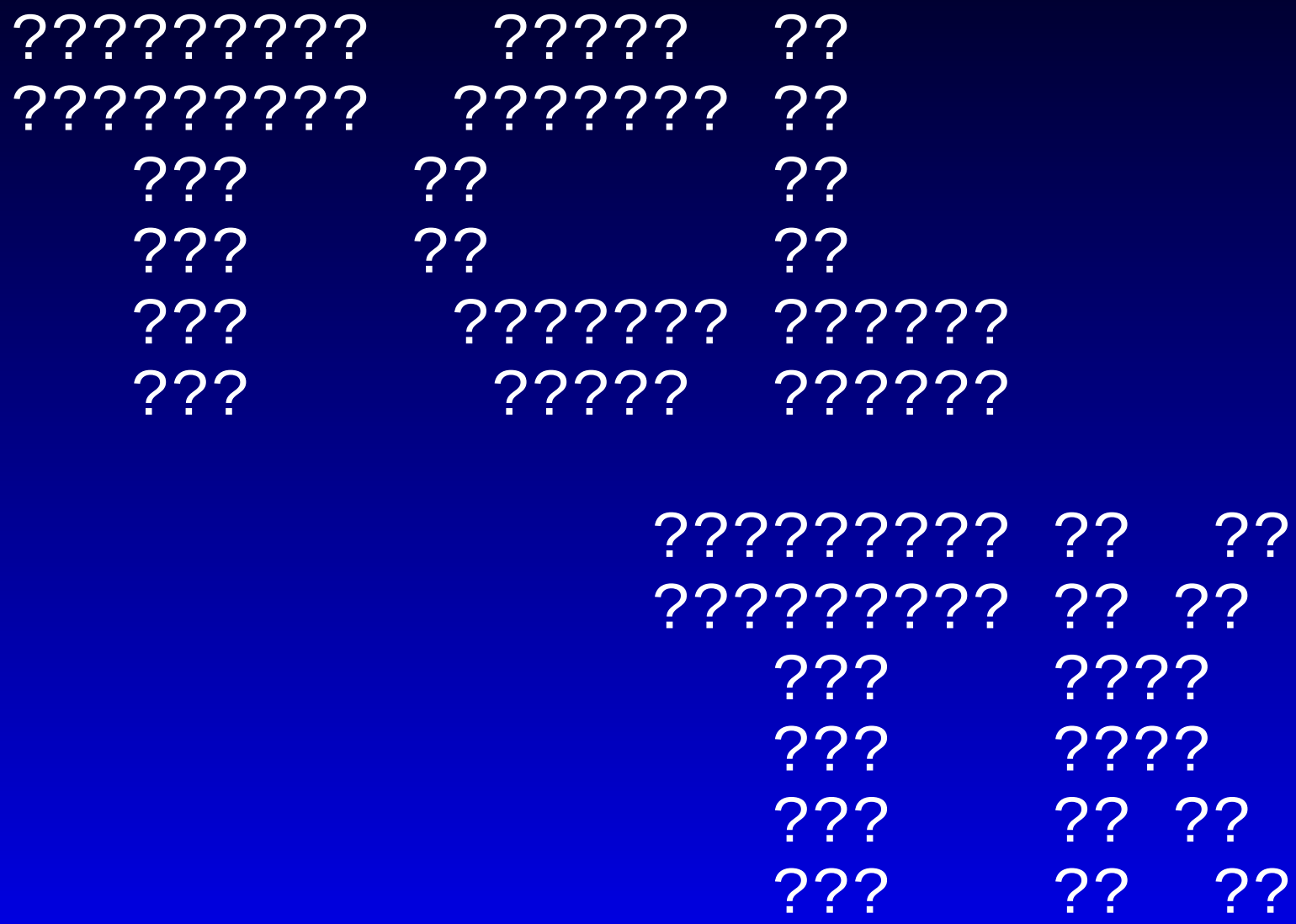
Tcl/TK és a Prolog

Hegyi Péter

hegyi@tmit.bme.hu



Tool Command Language

ToolKit

Mit kell tudni róla?

