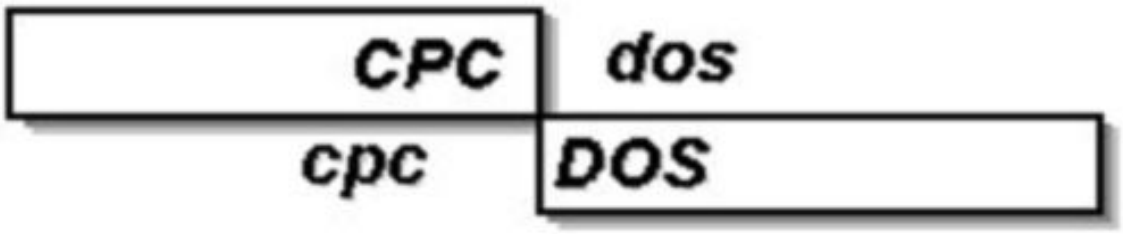


- Crée Pour Concevoir Des OS -

Microsf01 CpcdosC+

Manuel doc.1.2014 Alpha

(En cours de rédaction – 12/01/2014)

Apprendre la programmation CpcdosC+
pour le Kernel Cpcdos OSx

- Version OS2.0.5 Alpha 2.9 32Bits -

Intéressé(e) pour créer votre propre petit système d'exploitation ?

Programmation base en fichiers de commandes
et interprétation en console & interface utilisateur graphique.

- Tutoriels & exemples -

http://microsf01.fr.nf | http://cpcdos.fr.nf/
http://www.facebook.com/pages/Kernel-Cpcdos-OSx/479523255400921
Contact : sebastien.ordinateur@hotmail.fr | Bug&Info : http://forum-cpcdos.fr.nf/

Avertissement :

Autres informations :

Aucuns code source du noyau Cpcdos est dévoilé pour le moment.

Seul le code source du système d'exploitation CraftyOS basé Cpcdos est libre.

Vous avez le droit de produire vos projets librement, de le partager.
Mais en cas de vente, si vous incluez les binaires de Cpcdos, assurez-vous d'avoir l'autorisation au près de FAVIER Sébastien. *(Contact voir page de garde)*

Cette version de Cpcdos est totalement gratuite
Seul la version OS-SERVER sera payante.

OS-SERVER est un système lié à un système Windows.
Il permettra de créer un serveur de stockage, de prise en main à distance de Cpcdos et d'un téléchargement multi-tâche.

Sommaire

- Tester un réseau ou une machine - - - `ping/`
- Connecter un lecteur réseau - - - `connecter/`

  + Déconnecter un lecteur réseau - - `deconnecter/`
- **Les fonctionnalitées**
  + Fonctions mathématiques et autres
  + Les fonctions graphiques
  + Fonctions CpcdosC+ du noyau

5. Commandes avancées - - - - Chapitre V

- Gestion Multitâche
- Initialiser une interface - - - - `ini/`

  - Créer une Fenêtre - - - - `ini/ fenetre( )`
  - Créer un Bouton - - - - - `ini/ bouton( )`
  - Créer un Label - - - - - `ini/ label( )`
  - Créer une ImageBox - - - - `ini/ imagebox( )`
  - Créer un TexteBox - - - - `ini/ textebox( )`
  - Créer un Compteur - - - - `ini/ compteur( )`
- Créer une interface - - - - `creer/`

  + «Pirater» le SCI / modifier les propriétés
- Démarrer l'initialisation de l'OS - - - `demarrer/`
- Lancer l'interface graphique (l'OS) - - `iug/`
- Les événements - - - - - `ev/`
- Afficher un MSGBOX - - - - `msgbox/`
- Explorer un dossier - - - - `explorer/`
- Fermer une fenêtre ou objet - - - `fermer/`
- Actualiser une ou plusieurs propriétés - - `actualise/`
- Focus sur une fenêtre - - - - `focus/`
- Lancer le mode console - - - - `lc/`
- Lire & éditer le registre - - - - `reg/`
- **Variables d'environnements**
- **Résolutions d'écran**

- Configurer & Tester le système - - - `sys/`

  1. Résumé système (Application externe) - **/quick**
  2. Tester le Kernel - - - - **/krnltest**

Chapitre I – Projet Cpcdos ?

(Débuté le 15 Juillet 2011)

(Acronyme CPCDOS : Créé Pour Concevoir Des OS)

Cpcdos est un noyau monolithique modulaire multitâche coopératif 32Bit fonctionnant au-dessus du DOS, sur les machines type PC (80x86 et 80x64).
Ce noyau s'interprète et s'utilise qu'avec le langage ***CpcdosC+*** en fichiers "script" ou en console. Et requiert aucune compilation!
Pour permettre a tous de créer de toute pièce, une interface graphique et/ou une interface utilisateur très facilement, ou plus communément un système d'exploitation.

> Le but de ce projet, est pour amateurs désirant de créer leurs propre système exploitation en toute simplicité sans utiliser l'*Assembleur* , *C* ou autre que le **CpcdosC+**

Tout cela grâce à un système 32Bits, la séquence de démarrage , pilotes, création d'une interface graphique très avancée et simple allant jusqu'à **32Bits de couleurs** et une résolution maximale **2048x1536**

Ce système constitué d'un langage de programmation très simple avec des syntaxes , messages et commandes entièrement en **Français**, qui se nomme le **CpcdosC+.**

Vidéo présentation : https://www.youtube.com/watch?v=mhsKyaj9UEk
« Presentation Kernel Cpcdos OSx (2.0.5 A1.1)- FAVIER Sébastien 01 »

- - - - - - - - -
CpcdosC+
CpcdosCommande+
Initiales : CC+ / CCP
Utilisable en Console & en fichier Scripts (Batch) & Executable
- - - - - - - - -

Cpcdos est un Kernel Monolithique modulaire Multitâche Coopératif

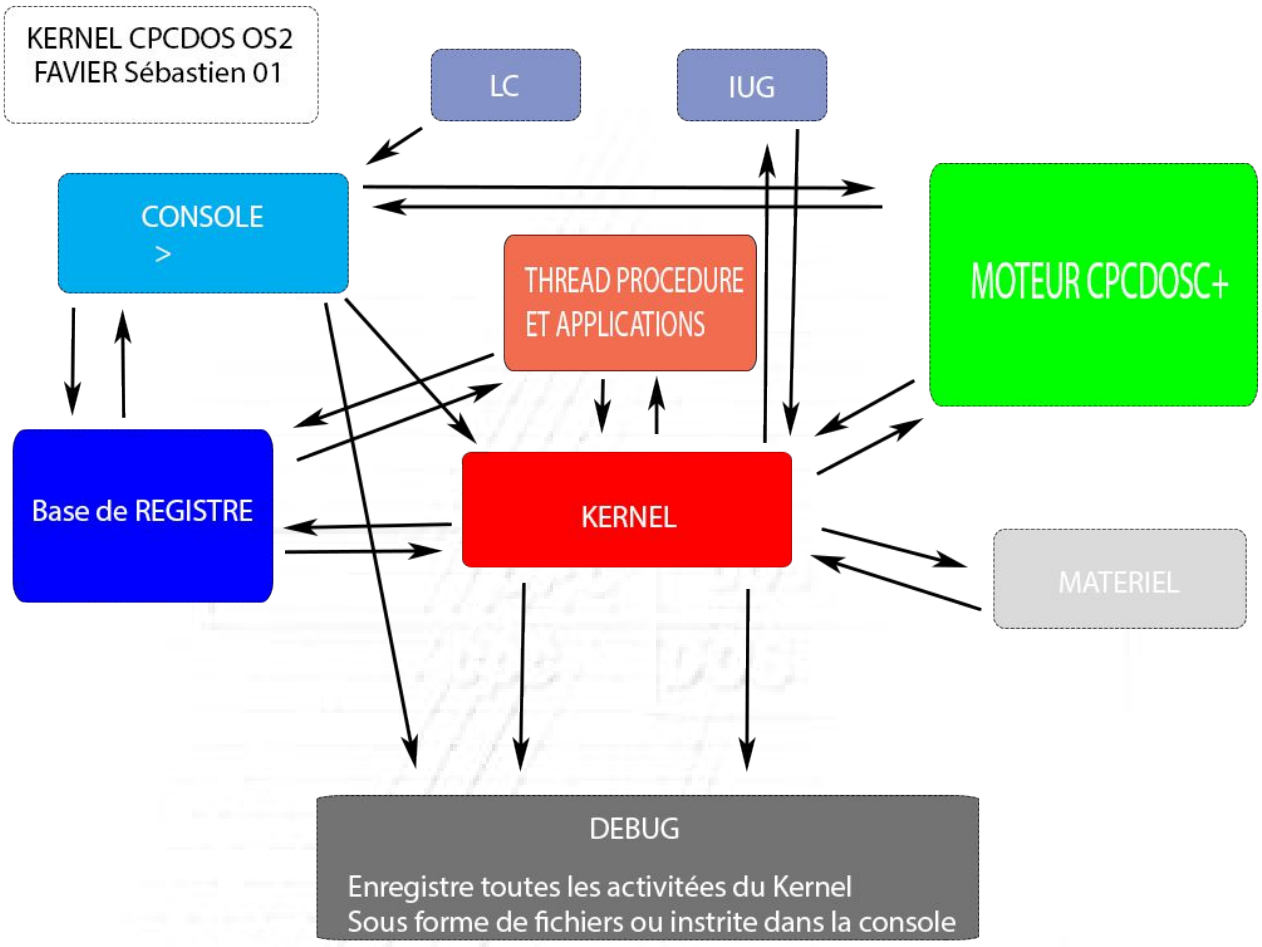
Monolithique modulaire : Les parties fondamentales du système sont regroupées en un bloc dans le code source.

Multitâche Coopératif : Gestions multitâche de plusieurs processus, qui communiquent en eux , son inconvénient c'est que si un des processus est défaillant , le système entier peux bloquer , *voir Section critique*

Son principe c'est d'exécuter du code CpcdosC+ en lignes de commandes (en Console) ou via des fichiers de format

- .CPB : **CCB** (Code Compilé Binaire *Non modifiable*) en utilisant le principe du ByteCode
- .CPC **CNB** (Code Non Binaire *Modifiable*|*Type texte*)

Schéma du fonctionnement du noyau



-

LC (Lignes Commandes) :

Partie en lignes de commandes semblable à la partie console
Elles permettent l'introduction de l'interface "Console" en ligne de commandes

IUG (Interface Utilisateur Graphique) :

Comme son nom l'indique , c'est tout simplement l'interface graphique où l'utilisateur interagit avec les objets etc...

Console :

Partie où l'on saisie les commandes CpcdosC+

Base de REGISTRE :

Partie du système , elle fournis et enregistre les informations et paramètres système du Kernel.
Il est aussi liée a la Console car on peu interagir au registre via la console avec la commande *REG/*

THREAD PROCEDURE ET APPLICATION :

Partie où les informations des propriétés et objets sont placées , et utilisées afin que la partie Kernel dessine
sur l'interface

MOTEUR CPCDOSC+ :

Partie du système où toutes les commandes CpcdosC+ sont analysée et exécutées par la partie KERNEL
C'est aussi ici où le système gère le multitâche des commandes pour donner les priorités au système

MATERIEL :

Partie où le Kernel gère le clavier , souris , affichage , imprimantes , USB etc...

KERNEL :

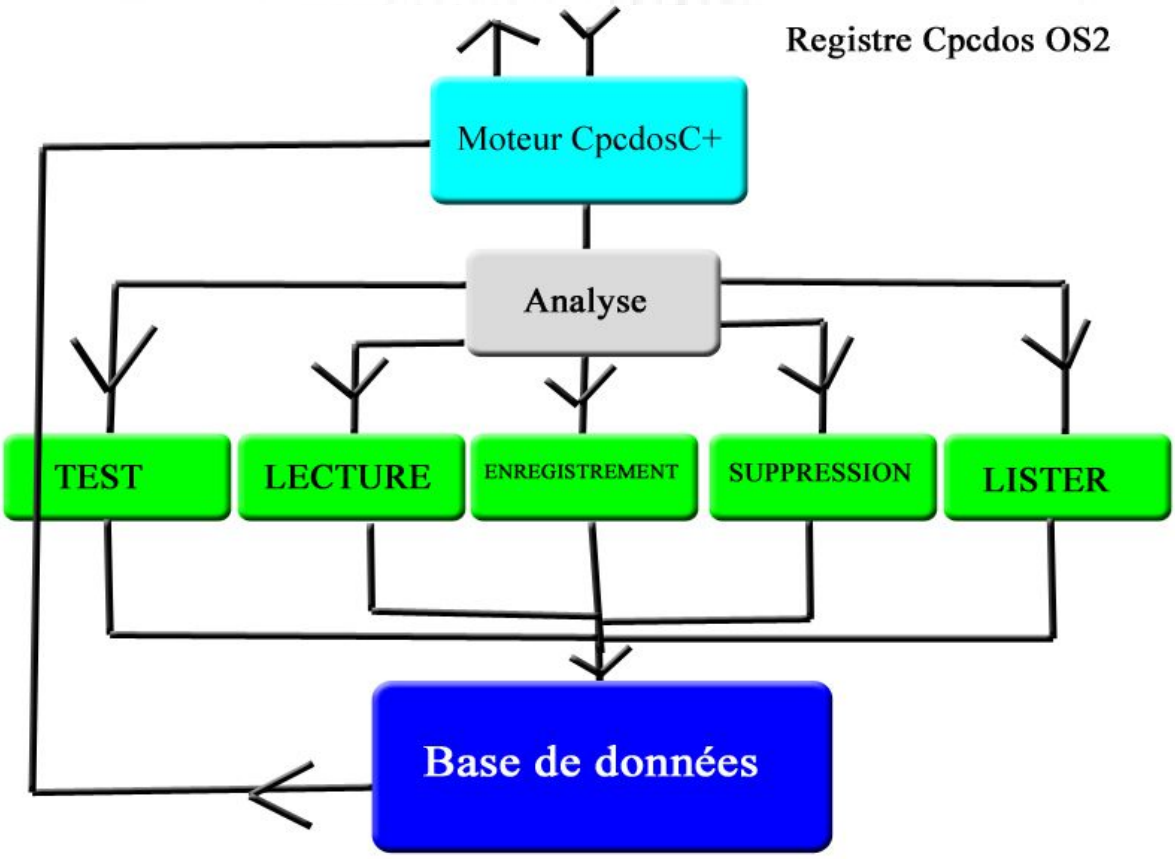
Partie NOYAU .. tout simplement celui "qui gère tout"..

DEBUG :

Partie assez importante qui enregistre ou affiche à la console , toutes le activités du Kernel !

Nb : Le Kernel est une surcouche au dessus du DOS

Schéma du registre :



L'utilisation du code CpcdosC+ s'utilise dans 2 façons différentes !

La première dite CCB (Code Compilé Binaire)

Puis , CNB (Code Non Binaire)

C’est-à-dire que le CCB deviendra un exécutable direct au Kernel qui lui le code CpcdosC+ sera convertit Assembleur puis deviendra un fichier Binaire.

Et le CNB est comme un fichier de paramètres , indiquant juste au Kernel les fonctions d’interface avec du code de type texte , modifiable , personnalisable etc ..

Par exemple le CNB que l’utilisateur créera , pourra être un MsgBox , une simple fenêtre etc ..

Chapitre II - Histoire du CpcdosC+

Le premier langage de programmation développé par Microsf01 était pour le 3eme Cpcdos sur Amstrad Cpc 464 il se nommait **CpcCmD** à la base il était écrite en Basic 1.0 en 2006 , ce langage ne servait pas a créer un programme , il était fait de façon à qu’il ressemble à l’interpréteur de commandes MS-DOS.

Le Second langage de programmation développé par était le CPCOMMAND sur Windows avec le l’OS virtuel Cpcdos 4.5 *écrit en Visual Basic 5 , Batch(ms-dos) , C++*, il servait de commutateur pour le programme cet à dire que Cpcdos 4.5 avait un noyau avec une interface préprogrammé le noyau était en liaison avec les API de Windows

Ce langage contient des commandes qui ont la base de :

- IO (gestions de fichiers , lecture/écriture)
- Réseau (gestion de serveur IP et utilisation des protocoles de Windows avec le dossier de partage etc ...)
- Création d'interface (fenêtres, événements...)

Le 3eme (aujourd'hui) c'est le **CpcdosC+**

acronyme de **Cpcdos C**ommande+

Cpcdos : Crée Pour Concevoir Des OS

initiales : CC+ ou CCP

Ce langage est utilisé dans 2 types

- CpcdosC+ pour systèmes d’opérations
- CpcdosC+ pour Jeux (Microsf01 Games 32 bit) (Projet actuellement abandonné)

Chapitre III – Exigences minimale du système

Avant tout , il faut faut bien évidement , un PC (et non un Mac !)

Configuration minimum requis pour le PC :

Processeur : 800 *mhz* Intel ou Amd

RAM : 128 Mo 133Mhz

Carte graphique : 8 Mo supportant le VGA , EGA , SVGA , VESA

Disque dur : au moins un 1 Go

Un processeur dans les alentours de

INTEL

Intel (P5) : Pentium

Intel (P6) : Pentium II , Celeron , Pentium III

Intel (&NetBurst) : Pentium 4

Intel (P6) Core 2 Duo

et plus...

