SYSTEME 411 VERSION 3

Cet article de gestion d'arborescence pour HP-28S nous est parvenu des U.S.A. par le réseau UNIX, reliant plusieurs milliers d'ordinateurs de par le monde. Thomas Affinito, son auteur, a donné l'autorisation aux Clubs de le publier. Paul Courbis l'a donc traduit, et c'est avec plaisir que nous vous le présentons ce mois-ci.

INTRODUCTION

Une mémoire à structure de stockage arborescente peut fournir une méthode agréable pour organiser la toujours croissante myriade de variables qu'un utilisateur actif crée pour son usage personnel. La plupart du temps, l'utilisateur passe trop de temps à utiliser ORDER, à monter et descendre les branches de l'arborescence pour chercher à retrouver à quel endroit sont situées certaines variables. Mon premier projet de programmation HP28S était de dompter cette hiérarchie à l'aide d'un système basé au sommet de l'arborescence de manière à libérer l'utilisateur des commandes transversales.

Mon idée était de mettre en oeuvre un concept d'espace de travail (comme en APL) avec des propriétés spécifiques :

1- l'utilisateur doit disposer de commandes simples pour créer (BUILD) et détruire (CRUSH) automatiquement un espace de travail.

2- l'espace de travail doit se comporter comme un banc d'essai pour le développement et les essais de nouveaux programmes, de nouvelles idées.

3- les programmes achevés et les variables doivent pouvoir être supprimés du menu USER (HIDE) et rangés dans un menu parent dans lequel il sera toujours accessible ; de plus les programmes, les variables et, plus généralement, les utilitaires pourront toujours être repoussés dans un menu parent plus élevé qui sera visible depuis tous les espaces de travail créés (STALL, abréviation de *install*).

4- les commandes du système principal peuvent être placées dans le menu *custom* pour en faciliter l'usage ; de plus, l'utilisateur peut créer des labels spécifiques qui seront toujours chargés quand l'espace de travail particulier sera activé (SLAB) ; les labels généraux (présents pour tous les espaces de travail) sont aussi possibles (GLAB) ; ainsi l'utilisateur peut-il cacher (HIDE) ou installer (STALL) tous ses programmes, et nommer de manière locale ou globale (SLAB ou GLAB) les applications principales, ce qui laissera tout en dehors de l'espace USER, gardera tous les utilitaires opaques et donnera une certaine transparence aux applications.

5- l'utilisateur peut passer d'un espace de travail à un autre avec des commandes simples ; les passages transversaux sont totalement éliminés.

6- le système ne prend pas trop de mémoire.

Le système à été minutieusement débogué et est plein d'utilitaires intéressants ; il est facilement personnalisable et occupe à peine 2 Ko (un faible prix à payer avec la 28S). Voici le système que j'utilise pour toute ma programmation. Je serai intéressé par tous les commentaires, toutes les critiques et tous les compléments à ce système...

VERSION 3 ?

Après avoir programmé sur la 28C pendant 6 mois, j'ai tressailli quand est sortie la 28S et je l'ai immédiatement achetée. 411 fut créé pour diriger mon style d'écriture de programmes. Après avoir lu le livre de Wlodek, j'ai réalisé que je devais placer mon système dans un sous-menu du menu HOME pour pouvoir conserver PEEK et POKE dans le menu principal (*). Ainsi naquit la version 2.

Il y a trois jours, j'ai acheté le livre de Bill Wickes, *HP 28 Insights* que je recommande vivement. C'est une extension des manuels, avec d'intéressantes suggestions... Certains de ces programmes étaient similaires aux miens : de manière à rendre 411 plus accessible, j'ai changé certains de mes programmes pour reprendre ses noms, ses algorithmes à la place des miens. A savoir :

UP page 83 de son livre ainsi que dans le manuel HP. Mon programme UP laissait dans la pile le chemin qu'il venait de quitter... Ainsi mes programmes avaient des paires de UP et de DOWN (qui exécutait le chemin dans la pile) quand j'effectuais des chemins transversaux.

DOPATH page 84. Mon programme faisant cela s'appelait DOWN mais j'ai changé le nom pour être en accord avec Bill ; Mon algorithme n'est pas le même... Le mien utilise des utilitaires non-inclus dans son livre et est plus général car il exécute une liste ou un nom.

