

Utiliser tkcon

auteur : David COBAC
date : 21 janvier 2003

Table des matières

1	Introduction	1
2	Installation	1
3	La fenêtre	2
a	Screenshot	2
b	Les menus	2
b.i	File	2
b.ii	Console	2
b.iii	Edit	3
b.iv	Interp	3
b.v	Prefs	3
b.vi	History	4
4	Les premières commandes	4
a	Les commandes pour naviguer dans l'arborescence	4
b	Quelques astuces	4
c	Les commandes pour tester un programme	5
5	Personnaliser tkcon	6
a	La ligne de commande	6
b	Personnalisation	6
b.i	Le fichier de configuration	6
b.ii	Les couleurs et la police	7
b.iii	Quelques autres options :	7
c	Faire afficher la valeur de toutes les couleurs et de toutes les options	7
6	Les commandes spécifiques de tkcon	8
7	La procédure tkcon	9
8	Conclusion	10

1 Introduction

tkcon est une « console » (une application permettant de dialoguer avec l'ordinateur) en langage Tcl/Tk. Elle permet notamment d'effectuer les opérations habituelles de maintenance sur les fichiers (création, destruction etc.) mais surtout tkcon permet au programmeur Tcl/Tk de tester les commandes du langage dans un mode interactif.

tkcon est actuellement développé par Jeffrey HOBBS et le site officiel du logiciel est <http://tkcon.sourceforge.net>. On trouvera aussi sur <http://mini.net/tcl> quelques informations sur son sujet. La version utilisée pour ce document est la version 2.3.

2 Installation

tkcon nécessite l'installation des langages Tcl et Tk. Avec la distribution ActiveTcl (téléchargeable gratuitement sur le site <http://www.activestate.com>), tkcon est installé automatiquement dans le répertoire C:\Tcl\bin sous windows.

L'appel de tkcon se fait en interprétant le script tkcon.tcl, c'est-à-dire en invoquant la commande wish tkcon.tcl, normalement sous windows, double-cliquer sur l'icône doit suffire à lancer le programme. Autrement dit tkcon n'est rien d'autre qu'une application Tcl/Tk qui embarque un interpréteur.

3 La fenêtre

a Screenshot

L'image représente l'application au démarrage. La première ligne rend compte de la réussite du chargement du fichier contenant l'historique des commandes déjà tapées dans les sessions antérieures; la deuxième rend compte de l'activation de la fenêtre elle-même et indique les versions de Tcl et de Tk utilisées, ici la version 8.3.4.

Ensuite vient la première ligne de saisie, ici numérotée 49 car l'historique a indiqué que la précédente ligne était numérotée 48 (si c'est la première fois que vous utilisez tkcon, la ligne sera numérotée 1).

Ce numéro est précédé de (david) qui est simplement le dernier répertoire dans l'écriture du chemin complet qui y mène; en effet nous verrons qu'actuellement nous sommes dans le répertoire /home/david (analogue linuxien du répertoire c:\windows\profile\david sous windows). Pour finir, le symbole % indique le début de la ligne de saisie : on tapera les commandes après ce symbole.