- gép és platform független
- interpreter alapú (script nyelv)
- GUI készítésre való
- egyebekkel bővíthető (pl. TCP/IP)
- `http://www.tcl.tk`

TCL

- „csak” egy script nyelv
- testvérei
  - Perl
  - UNIX C-shell
- DE bővíthető!

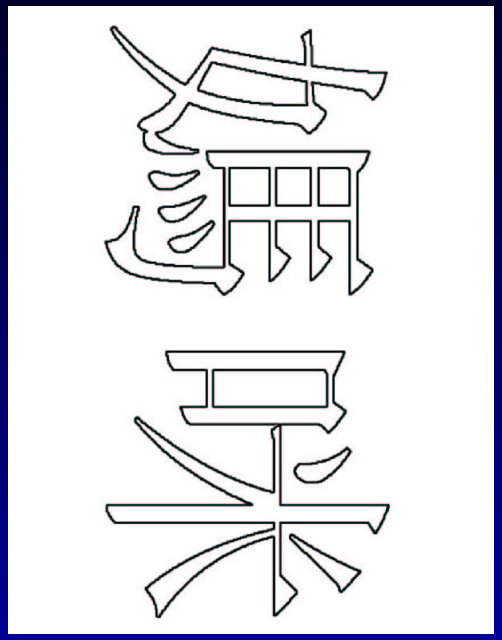
TCL futtatás

tclsh

- interaktív
- interinaktív

Speciális karakterek

- $ változók
- ; következő parancs
- # megjegyzés
- \
  - speciális karaktereknek
  - sorfolytatás
- [] azonnal végrehajtandó
- {}
  - nincs helyettesítés (kiv. újsor)
  - később értékeli ki
- "" semmit nem helyettesít



TCL szintaxis

szeparátorok: szóköz, soremelés, TAB

formátum: parancs [argumentumok]

Fordítás

1. szavakra bontás
2. elemzés, helyettesítések (ahol kell!)
   - balról jobbra
   - 1x olvas, 1x helyettesít!
3. végrehajtás
   - sorban első szót
   - a többi argumentum

Helyettesítés

```
set x 1
set a x
set b $$a
```

b = ?

Helyettesítés

```
set x 1
set a x
set b $$a
```

b = $x

Változók

- dinamikus
- set/unset
- asszociatív tömb: `uid(zoli) uid(peter)`
- műveletek: `incr, append...`

Kifejezések – *expr*

```
set uid(root) 1000
set uid(guest) 200
set login guest
set a [expr $uid($login)+1]
```

a=201

Listák

- elemek közt szóköz
- {}-k között
- lista is lehet elem:

```
lindex { a b { c d e } f } 2 ⇒
                              c d e
```

- `lindex, concat, llength, lappend, linsert, lreplace, lrange, lappend, split, join ...`

Stringek

- *string* parancs és alparancsai:
  `index, range ...`
- egyebek: `format, scan`

Vezérlési szerkezetek

- szokásosak
  - if–then–elseif
  - switch
  - while
  - for, foreach
  - break, continue
- DE ∀ {} között van:

```
for { set a 0 } { a < 5 } { incr a }
{ puts a }
```

Eljárások

`proc` *`eljárásnév { változók }`*
`{ függvénytörzs }`

- `return`
- `global`

Hibakezelés

- `catch` *`parancs változó`*
- `error` *`üzenet errorInfo hibakód`*

Állománykezelés

- szokásos parancsok:
  `open, gets, puts, seek, cd, pwd`
- állományleíró nélkül `stdin/stdout` a cél

TK (ToolKit)

