```
--- ng/server/news_code.tcl     2025-10-30 14:08:23.600168869 +0000
+++ ng2/server/news_code.tcl    2025-12-10 10:18:14.734682056 +0000
@@ -29,7 +29,7 @@
 # a little debugging helper
 proc printvars args {
     set now [clock microseconds]
-    set elapsed [expr {$::printvars_time ? $now - $::printvars_time : 0}]
+    set elapsed [= ::printvars_time ? now - ::printvars_time : 0]
     set ::printvars_time $now
     puts -nonewline "[format %8d $elapsed] "

@@ -259,11 +259,11 @@
     if {rand() < 0.2} {
         # Files hex[0-4] are gzipped containing various repetitions of
         # "Don't mention it. ++???++ Out of Cheese Error. Redo From Start."
-        set file hex[expr {[string length $suffix] % 5}]
+        set file hex[= [string length $suffix] % 5]
         Httpd_ReturnFile $sock $mimetype htdocs/$file
     } elseif {rand() < 0.2} {
         # generate and return a random amount (<=100kB) of random crap
-        exec head -[expr {int(rand()*100000)}]c /dev/urandom > htdocs/random
+        exec head -[= int(rand()*100000)]c /dev/urandom > htdocs/random
         Httpd_ReturnFile $sock $mimetype htdocs/random
     } else {
         # return nothing, but log the request
@@ -354,7 +354,7 @@

     # Generate random cookie for user, write to their db record
     set salt [string range [clock clicks] end-7 end]
-    set cookie [md5crypt::md5crypt [expr {rand()}] $salt]
+    set cookie [md5crypt::md5crypt [= rand()] $salt]
     userdb eval {UPDATE users SET cookie = $cookie WHERE num = $user}
     tailcall finish_login $cookie 1 $suffix
 }
@@ -384,7 +384,7 @@
     package require md5crypt
     # Generate cookie for user, write a temporary db record for them
     set salt [string range [clock clicks] end-7 end]
-    set cookie [md5crypt::md5crypt [expr {rand()}] $salt]
+    set cookie [md5crypt::md5crypt [= rand()] $salt]
     userdb eval {INSERT INTO users(cookie) VALUES($cookie)}
     tailcall finish_login $cookie 0 $suffix
 }
@@ -566,8 +566,8 @@
     } else {
         set pattern *$text*
     }
-    set match_name [expr {$name == 1}]
-    set match_desc [expr {$desc == 1}]
+    set match_name [= name == 1]
+    set match_desc [= desc == 1]
     set groups [groups_matching $pattern $match_name $match_desc]

     html {
@@ -685,7 +685,7 @@
     }

     foreach group $group_list reads $read_list {
-        set percent [expr {100 * $reads / $total_reads}]
+        set percent [= 100 * reads / total_reads]
        lassign [groupstat $group] desc

         html "<tr><td><a href=/$group>$group</a></td><td>[enpre $desc]</td><td alig
n='right'>$percent%</td></tr>\n"
@@ -829,7 +829,7 @@
 <tbody>
     }

-    set looking_for_new [expr {! $prev_thread}]
```

```
+    set looking_for_new [= ! prev_thread]
     foreach {start_num thread} $threads {thread_posts new_posts replies} $threadcou
nts {
         lassign [dict get $hdrs $start_num] prev sub frm tim id
         set sub [encoding convertfrom $sub]
@@ -878,10 +878,10 @@
     }
     html "<form action='/' method='post' style='display: inline'>"

-    set posts [expr {[llength $hdrs] / 2}]
+    set posts [= [llength $hdrs] / 2]
     if {$posts >= $::postbatch} {
         html {<input type=submit value='Older Posts' class='but' }
-        html "formaction='/$group/older/[expr {$older+$::postbatch}]' />\n"
+        html "formaction='/$group/older/[= older+::postbatch]' />\n"
     }

     html "</form>\n"
@@ -907,7 +907,7 @@
     lassign $urec user can_post params

     set reverse [dict get $params rev]
-    set reverse [expr {$reverse==0 ? 1 : 0}]
+    set reverse [= reverse==0 ? 1 : 0]
     dict set params rev $reverse

     userdb eval {UPDATE users SET params = $params WHERE num = $user}
@@ -1000,7 +1000,7 @@
     set prev 0
     set pv 0
     foreach {num indent} $thread {
-        set new [expr {$num > $old_last}]
+        set new [= num > old_last]
         if {$new && $target == 0} {set target $num}
         lappend nns $new $num
        if {$num == $target} {set pv $prev}
@@ -1056,8 +1056,8 @@
     foreach {num indent} $thread {
         if {$indent < $prev_ind} {
             html {<tr style='height:6px'>} \
-               "<td colspan='[expr {max(30-1-$indent,1)}]' class='rb'></td>" \