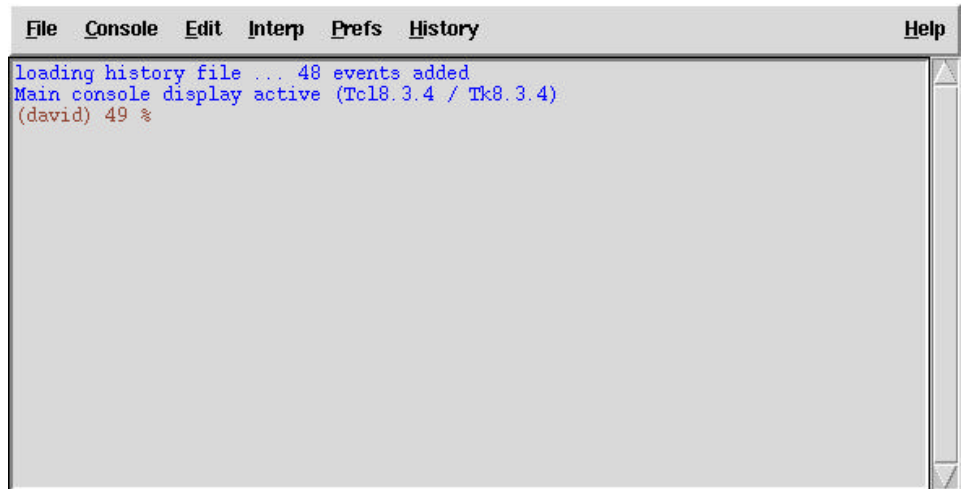


FIG. 1 – La fenêtre principale de l'application

b Les menus

Les menus de tkcon ne sont pas essentiels dans un premier temps. On peut donc très bien passer cette section et y revenir plus tard.

b.i File

Le menu File (fichier) permet :

- **Load File** : le chargement d'un fichier donc son exécution si c'est un script défini de manière solitaire ou le chargement de procédures/fonctions s'il s'agit d'un fichier contenant une bibliothèque de fonctions ;
- **Save ...** : l'enregistrement de tout ou d'une partie des résultats obtenus lors de votre session sous tkcon ;
- **Quit** : l'arrêt de tkcon et de tous les processus lui étant liés.

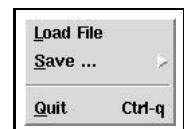


FIG. 2 – Le menu File

b.ii Console

Le menu Console permet de créer et de détruire des consoles esclaves de la console principale (celle qui s'ouvre à l'appel de l'application).

- **Main Console** n'est pas un bouton mais un affichage indiquant que vous travaillez avec la console principale (initiale) ;
- **New Console** permet de créer une console esclave de la console courante ;
- **Close Console** permet de quitter (fermer) la console courante ;
- **Clear Console** permet d'effacer toutes les lignes présentes dans la console et de commencer une nouvelle ligne de saisie en haut de la console ;
- **Make Xauth Secure ...**
- **Attach to ...** permet de lier la console courante à différents éléments :
 - à un interpréteur existant déjà, comme par exemple une application Tcl/Tk qui est déjà active ;
 - à un espace de nom listé dans le sous-menu ;

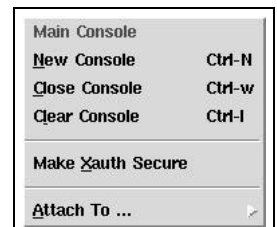


FIG. 3 – Le menu Console

- à une connexion (socket) ;
- à un affichage différent de l’affichage courant (cf. serveur d’affichage).

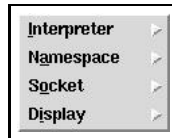


FIG. 4 – Le menu Attach to ...

b.iii Edit

Le menu Edit (Editer) contient les habituels « Couper–Copier–Coller ». Ce menu offre aussi une commande de recherche (Find) qui permet de trouver des séquences de mots dans la console, les résultats sont alors en surbrillance. Notons que la recherche peut se faire par l’intermédiaire d’expressions régulières avancées pour permettre des recherches plus élaborées.

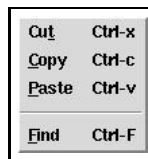


FIG. 5 – Le menu Edit

b.iv Interp

Le menu Interp (Interpréteur) propose quelques commandes pour gérer l’interpréteur embarqué dans la console courante :

- **SLAVE : slave** n’est pas un bouton mais un affichage indiquant que l’interpréteur actuellement en cours s’appelle slave ;
- **Show Last Error** permet d’afficher une fenêtre affichant le code complet de la dernière erreur (cette fenêtre est aussi accessible par hyperlien dans la console elle-même en cliquant sur l’erreur par la console) ;
- **Packages** permet d’afficher toutes les extensions du langage qui sont actuellement disponibles ;
- **Checkpoint State** insère une marque à partir de la ligne actuelle en annulant toute marque précédente, cette commande remet à zéro la commande traçant l’existence des variables et des procédures pour cette fonctionnalité ;
- **Revert State** permet de revenir à l’état avant la dernière marque ... ? ;
- **View State** affiche une fenêtre permettant de visualiser les procédures et variables définies dans l’interpréteur depuis la dernière marque ;
- **Send tkcon Commands ... ?**



FIG. 6 – Le menu Interp

b.v Prefs

Le menu Prefs (préférences) propose de changer quelques paramètres de la console, on notera le « calculator mode » qui permet de taper des formules mathématiques dans la console sans passer par la commande `expr`.