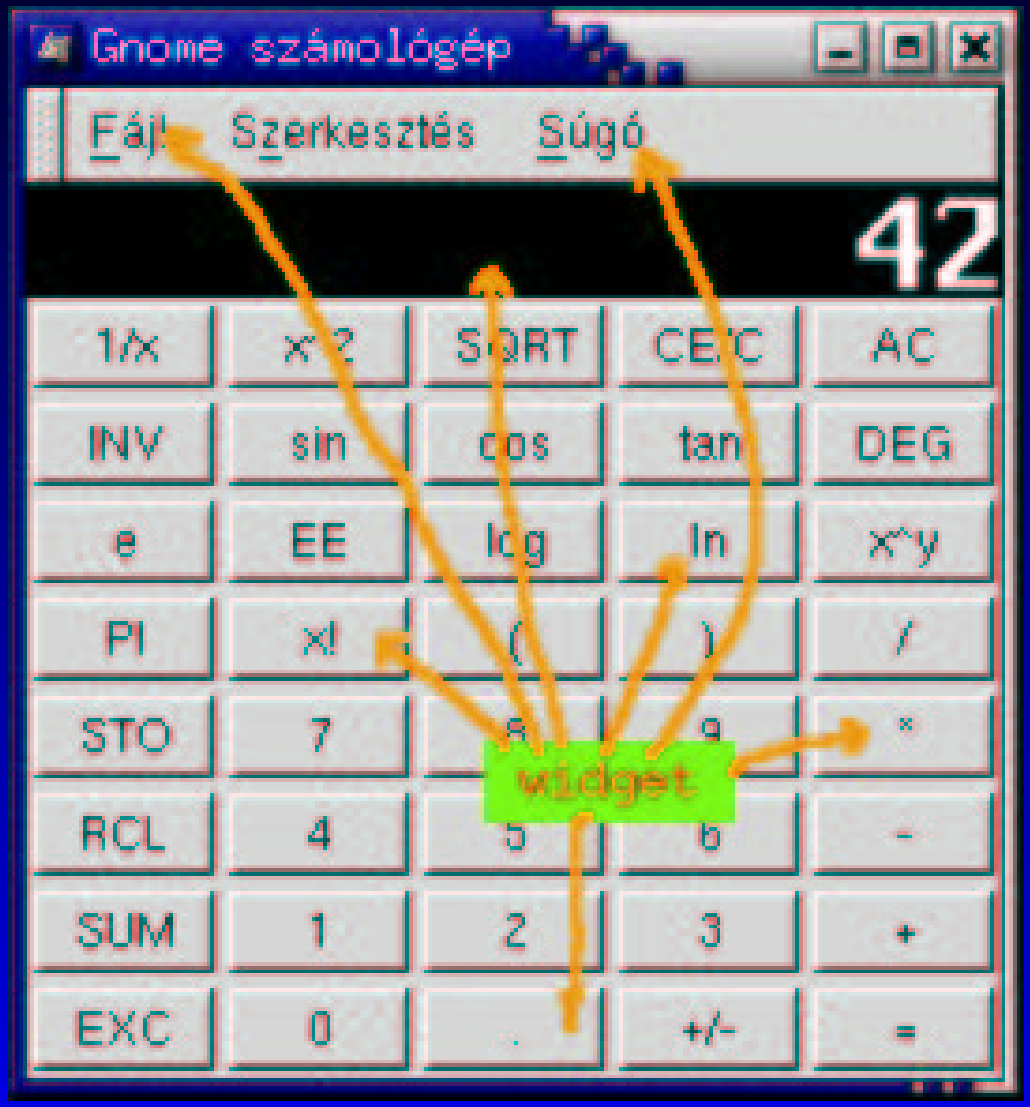
- TCL bővítése
- GUI készítéshez
- script nyelv ez is!!

- `/usr/lib/tk8.3/demos`

Widget


Gnome számológép
Fájl Szerkesztés Súgó
42
1/x x^2 SQRT CE/C AC
INV sin cos tan DEG
e EE log ln x^y
PI x! ( ) /
STO 7 8 9 *
RCL 4 5 6 -
SUM 1 2 3 +
EXC 0 . +/- =

Widget


Gnome számológép
Fájl
Szerkesztés
Súgó
42
1/x
x^2
SQRT
CE/C
AC
INV
sin
cos
tan
DEG
e
EE
log
ln
x^y
PI
x!
(
)
/
STO
7
8
9
*
RCL
4
5
6
-
SUM
1
2
3
+
EXC
0
.
+/-
=
widget