AMD

Fonctionnelle mais mal côtoyé au niveau de la syntax Intel

Amd Am386

Amd K5

et plus

Systèmes BIOS

Phoenix , Award Software

(Energy)

ARM : Ne fonctionne pas

Installation sur USB :

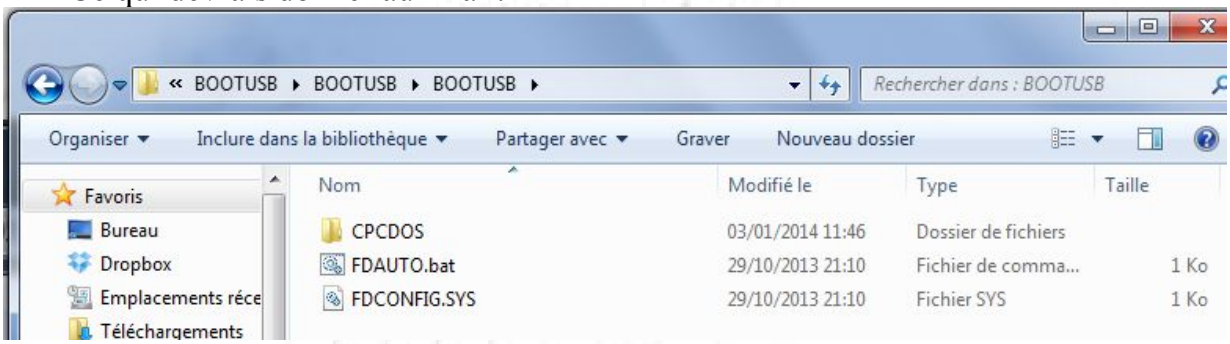
1. Pour commencer, téléchargez le logiciel *RMPrepUSB*
   sur **http://www.rmprepusb.com/documents/release-2-0**
   (en bas de la page)
   Version conseillée «Install_RMPrepUSB_Full_v2.1.709a.zip»

2. Extraire le contenu du fichier BOOTUSB.ZIP dans un dossier

3. Si vous avez déjà utilisé/installé le noyau sur votre ordinateur copier le dossier «CPCDOS» et collez-le dans le répertoire où vous avez extrait BOOTUSB.ZIP

   Si vous êtes un débutant/nouveau qui découvre le noyau
   Copier le contenu du répertoire «Racine a placer» dans le répertoire où vous avez extrait BOOTUSB.ZIP

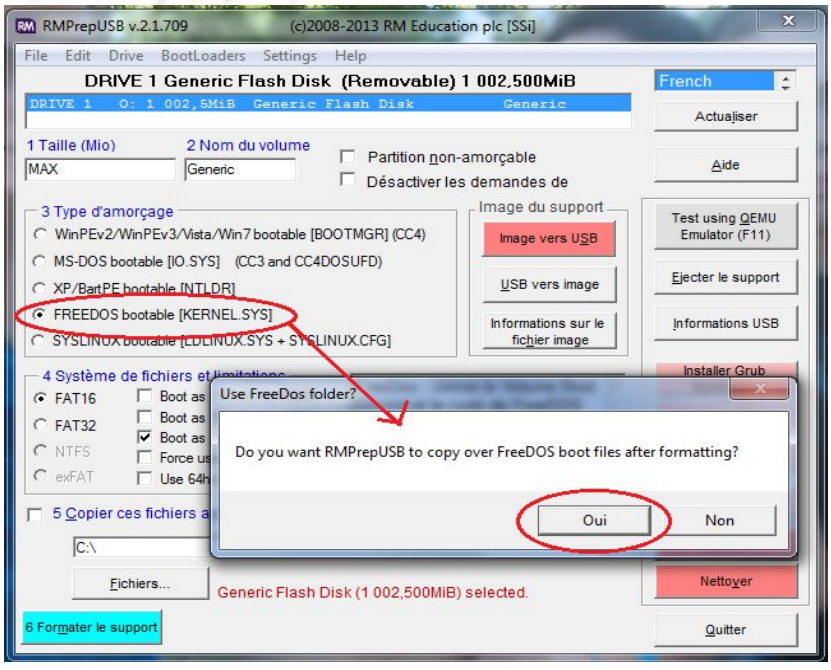
   Ce qui devrais donner au final :



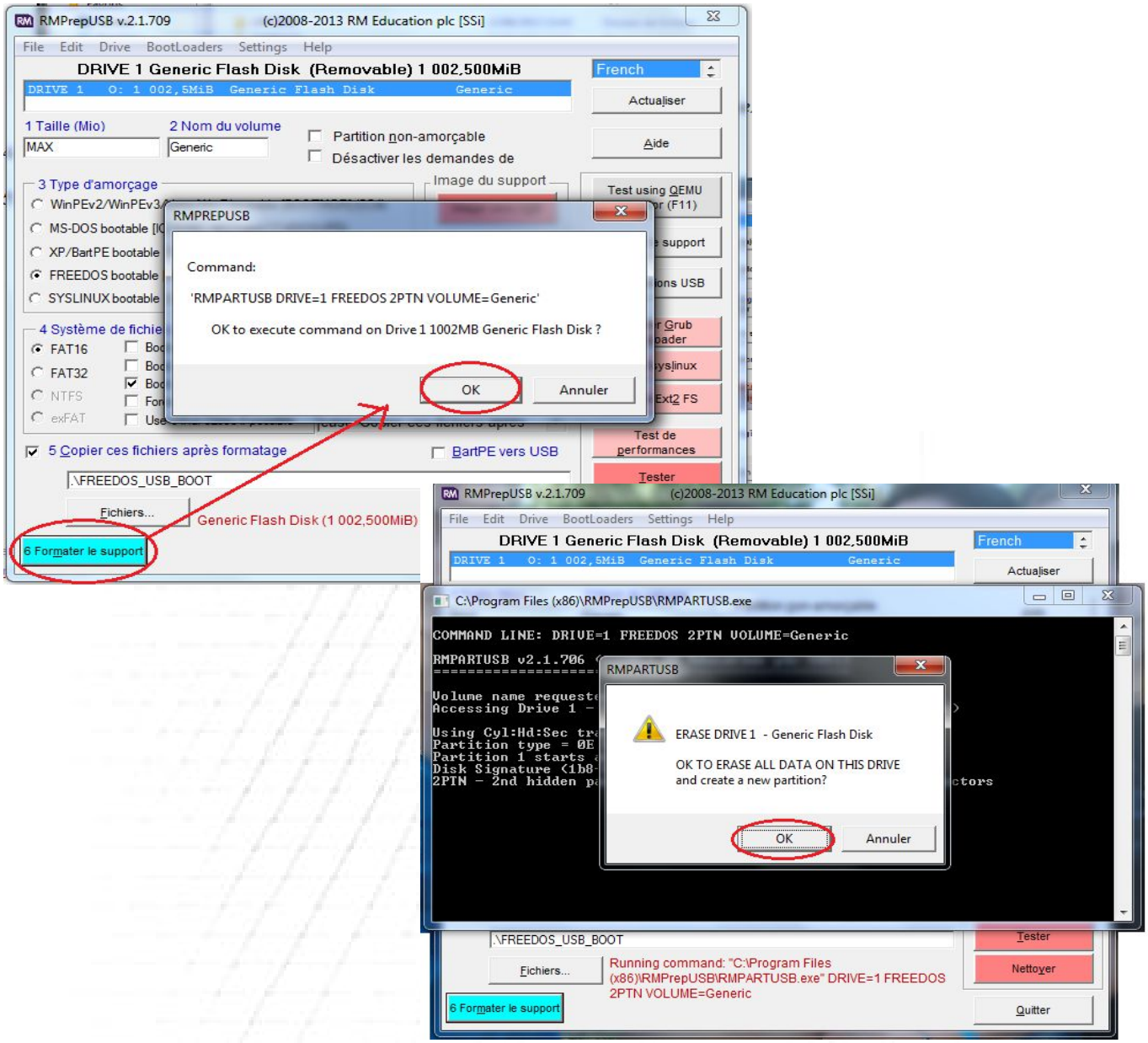
   Dans le dossier Cpcdos vous devrez avoir les dossiers comme «pilotes & Systeme»

4. Insérez votre clé USB (à formater) et ouvrez le logiciel *RMPrepUSB. Nous allons installer FreeDos*

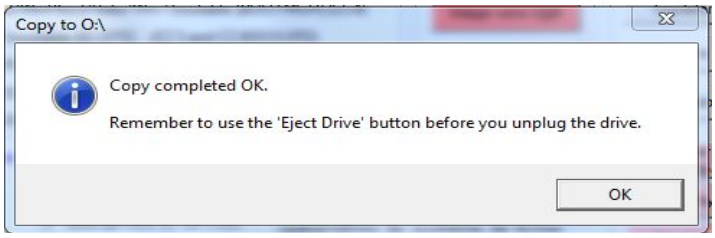
5. Sélectionnez «*FREEDOS Bootable [KERNEL.SYS]*» Et valider le message par Oui



6. Cliquez sur Formater le support
   et validez les deux messages

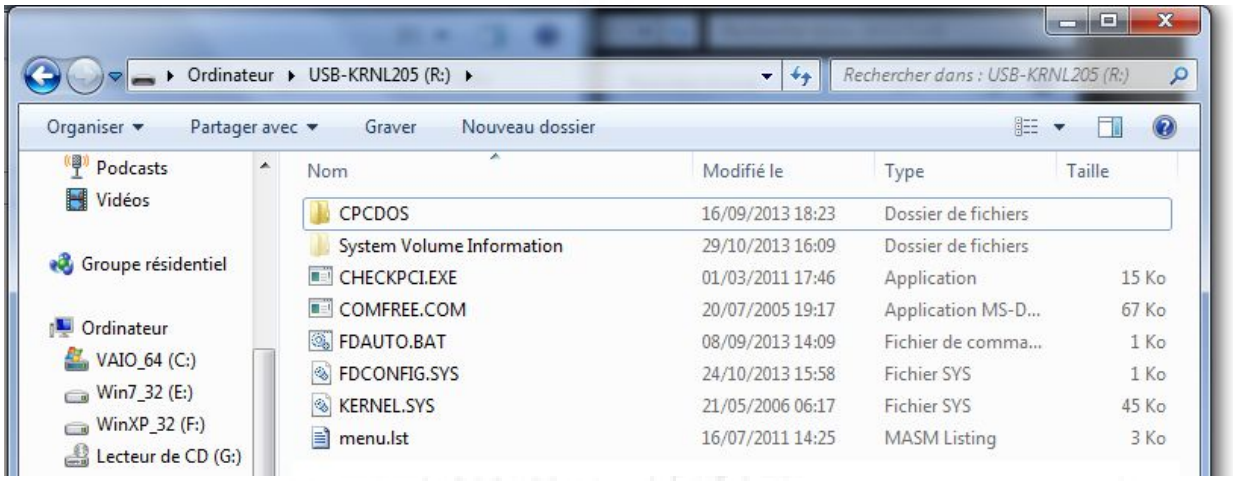


Vous devrez avoir ce message



7. Copier le contenu du répertoire où vous avez extrait BOOTUSB.ZIP dans votre clé USB et confirmer par oui pour remplacer 1 ou plusieurs fichiers

8. Vous devrez avoir ceci :



9. Branchez votre Clé USB sur votre machine, assurez-vous d'avoir paramétré le BIOS pour le BOOT en USB et démarrez votre machine et laissez la clé booter

Et puis voilà ! :)

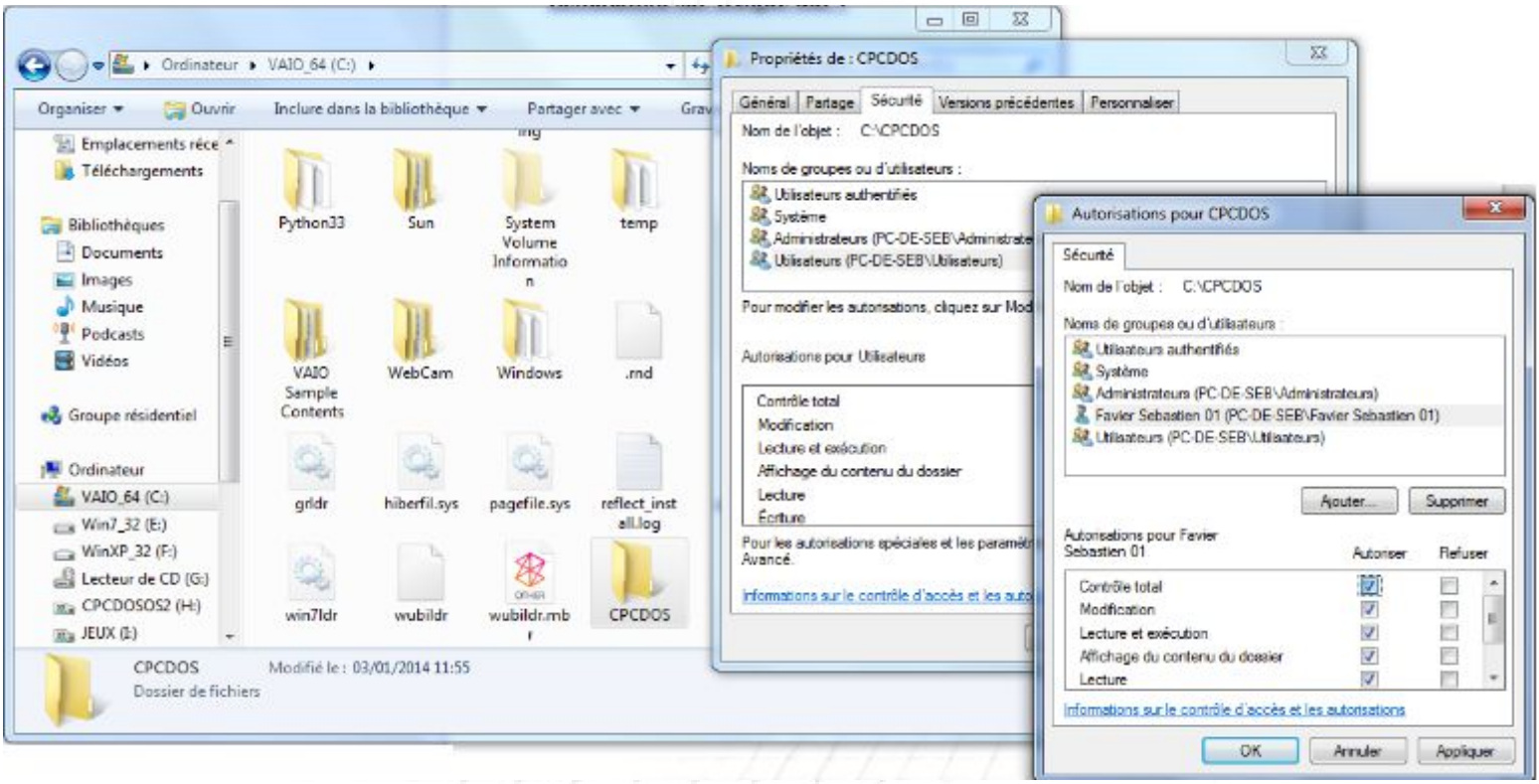
Installation sur Disque dur :

{ Page réservé à l'installation du Kernel sur un disque dur }
Prochainement sur le Doc 2.2014

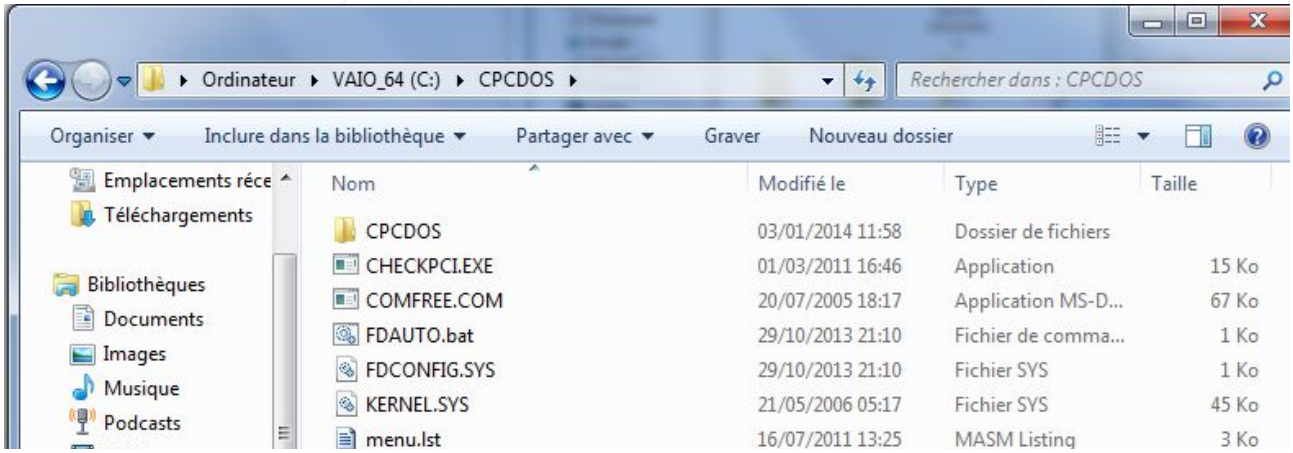
Installation avec DosBox :

Pour commencer, une fois que vous avez le Kernel en main
veuillez télécharger et installer D-Fend
ou à l'adresse suivante : **http://dfendreloaded.sourceforge.net/Download.html**

Une fois téléchargé, créez le répertoire nommé **DosBox** dans **C:\ (ou autre)**
Si vous êtes sur un système NTFS donnez lui tous le droits à ce dossier