- Les deux premières options permettent la coloration syntaxique des accolades et des commandes.
- Sélectionner **Hot Errors** permet de créer un hyperlien sur les erreurs vers une fenêtre montrant le code complet de l’erreur.
- Les trois dernières commandes permettent de personnaliser l’affichage de la fenêtre avec la possibilité de ne pas afficher les barres de menus et d’état, et celle de placer différemment l’ascenseur.

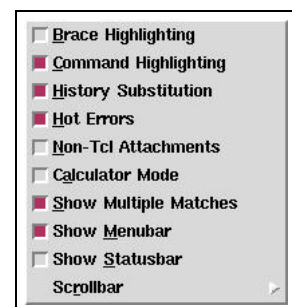


FIG. 7 – Le menu Prefs
 Lycée du Nord-Ober-Cobac-2003
 Initiation à la programmation

b.vi History

Le menu History (historique) affiche les dernières commandes saisies dans la console, on peut évidemment sélectionner une commande de l'historique pour la ré-évaluer dans la console.

```
48: 2+double(3)
47: 2+3
46: expr ${i}<=2
45: ${i}<=2}
44: ${i}<=2
43: i<=2
42: 2+3
41: 3+2
40: tkcon alias
39: set j 2
```

FIG. 8 – Le menu History

4 Les premières commandes

a Les commandes pour naviguer dans l'arborescence

La navigation du système de fichiers se fait évidemment grâce aux commandes Tcl :

- la commande `pwd` (*Print Working Directory*) renvoie le nom complet du répertoire actif ;
- la commande `cd` (*Change Directory*) permet de changer de répertoire, le nouveau répertoire étant en argument ; en l'absence de tout argument le changement s'effectue vers le répertoire personnel de l'utilisateur ;
- la commande `dir` (*DIRectory*) permet de lister le contenu du répertoire actif ;
- la commande `glob` renvoie la liste de tous les fichiers et sous-répertoires du répertoire actif satisfaisant un certain motif passé en argument.

La liste n'est évidemment pas complète puisque l'on peut ajouter notamment toutes les commandes assujetties à la commande `file` comme par exemple `file size` qui, prenant en argument le nom d'un fichier, renvoie sa taille en octets.

Voici un exemple de session `tkcon` utilisant ces commandes :

```
(david) 50 % pwd
/home/david
(david) 51 % cd /travail/david/tcltk/Mesapps/
(Mesapps) 52 % dir
.:
MesLibs          TkBM            TkD             TkDivPolynomes  TkDiviseurs     TkDvips         TkEdit
TkEtoile         TkEuclide      TkFEF          TkFEF-1.0.tcl ~  TkFractales     TkHorloge      TkLanceDe
TkMel           TkMultPolynomes TkParam        TkSacha         TkSelectCouleur TkStatlvar     TkVisuImages
TkVoyelles      Tk_Pres_CD.tcl collatz.tcl     editeur_bis.tcl ~ gestion_notes.tcl
(Mesapps) 53 % file size gestion_notes.tcl
5899
(Mesapps) 54 % glob Tk
no files matched glob pattern "Tk"
(Mesapps) 55 % glob Tk*
TkD TkBM TkMultPolynomes TkStatlvar TkFEF TkMel TkLanceDe TkSelectCouleur TkEdit TkEtoile TkVisuImages TkDiviseurs TkVoyelles
TkEuclide TkDivPolynomes Tk_Pres_CD.tcl TkDvips TkParam TkSacha TkHorloge TkFractales {TkFEF-1.0.tcl ~}
(Mesapps) 56 % glob Tk*s
TkMultPolynomes TkVisuImages TkDiviseurs TkVoyelles TkDivPolynomes TkDvips TkFractales
(Mesapps) 57 %
```

