MasterIT - Scripting Es4

Tcl/Tk

Sandro.Angius@lnf.infn.it

4/12/2002

crono.tcl

- Semplice cronometro con tempi parziali
  - Gestione:
    - Eventi
    - Button
    - RadioButton

```
wm title . "Tcl Crono"

#wm resizable . 0 0

frame  .m -bd 5 -relief groove

# Definizione Menu
menubutton .m.mfile -text File -menu .m.mfile.menu -relief ridge -width 16
menu .m.mfile.menu -tearoff 0

.m.mfile.menu add command -label Quit -accelerator Ctrl+Q -command "exit 0"

menubutton .m.mcmd -text "Commands..." -menu .m.mcmd.menu -relief ridge  -width 16
menu .m.mcmd.menu -tearoff 1

.m.mcmd.menu add command -label "Start Crono"  \
                    -accelerator Alt+S -command { setstart }
.m.mcmd.menu add command -label "Lap Time"     \
                    -accelerator Alt+L -command { setlapstop tlap }
.m.mcmd.menu add command -label "Stop Crono"   \
                    -accelerator Alt+S -command { setlapstop tstop }
.m.mcmd.menu add command -label "Clicks Mode"  \
                    -accelerator Alt+C -command { set mode 0 }
.m.mcmd.menu add command -label "Seconds Mode" \
                    -accelerator Alt+T -command { set mode 1 }
```

Crono.tcl - Pag. 1/5

```
# Definizione struttura di I/O

label .m.s1 -text " "
label .m.lclock -text "Program Started on:"
label .m.lstart -text "Start Time:"
label .m.llap   -text "Lap Time:"
label .m.lstop  -text "Stop Time:"
label .m.lmode  -text "Elapsed Time Mode:”
entry .m.clock   -width 60 -textvariable clock  -takefocus 0 -state disabled
entry .m.tstart  -width 60 -textvariable tstart -takefocus 0 -state disabled
entry .m.tlap    -width 60 -textvariable tlap   -takefocus 0 -state disabled
entry .m.tstop   -width 60 -textvariable tstop  -takefocus 0 -state disabled
label .m.s2 -text " "
radiobutton .m.mode0 -text "Clicks"  -variable mode -value 0
radiobutton .m.mode1 -text "Seconds" -variable mode -value 1
label .m.s3 -text " ”
button .m.start  -text "Start" -command { setstart }
button .m.lap    -text "Lap"   -command { setlapstop tlap  }
button .m.stop   -text "Stop"  -command { setlapstop tstop }
label .m.s4 -text " "


# Attiva gli "accelerator" precedentemente definiti

bind . <Control-q> { exit 0 }
bind . <Alt-s> { flip }
bind . <Alt-l> { setlapstop tlap }
bind . <Alt-c> { set mode 0 }
bind . <Alt-t> { set mode 1 }
```

```
# Mostra gli "objects" definiti

grid .m

grid .m.mfile  -row 0 -column 0
grid .m.mcmd   -row 0 -column 1

grid .m.s1     -row 1 -column 0 -columnspan 3

grid .m.lclock -row 2 -column 0
grid .m.lstart -row 3 -column 0
grid .m.llap   -row 4 -column 0
grid .m.lstop  -row 5 -column 0
grid .m.clock  -row 2 -column 1 -columnspan 2
grid .m.tstart -row 3 -column 1 -columnspan 2
grid .m.tlap   -row 4 -column 1 -columnspan 2
grid .m.tstop  -row 5 -column 1 -columnspan 2

grid .m.s2     -row 6 -column 0 -columnspan 3

grid .m.lmode  -row 7 -column 0
grid .m.mode0  -row 7 -column 1
grid .m.mode1  -row 7 -column 2

grid .m.s3     -row 8 -column 0 -columnspan 3

grid .m.start  -row 9 -column 0
grid .m.lap    -row 9 -column 1
grid .m.stop   -row 9 -column 2
grid .m.s4     -row 10
```

```
# Evento "Alt-S" determina stop o start

proc flip {} {
    global tstop
    if {$tstop == ""} {
        setlapstop tstop
    } else {
        setstart
    }
}

# Start cronometro
proc setstart {} {

    global tstart
    global tlap
    global tstop
    global ssec
    global sclicks
    global mode

    set ssec    [clock seconds]
    set sclicks [clock clicks]
    set tstart [clock format  $ssec -format "%A, %d %B %Y - %H:%M:%S"]
    if {! $mode} {
        append tstart " ($sclicks Clicks)"
    }
    set tlap  ""
    set tstop ""
}
```