Purge et Clusr. J'ai un seul programme (en fait une chaîne) qui fait un nettoyage complet d'un menu (appelé XCL pour *eXtra CLear*) Le Purge de Bill fait la même chose, ainsi ai-je utilisé ses noms et algorithmes... Avec l'exception que Clusr est placé dans une chaîne d'objets. J'ai fait cela comme protection ; avoir une touche qui peut vider toute la

mémoire est trop dangereux ! J'ai souvent perdu des choses lors de l'écriture de XCL ! Ainsi il faut faire Clusr STR→ pour lancer ce programme, ce qui ne peut être fait accidentellement par quelqu'un qui tâtonne...

QUE FAIRE ?

La fin de cet article contient tous les listings de programmes et les valeurs des variables par défaut. Il n'y a qu'à les taper ! Pour prévenir toute confusion, il faut savoir que :

- Dans le menu HOME, on stocke le programme ON411, et on crée le sous-menu →411 ; ainsi le menu HOME est vide pour accueillir les programmes devant résider en haut de Ram pour les programmes de Wlodek(*).

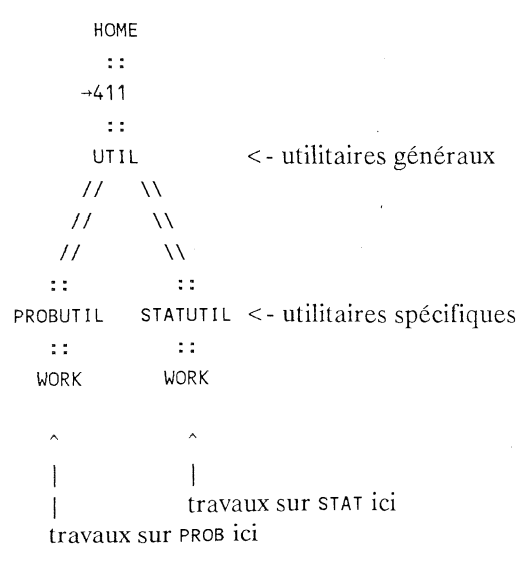
- Dans le menu →411 vont toutes les autres choses mais vous devez créer un sous-menu appelé ici UTIL et y stocker GU.LS.

- Les "" qui apparaissent dans certains programmes ne sont que deux guillemets que j'utilise comme séparateurs dans les menus *custom*.

- Les variables sont listées dans l'ordre dans lequel elles apparaissent dans mon espace USER : des commandes les plus importantes aux moins... Pour obtenir le même ordre : entrer les variables dans l'ordre inverse...

QUE FAIT LE PROGRAMME ?

Supposons que vous ayez créé les deux espaces de travail PROB et STAT (la manière de les créer est décrite dans la section suivante). 411 crée l'arborescence suivante pour vous :



411 automatise la création, la destruction, et les déplacement transversaux de cette structure. Il automatise aussi les mouvements de programmes et d'objets achevés dans les aires d'utilitaires spécifiques ou dans la zone d'utilitaires généraux. Il contrôle aussi les changements spécifiques dans le menu *custom* pour que les applications générales et spécifiques puissent rester transparentes en étant nommées dans ce menu.

Note : Ce type de structure est la même que celle suggérée par Wickes à la page 77 de son livre. Comme je n'ai fait que feuilleter le livre, j'ai eu peur qu'il contienne des programmes faisant la même chose que 411, rendant mes efforts inutiles. Fort heureusement, tel n'est pas le cas ! Ainsi j'espère que cet article sera un supplément et une illustration des idées présentées dans son livre, et qu'il aidera tous les utilisateurs à diriger leur arborescence d'une manière approuvée par le Maître...

COMMENT L'UTILISER ?

Pour commencer : MAIN, ON411

MAIN crée un menu *custom* de tous les espaces de travail, suivi par les commandes EXIT, BUILD et CRUSH. Depuis ce menu, on peut aller dans n'importe quel espace de travail en pressant le nom de cet espace... On peut aussi utiliser les trois commandes précitées.

MAIN est aussi valable dans le menu *custom* lorsqu'on est dans un espace de travail : presser MAIN est généralement fait quand on a fini d'utiliser un espace de travail et que l'on veut retourner au menu principal pour voir les autres espaces où l'on peut se rendre, ou pour utiliser BUILD, CRUSH ou EXIT...

Si vous êtes dans le menu HOME, activer 411 par ON411 vous enverra dans le sous-menu approprié et exécutera MAIN pour vous ; après être sorti de 411 grâce à EXIT et être retourné au menu HOME, ON411 est placé dans le menu *custom* par commodité.

Quitter 411 **:** EXIT

EXIT vous renvoie au menu HOME et affiche un menu *custom* contenant ON411.