b Quelques astuces

- Toutes les commandes déjà saisies (mais en nombre limitée tout de même) sont entreposées dans l'historique (cf. menu History) mais on peut les rappeler simplement en utilisant les touches « Haut » et « Bas » du pavé de flèches.
- Il est possible de faciliter la saisie d'une commande car `tkcon` les connaissant peut compléter automatiquement ce qui a déjà été écrit en appuyant sur la touche de tabulation, cette possibilité s'applique aussi à l'arborescence de l'ordinateur. Dans le cas où plusieurs complétions sont possibles, `tkcon` propose tout simplement toutes les possibilités.
- La commande `history clear` permet d'effacer l'historique, le numéro de la ligne suivante devient 2 (la ligne 1 étant la commande d'effacement) ; dans la même famille de commandes, citons `history keep` qui prend en argument un nombre qui sera le nombre de commandes à conserver dans l'historique.

- Cliquer droit sur la zone de saisie permet de faire apparaître un menu popup identique à celui de la barre de menu. Fonctionnalité très utile si on a désactivé la barre de menu dans le menu `Prefs`.

c Les commandes pour tester un programme

L'intérêt majeur de `tkcon` est qu'il permet de tester les commandes des langages `Tcl` et `Tk`. Ainsi on peut se familiariser sans effort avec de nouvelles commandes. Nous verrons comment on peut aussi utiliser `tkcon` pour tester nos applications en cours de développement.

Ce mode de fonctionnement interactif permet de comprendre que les procédures agissent sur les variables et changent éventuellement l'état global mais renvoient parfois un résultat. Ainsi certaines procédures en `Tcl` sont de vraies fonctions qui ne font que renvoyer un résultat alors que d'autres effectuent une action en changeant l'état global.

Par exemple la simple commande `set` peut être utilisée de manière très différente comme le montre l'exemple suivant :

```
(david) 64 % set i
can't read "i": no such variable
(david) 65 % set i 1
1
(david) 66 % set i
1
(david) 67 % set j $i
1
(david) 68 % set k 2
2
(david) 69 % set i [set k]
2
(david) 70 % set j
1
(david) 71 %
```

`tkcon` peut bien entendu servir de base à la construction d'une application ; bien que l'édition d'un fichier ne soit pas possible, on peut l'utiliser comme application annexe permettant d'exécuter le code et de tester certaines commandes.

Par exemple, après avoir créé un script `Tcl` nommé `galton.tcl` nous pouvons évaluer son contenu avec la commande `tclsh` qui est l'appel à un interpréteur `Tcl` comme le montre l'exemple suivant . Bien sûr à la place de l'interpréteur `tclsh`, on aurait pu invoquer `wish` interpréteur embarquant `Tk`. Il me semble que cette méthode est celle à privilégier pour tester ses programmes car comme nous le verrons dans la suite la commande `source` se destine plutôt au chargement d'un fichier de procédures/fonctions.

```
(david) 58 % cd /travail/david/lycee/atelierprog/
(atelierprog) 59 % tclsh galton.tcl
galton en tcl par david cobac (dcobac@free.fr) 2002
Erreur d'arguments dans la ligne de commande !
usage : galton nbre_billes nbre_rangees_clous
exemple : galton 1000 20
(atelierprog) 60 % tclsh galton.tcl 500 15
galton en tcl par david cobac (dcobac@free.fr) 2002
-----
billes : 500 et rangées : 15
-----
0
0
0
# 1%
#### 4%
##### 6%
##### 14%
##### 19%
##### 19%
##### 13%
##### 12%
#### 4%
## 2%
0
0
0
(atelierprog) 61 %
```

Pour des bibliothèques de commandes telles que `tcllib`, on pourra les charger avec le menu `Packages` du bouton `Interp`. Pour charger une bibliothèque personnelle, on utilisera la commande `source`, cette commande va évaluer le contenu du fichier dont le nom lui est passé en argument. Cette commande n'est pas (je pense) à utiliser pour interpréter des programmes en `Tcl` : ces programmes sont susceptibles de se voir adjoindre des arguments en ligne de commande et la commande `source`¹ ne pourrait pas alors les gérer comme tels comme le montre l'exemple suivant. La commande `source` est analogue au chargement du fichier avec `Load` dans le menu `File`. Par contre, les programmes graphiques, a priori moins susceptibles d'utiliser des arguments en ligne de commande, pourront utiliser la commande `source`. Hélas, ces programmes devront invoquer explicitement la bibliothèque graphique `Tk` ou alors l'utilisateur devra charger cette bibliothèque à l'aide du menu `Interp` (bouton `Packages`) ; l'inconvénient