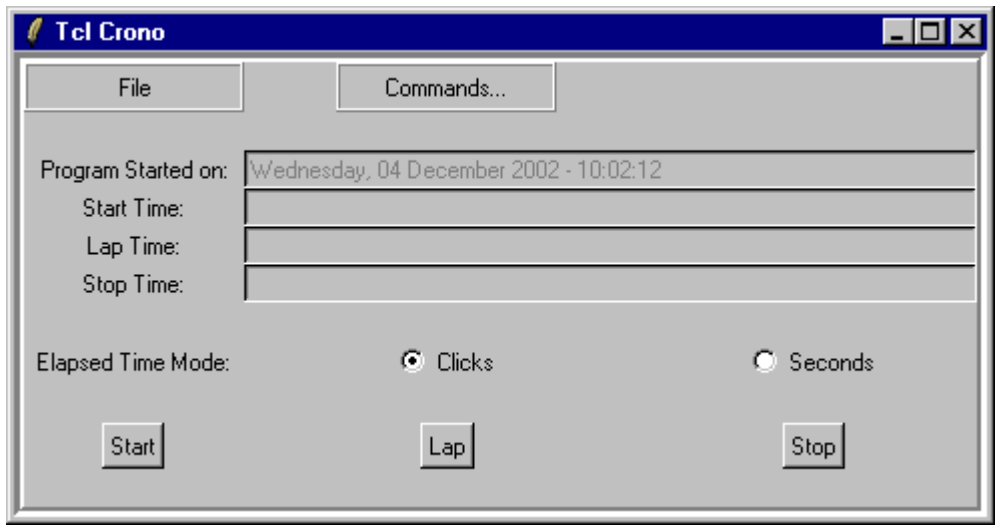
Crono.tcl - Pag. 4/5

```
# Tempo intermedio o stop del cronometro
proc setlapstop { vname } {

    global tstart
    global tlap
    global tstop
    global ssec
    global sclicks
    global mode

    if { ($tstart != "") && ($tstop == "") } {
        set nsec    [clock seconds]
        set nclicks [clock clicks]
        set $vname  [clock format  $nsec -format "%A, %d %B %Y - %H:%M:%S"]
        append $vname " (+"
        if {$mode} {
            append $vname [expr $nsec - $ssec]
            append $vname " Seconds)"
        } else {
            append $vname [expr $nclicks - $sclicks]
            append $vname " Clicks)"
        }
    }
}

# Fase di inizializzazione
set mode 0
set tstop " "
set bsec  [clock seconds]
set clock [clock format $bsec -format "%A, %d %B %Y - %H:%M:%S"]
```

Crono.tcl - Pag. 5/5

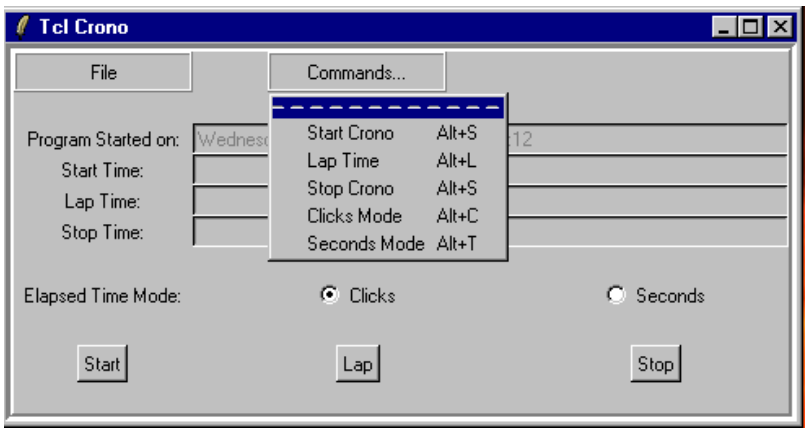
crono.tcl

- Cosí come Perl e Python anche TclTk puo' essere usato su sistemi Windows:



crono.tcl

- Il “tearoff 1” permette di creare una finestra separata per i comandi di un menu:



ipconv.tcl

- Conversione di numeri ip in decimale, esadecimale e binario
  - Gestione:
    - Eventi
    - Button
    - Entry

```
wm title . "IP Conversion"

#wm resizable . 0 0

frame  .ip -bd 5 -relief groove

# Definizione Menu
menubutton .ip.mfile -text File -menu .ip.mfile.menu -relief ridge -width 16
menu .ip.mfile.menu -tearoff 0

.ip.mfile.menu add command -label Quit -accelerator Ctrl+Q -command "exit 0"
menubutton .ip.mbmask -text "Bit Mask (1..31)" -menu .ip.mbmask.menu -relief ridge  -width 16
menu .ip.mbmask.menu -tearoff 1

for {set cnt 1} {$cnt < 32} {incr cnt} {
    set txt [format "%2s Bit" $cnt]
    if {$cnt > 1} { append txt "s" }
    .ip.mbmask.menu add command -label $txt -command "breset $cnt"
}

menubutton .ip.mother -text "Other..." -menu .ip.mother.menu -relief ridge -width 16
menu .ip.mother.menu -tearoff 1
.ip.mother.menu add command -label "Reverse Bits"     -accelerator Alt+R -command "toggleall"
.ip.mother.menu add command -label "/0  (Clear Bits)" -accelerator Alt+0 -command "breset 0"
.ip.mother.menu add command -label "/8  (Class A)"    -accelerator Alt+A -command "breset 8"
.ip.mother.menu add command -label "/16 (Class B)"    -accelerator Alt+B -command "breset 16"
.ip.mother.menu add command -label "/24 (Class C)"    -accelerator Alt+C -command "breset 24"
```

Ipconv.tcl - Pag. 1/8

```
label  .ip.iplabel  -text "IP:"           -width 16
label  .ip.hexlabel -text "IP (Hex):"     -width 16
label  .ip.binlabel -text "IP (Binary):"  -width 16
label  .ip.declabel -text "IP (Decimal):" -width 16
entry  .ip.decinput -textvariable decinput -width 16
button .ip.decpiu   -text "-" -command { DecIncr -1 } -width 3
button .ip.decmeno  -text "+" -command { DecIncr +1 } -width 3

grid  .ip
grid  .ip.mfile    -row 0 -column  0
grid  .ip.mbmask   -row 0 -column  2 -columnspan 10
grid  .ip.mother   -row 0 -column 13 -columnspan  8
grid  .ip.iplabel  -row 2 -column  0
grid  .ip.hexlabel -row 3 -column  0
grid  .ip.binlabel -row 4 -column  0
grid  .ip.declabel -row 5 -column  0
grid  .ip.decinput -row 5 -column  1 -columnspan  8
grid  .ip.decpiu   -row 5 -column 10 -columnspan  4
grid  .ip.decmeno  -row 5 -column 14 -columnspan  4

bind . <Control-q> { exit 0 }
bind . <Alt-r> { toggleall }
bind . <Alt-0> { breset  0 }
bind . <Alt-a> { breset  8 }
bind . <Alt-b> { breset 16 }
bind . <Alt-c> { breset 24 }
bind .ip.decinput <KeyRelease> {CheckDec}
```