Quitter un espace de travail : MAIN, WOFF

MAIN vous ramène au menu principal. WOFF (*Workspace OFF*) est utilisé à la fin du travail pour revenir au menu principal, nettoyer la pile et ré-initialiser les drapeaux à votre convenance.

Construire un espace de travail : BUILD

Taper le nom de votre espace, puis BUILD. Après quoi le menu de la commande MAIN sera activé, avec un nom pour votre nouvel espace. Retourner au menu MAIN par le menu *custom* en pressant MAIN.

Détruire un vieil espace : CRUSH

Taper le nom d'un espace de travail existant, puis CRUSH. Toutes les variables et sous-menus associés à cet espace seront détruits et le nom de l'espace sera retiré du menu *custom*.

Cacher des variables : HIDE, STALL, SEEK

Pour avoir des variables exécutables dans votre espace de travail, mais invisibles dans le menu USER, mettre tous leurs noms dans une liste et exécuter HIDE depuis votre espace de travail.
Pour avoir des variables exécutables depuis tous les espaces : utiliser STALL au lieu de HIDE.
Pour retrouver des variables cachées par HIDE ou STALL, utiliser une liste de noms de variables puis SEEK. SEEK range les variables récupérées dans le menu courant mais ne change aucun label spécifique ou local : vous devez utiliser LABOUT.

Contrôler le menu custom : SLAB, GLAB, LABOUT

SLAB prend une liste et ajoute les objets qu'elle contient à la liste spécifique de commandes. Quand vous êtes dans cet espace de travail, vous verrez toutes les commandes entrées par SLAB en première position dans le menu *custom*. Ceci est généralement utilisé ainsi : imaginons que j'ai un programme d'application spécifique appelé APP qui utilise les utilitaires spécifiques U1 U2 et U3. Je l'ai débogué et il est prêt à être utilisé : on le cache du menu USER par {APP U1 U2 U3} HIDE, puis on rend APP visible par {APP} SLAB (vous pouvez aussi taper 'APP' SLAB. HIDE, STALL, SEEK, SLAB, GLAB et LABOUT peuvent prendre un nom au lieu d'une liste à un élément. MOVE, DELL et VL→S aussi).
GLAB fonctionne de manière similaire pour une liste de commande globale qui affectera tous les espaces de travail.
LABOUT enlève une liste des listes de commandes générales et spécifiques.

Déplacer des variables : MOVE

Taper une liste de noms de variables, puis le nom de l'espace de travail où les envoyer puis MOVE. L'avantage de cette séquence sur la séquence { *<liste de nom de variables>* } VL→S 'nom d'un autre espace' EVAL S→VR est que la seconde séquence entraînera l'exécution de SWON et WON qui peuvent être modifiées pour faire des choses étranges (comme RESET) ce qui pourrait vider la pile. Note : MOVE ne détruit pas les copies originales des variables.

EXEMPLES ILLUSTRES

Vous avez tapé 411... Allons-y !

HOME

Nous sommes au niveau supérieur, ainsi si nous voulons taper PEEK et POKE, nous le pouvons. Si vous n'êtes pas familier de ce genre de chose dites adieu au menu HOME car vous n'aurez plus besoin d'y revenir(*).

ON411

Nous sommes à présent dans le système. Il n'y a pas encore d'espace de travail, mais les trois commandes sont présentes à côté du délimiteur qui sépare les noms des espaces de travail de ceux des commandes. Créons un espace de travail : PROB qui contiendra tous mes programmes de probabilité et de travail, ML pour quelques trucs en langage machine, TEST pour les brouillons et WIX pour les programmes du livre de Bill Wickes.

'PROB' BUILD 'ML' BUILD 'WIX' BUILD 'TEST' BUILD
Après avoir effectué ces commandes, les noms des espaces de travail apparaîtront dans le menu *custom*. Faisons un peu de probabilités...

PROB
Nous sommes soudainement placés dans le menu USER vide de notre nouvel espace de travail. Activez le menu *custom* et vous verrez les commandes HIDE, MAIN etc..., ainsi que les délimiteurs qui séparent les labels de PROB des labels généraux. Retournez au menu USER.

3 4 5 'A' STO 'B' STO 'C' STO
Nous créons quelques variables pour démontrer les possibilités de cacher des données. Lors d'une application réelle, nous cacherions des utilitaires et les applications utiles qui utilisent ces programmes.

'A' HIDE 'B' STALL A et B disparaissent du menu USER !