¹On pourra utiliser aussi la commande `package require` suivi du nom du fichier pour peu qu'un fichier `pkgIndex.tcl` existe

majeur de cette méthode est qu'en arrêtant l'évaluation de votre programme, vous allez détruire la fenêtre principale (mère de toutes les autres) dont le chemin est « . » et qu'il sera dès lors impossible de relancer un programme utilisant Tk.

```
(Mesapps) 71 % cd MesLibs/dcmaths/  
(dcmaths) 72 % source dcmaths.tcl  
(dcmaths) 73 % set dcmaths::pi  
3.141592653589793238  
(dcmaths) 74 % dcmaths::extrema {543 2 8.5 -5 -2e+1}  
-2e+1 543  
(dcmaths) 75 % cd /travail/david/lycee/atelierprog/  
(atelierprog) 76 % source galton.tcl  
galton en tcl par david cobac (dcobac@free.fr) 2002  
Erreur d'arguments dans la ligne de commande !  
usage : galton nbre_billes nbre_rangees_clous  
exemple : galton 1000 20  
(atelierprog) 77 % source galton.tcl 500 15  
wrong # args: should be "source fileName"  
(atelierprog) 78 %
```

La solution, si on veut toujours utiliser source à la place de wish, sera alors de sourcer dans un interpréteur esclave créé pour l'occasion (menu Interp).

Attention, la commande source est sensible à la commande exit, ainsi « sourcer » un fichier qui exécutera exit aura pour effet non seulement d'arrêter le programme sourcé mais aussi d'arrêter la session tkcon.

5 Personnaliser tkcon

a La ligne de commande

On peut passer des arguments à tkcon, lors de son invocation, grâce à la ligne de commande. Voici donc quelques exemples permettant de personnaliser son appel :

Chargement automatique d'un package : l'option **package** permet de charger une bibliothèque, exemple : tkcon -package math chargera la bibliothèque math de la bibliothèque tcllib. Notons que l'on peut utiliser -load à la place de package et que nous pouvons directement faire appel à Tk par tkcon -package Tk.

Utilisation d'une police : l'option **font** permet de préciser la police à utiliser dans la fenêtre, exemple : tkcon -font helvetica mais on pourra personnaliser encore plus avec tkcon -font {helvetica 14 bold}

Utilisation de couleurs particulières : l'option **color** permet d'utiliser d'autres couleurs que celles fixées par défaut, exemple : tkcon -font "helvetica 14 bold" -color,bg white permettra d'avoir un fond blanc².

Interprétation d'un script au démarrage : on peut directement invoquer un fichier à évaluer comme par exemple : tkcon galton.tcl ou encore si ce dernier nécessite des arguments tkcon galton.tcl 500 20 ; le résultat de l'interprétation sera alors lisible dans la fenêtre.

b Personnalisation

b.i Le fichier de configuration

L'utilisation systématique d'options en ligne de commande n'est pas très pratique. tkcon fournit une alternative intéressante : à son appel, il essaye de lire un fichier nommé tkcon.cfg sous windows ou Macintosh et .tkconrc sous unix. Ce fichier, s'il existe, contient les préférences de l'utilisateur de tkcon. Si le fichier n'existe pas, tkcon s'en passe et démarre ses options par défaut.

Ce fichier doit être situé à un endroit précis pour qu'il puisse être trouvé par tkcon : le répertoire env(HOME) pour windows et env(PREF_FOLDER) sous Macintosh. Pour connaître ce répertoire plus explicitement, taper sous unix et windows dans une console tkcon :

```
(atelierprog) 49 % set env(HOME)  
/home/david  
(atelierprog) 50 %
```

On adaptera l'appel sous Macintosh en tapant set env(PREF_FOLDER).