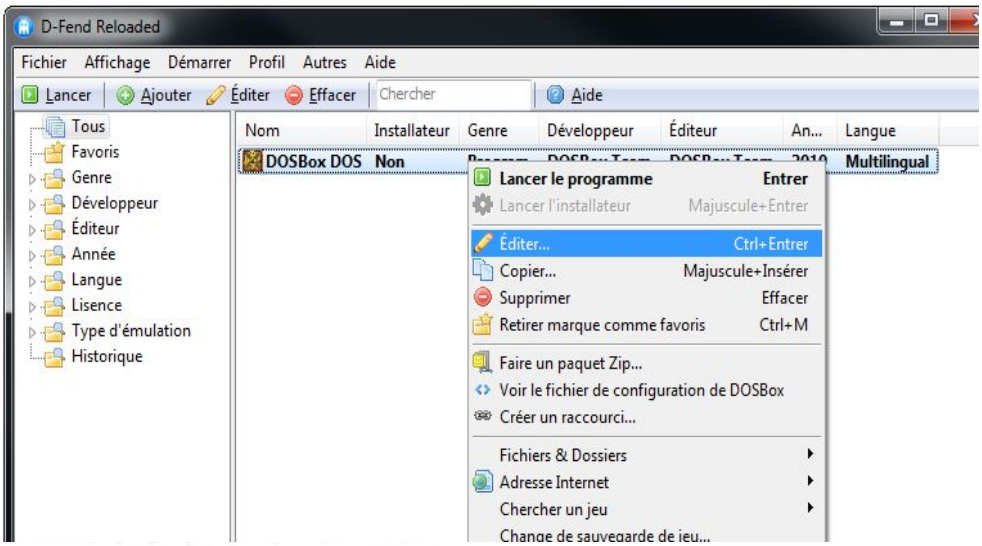


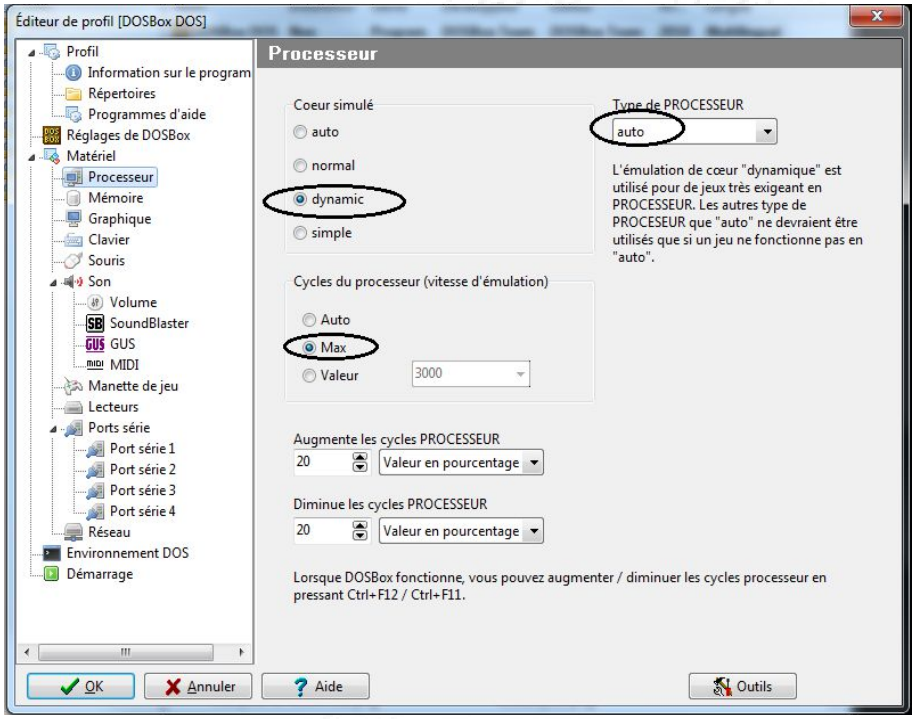
et copiez le contenu du dossier « RACINE A PLACER » dedans.
Vous devrez avoir ceci :



Puis lancez D-Fend

Clique droit sur DosBox, Éditer

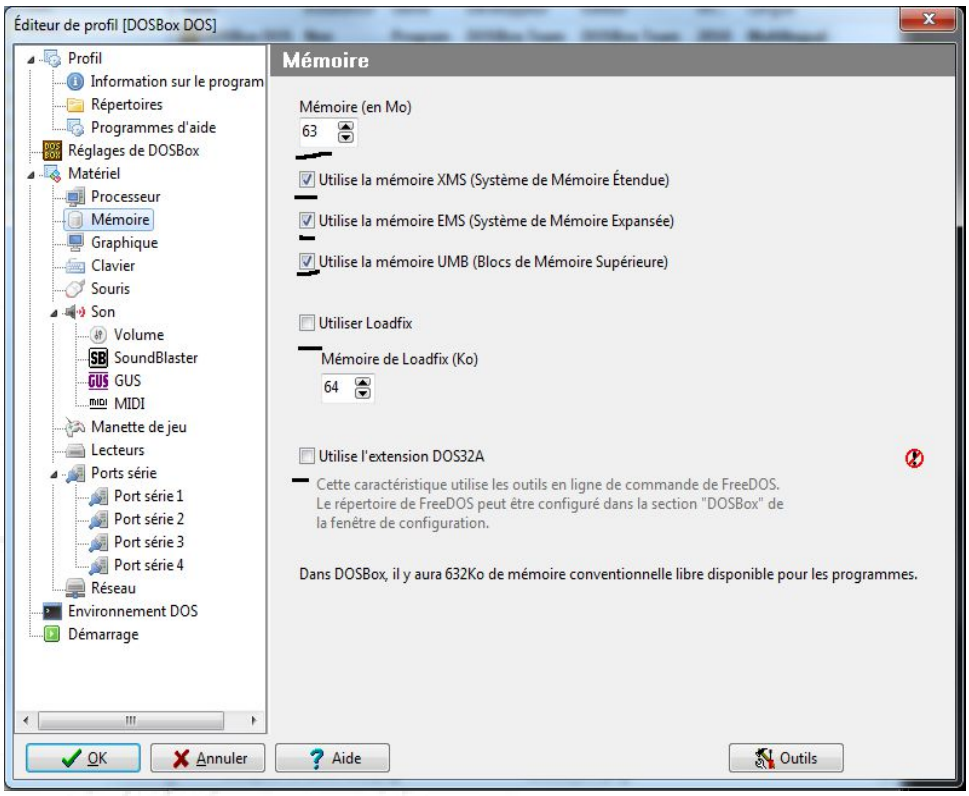




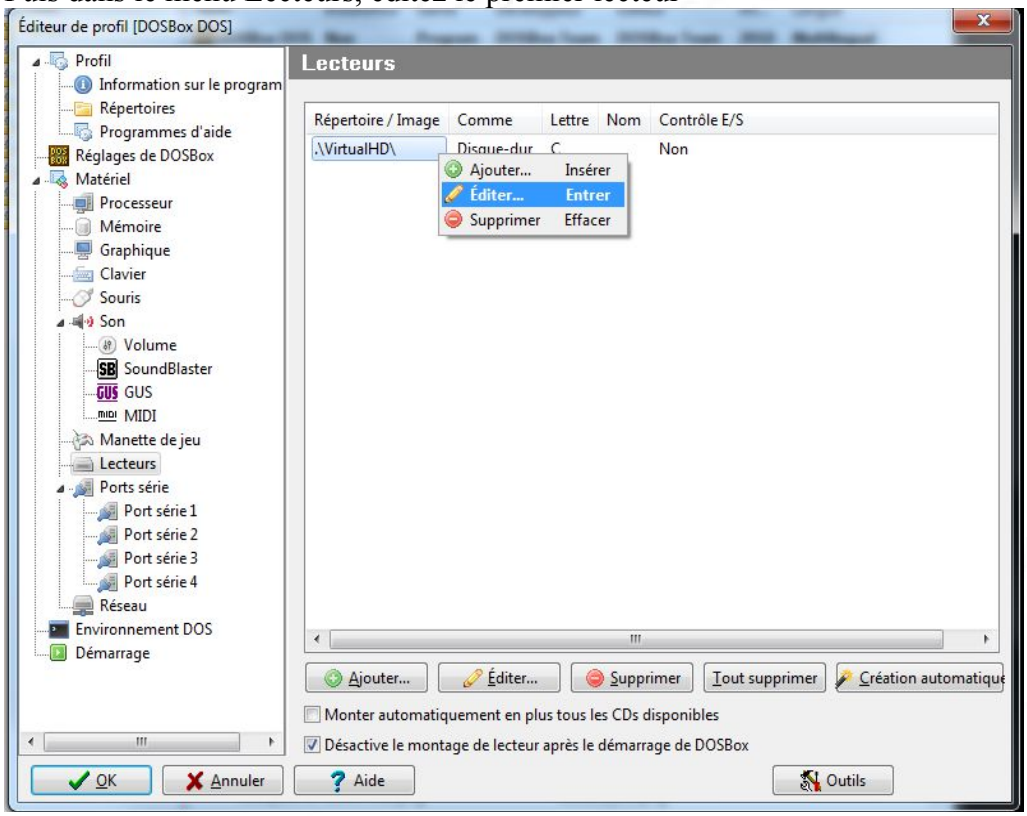
Choisissez et cochez

pour le cœur simulé, un **DYNAMIC** , les cycles du processeur à **MAX** et le type de processeur, un **AUTO**

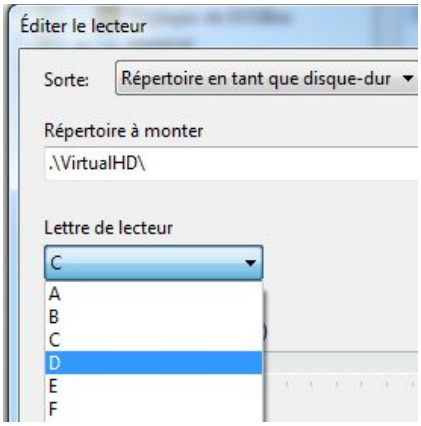
Choisissez la mémoire à 63 mo (maximum) puis cochez toutes les mémoire
XMS
EMS
et UMB
décochez LoadFix
et l'extension DOS32A



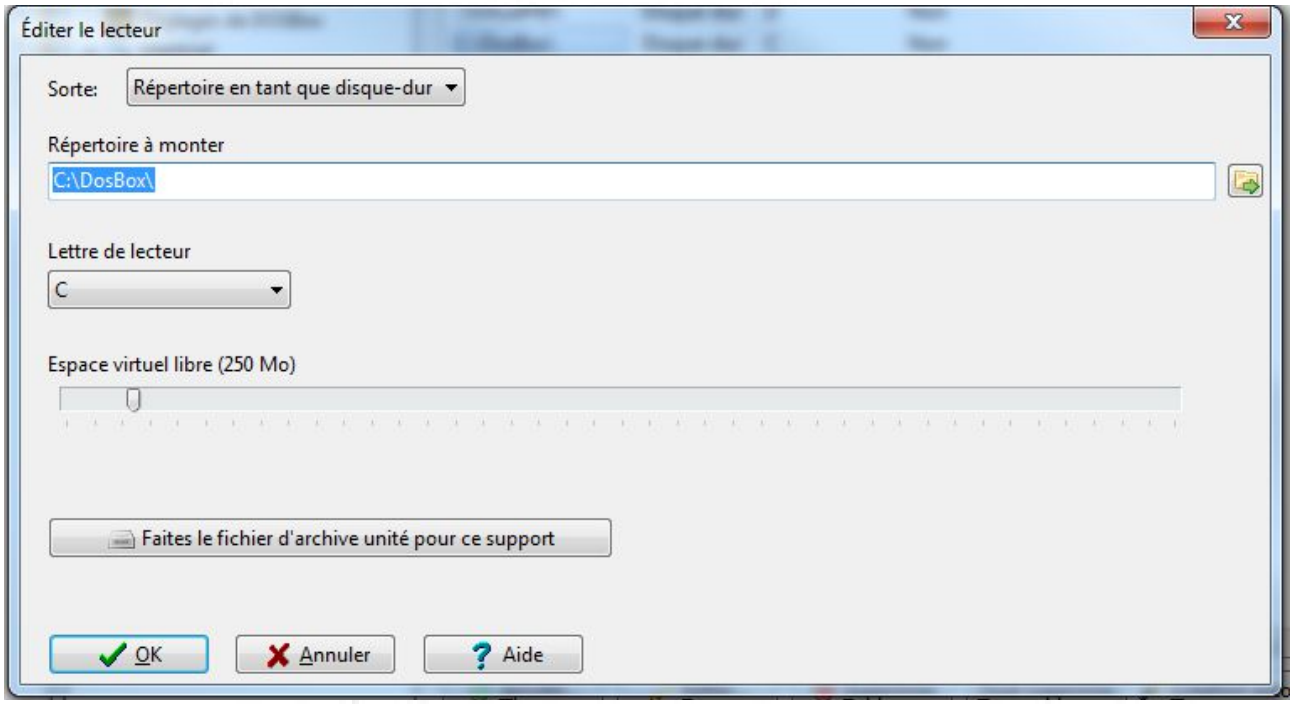
Puis dans le menu Lecteurs, éditez le premier lecteur



Choisissez tant que lecteur D
puis valider



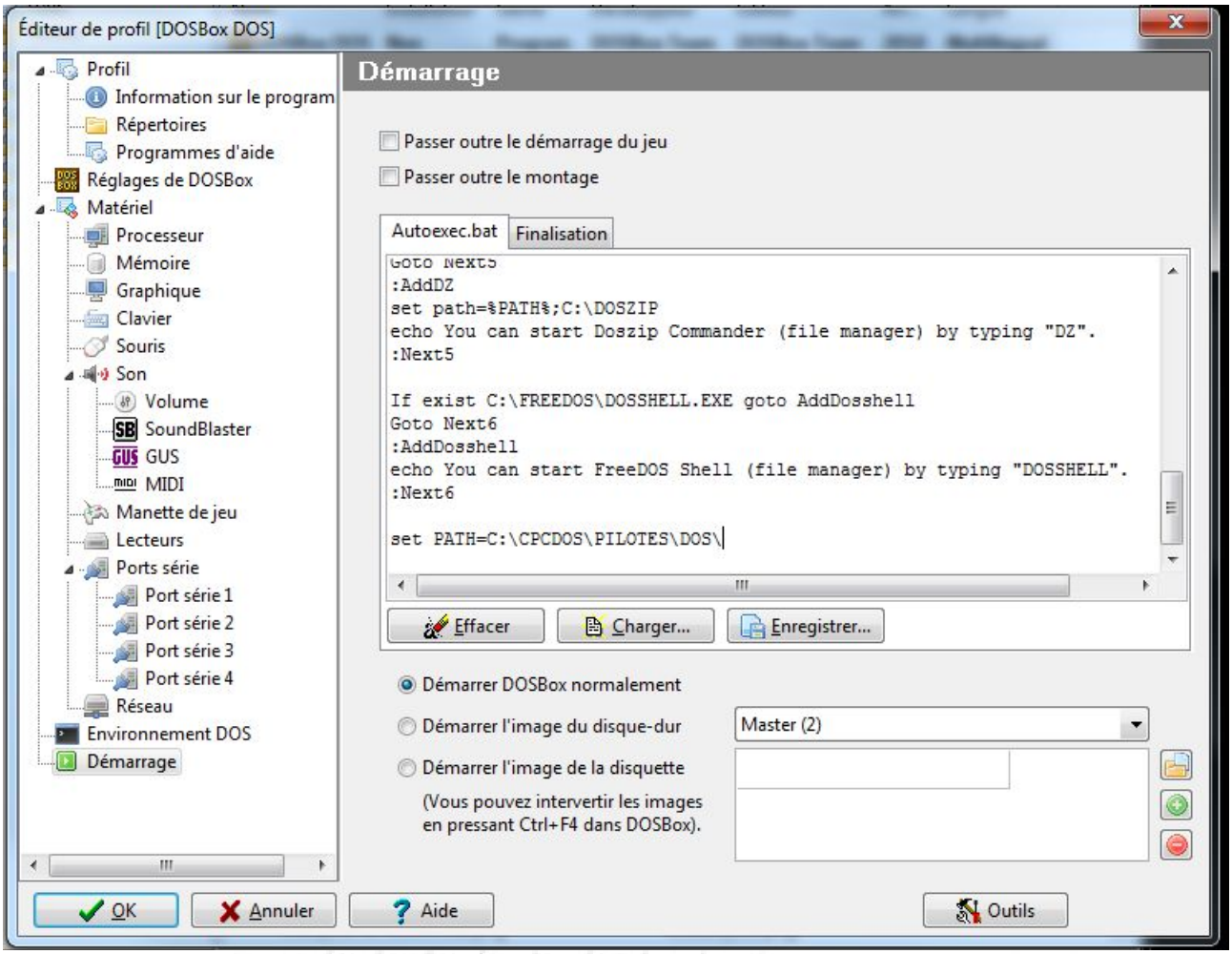
Puis Cliquez sur Ajouter
Créez un lecteur dont « répertoire tant que disque dur »
Ciblez le dossier que vous avez précédemment crée « DOSBOX »
et déclarez le comme un lecteur **C** puis validez



Puis dans le menu **Démarrage** ajoutez la dernière ligne

```
set PATH=C:\CPCDOS\PILOTES\DOS\
```

