
> Did you read my first request for this topic?
>
> I am thinking about the following application.
> A limited execution environment will be created that forbids to
> directly use standard file channels. For example, I must avoid that
> output is written to STDOUT by calls to the TCL function "puts" before
> the own data processing is finished. But these calls need a place to
> store the output.
> - How can the data be stored or redirected to a temporary file?
> or
> - All file system calls will be intercepted to store the data into a
> buffer that can be copied to a variable later.

Hmm, i would use a safe interpreter for that..., as you say limited execution environment...

```
proc buffered_puts { args } {
    global buffer
    switch -exact [llength $args] {
        0 { # handle error
            1 {
                append buffer(stdout) "[lindex $args 0]\n"
            }
            2 {
                if {[lindex $args 0] eq "-nonewline"} {
                    append buffer(stdout) "[lindex $args 1]"
                } else {
                    # check for error
                    append buffer([lindex $args 0]) "[lindex $args 1]\n"
                }
            }
        }
    }
}
```

comp.lang.tcl: Re: Redirect file output into variables

```
    }  
3 {  
    if {[lindex $args 0] eq "-nonewline"} {  
        # check error  
        append buffer([lindex $args 1]) "[lindex $args 2]"  
    } else {  
        # check for error  
        append buffer([lindex $args 1]) "[lindex $args 2]\n"  
    }  
    }  
  
    default {  
        # handle error  
    }  
}  
}
```

```
interp create safe foo  
interp alias foo puts {} buffered_puts  
# do similar things with read, gets etc.
```

```
set script "script you want to execute in a safe environment"  
foo eval $script
```

```
# print the buffered output
```

Michael Schlenker