Une fois ce fichier créé, son contenu sera évalué pendant la phase de démarrage de tkcon. Bien entendu, tkcon offre la possibilité de ne pas utiliser systématiquement ce fichier précis grâce à l'option **rcfile** qui prend en paramètre le nom du fichier³ à évaluer. Par exemple tkcon -rcfile mesprefs.moi lira le fichier mesprefs.moi situé dans le répertoire courant.

Une telle possibilité vous autorise donc plusieurs fichiers de préférences ... ne serait-ce que pour tester.

²Bizarrement cette commande n'a pas modifiée mon fond ... mais nous utiliserons une autre possibilité plus pratique pour le faire.

³Vous pouvez lui donner n'importe quel nom, avec n'importe quelle extension.

b.ii Les couleurs et la police

Le fichier peut contenir toutes les préférences pour les couleurs, 10 couleurs sont définissables :
Entre parenthèses, sont indiquées les valeurs par défaut :

bg fond (dépendant du système)
blink fond des accolades lors de la coloration (yellow)
cursor fond du curseur (noir)
disabled police des menus indisponibles (dark grey)
proc police de la commande proc lors de la coloration (dark green)
var fond des variables lors de la coloration (pink)
prompt ... ? (brown)
stdin police de saisie (black)
stdout police de sortie (blue)
stderr police des erreurs (red)

Pour changer ces couleurs, on écrira dans le fichier de configuration ce genre de choses :

```
| set ::tkcon::COLOR(bg) white  
| set ::tkcon::COLOR(stdin) {dark blue}  
| set ::tkcon::COLOR(stderr) black
```

Les autres restent inchangées et donc les valeurs par défaut sont adoptées.

Pour changer la police, on aura recours à la variable `::tkcon::OPT(font)` que l'on fixera avec par exemple :

```
| set ::tkcon::OPT(font) {helvetica 14}
```

b.iii Quelques autres options :

Taille à l'ouverture Le nombre de lignes et de colonnes est paramétrable par les options **rows** et **cols** qui sont par défaut fixées respectivement à 20 et 80. On pourra les changer par exemple de cette manière :

```
| set ::tkcon::OPT(rows) 40  
| set ::tkcon::OPT(cols) 100
```

Taille de l'historique Elle sera réglée par l'option **history**, on peut fixer à 20 le nombre d'éléments à retenir :

```
| set ::tkcon::OPT(history) 20
```

Menus visibles et barre d'état Pour rendre la barre de menu invisible, on fixera l'option **showmenu** à 0, par défaut la valeur est fixée à 1 et le menu est bien entendu visible.

```
| set ::tkcon::OPT(showmenu) 0
```

De la même manière, on pourra supprimer la barre d'état :

```
| set ::tkcon::OPT(showstatusbar) 0
```

Bien sûr de nombreuses autres options sont configurables ; a priori, toutes les choses accessibles par le menu sont d'une manière ou d'une autre configurable via le fichier de configuration.