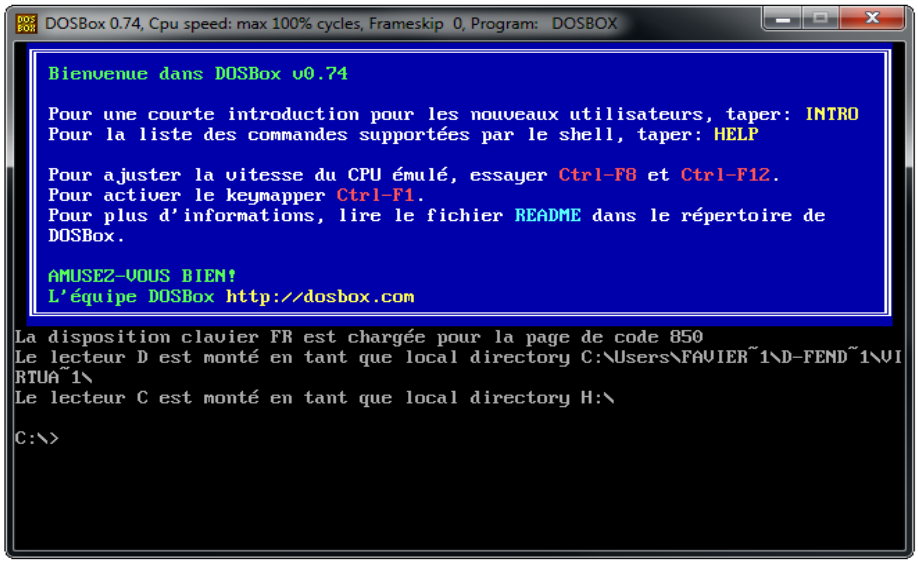
Afin de permettre d'utiliser les commandes et programmes DOS où que vous soyez



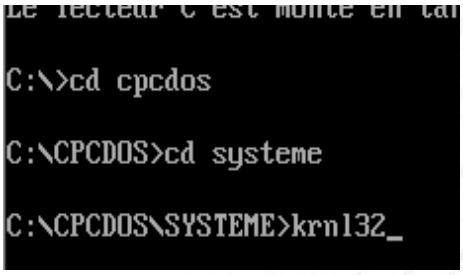
Puis validez le tout

Lancez DosBox (double clic)

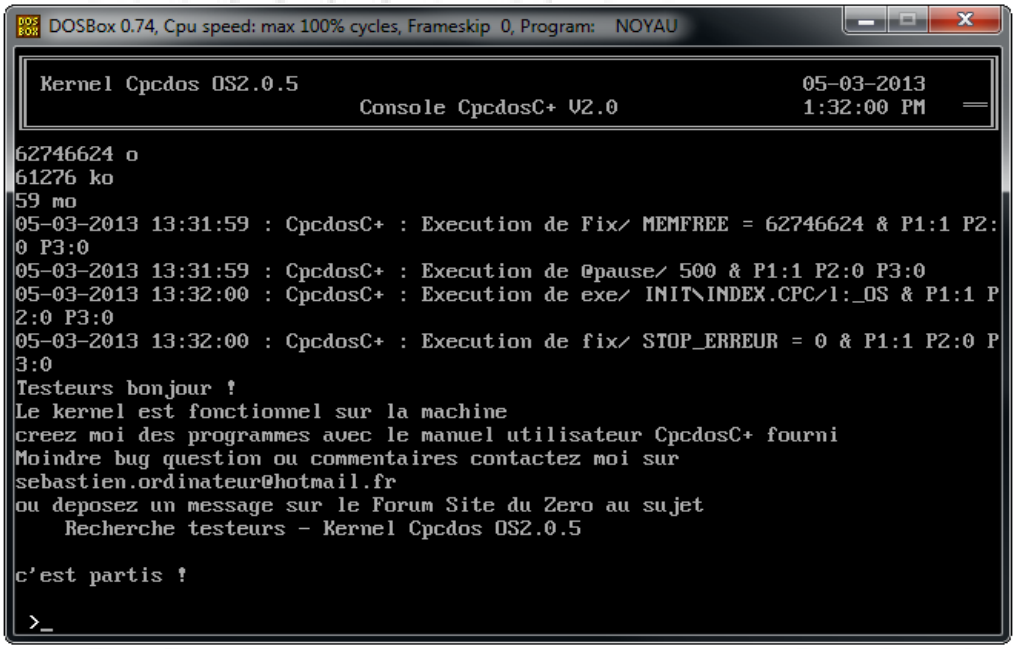
Vous devrez avoir ceci :



Puis tapez ces commandes



Le Kernel s'initialise et vous devrez tomber sur ceci :



C'est que le Kernel est fonctionnel !
Maintenant pour créer vos programmes, allez dans le répertoire que vous avez créer par défaut **C:\DosBox**

puis dans cpcdos\systeme
Créez un fichier de nom que vous voulez, qui servira pour tester et programmer par exemple : « PROG.CPC »

(Attention que le nom dépasse pas 8 caractères et l'extension, 3 caractères)

Ce fichier s'ouvre avec un simple éditeur de textes, utilisez par exemple Bloc-Notes de Windows

puis pour exécuter votre fichier pour tester votre programme, sur DosBox appuyez sur CTRL+F4 pour mettre à jour les fichiers et dossiers dans l'émulateur dosbox et lancez le Kernel (si c'est pas déjà fait)
Puis taper à la console

```
exe/ PROG.CPC
```

Puis voilà suivez bien le manuel afin d'être pas paumé hein, vous avez toutes les commandes expliqués !

Moindre questions ou beugues ou fautes sur le manuel, contactez moi par E-mail :
sebastien.ordinateur@hotmail.fr
ou sur le Forum **http://forum-cpcdos.fr.nf/**

Chapitre IV - Commandes de bases

J'ai dédié ce manuel de ce langage uniquement pour le noyau Cpcdos OS2 , donc si vous voulez être développeur Cpcdos , vous pouvez apprendre a l'utiliser ici.

Cette partie IV sera les commandes de base que l'on utilise principalement en mode console

Avant tout,

A LIRE (Important) :

- *Pas de différenciation entre les commandes et paramètres en majuscules/minuscules*
- *Autant de tabulation ou espace entre le début de ligne et la commande*
- *Utilisez des nom de propriétés différent et toujours en majuscules*
- *L'attribution des valeurs de couleurs R.V.B doivent être de 3 caractères numériques*

  ex : Ne pas mettre « 1,50,30 » . *mais* « **00**1,**0**50,**0**30 » *(rajouter des zéros au début pour faire 3 caractères de chaque) vous pouvez choisir d'autres caractères que des virgules « , »*
- *LC : Lignes de Commandes*
- *IUG : Interface Utilisateur Graphique*
- *CCB : Code Compilé Binaire*
- *CNB : Code Non Binaire*
- *SCI : Service Création Initialisation, le service créateur/dessinateur d'interface graphique*
- *Moteur CCP : Moteur de traitement des commandes CpcdosC+ et liaisons du Kernel*
- *Ajoutez «* **fix/ DEBUG = 0** *» au début de votre programme afin d'éviter d'afficher les informations d'exécution de commandes*
- *Ajoutez «* **fix/ LOG = 0** *» pour permettre l'enregistrement des informations de débogage dans SYSTEME\DEBUG.LOG (ralenti un peux le système)*
- *Les caractères suivants : |» veulent dire que la ligne continue avec celle de en-dessous (manque de largeur du manuel..)*
- *Pour les caractères comme « é è â ä »... utiliser le format ANSI / ASCII*
- *Pour utiliser une commande fantôme, il suffit de placer « @ » juste au début de votre commande ex :* **@txt/ Ma commande fantôme**
  Rediriger les sorties ex : ***@#VARIABLE SYS/ /TESTECR 16*** *(enregistre dans VARIABLE)*
- *Pour récupérer les propriétés dans une* `variable` *, il suffit d'utiliser «#%» exemple :*
  `ini;propriété = "#%VARIABLE"`
- *Pour ré-exécuter la dernière commande tapé, exécutez* **>** *ou l'avant dernière* **>>** ***ou*** **>>>**
- *Pour cette version, les virgules « , » ne fonctionne pas en interprétation sur la console*

AFFICHER L'AIDE

La commande qui le permet est :

```
aide/
```

Vous pouvez aussi demander la fonctionnalité la commande exemple :

```
aide/ fix/
```

```
aide/ txt/
```

EFFACER L'ECRAN

La commande qui le permet est :

```
cls/
```

(Clean Screen **)**

Permet d'effacer l'écran,

Exemple :

```
txt/ Appuyez sur une touche pour effacer l'écran
Pause/
cls/
txt/ Écran efface !
```

COMMENTAIRES

La commande qui permet ceci est :

```
rem/
```

Cette commande permet tout simplement au développeur d'écrire des commentaires dans son code pour se repérer etc …

Aucunes influence sur le Kernel, vous pouvez mettre autant de conneries que vous voulez !

AFFICHAGE TEXTE

La commande qui permet ceci est :

```
txt/ {texte}
```

(**TeXTe**)

Cette commande est utilisable uniquement en mode LC , il permet l'affichage de caractères ASCII de codage DOS dans une plage de caractères allant de 0 à 255

par exemple :

Écriture basique :

```
txt/ Hello word
```

Afficher le contenue d'une variable :

```
fix/ VAR1 = utilisateur
fix/ VAR2 = de l'ordinateur
txt/ Coucou %VAR1% %VAR2%
```

Sortie :

Coucou utilisateur de l'ordinateur

Des couleurs ?

```
Couleurf/ 2
couleurp/ 1
txt/ Hello word !
```

Affiche Hello Word ! En bleu (1) sur vert (2)

POSITIONNER CURSEUR (CONSOLE)

La commande qui permet ceci est :

```
posx/ {valeur}
```

et

```
posy/ {valeur}
```

Cette commande permet de positionner le curseur de la console à une zone définit par l'utilisateur

ex :

```
txt/ Position d'origine x:%CURPOSX% et y:%CURPOSY%
posx/ 5
posy/ 6
txt/ Nouvelle position x:%CURPOSX% et y:%CURPOSY%
```

NB : Si vous voulez masquer le menu de la console, il suffit de fixer la variable CONSMENU à 0
(1 : affiché)

```
fix/ CONSMENU = 0
```

CREER UNE VARIABLE/TABLEAU

La commande qui le permet est :

```
fix/ {variable} = {Données}
```

(**Fix**er (Set))

Cette commande permet de créer une variable locale , (limité à 255 Ko)

Pour déclarer une variable sans contenu, il faut que la variable sois égale à « ***#NUL*** »

Puis pour « poser une question » Il faut utiliser ce paramètre :

```
fix/ /q {Variable}
```

(q : Question)
La variable aura le contenu que l'utilisateur à tapé au clavier puis validé avec la touche ENTRER
Bloque si l'IUG est en exécution, tapez un contenu dans le vide et valider ENTRER

Pour supprimer une variable de la mémoire :

```
fix/ /s {Variable}
```

(s : Supprimer)

Pour lister les variables de la mémoire :

```
fix/ /liste {Variable}
```

Et si la liste est grande, vous pouvez lister 1 par 1 pressez ESPACE pour lister et ECHAP pour arrêter :

```
fix/ /liste /pause
```

Exemple :

```
fix/ NOM = Thomas
txt/ Bonjour %NOM% quelle age avez-vous ?
Fix/ /q AGE
si/ %AGE% > 17 (:txt/ Vous etes majeur %nom% !:)
si/ %age% < 18 (:txt/ Vous etes encore jeune %nom%:)
@fix/ /s NOM
@fix/ /s AGE
```

NB : le caractère « @ » permet d'exécution fantôme de la commande

Et pour créer un tableau, exemple :

```
fix/ index = 2
fix/ MON_TABLEAU(%index%) = Blablabla123

rem/ Puis pour afficher :
txt/ %MON_TABLEAU(index)%
```

OU

```
fix/ MON_TABLEAU(2) = Blablabla123

rem/ Puis pour afficher :
txt/ %MON_TABLEAU(2)%
```

EXECUTER UN FICHIER CPCDOSC+

La commande qui le permet est :

```
exe/ {fichier}
```

(**exe**cuter)

Cette commande permet d 'exécuter un fichier de commandes CpcdosC+ uniquement

Le paramètre « /l: »

```
exe/ {fichier} /l:
```

Ce paramètre permet d'indiquer a partir de quelle label ou de quelle ligne le code s'exécute

Exemple :

```
exe/ Fichier.cpc /l:monlabel
```

```
exe/ Fichier.cpc /l:#3                    {#3 = Ligne N°3}
```

Et dans Fichier.cpc :

```
txt/ texte1
monlabel:
txt/ texte2
```

Sortie :

texte2

Exécuteur a **double** Threads

Il existe un paramètre permettant d'exécuter en second thread, un 2eme fichier CCP jusqu'à la fin et reprendre l'exécution du premier exécuté il suffit d'utiliser le caractère AND « & » :

```
exe/ & Fichier.cpc
```

ARRÊTER L'EXECUTION D'UN FICHIER CPCDOSC+

La commande qui le permet est :

```
stop/
```

Cette commande permet d'arrêter l'exécution d'un fichiers de commandes CpcdosC+ en cours

Il devrait être utilisé dans un fichiers.

Une autre commande est semblable , ne pas confondre, il sert a stopper complètement l'exécution du Kernel sans prévenir le SCI et la procédure de vidage mémoire et arriver directement à l'interpréteur de commandes ms-dos ou FreeDos.

La commande qui le permet est :

```
stopk/
```

(**stopk**ernel)

CONDITIONS

La commande qui le permet est :

```
si {interrogé} {Condition =/N/</>} {Opérateur} (:{Commande}:)
```

Cette commande est une instruction conditionnelle.

Exemples :

```
fix/ VAR1 = 5
fix/ VAR2 = 2
fix/ RES = /c %VAR1% + %VAR2%
si/ %RES% = 7 (:txt/ %VAR1% + %VAR2% est egale a 7 !:)
```

NB : `/c %VAR1% + %VAR2%` permet de calculer les deux variables, voir partie **FONCTIONS**

Les signes de conditions utilisables sont :

- **Égale à** de signe « = »
- **N'est pas égale à**, de signe « N »
- **Supérieur à**, de signe « > »
- **Inférieur à**, de signe « < »

EXECUTER UN FICHIER EXECUTABLE DOS/WIN32

La commande qui le permet est :

```
shell/ {fichier exécutable}
```

Cette commande exécute des fichier de format MZ (*.exe .com)* et interpréteur DOS *.BAT*

Exemple :

```
shell/
```

Pour exécuter un programme WIN32, il suffit d'indiquer le paramètre suivant :

Exemple :

```
shell/ /win32 {fichier exécutable}
```

A prevoir que le Kernel n'est pas en mesure d'exécuter des programme ayant une interface graphique Windows

Mais compatible uniquement en format Console pour le moment

Vous pourrez exécuter des programme type console codé en

C++ , Code::Block, turboC etc ...

Une commande voisine existe et remplit presque la même fonction :

```
dos/
```

RECUPERER LES ENTREES AU CLAVIER

La commande qui le permet est :

```
touche/ {variable}
```

Cette commande exécuté sur suite enregistre dans la variable définit la touche que l'utilisateur à pressé

Elle ne met pas en attente le système jusqu'à interaction au clavier

Pour permettre l'attente, il suffit d'ajouter ce paramètre (*avant la variable)*

```
touche/ /p {variable}
```

Ce qui permet au système se mettre en pause, jusqu'à interaction au clavier
puis une fois la touche pressé, le « caractère » ASCII est enregistré dans la variable définit.
Si on appuie sur la touche a il affiche un message
Si on appuie autre chose que a il affiche autre chose

Exemple :

```
fix/ debug = 0
fix/ variable = 0
touche/ /p variable
Si/ %variable% = a (:txt/ pas touche au A!!:)
si/ %variable% N 0 (:txt/ Touche %variable% pressée!:)
```

Un autre exemple, mais sans le paramètre « /p » :

Dans cet exemple, si on appuie sur autre chose que 0 il affiche la touche qu'on a pressé
Si on appuie sur q le programme est stoppé

```
fix/ debug = 0
:debut:
fix/ variable = 0
touche/ variable
si/ %variable% N 0 (:txt/ Touche %variable% pressée!:)
si/ %variable% = q (:stop/:)
aller/ debut
```

Le programme fait une boucle

NB : Reste à noter que pour la version actuel, les touches < = > et N risquerons de ne pas fonctionner.

METTRE EN PAUSE LE SYSTEME

La commande qui le permet est :

```
pause/
```

Cette commande en console permet de mettre en pause le système complètement jusqu'à que l'utilisateur appuie sur une touche au clavier

un paramètre est disponible, c'est celui du temps de pause

exemple :

```
pause/ 1500
```

Ce paramètre met en pause 1 secondes et 500 millisecondes
Le temps s'écoule directement si l'utilisateur appuie sur une touche

PRENDRE UN SCREENSHOT (IUG / LC)

La commande qui le permet est :

```
src/
```

Cette commande permet de prendre un screenshot de l'écran de format BMP 24bit à la résolution actuelle
la photo est enregistré dans le répertoire *SYSTEME\SCR*

Pour prendre un screenshot manuellement en mode IUG, presser la combinaison des touches ALT+S

ps : Crash en mode LC *[en correction]*

EXECUTER UNE COMMANDE D'ENTREE

La commande qui le permet est :

```
cmd/
```

Cette commande donne la possibilité d'exécuter une commande qui se trouve dans une variable

exemple :

La commande qui le permet est :

```
fix/ MA_COMMANDE = txt/ Coucou !
cmd/ %MA_COMMANDE%
```

ou vous pouvez bien sur l'exécuter directement sans variables !