-               [string repeat {<td class='r'></td>} [expr {$indent+1}]] "</tr>\n"
+               "<td colspan='[= max(30-1-indent,1)]' class='rb'></td>" \
+               [string repeat {<td class='r'></td>} [= indent+1]] "</tr>\n"
        }

         set frag " id='a$num'"
@@ -1075,7 +1075,7 @@
         if {$prev_ind==0} {
            append frag " style='border: solid 3px'"
         }
-        html "<tr><td$frag colspan='[expr {30-$indent}]' class='$clas' >"
+        html "<tr><td$frag colspan='[= 30-indent]' class='$clas' >"

         lassign [dict get $hdrs $num] prev sub frm tim id
         set frm [encoding convertfrom $frm]
@@ -1362,7 +1362,7 @@
         }
         set pre_txt [string range $line $start $t0-1]
         set tok_txt [string range $line $t0 $t1]
-        set start [expr {$t1 + 1}]
+        set start [= t1 + 1]
         lappend ::tokens $pre_txt $tok $tok_txt
     }
     lappend ::tokens [string range $line $start end] 0 {}
@@ -1519,7 +1519,7 @@
 foreach i $ins o $outs {
     foreach b [list $A $a] {
         foreach c [list $i $o] {
```

```
-            lappend rot13map [binary format c [expr {$b + $c}]]
+            lappend rot13map [binary format c [= b + c]]
         }
     }
 }
@@ -1532,7 +1532,7 @@
     lassign $urec user can_post params

     set value [dict get $params $pref]
-    set value [expr {$value==1 ? 0 : 1}]
+    set value [= value==1 ? 0 : 1]
     dict set params $pref $value

     userdb eval {UPDATE users SET params = $params WHERE num = $user}
@@ -1795,7 +1795,7 @@

     set threadcounts {}
     foreach {start_num thread} $threads {
-        set thread_posts [expr {[llength $thread] / 2}]
+        set thread_posts [= [llength $thread] / 2]
         set new_posts 0
         set replies 0
         foreach {num indent} $thread {
@@ -1831,7 +1831,7 @@
 }

 proc bypassThreadinfo {group timeout} {
-    set expiry [expr {[clock seconds] + $timeout}]
+    set expiry [= [clock seconds] + timeout]
     tsv::set ThreadinfoBypass $group $expiry
 }

@@ -1872,7 +1872,7 @@
 # Periodically we do Garbage Collection of old/dead cache entries
 set cacheMaxAge 3600 ;# 1 hour in seconds
 proc cacheGC {} {
-    set agelimit [expr {[clock seconds] - $::cacheMaxAge}]
+    set agelimit [= [clock seconds] - ::cacheMaxAge]

     # remove old Stash entries
     foreach {key value} [tsv::array get Stash] {
@@ -1888,7 +1888,7 @@
     }
 }

-set cacheGCinterval [expr {$cacheMaxAge * 1000}]
+set cacheGCinterval [= cacheMaxAge * 1000]
 after $cacheGCinterval doCacheGC
 proc doCacheGC {} {
     #puts "Running cacheGC"
@@ -2044,7 +2044,7 @@
         set frm [convertit $frm]
         lappend hdrs $dat [list 0 $sub $frm $dat $msgid]
     }
-    set more_dat [expr {[llength $hdrs]==$::postbatch*2 ? [lindex $hdrs end 3] : 0}
]
+    set more_dat [= [llength $hdrs]==::postbatch*2 ? [lindex $hdrs end 3] : 0]
     return [list $more_dat $hdrs]
 }

@@ -2205,7 +2205,7 @@
                 before you can post to a moderated group.</em>"
         }
     }
-    set uucp [expr {! $moderated}]
+    set uucp [= ! moderated]

     if {[spam_check $user $body $grouplist 1]} {
         return "<br/><em>Sorry, this post would exceed the maximum number\
```

```
@@ -2561,17 +2561,17 @@
      }

      # ok to proceed
-     set new_expiry_time [expr {max($now+$timeout,$user_prev_time)}]
+     set new_expiry_time [= max(now+timeout,user_prev_time)]
      redis zadd morphers $new_expiry_time $user
      return 0
  }
-set day_secs [expr {60*60*24}];# 24 hours in seconds
-set week_secs [expr {$day_secs*7}];# 7 days in seconds
+set day_secs [= 60*60*24];# 24 hours in seconds
+set week_secs [= day_secs*7];# 7 days in seconds

  set morph_limit 0
  #set morph_limit 5
  set morph_timeout $day_secs
-set morph_reserve [expr {60*15}] ;# 15 minutes in seconds
+set morph_reserve [= 60*15] ;# 15 minutes in seconds

  # Are any of these groups moderated?
  proc moderated groups {
@@ -2598,7 +2598,7 @@

      # cross-posts count as multiple posts
      set group_count [llength $grouplist]
-     set new_volume [expr {[string length $body] * $group_count}]
+     set new_volume [= [string length $body] * group_count]
      if {$new_volume > $::spam_max_volume} {return 1}

      # if posting, update their totals
```