c Faire afficher la valeur de toutes les couleurs et de toutes les options

Les variables que l'on a rencontrées sont toutes situées dans 2 tableaux : `::tkcon::COLOR` pour le choix des couleurs et `::tkcon::OPT` pour les options. Pour visualiser ces valeurs, il faut d'abord accéder à l'interpréteur du script tkcon, et non pas celui que l'on manipule habituellement dans la console. Pour ce faire, nous allons « attacher » la console à l'interpréteur de tkcon par la commande `tkcon attach main` ; ensuite il suffit de demander l'affichage complet des deux tableaux avec la commande `parray` de Tcl :

```
(atelierprog) 37 % parray ::tkcon::COLOR  
can't access "::tkcon::COLOR": parent namespace doesn't exist  
(atelierprog) 38 % parray ::tkcon::OPT  
can't access "::tkcon::OPT": parent namespace doesn't exist  
(atelierprog) 39 % tkcon attach main  
>Main< (atelierprog) 40 % parray ::tkcon::COLOR  
::tkcon::COLOR(bg) = white  
::tkcon::COLOR(blink) = #FFFF00  
::tkcon::COLOR(cursor) = #000000  
::tkcon::COLOR(disabled) = #4D4D4D  
::tkcon::COLOR(proc) = #008800
```

```

::tkcon::COLOR(prompt) = #8F4433
::tkcon::COLOR(stderr) = black
::tkcon::COLOR(stdin) = blue
::tkcon::COLOR(stdout) = #0000FF
::tkcon::COLOR(var) = #FFC0D0
>Main< (atelierprog) 41 % parray ::tkcon::OPT
::tkcon::OPT(autoload) =
::tkcon::OPT(blinkrange) = 1
::tkcon::OPT(blinktime) = 500
::tkcon::OPT(buffer) = 512
::tkcon::OPT(calcmode) = 0
::tkcon::OPT(cols) = 100
::tkcon::OPT(dead) =
::tkcon::OPT(debugPrompt) = (level \#$level) debug [history nextid] >
::tkcon::OPT(exec) = slave
::tkcon::OPT(expandorder) = Pathname Variable Procname
::tkcon::OPT(font) = helvetica 14
::tkcon::OPT(gc-delay) = 60000
::tkcon::OPT(gets) = congets
::tkcon::OPT(history) = 20
::tkcon::OPT(hoterrors) = 1
::tkcon::OPT(library) =
::tkcon::OPT(lightbrace) = 1
::tkcon::OPT(lightcmd) = 1
::tkcon::OPT(maineval) =
::tkcon::OPT(maxmenu) = 15
::tkcon::OPT(nontcl) = 0
::tkcon::OPT(prompt1) = ([file tail [pwd]]) [history nextid] %
::tkcon::OPT(rows) = 40
::tkcon::OPT(scrollypos) = right
::tkcon::OPT(showmenu) = 0
::tkcon::OPT(showmultiple) = 1
::tkcon::OPT(showstatusbar) = 0
::tkcon::OPT(slaveeval) =
::tkcon::OPT(slaveexit) = close
::tkcon::OPT(subhistory) = 1
::tkcon::OPT(usehistory) = 1
>Main< (atelierprog) 42 % tkcon attach slave
(atelierprog) 43 %

```

On remarquera que les tableaux ne sont pas accessibles dans l'interpréteur embarqué (nommé slave) et qu'il faut bien accéder à l'interpréteur de tkcon. À cette occasion, la ligne de saisie change d'allure avec >Main< qui précède la ligne. Pour revenir à l'interpréteur esclave, il suffit alors de réattacher la console à celui-ci : \tkcon attach slave

6 Les commandes spécifiques de tkcon

tkcon introduit des commandes qui peuvent s'avérer utiles et qui augmentent ces capacités. Voilà donc une petite liste non exhaustive avec quelques exemples :

alias permet de rediriger une commande vers une autre :

```

(atelierprog) 34 % alias chemin pwd
chemin
(atelierprog) 35 % chemin
/travail/david/lycee/atelierprog
(atelierprog) 36 %

```

Bien entendu, l'utilisateur va pouvoir non seulement créer des commandes, mais aussi en modifier d'existantes ... à vos risques et périls. On peut supprimer un alias avec la commande unalias suivi du nom de l'alias précédemment créé.

clear permet d'effacer le contenu de la console (peut prendre en argument un pourcentage, qui par défaut est 100)

dir qui avec les options -full ou -long et sa capacité à prendre en charge les motifs (comme glob) est une commande très puissante ; la commande ls est un alias de dir -full.

edit prend en argument le nom d'un fichier. La commande ouvre alors un petit éditeur assez modeste cf. figure 9 qui permet d'éditer le fichier et d'en évaluer le contenu dans l'interpréteur que l'on vient de quitter. Deux autres commandes sont redirigées vers edit : more et less.

what suivi d'un nom permet de reconnaître le type du nom :

```

(atelierprog) 41 % dir *.tcl
./:
code_courrier9.tcl    couleurcanvas.tcl    courrier1codetcl.tcl  essai_courrier3.tcl
essaidicho.tcl      galton.tcl            galtoncomplet.tcl    montecarlo.tcl
montecarlopro.tcl   recursivite.tcl       tirageloto.tcl       tiragessremise.tcl
(atelierprog) 42 % what galton.tcl
file executable