INTERACTION CPCDOS

La commande qui le permet est :

```
cpc/
```

Cette commande contient toute sortes de paramètres utilisable pour interagir sur cpcdos

Les paramètres disponibles pour cette version sont :

```
cpc/ /ARRETER
```

Pour arrêter le noyau avec un arrêt du système d'exploitation en cours

```
cpc/ /REDEMARRER
```

Pour redémarrer le noyau avec un arrêt du système d'exploitation en cours

```
cpc/ /CONSOLE
```

Pour lancer la console graphique **(Non terminée)**

```
cpc/ /REDUIRE {/TOUT | nom de fenetre}
```

Pour réduire une fenêtre
le paramètre /TOUT permet de réduire toutes les fenêtres ayant pas le paramètre R0
Vous pouvez aussi préciser 1 seule fenêtre, pour la réduire.

```
cpc/ /RESTAURER {/TOUT | nom de fenetre}
```

Pour restaurer une fenêtre
le paramètre /TOUT permet de restaurer toutes les fenêtres ayant pas le paramètre R0
Vous pouvez aussi préciser 1 seule fenêtre, pour la restaurer.

LIRE & ECRIRE DANS UN FICHIER (+BINAIRE)

La commande qui le permet est :

```
fichier/
```

Paramètres & exemples :
(Le numéro de canal c'est a votre choix! De 1 à 80)
n'oubliez pas de fermer le n° de canal avant de réutiliser le même ou.. utilisez-en un autre !
Ouvrir un canal pour écriture dans un fichier

```
fichier/ /SORTIE #{N°canal};{Fichier}
```

Ouvrir un canal pour écriture à la suite dans un fichier (Garde le contenu actuel et écrit à la suite)

```
fichier/ /SORTIEA #{N°canal};{Fichier}
```

uvrir un canal pour écriture binaire dans un fichier

```
fichier/ /SORTIEB #{N°canal};{Fichier}
```

Ouvrir un canal pour lire dans un fichier

```
fichier/ /ENTRE #{N°canal};{Fichier
```

Ouvrir un canal pour lecture binaire dans un fichier

```
fichier/ /ENTREB #{N°canal};{Fichier}
```

Lire un fichier (ligne par ligne)

```
fichier/ /LIRE #{N°canal};{Variable de sortie}
```

Lire un fichier binaire (ligne par ligne)

```
fichier/ /LIREB #{N°canal}.{position};{Variable de sortie}
```

Écrire dans un fichier (ligne par ligne)

```
fichier/ /ECRIRE #{N°canal};{Texte}
```

Écrire un fichier en binaire (ligne par ligne)

```
fichier/ /ECRIREB #{N°canal};{donnees}
```

Fermer un canal/fichier

```
fichier/ /FERMER #{N°canal}
```

Exemples :

```
REM/ Ecrire dans un fichier
FICHIER/ /SORTIR #1;C:\CPCDOS\SYSTEME\OS\FICHIER.TXT
FICHIER/ /ECRIRE #1;Coucou toi !
FICHIER/ /ECRIRE #1;Il est %TIME% et on est le %DATE%
FICHIER/ /ECRIRE #1;Bye !
FICHIER/ /FERMER #1
```

```
REM/ Lire dans un fichier en recuperant toutes les lignes
FIX/ RESULTAT = #NUL
REM/ Pour le retour chariot. Code ASCII 10
FIX/ CH10 = /C CHR >10
FICHIER/ /ENTRE #1;C:\CPCDOS\SYSTEME\OS\FICHIER.TXT
:BOUCLE:
REM/ EOF: End-Of-File 0:Fin non atteint -1:Fin atteint
REM/ Le numero 1 de la commande ci-dessous correspond au canal
FIX/ FIN = /C EOF >1
SI/ %FIN% = -1 (:Aller/ FIN:)
FICHIER/ /LIRE #1;MA_VARIABLE
```

```
FIX/ RESULTAT = %RESULTAT%%CH10%%MA_VARIABLE%
REM/ Enf-Of-File
Aller/ BOUCLE
:FIN:
FICHIER/ /FERMER #1
REM/ Afficher les lignes
TXT/ %RESULTAT%
```

ACTIVER/DESACTIVER UN SERVICE

La commande qui le permet est :

```
service/
```

La commande toute seule, elle affiche la liste des service installés et indique si ils sont en cours d'exécution ou en arrêt.

```
service/ {Nom du service}
```

Affiche si le service précisé est en cours d'exécution ou pas

```
service/ /? {Nom du service}
```

Affiche la description du service

```
service/ /ACTIVER {Nom du service}
```

Active le service

```
service/ /DESACTIVER {Nom du service}
```

Désactive le service

Vous pouvez créer votre propre service, il suffit de créer votre programme dans le dossier *CPCDOS\SYSTEME\KRNL\SERVICES*

Puis vous avez un fichier *Exemple.cpc* qui vous donne un exemple de la structure d'un fichier de services

Il faut bien respecter du fait qu'un service est un COMPTEUR avec un nom commençant par « SVC_ »

Et que selon le cycle, l'intervalle donné il exécute la procédure qui se trouve sur le fichier lui même.

Qu'il contient obligatoirement les labels suivants :

Quand le noyau fait une interaction il passe par ces labels :

```
:DESACTIVER:
:ACTIVER:
:DESCRIPTION:
```

DONNER LA PRIORITE AU SYSTEME

La commande qui le permet est :

```
doevents/
```

Cette commande met en pause l'exécution du code, et laisse une boucle au systèmes, services et timers du noyau afin d'éviter les «bloquages» lors ce que le code effectue par exemple une boucle un peux trop longue …
Mettez cette commande dans vos boucles ! :) (Utilisable uniquement si le IUG est lancé)
Tapez cette commande en commande @fantôme pour éviter l'affichage du message «Utilisable uniquement en IUG » ex :

```
@DOEVENTS/
```

TESTER UN RESEAU OU UNE MACHINE

La commande qui le permet est :

```
ping/
```

Cette commande envoie un paquet de donnés sur une adresse définit et attend une réponse sur un délais de 4 secondes
Aucuns système de serveur DNS à été programmé sur cette version, vous devez taper en IP

Exemple sur google.fr:

```
ping/ 74.125.132.94
```

CONNECTER UN LECTEUR RESEAU

La commande qui le permet est :

```
connecter/
```

Utilisant le protocole SMB, cette commande permet via le répertoire d'une adresse serveur, d'y être connecté tant qu'un lecteur

Exemple :

```
connecter/ e: \\SERVEUR\USER\SEB01\PARTAGE
```

Ceci crée le lecteur E: tant que racine **SERVEUR\USER\SEB01\PARTAGE**

Et puis pour déconnecter le lecteur E:

```
deconnecter/ e:
```

Cette version de cpcdos ne prend pas encore en charge les authentifications
Donc évitez d'utiliser un serveur ou répertoire protégé pas un mot de passe, ou d'un account-ID

LES FONCTIONNALITEES

Fonctions mathématiques et autres :

Les fonctions sont utilisé pour les calculs, modification, informations etc..

Voici la liste pour cette version de Cpcdos :

- \+ - / * ^ (Opération de calculs)
- SQR (La racine carré)
- COS (Cosinus)
- SIN (Sinus)
- TAN (Tangente)
- ATAN (ArcTengeante [TAN^-1])
- INT (Arrondir les valeurs)
- LEN (Obtenir le nombre de caractères dans une chaîne ou variable)
- LOG (Logarithme)
- MAJ (Mettre une chaîne de caractères ou variables en MAJuscules)
- MIN (Mettre une chaîne de caractères ou variables en MINuscules)
- HEX (Convertir une valeur en Hexadécimale)
- HEX{1-8} (Idem, nombre entre 1et 8 contre la fonction pour le nombre de digits)
- VAL (Convertir le binaire **&B** Octal **&O** ou Hexadécimale **&H** en décimale)
- EXP (Exponentielle)
- FRE (Mémoire disponible / pile)
- CHR (Convertir valeur ASCII en caractère ASCII)
- ASC (Convertir des caractères ASCII en valeur ASCII)
- INS (Obtenir la position d'une chaîne de caractères INString)
- CAP (CAPturer une chaîne de caractères)
- EOF (End-Of-File, Indique si le canal en lecture a atteint la fin -1 sinon 0)
- DIMX (Dimension X d'un fichier .BMP)
- DIMY (Dimension Y d'un fichier .BMP)

Comment les utiliser ?

Il suffit d'utiliser

```
/c {VALEUR ou FONCTION} {ATTIBUTION FONCTION OU OPERATION} {VALEUR}
```

Utilisable dans la commande **fix/ VARIABLE = /c …**
Ou en texte **txt/ /c …**

Quelques exemples :

Utilisation de la fonction INS et CAP

Fonction INS (Retourne la position du caractère a partir de la gauche) exemple :

```
TXT/ /C INS >Coucou le monde!;m
```

Je cherche le caractère «m» et donc résultat en position « *11* »

Fonction CAP (Capture une zone de texte) exemple :

```
TXT/ /C CAP >Coucou le monde!;8-5
```

Je capture du 8eme caractère jusqu'au 5eme donc résultat : « *le mo* »

Fonction VAL (Convertir le binaire **&B** Octal **&O** ou Hexadécimal **&H** > en décimal) exemple :

```
TXT/ /C VAL >&H4D
```

Résultat : 77

```
TXT/ /C VAL >&B1010011
```

Résultat : 83

Voici un jolie programme CCP qui montre l'utilisation claire d'une fonction + calculs TESTEZ LE !!

```
fix/ debug = 0
txt/ Appuie sur une touche pour continuer sur chaque phases !
Pause/
txt/ Ecrit une phase perso (sans virgules ni 2 points)
fix/ /Q Phrase
txt/ En majuscules ca donne /c MAJ >%Phrase%
txt/ En minuscules ca donne /c MIN >%Phrase%
Pause/
fix/ Taille = /c LEN >%Phrase%
txt/ Dans cette phase il y a %TAILLE% caracteres
txt/ Si on converti %TAILLE% en Hexadecimale sa donne /c HEX >%TAILLE%
Pause/
fix/ resultat = /c %TAILLE% * 2
txt/ %TAILLE% multipliee par 2 donne %resultat%
Pause/
fix/ resultat2 = /c %taille% * %RND%
txt/ Et ce dernier fois un nombre au hazard donne %resultat2%
fix/ resultat2 = /c %resultat2% + 5.20
txt/ Plus 5.20 donne %resultat2%
Pause/
txt/ Puis si on arrondi cette valeur cela donne /c INT >%resultat2%
txt/ Cette valeur en caractere ASCII donne /c CHR >%resultat2%
txt/ Travaille terminee Aurevoir !
Pause/
stop/
```

Observez bien l'utilisation des fonctions !

Fonctions graphiques :

Entrer dans la console (Mode LC) Pressez la touche **F12**

Fermer l'IUG pressez la touche **F6** *(Touche temporaire pour les développeurs)*

Stopper net le noyau pressez la touche **F5** *(Touche temporaire pour les développeurs)*

Pour basculer de fenêtre en fenêtre, pressez **ALT + ²**
(Pourquoi pas ALT+TAB ? Le problème serais si vous utilisez dosbox Windows prend le relais)

Pour réduire toute les fenêtres, pressez ALT + B ***(Fonction non finalisé [réduction parent])***

Pour fermer une fenêtre sélectionnée, pressez **ALT + F4**

Pour prendre un Screenshot, pressez **ALT+S** (Répertoire de destination *SYSTEME\SCR*)

Pour déplacer une fenêtre presser **CLIC DROIT** sur la barre de titre

Pour afficher l'heure sur un texte d'un label, ajoutez à la fin du nom « **HEURE*** »

Pour afficher la DATE sur un texte d'un label, ajoutez à la fin du nom « **DATE*** »

Pour afficher l'FPS sur un texte d'un label, ajoutez à la fin du nom « **FPS*** »

Pour afficher l'ACTIVITE sur un texte d'un label, ajoutez à la fin du nom « **ACT*** »

Pour afficher la mémoire restante sur un texte d'un label ajouter à la fin du nom « **MEM*** »

Pour afficher la mémoire restante en %sur un texte d'un label ajouter à la fin du nom « **MEMP*** »

Pour débloquer une application qui ne répond pas **ALT+D**

Fonctions CpcdosC+ du noyau :

Le code source de quelques fonctions CpcdosC+ se trouve dans le dossier *SYSTEME\KRNL*

Leurs fonctionnalité et leurs utilisation :

Nom :	Fonctionnalité(s) :	Utilisation :
AP_IMG	Affiche le programme de visualisation d'image	Compléter la variable : VISUALIS_IMG = {Cible d'un BMP ou JPG} Puis être interprété comme ceci : *EXE/ KRNL\AP_IMG\VISUALIS.CPC*
CONS_F01	Affiche une console graphique *(non terminée)*	Peut être directement interprété comme ceci : *EXE/ KRNL\CONS_F01\CONSOLE.CPC* *ou* *CPC/ /CONSOLE*
EXP_F00	Affiche l'explorateur en 1 seul instance en utilisant les fonctions EXP_F** ci-dessous	Compléter les variables : Couleurs de **F**ond Puis des **C**aractères EXP_COULEUR_F_R ={Fond Rouge} EXP_COULEUR_F_V = {Fond Vert} EXP_COULEUR_F_B = {Fond Bleu}

		EXP_COULEUR_C_R = {Caractères Rouge} EXP_COULEUR_C_V = {Caractères Vert} EXP_COULEUR_C_B = {Caractères Bleu} Dimensions EXP_DIM_X = { Taille x } EXP_DIM_Y = {Taille Y} Dossier cible par défaut EXP_CIBLE_DEF = {chemin dossier} Puis être interprété comme ceci : *EXE/ KRNL\EXP_F00\EXPLORE.CPC* *ou* *EXPLORER/*
EXP_F01	Crée un scroll dans une fenêtre avec comme repère de position, une variable *Fonction principalement utilisé pour l'explorateur de fichiers*	Compléter les variables : EXP_FENETRE = {Fenêtre cible} EXP_SCROLPX = {Position x du scrol} EXP_SCROHPY = {Position y du scrol haut} EXP_SCROBPY = {Position y du scrol bas} La valeur de retour de position (si vous cliquez *nombre fois* haut ou bas) est placé dans la variable *EXP_SCROL* ET puis interprété comme ceci : *EXE/ KRNL\EXP_F01\SCROL.CPC*
EXP_F02	Crée un bouton « Précédent » pour l'explorateur et retourne une valeur pour prévenir que l'utilisateur a cliqué « 1 » ou non « 0 »	Compléter la variable : EXP_FENETRE = {Fenetre cible}
EXP_F03	Événement du textebox de l'explorateur	Utilisable qu'a partir d'un événement Variable a communiquer : EXP_CIBLE
NUL_0	Fonction plutôt … inutile pour le moment ;-)	
RED_F01	Fonction de réduction et restauration des application via une barre des tâches Processus automatique si IUG_REDUCTION est définit Important : Il faut préciser le nom de la fenêtre qui fait d'office de barre des tâches dans la variable %IUG_REDUCTION% Une seule est possible ! Puis préciser les position et si votre barre est verticale ou horizontale Le noyau génère automatiquement les variables suivantes : IUG_REDUCTION_APP (Application en cours) IUG_REDUCTION_ACTION	Compléter les variables : IUG_REDUCTION =Nom de la fenêtre d'office d'une barre des taches IUG_REDUCTION_APP = Nom de l'application IUG_REDUCTION_ACTION = 1:Créer une icône / 2:Effacer IUG_REDUCTION_POS = 1:Horizontale / 2:Verticale IUG_REDUCTION_POSX = Position d'origine x IUG_REDUCTION_POSY = Position d'origine y IUG_REDUCTION_ESPACE = Espace entre les icônes ET puis interprété comme ceci :

	(Si il s'agit d'une création « 1 »ou suppression « 2 »)	*EXE/ KRNL\EXP_F01\SCROL.CPC*

Chapitre V - Commandes avancées

Nous allons voir ici les commandes plus approfondies, cet à dire des commandes pour contrôler & créer une interface utilisateur et graphique.

IUG

Interface Utilisateur Graphique

GESTION MULTI-TÂCHE

Rappel que le noyau est un noyau monolithique modulaire multitâche coopératif

La gestion du multitâche se fait par rapport à une valeur d'une variable

C'est le moteur CpcdosC+ qui gère l'intégralité du multitâche

cette variable qui la contrôle est %MULTI_TACHE% qui est par défaut à 10

La valeur 10 représente 10 boucles prioritaire pour le moteur qui lit code CpcdosC+ en exécution et 1 boucle pour l'IUG

- Plus la valeur est petite plus le système d'exécution devient lent mais beaucoup plus fluide au niveau du curseur, des synchronisation (Date, Heure, activités %, mémoire), pour gérer d'autres tâches en même temps etc. *L'exécution du code CpcdosC+ devient plus lent mais plus fluide.*
- Plus la valeur est grande, moins le système deviens fluide, mais l'exécution du code CpcdosC+ sera beaucoup rapide.
  Maximum conseillé : « 32 »
- Si la valeur est égale à 0 alors le système de priorité système pour le multitâche sera désactivé donc ce qui veux dire que si le code CpcdosC+ est exécuté, l'interface IUG sera bloqué jusqu'à la fin des opérations.
  A moins que la commande DOEVENTS/ soit écrite quelque pars dans le code ! ;-)
- Valeur « 1 » est impossible elle sera automatiquement mise à « 2 »
- Valeur stable et conseillée : « 10 » et pour Émulateurs (DosBox) « 15 » à « 20 »

Pour changer la valeur par exemple pour 20, il suffit de taper cette commande :

```
FIX/ MULTI_TACHE = 20
```

A la place de 20 mettez votre valeur !