A B
Les nombres 1 et 4 sont renvoyés ! A et B existent toujours... ils sont juste cachés. Activez le menu *custom* puis :

MAIN WIX

Une nouvelle fois un menu vide nous fait face... Nous sommes dans l'espace de travail WIX...

```
A B C
```

A renvoie A car l'espace WIX ne peut pas voir A qui a été caché à l'aide de HIDE par PROB ; de même WIX ne peut voir 'C' qui existe dans PROB. Par contre WIX peut accéder à B qui a été caché comme objet général par STALL. Nous avons appris trois choses : 1) les espaces de travail sont séparés et indépendants les uns des autres, 2) HIDE cache les objets qui restent spécifiques à leur espace de travail, 3) STALL les cache tout en les rendant exécutables par tous les espaces de travail. Retournons à PROB par la voie rapide :

```
PROB
```

Le menu user PROB réapparaît. Maintenant supposons que A et B soient des applications importantes et que nous voulons les exécuter par une touche depuis PROB.

```
{A B} SLAB
```

Le menu *custom* est activé et vous voyez A et B apparaître sur les touches. Ils apparaîtront toujours comme touche dans PROB jusqu'à ce que vous les enleviez ou que vous les détruisiez PROB par CRUSH.

```
WIX
```

Allez au menu *custom* et vous ne trouverez ni la touche A, ni la touche B... SLAB n'effectue que des appellations spécifiques à un espace de travail.

```
'B' LABOUT
```
Le menu *custom* est activé et le nom B a disparu. La variable existe toujours.

```
'B' GLAB
```
Remarquez comme la position de B a changé... Il est dans les variables globales.

```
WIX
```
Allez au menu *custom*. B y est ! Les labels globaux apparaissent dans tous les espaces de travail. Toute chose pouvant être mise dans une liste peut devenir un label. Il faut remarquer que le fait que B soit un label global ne garantit pas le fait que B soit une variable globale. L'affectation de labels (GLAB, SLAB, LABOUT) est indépendante du masquage des variables (HIDE, STALL, SEEK). Nettoyons à présent la mémoire :

```
'B' LABOUT 'B' SEEK
```

Ceci supprime le label global et l'objet global, qui continuerait à exister si on s'était contenté de détruire l'espace de travail spécifique. C'est une bonne chose qu'ils demeurent... nous ne les ramenons à un espace spécifique que pour faire la démonstration de la destruction d'une telle variable.

```
WOFF 'PROB' CRUSH 'WIX' CRUSH 'TEST' CRUSH 'ML' CRUSH
```

La meilleure façon d'apprendre à se servir de 411 est de créer d'autres exemples et de jouer avec, par vous-même. Il est aisé d'apprendre à utiliser 411 si vous en prenez le temps, temps que vous regagnerez grâce aux possibilités d'organisation qu'il fournit...

NOTES ET RESUMES

Après avoir appris les commandes, vous vous êtes sans doute rendu compte qu'il était plus facile de taper les commandes que de passer par le menu *custom*. Ce menu *custom* est là pour l'exécution en une touche des utilitaires que vous avez sélectionnés.

Pour aller d'un espace de travail à un autre, il n'est nul besoin de passer par MAIN ou par WOFF... Il suffit de taper le nom de cet espace et vous y êtes... Le menu MAIN est un aide-mémoire pour se souvenir de la liste de ces noms.

Toutes les commandes fonctionneront toujours si vous avez créé un sous-menu dans votre zone de travail, ou un sous-menu dans votre sous-menu, etc. Le seul problème est que si vous cachez, par HIDE ou par STALL, des variables qui ne sont pas dans le menu courant, dans le menu de travail ou dans le menu des utilitaires spécifiques, alors elles seront cachées correctement mais leur copie originale continuera à exister dans le sous-menu où elles ont été créées. Ce problème ne se produira jamais si vous utilisez correctement cet utilitaire (le seul besoin de sous-menu est pour le stockage simultané de solutions obtenues par SOLVER d'une équation donnée.).

BUILD et CRUSH vous ramènent au menu principal. Ces commandes sont supposées être peu fréquentes... Les espaces de travail sont prévus pour que l'on y travaille...

Pour montrer les possibilités de 411 à vos amis, créez l'espace de travail DEMO où vous pourrez faire tous vos essais... Après quoi il sera facile de tout détruire par WOFF 'DEMO' CRUSH!

UTILITAIRES UTILES

BOOP beep d'erreur standard.