Ipconv.tcl - Pag. 2/8

```
# Definizione, Attivazione, Bind dei campi Dotted Decimali, Esadecimali e Binari

set ccol 1
set bcol 1

for {set cnt 0} {$cnt < 4} {incr cnt} {
    entry .ip.byte$cnt -textvariable byte$cnt -width 16
    entry .ip.hex$cnt  -textvariable hex$cnt  -width 16
    grid  .ip.byte$cnt -row 2 -column $ccol -columnspan 8
    grid  .ip.hex$cnt  -row 3 -column $ccol -columnspan 8
    bind  .ip.byte$cnt <KeyRelease> {CheckIP "byte" "%%e%%s"}
    bind  .ip.hex$cnt  <KeyRelease> {CheckIP "hex"  "%%x%%s"}
    incr ccol 8
    for {set bits 0} {$bits < 8} {incr bits} {
        set cbit bit[expr $bits + 8 * $cnt]
        set $cbit 0
        entry .ip.$cbit -width 1 -textvariable $cbit -takefocus 0 -state disabled
        grid  .ip.$cbit -row 4 -column $bcol
        bind  .ip.$cbit <ButtonPress>  { toggle %W }
        incr bcol
    }
    if {$cnt < 3} {
        label .ip.ipmark$cnt  -text "." -width 2
        label .ip.hexmark$cnt -text "." -width 2
        label .ip.binmark$cnt -text "." -width 2
        grid .ip.ipmark$cnt  -row 2 -column $ccol
        grid .ip.hexmark$cnt -row 3 -column $ccol
        grid .ip.binmark$cnt -row 4 -column $bcol
        incr ccol
        incr bcol
    }
}
```

Ipconv.tcl - Pag. 3/8

```
# Complementa il bit $inbit
proc toggle {inbit} {
    upvar [string range $inbit 4 end] cbit
    set cbit [ expr ($cbit + 1) % 2 ]
    UpdateByBin
}

# Complementa tutti i bit
proc toggleall {} {
    for {set cnt 0} {$cnt < 32} {incr cnt} {
        upvar bit$cnt cref
        set cref [ expr ($cref + 1) % 2 ]
    }
    UpdateByBin
}

# Reset degli ultimi (32 - $cnt) bits
proc breset {nbit} {
    for {set cnt $nbit} {$cnt < 32} {incr cnt} {
        global bit$cnt
        set bit$cnt 0
    }
    UpdateByBin
}

# "Incrementa" o "Decrementa" il numero IP
proc DecIncr {indata} {
    global decinput
    set decinput [expr "${decinput}.0" + "${indata}.0"]
    CheckDec
}
```

Ipconv.tcl - Pag. 4/8

```
# Controlla l'input decimale
proc CheckDec {} {
    global decinput
    set decinput [CheckNum "%e%s" $decinput 4294967295]
    UpdateBin
    UpHexAndByte
}


# Aggiorna le entry "binarie"
proc UpdateBin {} {

    global decinput
    global bit0

    if { "${decinput}.0" > 2147483647.0 } {
        set bit0 1
        set indata [expr int("${decinput}.0" - 2147483648.0)]
    } else {
        set bit0 0
        set indata $decinput
    }

    for {set cnt 31} {$cnt > 0} {incr cnt -1} {
        global bit$cnt
        set bit$cnt [ expr $indata % 2 ]
        set indata  [ expr int($indata/2) ]
    }
}
```

Ipconv.tcl - Pag. 5/8

```
# Aggiorna le entry dotted decimali e esadecimali
proc UpHexAndByte {} {

    for {set byte 0} {$byte < 4} {incr byte} {
        global hex$byte
        global byte$byte
        set parz 0
        for {set bit 0} {$bit < 8} {incr bit} {
            set cbit [expr 8*$byte + $bit]
            upvar #0 bit$cbit vbit
            set parz [expr ($parz << 1) + $vbit]
        }
        set byte$byte $parz
        set hex$byte  [format "%02X" $parz]
    }
}

# Controlla input dotted decimali e esadecimali
proc CheckIP {tgt format} {

    for {set byte 0} {$byte < 4} {incr byte} {
        global byte$byte
        global hex$byte
        upvar #0 $tgt$byte ipbyte
        set vtmp [expr int([CheckNum $format $ipbyte 255])]
        set byte$byte $vtmp
        set hex$byte [format "%02X" $vtmp]
    }
    UpdateDec $byte0 $byte1 $byte2 $byte3
    UpdateBin
}
```