```

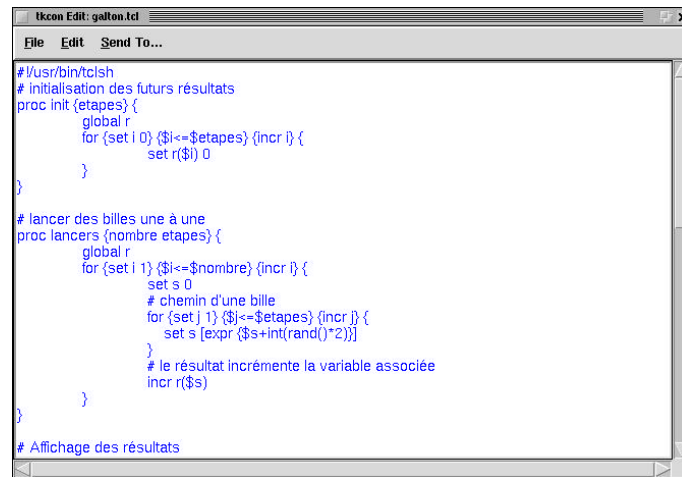


FIG. 9 – L'éditeur embarqué de tkcon

```

(atelierprog) 43 % what .
directory
(atelierprog) 44 % dir *.txt
./:
la_tcllib.txt
(atelierprog) 45 % what la_tcllib.txt
file
(atelierprog) 46 % set i 1
1
(atelierprog) 47 % what i
scalar variable
(atelierprog) 48 % what dir
procedure executable
(atelierprog) 49 % what pwd
command executable
(atelierprog) 50 %

```

which est la commande analogue à celle d'unix, elle permet de chercher le chemin complet d'une commande exécutable :

```

(atelierprog) 50 % which pwd
pwd: internal command
/bin/pwd
(atelierprog) 51 % which tkcon
tkcon: aliased to tkcon
/usr/bin/tkcon
(atelierprog) 52 % which acroread
/usr/bin/acroread
(atelierprog) 53 % which latex
/usr/bin/latex
(atelierprog) 54 % source ../../tcltk/Mesapps/MesLibs/dcmaths/dcmaths.tcl
(atelierprog) 55 % which dcmaths::extrema
dcmaths::extrema: procedure
(atelierprog) 56 % which lacommandequinexistepas
lacommandequinexistepas: command not found
(atelierprog) 57 %

```

On remarque que la commande indique toutes les correspondances même externes.

7 La procédure tkcon

tkcon introduit aussi une procédure éponyme permettant de gérer tkcon et sa fenêtre. Nous avons déjà rencontré cela lorsque nous avons affiché les codes des couleurs avec la commande `tkcon attach`.

Mais cette commande va plus loin avec une foultitude de possibilités :

tkcon close permet de quitter la console actuelle.

tkcon font permet de récupérer le nom de la police utilisée, on peut adjoindre en argument une police qu'elle fixera alors comme police de fond.

tkcon hide permet de cacher la fenêtre ... attention d'être alors certain de pouvoir faire la commande suivante :

tkcon show permet d'afficher la fenêtre qui était cachée (cf. commande précédente).

tkcon history affiche la liste des éléments de l'historique.

tkcon new crée une nouvelle console donc un nouvel interpréteur esclave.

tkcon title permet de donner un nom à la fenêtre de votre application tkcon s'il est suivi d'un argument, sinon la commande renvoie le nom actuel. Ainsi `\tkcon title {Mon application TkCon}` changera le titre de la fenêtre.



Bien sûr, la procédure tkcon inclut d'autres fonctionnalités dont on peut récupérer la liste en se trompant volontairement :

```
(atelierprog) 25 % tkcon mauvaiseoption
bad option "mauvaiseoption": must be attach, bgerror, close, console, deiconify, destroy, font, hide, iconify, load,
main, master, new, save, show, slave, title, version
(atelierprog) 26 %
```

8 Conclusion

Avec un peu d'habitude, tkcon est vraiment l'outil idéal pour développer et se familiariser avec Tcl/Tk. On trouvera sur le Wiki ([\http://mini.net/tcl](http://mini.net/tcl)) une petite astuce pour pouvoir embarquer dans une application Tcl/Tk cette console ce qui peut s'avérer très pertinent pour la conception d'un éditeur Tcl/Tk qui permettra en plus tous les avantages de la console tkcon.