S→N effectue une conversion chaîne→nom ou algebraic (si le contenu de la chaîne le permet)

O→S Prend un objet délimité et le renvoie dans une chaîne sans les délimiteurs... Fonctionne pour les noms, les listes, les vecteurs.

Purge détruit toute variable... Même un sous-menu non vide.

Clusr détruit toutes les variables dans le menu courant, y compris les sous menus non vides. Il doit être exécuté avec STR→.

DELL prend une liste ou un nom au niveau 2, et une liste ou un nom au niveau 1 et supprime toute apparition des objets du niveau 2 de la liste du niveau 1. Rien n'est renvoyé dans la pile.

RESET vide la pile et redonne les valeurs par défaut aux drapeaux.

VL→S prend une liste de variables (ou un nom) et rappelle chacune des variables dans la pile, le contenu étant suivi par le nom. A la fin, le nombre de variables rappelées est placé dans la pile.

S→VR prend la sortie de VL→S et le transforme en variables dans le menu courant. La combinaison de VL→S et S→VR rend facile le déplacement de programmes depuis le menu HOME dans un espace de travail et vice versa... bien que ceci soit rare.

UP passe au menu père.

DOPATH prend une liste de noms et les exécute. Ceci est généralement utilisé pour revenir à un menu dont la position a été sauvée dans la pile grâce à la commande PATH.

PERSONNALISATION DU SYSTEME

Je m'excuse : j'aurai dû commenter mes programmes mais je voulais poster ceci aussi vite que possible, ayant peur de commencer une version 4... Mes programmes ne sont pas faits que d'astuces, et vous n'êtes pas obligés de les comprendre pour les utiliser. Voici quelques solutions simples pour personnaliser 411 :

Pour changer le comportement en entrée de tout espace de travail : modifiez WON (*Workspace ON*) : tout ce que fait WON pour le moment est de créer le menu *custom* pour l'espace de travail. Si on veut ajouter un beep à l'entrée il suffit de remplacer WON par « UCML BOOP ».

Pour changer le comportement en entrée d'un espace précis : modifier SWON (WON spécifique) qui est situé dans le menu utilitaires au dessus de la zone de travail. Le SWON standard ne fait qu'appeler WON. Imaginons que vous êtes dans l'espace PROB et que vous vouliez effectuer un RESET : il faut changer SWON en « WON RESET » (faire UP pour pouvoir modifier SWON).

Si vous n'aimez pas les "" comme délimiteurs de menus : enlevez les "" ainsi que les + qui les suivent dans MAIN et ACML.

Il y a des commandes très utiles en sortie (EXIT) : modifiez la liste dans EXIT pour inclure les commandes que vous désirez (par exemple ON411).

Si l'état standard des drapeaux est différent du mien : mettez le calculateur dans l'état souhaité puis RCLF 'FVFLG' STO (dans →411). RESET et WOFF mettront dorénavant la machine dans cet état standard.

Vous voulez que vos programmes modifient le menu *custom* puis le remettent plus tard à son état standard : il suffit de mettre dans votre programme la séquence CM.M MENU.

Accéder directement au menu des labels : la variable SU.LS (dans le menu père du menu WORK, ou menu de travail) contient ces labels. GU.LS (dans le menu UTIL) contient la liste des labels globaux.

411

Menu HOME

→411 directory

ON411 active 411
« HOME →411 MAIN »

Menu →411

UTIL directory contenant GU.LS

GU.LS general utility list
{}

WK.LS liste des espaces de travail
{}

CM.LS liste des commandes du menu MAIN
{ EXIT BUILD CRUSH }

CU.LS liste des commandes utilitaires
{ MAIN WOFF "" HIDE SLAB "" STALL GLAB
"" SEEK LABOUT "" MOVE }

FVFLG état standard des drapeaux

```
#8081FFD40000000h
```

MAIN aller au menu principal

```
« HOME →411 WK.LS "" + CM.LS + MENU »
```

BUILD créer un espace de travail
(entrée : un nom)

```
« HOME →411 DUP O→S "UTIL" + → n n.s
  « IF WK.LS n POS
    THEN "Name In Use" 1 DISP BOOP
    ELSE 'WK.LS' DUP EVAL n + SWAP STO
      "«UTIL " N.S + " WORK SWON»" +
      STR→ n STO UTIL n.s DUP S→N
      CRDIR « WON » SWON STO {} SU.LS
      STO MAIN
    END
  »
»
```