Pour désactiver ce système :

```
FIX/ MULTI_TACHE = 0
```

INITIALISER UNE INTERFACE

{ l'utilisation de ces commandes dans les pages suivantes }

Les commandes qui le permettent sont :

```
ini/
```

Et

```
ini;
```

Ces commandes permettent la création d'un tableau pour la création de fenêtres ou d'objets

Il faut bien sûre compléter la création avec la commande ***créer/***

ini/ est une boucle d'initialisation

Pour créer une fenêtre :

```
ini/ fenetre(
ini/ fenetre)
```

Pour créer un bouton :

```
ini/ bouton(
ini/ bouton)
```

Pour créer un label :

```
ini/ label(
ini/ label)
```

Pour créer une imagebox :

```
ini/ imagebox(
ini/ imagebox)
```

Pour créer un textebox :

```
ini/ textebox(
ini/ textebox)
```

Pour créer un compteur :

```
ini/ compteur(
ini/ compteur)
```

Pour créer un :

```
ini/ (
ini/ )
```

Puis la commande ***ini;*** doit être utilisé uniquement dans une boucle d' INItialisation (vu ci-dessus) il permet de remplir des valeurs dans un tableau du Kernel permettant la création d'une interface

```
ini;nom = "NOM"
ini;texte = "TEXTE
ini;type = ""
ini;couleur = "Rouge,Vert,Bleu"  ‘ Couleur base (genre fenêtre)
ini;couleurf = "Rouge,Vert,Bleu" ‘ Couleur de Fond
ini;couleurp = "Rouge,Vert,Bleu" ‘ Couleur Premier Plan
ini;tx = "Taille X"
ini;ty = "Taile Y"
ini;px = "Position X"
ini;py = "Position Y"
ini;intervalle = "5"
ini; etc...
etc...
```

Nous allons dans les pages suivantes voir comment utiliser tout cela ! ;-)

Pour savoir comment créer un événement, RDV sur le chapitre ***VI - Les événements***

NB :

Pour lister les propriétés en mémoire il suffit de taper :

```
ini/ /liste
```

Pour tester si une fenêtre ou un objet est existant :

```
ini/ /test MA_FENETRE
```

```
ini/ /test MON_LABEL
```

CREER UNE FENETRE

La commande qui le permet est :

```
ini/ fenetre( )
```

Cette commande , accompagné de paramètres obligatoires , permet de créer une fenêtre graphiquement de ce type (Exemple):

Barre de titre , fond de couleur , position , taille , type , déplaçable , refermable.. et un contenu

Exemple :

```
ini/ fenetre(
   ini;nom = "FENETRE_1"
   ini;texte = "Ma petite fenetre !"
   ini;type = "x;MxAxRxFxTxFPx..."
   ini;img = "MONICONE.BMP"
   ini;couleur = "087,215,186"
   ini;tx = "300"
   ini;ty = "250"
   ini;px = "MX"
   ini;py = "MY"
   Creer/
ini/ fenetre)
```

Explications !

```
ini/ fenetre(
```

Permet d'informer pour INItialiser / créer une nouvelle fenêtre

```
ini;nom = "FENETRE_1"
```

Donner au noyau le nom d'Objet de la fenêtre

Si vous nommez une autre fenêtre au même nom, la fenêtre sera alors remplacée

```
ini;texte = "Ma petite fenetre !"
```

Titre de la fenêtre

```
ini;img = "MONICONE.BMP"
```

Paramètre optionnel qui permet de personnaliser l'icône de l'application situé dans la barre de titre

```
ini;type = "1"
```

Type de la fenêtre

1:Normal 2:Sans Barre de titre 3:transparente


Il y a aussi d'autres paramètres attribuables comme **(par défaut ils sont tous à 1)**

M ovable? (Déplaçable ?)

A tive? (Interaction possible ?)

V isible (Visible ?)

R eductable?(Reduire)

F ermable?

T ache?(Affiché dans la barre des tâches ?)

FP **F**enetre**P**rioritaire ? (+entourée d'un fond gris)

C ouleur généralisé + translucidité ---------->

O mbre ? (Affichage de l'ombre sous la fenêtre ou pas)

Exemple avec touts les paramètres en « active », donc 1:

```
ini;type = "1;M1A1R1F1T1FP1"
```

La lettre du paramètre + une valeur booleen (1 ou 0)
si un(s) des paramètres est omit il est activé «1» par défaut
Pour le paramètre Couleur généralisé + translucidité il y a
-C0 (Barre seul) -C1 (Généralisé) -C2 **Par Défaut** (Image *os\media\iug\BAR_T.BMP*)

```
ini;couleur = "000,000,000"
```

Couleur de fond de la fenêtre en R.V.B (bien mettre 3 caractères sur chaque couleurs) /!\

Si le paramètre « **C1** » est placé » dans **ini;type**, alors cette couleur sera généralisée et transparente

```
ini;tx = "300"
```

Taille X de la fenêtre
Acronyme : TX = Taille X

```
ini;ty = "250"
```

Taille Y de la fenêtre
Acronyme : TY = Taille Y

```
ini;px = "MX"
```

Position X de la fenêtre
Acronyme : PX = Position X
La valeur "MX"(Milieu X) permettant que la fenêtre sois centré horizontalement ,vous pouvez mettre une valeur numérique à la place

```
ini;py = "MY"
```

Position X de la fenêtre
Acronyme : PY = Position Y
La valeur "MY"(Milieu Y) permettant que la fenêtre sois centré Verticalement ,vous pouvez mettre une valeur numérique à la place

```
creer/
```

Permettant de créer l'objet ou la fenêtre qui se trouve en mémoire dans le tableau du SCI
(Attention, bien le mettre avant la prochaine création d'objet ou de fenêtre , car sinon , les valeurs risquerons d'être remplacé par les nouvelles.)
Il faut obligatoirement l'écrire juste avant la fin de boucle «**ini/ fenetre)**»
sinon rien ne se passera.

```
ini/ fenetre)
```

Indique la fin de l'INItialisation de la fenêtre.

CREER UN BOUTON

La commande qui le permet est :

```
ini/ bouton( )
```

Cette commande , accompagné de paramètres obligatoires permet de créer un bouton graphiquement ce type (Exemple):

Voici un exemple : (il faut bien sûre créer une fenêtre –> voir **CREER UNE FENETRE** ci dessus).

```
ini/ bouton(
        ini;nom = "BOUTON1"
        ini;fenetre = "FENETRE_1"
        ini;texte = "Clique moi !"
        ini;img = "0"
        ini;couleurf = "255,255,255"
        ini;couleurp = "255,000,000"
        ini;tx = "180"
        ini;ty = "30"
        ini;px = "30"
        ini;py = "80"
        creer/
ini/ bouton)
```

Attention a bien mêttre le nom de la fenetre où vous voulez placer votre bouton dans le paramètre « `ini;fenetre` »

Alors bien sûre si vous cliquez dessus, rien ne se passera, il faut créer un événements !

RDV donc au chapitre ***VI – Les événements***

Et voici le résultat :

Explications !

```
ini/ bouton(
```

Permet d'informer au Kernel , la création d'un objet (Bouton).

```
ini;nom = "BOUTON_1"
```

Permet de nommer la propriété.

```
ini;fenetre = "FENETRE_1"
```

Indique au Kernel , sur quelle fenêtre allons-nous créer ce bouton.
Là , il va créer ce bouton sur **FENETRE_1.**

```
ini;texte = "Clique moi !"
```

Permet d'écrire le texte du bouton.

```
ini;type = "1"
```

Paramètres disponibles

V0: Invisible (V1 par défaut)

```
ini;img = "3"
```

Permet de définir une image de fond sur un bouton avec son index

numéro entre 0-7

```
ini;img = "1"
```

```
ini;img = "2"
```

```
ini;img = "3"
```

```
ini;img = "4"
```

```
ini;img = "5"
```

```
ini;img = "6"
```

```
ini;img = "7"
```

Les images sont stocké dans la cible où vous avez définit la variable MEDIA dans OS.CPC

Par défaut , il se situe dans « `C:\CPCDOS\SYSTEME\OS\Media\IUG` »

Souces images BMP personnalisables [8,16, 24 et 32bit]

```
ini;couleurf = "100,100,250"
```

Permet de définir une couleur en R , V , B de fond du bouton
Uniquement si l'option `ini;img = "0"`

```
ini;couleurp = "200,000,255"
```

Permet de définir la couleur des caractères du texte du bouton

```
ini;tx = "150"
```

Défini la taille de l'axe X du bouton

```
ini;ty = "30"
```

Défini la taille de l'axe Y du bouton
(par défaut&conseillé ,mettez la valeur à 30)

```
ini;px = "10"
```

Défini la position de l'axe X sur la fenêtre définit (valeur négatifs autorisés)

```
ini;py = "170"
```

Défini la position de l'axe Y sur la fenêtre définit (valeur négatifs autorisés)

```
creer/
```

Permet de créer l'interface avec les valeurs que vous avez fournis ci-dessus

```
ini/ bouton)
```

Ferme la boucle..

CREER UN LABEL

La commande qui le permet est :

```
ini/ label( )
```

Cette commande , accompagné de paramètres obligatoires , permet de créer un label graphiquement de ce type (Exemple):

Possibilité d'avoir bien sûr, le fond transparent !

Voici un exemple : (il faut bien sûre créer une fenêtre –> voir **CREER UNE FENETRE** ci dessus).

```
ini/ label(
  ini;fenetre = "FENETRE_1"
  ini;nom = "LABEL_1"
  ini;texte = "Hello word !"
  ini;couleurf = "250,010,010"
  ini;couleurp = "000,255,100"
  ini;transparent = "0"
  ini;type = "0"
  ini;tx = "200"
  ini;ty = "20"
  ini;px = "10"
  ini;py = "15"
  Creer/
ini/ label)
```

Explications !

```
ini/ label(
```

Permet d'informer au Kernel , la création d'un objet (Label).

```
ini;fenetre = "FENETRE_1"
```

Indique au Kernel , sur quelle fenêtre allons-nous créer ce bouton.
Là , il va créer ce bouton sur **FENETRE_1.**

```
ini;nom = "LABEL_1"
```

Permet de nommer la propriété. (l'objet)

```
ini;texte = "Hello word !"
```

Permet de définir du texte graphiquement dans le label.
Vous pouvez écrire plusieurs lignes dans 1 seul label et pour ça il suffit d'utiliser le caractère ASCII 11
Pour ceci, aller voir à la page suivante

```
ini;couleurf = "210,225,240"
```

Définir la couleur de fond si **ini;transparent = "0"**

```
ini;couleurp = "010,010,010"
```

Définir la couleur des caractères.

```
ini;transparent = "0"
```

Permet d'avoir les couleur d'arrière plan « entre les caractères »
la transparence arrière plan. (1 = activé / 0 = opaque)
(assurez-vous que ini;type est égale sois à 3 uniquement)

```
ini;type = "0"
```

Permet de définir si la taille du label est réglé automatiquement par rapport a son contenu **"0"** donc les paramètres **ini;TX** et **ini;TY** sont inutiles
SI **"1"** alors il faut définir en valeur les paramètres **ini;TX** et **ini;TY.**
SI **"3"** alors **ini;TX** et **ini;TY** sont inutiles et fond opaque utilisable

```
ini;tx = "200"
```

Permet de définir la taille sur l'axe X uniquement si **ini;type = "0"**

```
ini;ty = "20"
```

Permet de définir la taille sur l'axe Y uniquement si **ini;type = "0"**

```
ini;px = "10"
```

Permet de positionner sur l'axe X le label dans la fenêtre

```
ini;py = "15"
```

Permet de positionner sur l'axe Y le label dans la fenêtre

```
Creer/
```

Permet de créer l'interface avec les valeurs que vous avez fournis ci-dessus

```
ini/ label)
```

Ferme la boucle..

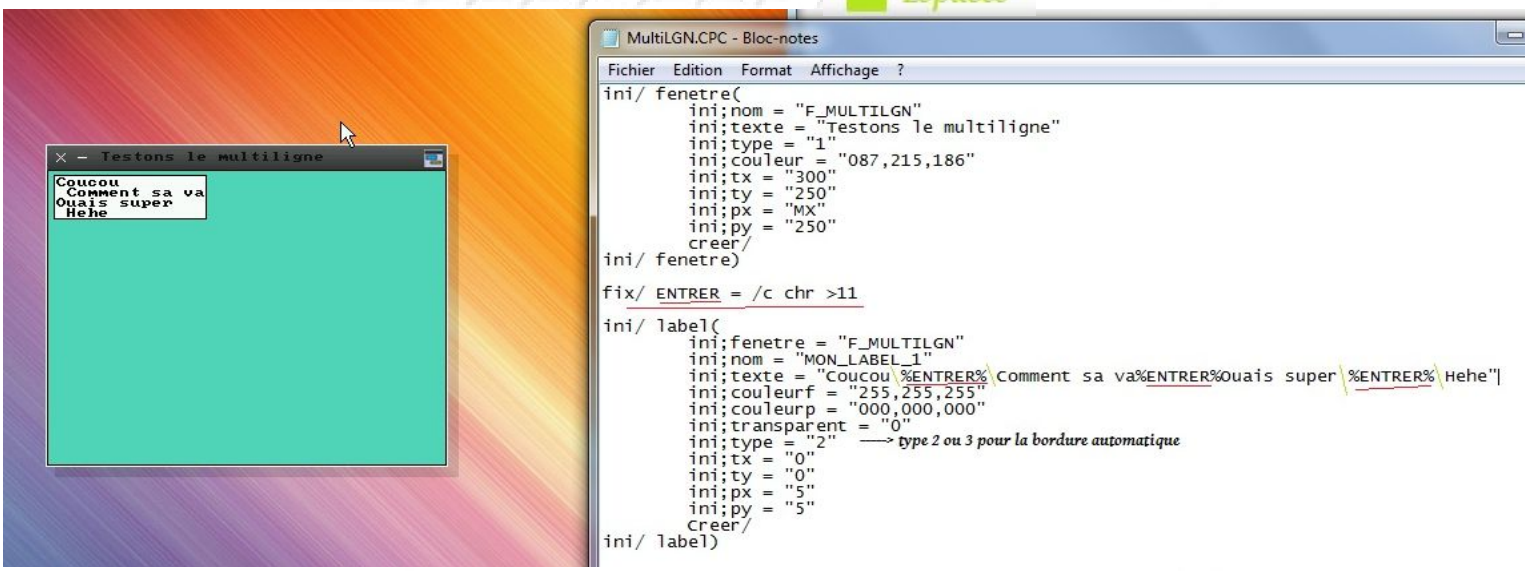
Exemple pour écrire plusieurs lignes dans 1 seul label

```
ini/ fenetre(
        ini;nom = "F_MULTILGN"
        ini;texte = "Testons le multiligne"
        ini;type = "1"
        ini;couleur = "087,215,186"
        ini;tx = "300"
        ini;ty = "250"
        ini;px = "MX"
        ini;py = "250"
        creer/
ini/ fenetre)
fix/ ENTRER = /c chr >11
ini/ label(
        ini;fenetre = "F_MULTILGN"
        ini;nom = "MON_LABEL_1"
        ini;texte = "Coucou %ENTRER%Comment sa va?%ENTRER% Ouais super%ENTRER% Hehe"
        ini;couleurf = "255,255,255"
        ini;couleurp = "000,000,000"
        ini;transparent = "0"
        ini;type = "2"
        ini;tx = "0"
        ini;ty = "0"
        ini;px = "5"
        ini;py = "5"
        Creer/
ini/ label)
```

Voici le résultat :

Important

Espaces



CREER UNE IMAGEBOX

La commande qui le permet est :

```
ini/ imagebox( )
```

Cette commande , accompagné de paramètres obligatoires , permet de créer une image dans une fenêtre (Exemple):


```
ini/ imagebox(
      ini;nom = "MON_IMGBOX"
      ini;fenetre = "FENETRE_1"
      ini;couleur = "000,000,000"
      ini;couleurf = "001,001,001"
      ini;type = "0"
      ini;image = "os\prog\image.BMP"
      ini;px = "10"
      ini;py = "30"
      ini;tx = "400"
      ini;ty = "280"
      creer/
ini/ imagebox)
```

Explications !

```
ini/ imagebox(
```

Permet d'informer au Kernel , la création d'un objet (image).

```
ini;fenetre = "FENETRE_1"
```

Indique au Kernel , sur quelle fenêtre allons-nous créer cette image.
Là , il va créer cet image sur **FENETRE_1.**

```
ini;nom = "MON_IMGBOX"
```

Permet de nommer la propriété. (l'objet)

```
ini;couleur = "210,225,240"
```

Définir la couleur de fond si couleurf est à « 000,000,000 »

```
ini;couleurf = "001,001,001"
```

Activer ou pas le color Mask (supprimer la couleurs magenta RVB [255,0,255]) et permettre ainsi avoir une image transparente par rapport à l'arrière plan d'origine. Paramètre utilisable uniquement si **ini;type = 0**

```
ini;type = "0"
```

Permet de créer un cadre autour de l'image 0:Oui 2:Non

```
ini;image = "os\prog\image.BMP"
```

Définir l'image

Sur la version 2.0.5, pas d'ajustement possible

```
ini;tx = "200"
```

Permet de définir la taille sur l'axe X uniquement si **ini;type = "0"**

```
ini;ty = "20"
```

Permet de définir la taille sur l'axe Y uniquement si **ini;type = "0"**

```
ini;px = "10"
```

Permet de positionner sur l'axe X le label dans la fenêtre

```
ini;py = "15"
```

Permet de positionner sur l'axe Y le label dans la fenêtre

```
Creer/
```

Permet de créer l'interface avec les valeurs que vous avez fournis ci-dessus

```
ini/ imagebox)
```

Ferme la boucle..