Widgetek

- keretek
- gombok
- rádiógombok
- checkboxok
- címkék
- menük
- vászon
- csúszka
- beviteli mezők
- …

Widgetek – neveik TCL-ben

- keretek – `frame`
- gombok – `button`
- rádiógombok – `radiobutton`
- checkboxok – `checkbutton`
- címkék – `label`
- menük – `menu`
- vászon – `canvas`
- csúszka – `scrollbar`
- beviteli mezők – `entry`
- …

Widgetek – Demo

- `/usr/lib/tk8.3/demos/widget`
- példa gombokra és beviteli mezőkre

Felépítés

- fastruktúra
- tetején a *main window*
- *main window* kihelyezése automatikus
- külön ablakok is gyerekek

Nevek

- fastruktúrában
- UNIX-szerűen
- /=.

Alkossunk!

1. készítsük el a widgetet:
   widgettípus elhelyezés -opciónév opcióérték ...
   `button .b -text "Halihó!" -fg red`
2. helyezzük el!
   `place/pack` ...

Elhelyezés

- *placer*
  - pontos elhelyezés
  - de kézi :(
- *packer*
  - automatikus elhelyezés
  - csak irányítani kell
    - `-side` *oldal*
      - *frame*-et levág *oldal*-ról
      - mérete növelhető: `-ipadx, -ipady`
      - `-fill` *irány* / kicsinyítés
    - `-anchor` irány
      - `-padx, -pady`
    - `-in`-nel szülő váltható

Elhelyezés – Demo

- `/usr/lib/tk8.3/demos/widget`
- elhelyezés bemutatása a címkés példán

Futás

- eseményvezérelt
- eseménykezelő hurok
- `-command` opció
  - eljárások, parancsok
  - GUI módosítás – `configure` parancs
  - `destroy`

Futás – Demo

- `/usr/lib/tk8.3/demos/widget`
- futtatás bemutatása a gombos példán

Prolog?

Prolog?

Tcl/TK-hoz számos nyelv tartalmaz beépített interpretert!

Prolog?

Tcl/TK-hoz számos nyelv tartalmaz beépített interpretert!

Prolog is!

De melyik Prolog?

- SICStus
- Ciao
- SWI
- ALS
  - `tcl_call(`*`interp., parancs, eredm.`*`)`

Prolog és TCL/TK ⇒

1. interpretert alkotunk
   `tcl_new(-TclInterpreter)`
2. futtatunk
   `tcl_eval(+TclInterpreter, +Command, -Result)`
3. interpretert kitöröljük :)
   `tcl_delete(-TclInterpreter)`

Command

- Atom
- `write(`*`Term`*`)`
- `format(`*`Fmt, Args`*`)`
- fentiek zárójelezése:
  - `dq(`*`Command`*`)` – ””
  - `br(`*`Command`*`)` – {}
  - `sqb(`*`Command`*`)` – []
- fentiek listája

Például:

```
T = "set"
[format('~s',[T]),x,br([a,b,c])] ⇒
```

set x {a b c}

Prolog és TCL/TK ⇐

TCL-ből visszanyúlhatunk Prologba

- `prolog` *`Goal`*
  - visszatérés: 1, 0, exception
  - `prolog_variables` – asszociatív tömb
- `prolog_event` *`Terms`*

Demo: egyszerű telefonkönyv grafikus felülettel

Prolog és TK ⇒

- közvetlenül TK-val is van kapcsolat
- `tk_new(+Options, -TclInterpreter)`
  - `name(+név)`
  - `display(+képernyő)`
- `tk_main_window(+TclInterpreter, -TkWindow)`
- `tk_destroy_window(+TkWindow)`

Vezérlés

- `tk_do_one_event`
- `tk_do_one_event(+List)`
- `tk_main_loop`
- `tk_next_event(+Interpr., -Event)`

Demo: eseményszámláló

Miért jó ez az egész? ($\sum$)

(összefoglalás)

- gyors fejlesztési lehetőség
- programok kommunikációját segíti
- áttekinthető, tanulható