Ipconv.tcl - Pag. 6/8

```
# Aggiorna valore "decimale" da dotted decimali o esadecimali
proc UpdateDec {x0 x1 x2 x3} {
    set base 256.0
    set iptmp [expr $base * ($x2 + $base * ($x1 + $base * $x0)) + $x3]
    global decinput
    set decinput [string range $iptmp 0 [expr [string length $iptmp] - 3]]
    return $decinput
}

# Controlla limiti e validita' campi numerici decimali e esadecimali
proc CheckNum {format num max} {

    set flag [scan $num $format vtmp dummy]
    if { $flag < 1 } {
        bell
        set vtmp 0.0
    } else {
        if { $flag > 1 } {
            bell
        }
        if { $vtmp < 0 } {
            bell
            set vtmp 0.0
        }
        if { $vtmp > "${max}.0" } {
            bell
            set vtmp "${max}.0"
        }
        set vtmp [expr double($vtmp)]
    }
    return [string range $vtmp 0 [expr [string length $vtmp] - 3]]
}
```

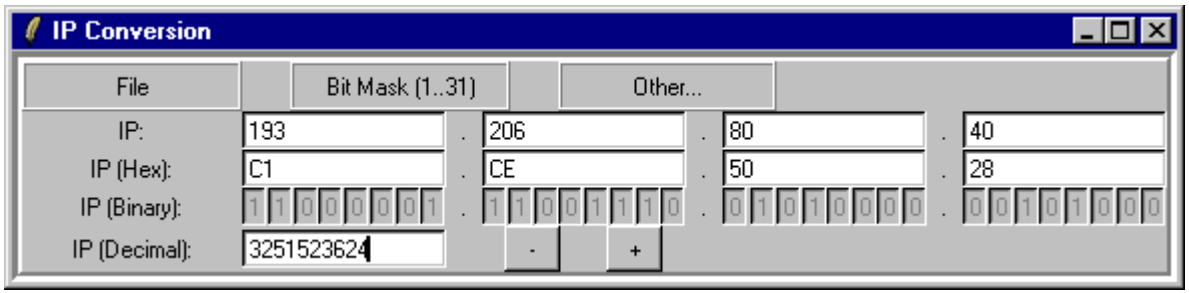
Ipconv.tcl - Pag. 7/8

```
# Aggiorna tutti i campi usando le entry binarie
proc UpdateByBin {} {

    for {set byte 0} {$byte < 4} {incr byte} {

        set tmp 0

        for {set bit 0} {$bit < 8} {incr bit} {
            set bptr [expr (8 * $byte) + $bit]
            upvar #0 bit$bptr cbit
            set tmp [expr ($tmp << 1) + $cbit]
        }

        global byte$byte
        global hex$byte
        set byte$byte $tmp
        set hex$byte [format "%02X" $tmp]
    }

    UpdateDec $byte0 $byte1 $byte2 $byte3
}

# Inizializzazione
set decinput 0
UpdateBin
UpHexAndByte
```

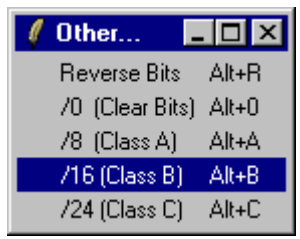
Ipconv.tcl - Pag. 8/8

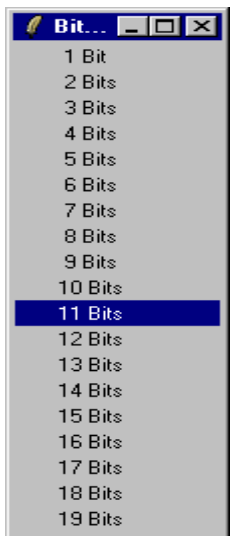
ipconv.tcl

- Window principale:



- Menu Windows (tearoff 1):





Bibliografia

- http://www.tcl.tk/doc/
- TclTk package: http://www.tcl.tk/software/
- Esempi sw: http://resource.tcl.tk/resource/
- Copia degli scripts si trova in: /afs/lnf.infn.it/project/master.it/doc/Scripting/EsScripts/