CREER UN TEXTEBOX

La commande qui le permet est :

```
ini/ textebox( )
```

Cette commande , accompagné de paramètres obligatoires , permet de créer une zone de saisie dans une fenêtre (Exemple):

NB :

Pas de sélection ou de placement avec le curseur pour cette version
Déplacement gauche/droite avec les flèches
Touches Début et Fin retour arrière et SUPPR utilisables

```
ini/ textebox(
      ini;nom = "MON_TEXTEBOX"
      ini;fenetre = "FENETRE_1"
      ini;type = "1"
      ini;couleurf = "255,255,255"
      ini;couleurp = "000,000,000"
      ini;texte = "Mon textebox !"
      ini;px = "20"
      ini;py = "20"
      ini;tx = "210"
      ini;ty = "17"
      creer/
ini/ textebox)
```

Explications !

```
ini/ textebox(
```

Permet d'informer au Kernel , la création d'un objet (image).

```
ini;fenetre = "FENETRE_1"
```

Indique au Kernel , sur quelle fenêtre allons-nous créer cette image.
Là , il va créer cet image sur **FENETRE_1.**

```
ini;nom = "MON_TEXTEBOX"
```

Permet de nommer la propriété. (l'objet)

```
ini;couleurf = "255,255,255"
```

Définit la couleur de fond RVB

```
ini;couleurp = "000,000,000"
```

Définit la couleur des caractères saisies RVB

```
ini;type = "B1"
```

Pour cette version laissez «B1»
Paramètres disponibles :

V0 : Invisible (V1 par défaut)
O0 : Sans ombres (O1 par défaut)
P1 : Protéger par des « * »

```
ini;texte = "Mon textebox !"
```

Écrit le texte initiale du textebox
Pour que le textebox sois « vide » insérez seul, le code caractère ASCII 255 (ALT+255)

```
ini;tx = "210"
```

Permet de définir la taille sur l'axe X

```
ini;ty = "17"
```

Permet de définir la taille sur l'axe Y *(17 étant la taille idéale)*

```
ini;px = "20"
```

Permet de positionner sur l'axe X le label dans la fenêtre

```
ini;py = "20"
```

Permet de positionner sur l'axe Y le label dans la fenêtre

```
Creer/
```

Permet de créer l'interface avec les valeurs que vous avez fournis ci-dessus

```
ini/ textebox)
```

Ferme la boucle..

NB : si vous voulez récupérer le contenu du texte pour y placer dans une variable il suffit de récupérer la propriété avec *#%*

Exemple :

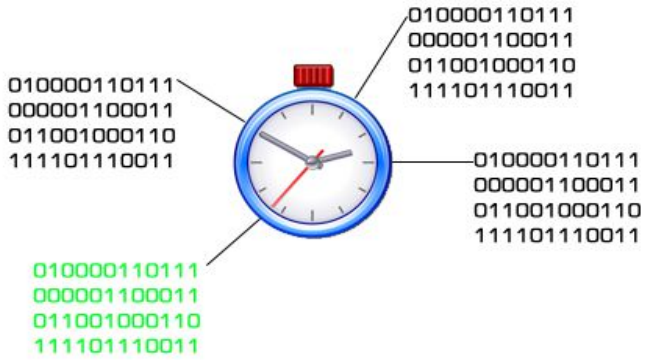
```
ini/ textebox(
      ini;nom = "MON_TEXTEBOX"
      ini;texte = "#%MA_VARIABLE"
ini/ textebox)
txt/ %MA_VARIABLE%
```

CREER UN COMPTEUR

La commande qui le permet est :

```
ini/ compteur( )
```

Cette commande, accompagné de paramètres obligatoires , permet de créer un compteur capable d'exécuter des commandes (dans des fichiers événements) dans un cycle de temps donné




```
ini/ compteur(
      ini;nom = "COMPTEUR_1"
      ini;fenetre = "FENETRE_1"
      ini;intervalle = "3"
      ini;active = "1"
      creer/
      ev/ FICHIER.CPC        >    {Pas d'événements, le compteur ne sert strictement a rien}
ini/ compteur)
```

Explications !

```
ini/ compteur(
```

Permet d'informer au Kernel , la création d'un objet (image).

```
ini;fenetre = "FENETRE_1"
```

Indique au Kernel , sur quelle fenêtre allons-nous créer cette image.
Là , il va créer cet image sur **FENETRE_1**

```
ini;intervalle = "3"
```

Indique l'intervalle en secondes (Exécuter l'événement toutes les x secondes)

```
ini;active = "1"
```

Indique si le timer est activé « 1 » ou pas « 0 »

```
Creer/
```

Permet de créer l'interface avec les valeurs que vous avez fournis ci-dessus

```
ev/ FICHIER.CPC
```

Indique le fichier événement cible dans le cycle du comptage

exemple du contenu de FICHIER.CPC

(exécutez « **FIX/ MON_CYCLE1 = 0** » avant tout sinon la variable est introuvable)

```
PROC/ COMPT_1(CYCLE)
     FIX/ MON_CYCLE1 = /C %MON_CYCLE1% + 1
     MSGBOX/ /TEXTE=Cycle %MON_CYCLE1% /TITRE=123 /MODE=1 /ALERTE=1
FIN/ PROC
```

```
ini/ compteur)
```

Fin de la boucle...

CREER UNE INTERFACE

La commande qui le permet est :

```
creer/
```

Cette commande permet de créer une fenêtre ou un objet dans la boucle de création d'interface à partir de valeurs complétées (d'un tableau)

Vous l'avez déjà vu dans les pages précédentes

Exemple d'utilisation :

```
ini/ fenetre(
   ini;nom = "FENETRE_1"
   ini;texte = "Ma première fenêtre Cpcdos !"
   ini;type = "1"
   ini;couleur = "255,255,255"
   ini;tx = "300"
   ini;ty = "250"
   ini;px = "200"
   ini;py = "150"
   creer/
ini/ fenetre)
```

Si vous ne le mettez pas avant la commande `ini/ fenetre)` le tableau de valeurs sera vidé

Toujours dans la boucle d'initialisation et à la fin !

«PIRATER» le SCI / MODIFIER LES PROPRIETES

Je ne vais pas vous prendre tous pour des cons, mais … il suffit juste de réutiliser la boucle ini.. *:-P*

A une différence près, c'est que vous n'êtes pas obligé de réécrire toutes les propriétés si le contenu sera de toute manière, le même.
Exemple si dessus, il y a un code pour créer une fenêtre « Ma première fenêtre cpcdos »
vous voulez changer le titre PUIS la position X, comment faire, et bien voici un exemple :

```
ini/ fenetre(
   ini;nom = "FENETRE_1"
   ini;texte = "Mon nouveau titre de ma fenetre!"
   ini;px = "50"
   creer/
ini/ fenetre)
```

Vous pouvez faire ce genre de modification de propriété avec tous les objets et fenêtres !
Il suffit indiquer le nom de propriété « *ini;nom* = .. » **(Obligatoire)** Puis réutiliser les propriétés.

Attention vous réutilisez une propriété, vous effacez tous le contenu par le nouveau.
Prenez égard avec INI;TYPE vous vous ajoutez, ou modifiez un paramètre faudra tout retaper !

DEMARRER L'INITIALISATION D'UN OS

La commande qui le permet est :

```
demarrer/
```

Cette commande permet tout simplement de démarrer le fichier qui se trouve dans la variable %BOOTOS% qui est normalement dans le chemin « OS\INDEX.CPC »

Donc en gros, il permet juste d'exécuter le fichier d'index qui permet l'exécution de l'OS.

LANCER L'INTERFACE GRAPHIQUE (OS)

La commande qui le permet est :

```
iug/
```

Cette commande permet de passer du mode LC au mode IUG,
elle exécute le service IUG, L'interface utilisateur Graphique qui lui gère le fond d'écran, les fenêtre et objets puis l'interaction utilisateur (événements) , timer etc ..

```
iug/ /console
```

Permet de lancer l'IUG en arrière plan tout étant en mode LC (Visionnage du debug)
Ce qui veux dire que les touches comme F12 fonctionne

```
iug/ /safe
```

Lance l'IUG en SAFE MODE *(Voir partie ci-dessous)*

```
iug/ /reset
```

Peut être tapé à la console avant ou même après l'exécution de l'OS
Permet tout simplement de recharger le code des fichiers indexés au tableau %IUG_RESET(x)% (x étant une valeur entre 0 et 8)
Donc 8 fichier .CPC rechargeable possible

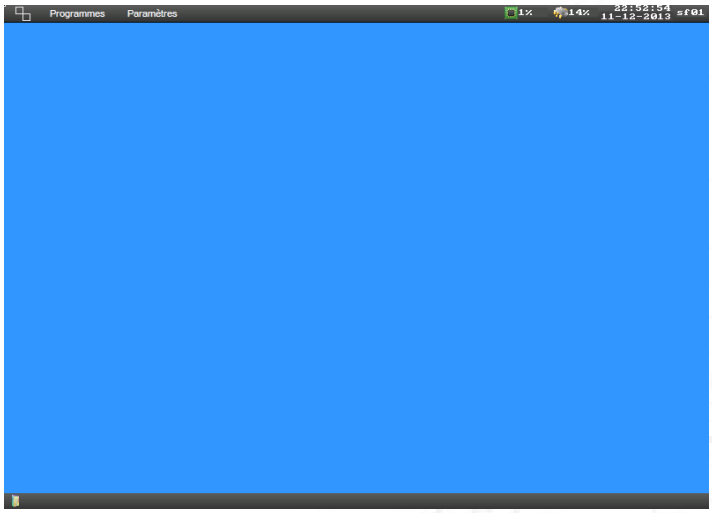
Une tactique assez utile peut être utilisée sur votre OS, par exemple si vous changer de résolution d'écran et que votre bureau, vos barres, vos menus etc.. qui eux se règle du chargement par rapport à la taille de l'écran ; si ils sont tous décalés vous pouvez utiliser IUG/ /RESET en plaçant dans le tableau IUG_RESET(x) les fichiers nécessaire qui permette de recharger a nouveau vos menus et barres par rapport a la nouvelle résolution d'écran. C'est pas genre pète ça mère ça ?

Explication page suivante !

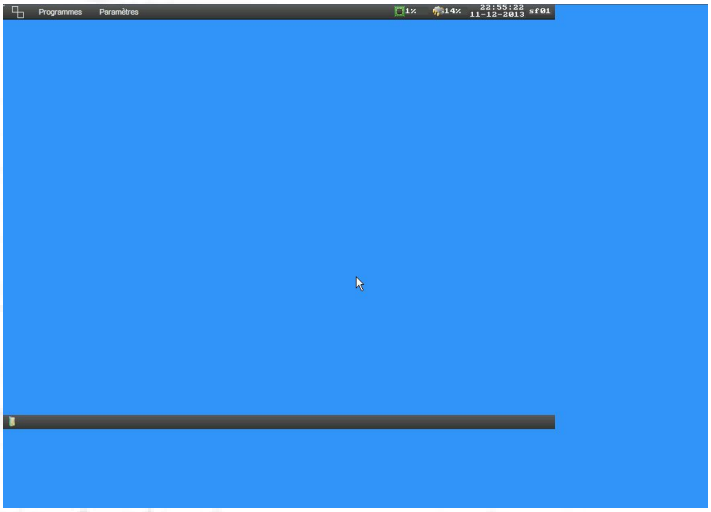
Comment faire ?

Il est conseillé de faire un bureau adaptatif aux résolutions d'écran

Prenons exemple avec le bureau d'un OS quelconque, CraftyOS, vous avez fait un programme qui change de résolution d'écran, vous avez actuellement 800x600 :



Jusqu'à là pas de soucis, maintenant vous changez de résolution d'écran vous mettez en 1024x768 et vous avez ça :



Pas de panique c'est normal, vos menus ont gardé les mêmes tailles.

Pour recharger les fichiers qui crée le bureau avec IUG/ comment faire ?

Tout simplement utilisez le tableau

%IUG_RESET(x)%

x étant une valeur entre 1 et 8

Par exemple :

Créez un fichier *FERM.CPC :*

```
REM/ Ce fichier permet de fermer les menus du bureau (les processus)
fermer/ mon_menu_barre_1
fermer/ blablabla
```

Dans un autre fichier qui change votre résolution d'écran **ou** à la console :

```
fix/ SCR_BAS = 1024x768                 <(La résolution d'écran choisie)
fix/ IUG_RESET(1) = FERM.CPC            <(Étape 1, Ferme les processus)
fix/ IUG_RESET(2) = MONBUREAU.CPC       <(Le fichier où se trouve vos menus..)
LC/                          <(Ferme l'interface IUG [obligatoire] :scintillements possible)
IUG/ /RESET                  <(Cette commande exécute %IUG_RESET(1 et 2)% puis l'IUG)
```

LANCEMENT SAFE MODE

Pour lancer l'OS en mode sans échec, il suffit que cette commande sois exécutée :

```
demarrer/ /safe
```

Si l'OS ne démarre pas correctement, cette commande permet de charger l'OS avec des limitations dans certains paramètres du noyau

Si cette commande a été exécuté, cpcdos bloquera la commande `IUG/` si elle n'est pas accompagné du paramètre `/SAFE`

Puis si c'est pas déjà fait, pour lancer l'IUG :

```
iug/ /safe
```

Changements apportés : Résolution limitée à 800x600x16b , pas de fond d'écran, pas d'ombres, pas de transparence des fenêtres hors mis l'assombrissement générale pour des fenêtres prioritaires.

Et je vous dis quand même, votre OS est plus performant comme ça ! ;-)

LES EVENEMENTS

Dans cette partie nous allons voir comment créer un événement

Que doit faire le Kernel si l'utilisateur clique sur t-elle bouton ?

Que doit faire le Kernel si l'utilisateur essai de fermer une fenêtre etc..

Avant ce coder la procédure, assurez-vous d'avoir insérer le fichier d'événement dans une boucle INI

Oups ! Comment faire ?

Exemple, tout simplement créez un fichier texte extension .cpc nommée MON_EV.CPC dans « **OS\PROG\MON_EV.CPC** »

qui lui sera le fichier d'événements

(si c'est pas déjà fait)

et ajouter la juste après la commande « créer/ » dans les boucles INI :

« ev/ OS\PROG\MON_EV.CPC »

Par exemple pour un bouton, ça donne ceci :

```
ini/ bouton(
        ini;nom = "BOUTON_1"
        ini;fenetre = "FENETRE_1"
        ini;texte = "Clique moi !"
        ini;img = "5"
        ini;couleurf = "255,255,255"
        ini;couleurp = "255,000,000"
        ini;tx = "180"
        ini;ty = "30"
        ini;px = "30"
        ini;py = "80"
        creer/
        ev/ os\prog\mon_ev.cpc
ini/ bouton)
```

La commande **EV/** indique au Kernel que si il y a interaction, qu'il va chercher la procédure et l'événement d'interaction correspondant (CLIC, DLBCLIC, CLICG, DLBCLICG , FOCUS, FERME etc ..) dans le fichier indiqué

Pour un premier exemple, créez **une** fenêtre et **un** bouton
(voir dans la partie CREER UNE FENETRE et CREER UN BOUTON)

Ajoutez dans la boucle ini de votre bouton, juste après la commande « **creer/** »
(Normalement c'est déjà fait ;-))

```
ev/ OS\MON_EV.CPC
```

Après ouvrez ce fichier avec un éditeur de textes, puis ajoutez ces commandes :

```
Proc/ BOUTON_1(CLIC)
msgbox/ /texte=Tu m'a clique dessus ! /titre=Information /mode=1 /alerte=1
Fin/ Proc
```

Vous pouvez faire la même chose avec tous les autres noms d'objets, et même la fenêtre

Pour lister les événements en mémoire, il suffit de taper cette commande :

```
ev/ /liste
```

Pour simuler (exécuter) un événements en mémoire, il suffit de taper cette commande :

```
ev/ /exe {objet/fenêtre}:{événement}
```

Exemple après avoir créer un bouton «BOUTON_1» + événement d'un clic vous pouvez utiliser :

```
ev/ /exe BOUTON_1:CLIC
```

Et le noyau va exécuter le code où se trouve votre procédure d'événement dans le fichier comme si vous avez réellement cliqué sur ce bouton par exemple :

Événements disponibles sur cette version :

- CLIC (Si l'utilisateur clique sur ..)
- CLICD (Si l'utilisateur clique droit sur ..)
- DLBCLIC (Si l'utilisateur double clique sur ..)
- DLBCLICD (Si l'utilisateur double clique droit sur ..)
- FOCUS (Si l'utilisateur sélectionne la fenêtre ..)
- PFOCUS (Si l'utilisateur fait perdre la sélection de la fenêtre ..)
- ENTRER (Si l'utilisateur presse ENTRER sur un TexteBox)
- CHANGE (Si l'utilisateur écrit sur un TexteBox)
- FERME (Si l'utilisateur ferme l'objet ou la fenêtre ..)
- REDUIRE (Si l'utilisateur clique sur réduire)
- CYCLE (Si le temps du cycle d'un Compteur s'est écoulé)
- SOURIS (Si l'utilisateur passe la souris au dessus d'un objet ou fenêtre)
- SOURISQ (Si l'utilisateur quitte la zone où il passe la souris au dessus ..[IDEM])

AFFICHER UN MSGBOX

La commande qui le permet est :

```
msgbox/ /texte={TEXTE} /titre={TEXTE} /mode={1/2} /alerte={0/1/2/3} /nom={TEXTE}
```

Cette commande permet de créer un message graphique

Le paramètre **/texte** indiquant le texte du message, ci dessus c'est « HELLO World ! »

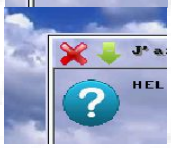
Le paramètre **/titre** indique le titre de la fenêtre, ci dessus c'est « J'ai un message pour toi »

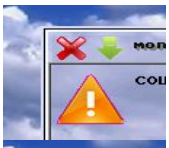
Le paramètre **/mode** permet de choisir entre un message avec 1 le bouton « OK »

ou 2 avec les boutons « Oui » et « Non »

Le paramètre **/alerte** définit le niveau d'alerte (icônes par défaut, *Oui vous pouvez les modifier!*)

0 : message de validation 

1 : Question 

2: Avertissement 

3: Erreur 

Le paramètre **/nom** est optionnel si il est omit, par défaut il se nomme « MSGCONSOLE»

Exemple :

```
msgbox/ /texte=HELLO Word ! /titre=Message /mode=1 /alerte=0
```

Pour créer un événement sur le bouton OUI & NON du msgbox suivez cet exemple : (/**mode=2**)

```
msgbox/ /texte=Que faire ? /titre=test /mode=2 /alerte=1 /nom=MSG1
ini/ bouton(
      ini;nom = "BOUI.MSG1"
      ev/ EVENEMT.CPC
ini/ bouton)
ini/ bouton(
      ini;nom = "BNON.MSG1"
      ev/ EVENEMT.CPC
ini/ bouton)
```

Dans le fichier EVENEMT.CPC :

```
PROC/ BOUI.MSG1 (CLIC)
      Fermer/ MSG1
      msgbox/ /texte=Tu as choisis OUI /titre=123 /mode=1 /alerte=0
FIN/ PROC
PROC/ BNON.MSG1 (CLIC)
      Fermer/ MSG1
      msgbox/ /texte=Tu as choisis NON /titre=123 /mode=1 /alerte=0
FIN/ PROC
```

Puis pour le /**mode=1** cet à dire le bouton «OK» :

```
msgbox/ /texte=Compris ? /titre=test /mode=1 /alerte=1 /nom=MSG1
ini/ bouton(
      ini;nom = "BOK.MSG1"
      ev/ EVENEMT.CPC
ini/ bouton)
```

Dans le fichier EVENEMT.CPC :

```
PROC/ BOK.MSG1 (CLIC)
      Fermer/ MSG1
      msgbox/ /texte=Tu as clique sur OK /titre=123 /mode=1 /alerte=0
FIN/ PROC
```

EXPLORER UN DOSSIER

La commande qui le permet est :

```
explorer/
```

(1 seule instance du processus autorisée pour cette version, pour le moment...)