CRUSH détruit un espace de travail
(entrée : un nom)

```
« HOME →411 DUP O→S "UTIL" + → n n.s
  « IF WK.LS n POS
    THEN n PURGE n 'WK.LS' DELL UTIL
      n.s S→N Purge MAIN
    END
  »
»
```

EXIT quitter 411

```
« HOME { "" "" ON411 "" "" "" } MENU 23 MENU »
```

HIDE cacher le variables de la liste
(entrée : liste de noms)

```
« → ls
  « PATH ls VL→S ls PURGE WORK
    ls PURGE UP S→VR DOPATH
  »
»
```

STALL cacher des variables générales
(entrée : liste de noms)

```
« → ls
  « PATH ls VL→S ls PURGE WORK ls PURGE
    UP ls PURGE UTIL S→VR  DOPATH
  »
»
```

SEEK retrouver des variables
(entrée : liste de noms)

```
« → ls
  « ls VL→S PATH WORK ls PURGE UP ls PURGE
    UTIL ls PURGE DOPATH S→VR
  »
»
```

SLAB mettre une liste dans un menu spécifique
(entrée : liste)

```
« SU.LS + PATH SWAP WORK UP 'SU.LS' STO
  DOPATH UCML
»
```

GLAB mettre une liste dans un menu général
(entrée : liste)

```
« GU.LS + PATH SWAP UTIL 'GU.LS' STO
  DOPATH UCML
»
```

LABOUT retirer les labels de la liste des menus spécifiques et généraux.
(entrée : liste)

```
« → ls
  « PATH WORK UP ls 'SU.LS' DELL
    UTIL ls 'GU.LS' DELL DOPATH UCML
  »
»
```

MOVE déplacer les variables de la liste dans l'espace de travail spécifié.
(entrée : liste des noms, nom espace)

```
« → nm
  « PATH SWAP VL→S UTIL nm O→S
    "UTIL" + STR→ WORK S→VR DOPATH
  »
»
```

WON activation de l'espace de travail

```
« UCML 23 MENU »
```

WOFF désactivation de l'espace de travail

```
« RESET MAIN »
```

UCML crée le menu *custom*

```
« PATH WORK UP SU.LS "" + GU.LS + "" +
  CU.LS + DUP 'CM.M' STO MENU DOPATH
»
```

UP aller au menu père

```
« PATH DUP SIZE 1 - 1 MAX GET EVAL »
```

DOPATH exécute une liste de noms
(entrée : liste de nom ou un nom)

```
« O→S STR→ »
```

VL→S liste de variables -> variable dans la pile
(entrée : liste de noms)
(sortie : obj, nom, obj ... nom, obj, nom, #obj)

```
« {} + → ls
  « 1 ls SIZE
    FOR j ls j GET DUP RCL SWAP NEXT
    ls SIZE
  »
»
```

S→VR pile de variables --> variables
(entrée :obj,nom,obj...nom,obj,nom,#obj)

```
« 1 SWAP START STO NEXT »
```

RESET vide la pile, initialise les modes

```
« CLEAR FVFLG STOF »
```

DELL enlève des objets d'une liste donnée par son nom
(entrée : {obj obj...obj } 'nom liste'
ou : obj 'nom liste')

```
« SWAP {} + → lsnm erls
  « 1 erls SIZE
    FOR j
      erls j GET lsnm EVAL
      WHILE
        DUP 3 PICK POS DUP
      REPEAT
        SWAP LIST→ DUP DUP 3 + ROLL - 2
        + ROLL DROP 1 - →LIST
      END
      DROP lsnm STO DROP
    NEXT
  »
»
```

Purge détruit toutes les variables
(entrée : nom)

```
« 31 SF IFERR PURGE
        THEN DUP EVAL Clusr STR→ UP PURGE
        END
»
```

Clusr vide un menu entier

```
"VARS LIST→ 1 SWAP START Purge NEXT"
```

O→S objet délimité --> chaîne
(exemples : 'nom' --> "nom" ou {liste} --> "liste")

```
« →STR DUP SIZE 1 - 2 SWAP SUB»
```

S→N chaîne --> nom
(entrée : chaîne)

```
« "'" SWAP + STR→ »
```

BOOP son d'erreur

```
« 1400 .07 BEEP »
```

J'espère que vous apprécierez cet utilitaire autant que moi...

Thomas J. Affinito
traduction Paul Courbis (392)

(*) : Note de la Rédaction : les restrictions citées par Thomas ne sont pas applicables avec les programmes publiés par Paul Courbis. Voir les anciens numéros de JPC, et plus particulièrement JPC 51 et JPC 56.