Cette commande permet d'ouvrir l'explorateur de fichiers dans le dossier indiqué, et pouvoir ouvrir des fichiers comme
fichiers exécutable *.exe .com .bat .cpc*
fichiers textes *.txt .log* etc …

Exemple pour aller au dossier PROG :

```
explorer/ C:\CPCDOS\SYSTEME\OS\PROG
```

Dans une prochaine version vous pouvez paramétrer votre propre extension de fichier pour choisir avec quelle programme & paramètre voulez vous ouvrir le fichier sélectionnée

>> Et avoir accès au mode «grandes icônes» <<

Voir les partition et lecteurs du PC :

```
explorer/ :
```

Si vous n'avez pas le pilotes NTFS vous ne pourrez sûrement pas lire vos lecteur contenant Windows

FERMER UN(E) OU PLUSIEURS OBJETS/FENETRES

La commande qui le permet est :

```
fermer/
```

Cette commande permet de fermer (décharger) en mémoire toutes les propriétés associé au tableau qui permettant l'affichage graphique & interagir des événements des données de propriété.

Pour fermer un objet il faut ajouter le paramètre `/OBJET`

Nb : nom en majuscules !

Exemples :

```
fermer/ /OBJET MON_BOUTON
fermer/ FENETRE_1
fermer/ A1
```

Il existe aussi un paramètre, (a utiliser avec précautions) , qui permet de fermer TOUS les objets et fenêtres.

La commande qui le permet est :

```
fermer/ /tout
```

Si elle est exécute , vous n'aurez plus de bouton , de label , de textebox etc...
Le mode IUG sera automatiquement fermé et le mode LC sera exécuté. (console)

ACTUALISER UNE OU PLUSIEURS FENETRES

La commande qui le permet est :

```
actualise/
```

Cette commande permet d'actualiser l'interface

Exemple :

```
actualise/ MA_FENETRE_1
```

Le paramètre **/tout** permet d'actualiser toutes les fenêtres

Exemple :

```
actualise/ /tout
```

A noter que ce paramètre actualise aussi le fond d'écran

```
actualise/ /fond
```

Permet d'actualiser a nouveau le fond d'écran en le stockant dans la RAM
Paramètre utile si vous changer le fond d'écran via la variable *SCR_FOND*

FOCUS/SELECTIONNER UNE FENETRE

La commande qui le permet est :

```
focus/
```

Cette commande permet de sélectionner une fenêtre et la dessiner au premier plan.

Exemple :

```
focus/ MA_FENETRE_1
```

LANCER LA CONSOLE

La commande qui le permet est :

```
lc/
```

(Ligne de **C**ommandes)

Cette commande permet de passer du mode IUG au mode LC tout ayant l'IUG en arrière plan
Un peux le même principe que `IUG/ /CONSOLE`

Mais pour **UTILISER AU CLAVIER** la console, il faut ajouter le paramètre suivant :

```
lc/ /console
```

Sinon le IUG sera en exécution en arrière plan

Si vous avez malencontreusement oublié ce paramètre, pas de panique appuyer sur la touche F12 pour pouvoir taper les commandes ! ;-)

LES VARIABLES D'ENVIRONEMENTS

Chapitre assez intéressant qui vous permet de savoir les variables *modifiables* utilisé par le Kernel Vous pouvez par exemple modifier les couleurs par défaut des fenêtres, changer le fond & résolutions d'écran, utiliser les ombres ou pas etc...

En bleu, les variables déjà définies
En rouge, non modifiables
En noir, non définies et A FAIRE
En gris, non définies.

Voici la liste et leurs utilités :

- Information du système d'exploitation :

- OS	[Nom de votre Système d'exploitation]
- VERSION	[Version de votre Système d'exploitation]
- ORG	[ORGanisation/Entreprise qui le crée]
- SOURCE	[Votre site internet principal *ex : cpcdos.fr.nf*]
- CONTACT	[Votre E-Mail de contact]
- BOOTOS	[Cible du fichier « BOOT » de l'OS Par défaut *«OS\INDEX.CPC /l:Autoexec»*]

- Répertoires systèmes :

- PROG	[Répertoire où se trouve les programmes]
- MEDIA	[Répertoire des fichiers media (image, son etc..)]
- SYSTEME	[Répertoire du système]

- Interface :

- SCR_FOND	[Cible du fichier du fond d'écran du bureau Si = 0 alors couleur de fond]
- FOND_COULEUR_R - FOND_COULEUR_V - FOND_COULEUR_B	[Couleur de fond du bureau si il y a pas d'image de fond Si l'image est indisponible ou si le safe mode est lancé]
- SCR_BAS	[Résolution d'écran – *(Base 800x600 & 1024x768)*]
- SCR_BIT	[Bit de couleur – 8 16 24 et 32 bits]
- ECRX	[Taille x de l'écran initialisé]
- ECRY	[Taille y de l'écran initialisé]
- CURPOSX	[Position actuel du curseur console X]
- CURPOSY	[Position actuel du curseur console Y]

- CONSMENU [Affichage du menu de la console *1 sinon 0*]
- CONSCERR [Couleur message d'erreur de la console – Rouge par défaut]
- CONSCAVT [Couleur message d'avertissement de la console – Jaune par défaut]
- ANTI_DEB_X [ANTI DEBordement horizontale des fenêtres]
- ANTI_DEB_Y [ANTI DEBordement verticales des fenêtres]
- EXP_SCROL [Valeur du scrolling de l'explorateur *(Valeur 0 à ..)*]
- EXP_DIM_X [Taille x et y de l'explorateur taille Y minimum = 300]
- EXP_DIM_Y
- EXP_COULEUR_F_R [Couleur R,V,B du fond de l'explorateur]
- EXP_COULEUR_F_V
- EXP_COULEUR_F_B
- EXP_COULEUR_C_R [Couleur R,V,B des caractères de l'explorateur]
- EXP_COULEUR_C_V
- EXP_COULEUR_C_B
- EXP_CIBLE [Cible du chemin d'accès de l'explorateur]
- EXP_CIBLE_DEF [Cible par défaut si ouverture sans paramètre de l'explorateur]
- EXP_RETOUR [Variable communicante pour le bouton «retour» de l'explorateur *(Valeurs 1 ou 0)*]
- FENETRE_ALPHA [Tolérance de transparence de la barre de fenêtre *(valeurs 0 à 255) par défaut 100*]
- FENETRE_OMBRE [Tolérance de l'assombrissement de l'ombre de la fenêtre *(valeurs 0 a 255) par défaut 30*]
- FENETRE_COULEUR_R [Couleurs RVB de la barre de la fenêtre *si Type=C1*
- FENETRE_COULEUR_V *(valeurs pour chaque 0 à 255)*
- FENETRE_COULEUR_B *par défaut 200 200 et 250*]
- FENETRE_TITRE_COULEUR_R [Couleur RVB du titre de la fenêtre]
- FENETRE_TITRE_COULEUR_V
- FENETRE_TITRE_COULEUR_B
- TAILLE_TITRE [Taille Y de la barre de titre]
- TXT_PROTECT [Caractère remplaçant du textebox protégé (ex. mot de passe) par défaut «*»]
- ANNIMATION [1 :Lance l'écran du Loading Screen 0:Ferme l'écran Assurez-vous d'avoir placé une image bmp « FOND.BMP » dans le dossier INIT/ et d'avoir répartis la variable suivante]
- BAR_PROGRESSION [Indice en pourcentage de la progression du chargement Par défaut elle est déjà répartit dans les fichiers afin de connaître l'état du chargement du noyau + OS (Valeur : 0 à 100)]

- ANNIM_TYPE_BARRE [Type de barre de chargement
0 :Aucune 1 :Progression 2:Style de Windows XP
3 : Les deux]

- ANNIM_COULEUR_F_R [Couleur RVB de fond de la barre de progression du
- ANNIM_COULEUR_F_V loading Screen]
- ANNIM_COULEUR_F_B

- Kernel :

- KRNL_PROC_NB [Nombre de processus en cours]
- KRNL_PROC_LST [Liste des processus en cours]
- MEMFREE [Mémoire disponible en octets]
- EXE_EN_COURS [Fichier .CPC en cours d'exécution]
- AEXE_EN_COURS [Ancien fichier .CPC exécuté précédemment]
- CLK [CLOCK – vitesse de traitement des commandes en pourcentage 1-100]
- DEBUG [Affichage des information de traitement et exécutions en console *1:active 0:désactivé*]
- LOG [Enregistrement + précisions des informations de débogage au fichier DEBUG.LOG]
(Utile pour connaître les causes des plantages)
- SYS_STACK [Mémoire STACK – 4096 ou 8128 par défaut]
- ACTMENU [Affichage du menu d'activité du système graphique *(Valeurs 1 ou 0)*]
- VER_KERNEL [Version Majeur et mineure du noyau]
- VER_DATE [Date de sortie du noyau]
- MULTI_TACHE [Permet de donner la priorité multitâche du système
Valeur entre 0 et infini (valeur 10 conseillée)
Plus la valeur est petite plus c'est lent mais plus fluide
| 0 = Désactivé]
- IUG_RESET(x) ['x' étant une valeur entre 1 et 8. Tableau des fichiers CPC à charger après l'utilisation de la commande IUG/ /RESET]

Ces variables suivantes se génère après l'utilisation de la commande :
SYS/ /DISQUE_INFO *{C, D, E... Z}*

- SYS_DISQUE_NUM [Numéro de série du lecteur]
- SYS_DISQUE_LABEL [Nom du lecteur]
- SYS_DISQUE_SYS [Système de fichier]
- SYS_DISQUE_OPS [Octets par secteur]
- SYS_DISQUE_SPC [Secteur par cluster]
- SYS_DISQUE_CTOTAL [Total de clusters]
- SYS_DISQUE_CDISPO [Clusters disponibles]

- SYS_DISQUE_OTOTAL [Octets total (Calcul C/H/S automatique)]
- SYS_DISQUE_OLIBRE [Octets libre]

Pour les modifier, il suffit d'utiliser la commande **FIX/**

Par exemple :

Si vous voulez modifier le fond d'écran, accédez a la console puis taper :

```
fix/ SCR_FOND = OS\MEDIA\FOND\IMAGE.BMP
```

Mettez bien sûre autre chose que cette cible, un fichier existant !
Si vous êtes sur une console graphique il faut taper LC/ /CONSOLE puis IUG/ afin de réinitialiser l'affichage du fond d'écran.

Modifier la résolution d'écran :

```
fix/ SCR_BAS = 800x600
```

Voir la liste des résolutions d'écran ci dessous

Utile :

Pour mettre le résultat d'une commande dans une variable exemple :

```
@#MA_VARIABLE SYS/ /TESTECR 16
```

Cette exemple place toutes les résolution d'écran dans la variable «MA_VARIABLE»

ou encore :

```
@#MA_VARIABLE SYS/ /TESTVESA
```

Cette exemple place dans la variable «MA_VARIABLE» si le VESA est supporté ou non.

Rappel :

Pour afficher la liste des variables en mémoire :

```
fix/ /liste
```

Si il y en a trop à l'affichage, ajouter le paramètre /PAUSE

```
fix/ /liste /pause
```

Pour afficher directement sur la console une variable en mémoire :

```
txt/ %VARIABLE%
```

Mettez bien sûre autre chose que VARIABLE, donc une variable qui existe !

LES RESOLUTIONS D'ECRAN

Voici la liste des résolutions que le Kernel est capable de gérer, après a voir si votre carte graphique peux les supporter.

320x200, 320x240, 320x400, 320x480, 400x300, 512x384, 640x350, 640x400, 640x480, 640x640, 848x480, 720,480, **800x600**, **1024x768**, 1080x1050, 1152x864, 1280x854, 1280x720, 1280x960, 1280x1024, 1366x768, 1440x900, 1440x960, 1440x1050, 1600x900 **1600x1200**, 1920x1200, 1920x1440, 2048x1536

En **GRAS** les mieux et les plus rependus et les plus fonctionnels

Si aucunes de ces résolutions correspond a votre choix, Cpcdos choisira celui de base : 800x600x16b

Pour tester des résolutions tapez :

```
sys/ /TESTECR 16
```

« 16 » indiquant 16Bits, il y à aussi 8 24 et 32Bits
Puis vous avez la liste des résolutions supporté par la carte graphique selon le Bit de couleur choisissez la meilleure, pour votre OS (à configurer dans la variable « SCR_BAS »)

Si vous avez une résolution affiché, mais pas supporté par le noyau, veuillez le signaler sur le forum cpcdos afin de l'ajouter ! (voir page de garde)

Modifier la résolution d'écran par exemple pour 1024x768 tapez :

```
fix/ SCR_BAS = 1024x768
```

En 32 bit ? Tapez :

```
fix/ SCR_BIT = 32
```

CONFIGURER & TESTER LE SYSTEME

La commande qui le permet est :

```
sys/
```

Elle ne peux être exécutée toute seule, elle lui faut des paramètres, comme

Tester le CPU :

```
sys/ /TEST
```

Tester le VESA :

```
sys/ /TESTVESA
```

Affiche un message si oui ou non le VESA est compatible

Tester le Kernel :

```
sys/ /KRNLTEST
```

Lister les modes graphiques :

```
sys/ /TESTECR {Bit}
```

A la place de *{BIT}* tapez 8, 16, 24 ou 32

Mémoire disponible:

```
sys/ /MEM
```

Charger un pilote DOS :

```
sys/ /device {chemin fichier}
```

Booter sur un serveur ou local :

```
sys/ /boot:{local/reseau \\serveur}
```

Pour cette version l'utilisation de cette commande n'est pas conseillée

Créer une image boot virtuel RAM

```
sys/ /CREERIMG {Lecteur:}
```

Le *lecteur* correspond au lecteur de destination où le fichier image est enregistré (C: par défaut)
> Puis pour booter, configurez le boot du noyau avec la commande *SYS/ /BOOTIMG {lecteur:}* <

(Conseillé d'utiliser FreeDos)

Booter sur une image virtuel :

```
sys/ /BOOTIMG {0|RESEAU|DOSBOX|lecteur:}
```

Le paramètre */BOOTIMG 0* permet de restaurer le boot pour PC par défaut
Le paramètre */BOOTIMG DOSBOX* permet de restaurer le boot pour DosBox
Le paramètre */BOOTIMG **lecteur**:* permet d'installer le boot sur le ***lecteur*** virtuel de destination
Le paramètre */BOOTIMG RESEAU lecteur:* ***SERVEUR*** Idem, mais en récupérant le boot sur le **serveur** distant
Il faut que la cible du SERVEUR sois placé dans le dossier racine où se trouve le dossier CPCDOS

> Par exemple si sur le serveur le dossier SYSTEME se trouve dans
\\SERVEUR\PARTAGE\CPCDOS
et le lecteur virtuel installé sur la machine est d:

Alors dans la commande vous taperez :

```
SYS/ /BOOTIMG RESEAU d: \\SERVEUR\PARTAGE\
```

Il faut mettre le dossier racine où se trouve le dossier CPCDOS
Et ensuite le noyau va chercher KRNL_205.IMG

Cette technique de boot virtuel améliore la vitesse de votre OS presque x20 plus rapide !!

Si vous voulez booter normalement vous avez le choix « n » dans le menu sélectif
ou si vous êtes sous l'interpréteur DOS, taper

```
KRNL32 NOVIRT
```

Beeper le système :

```
sys/ /beep {Fréquence}-{Durée en ms}
```

Exemple :

```
sys/ /beep 1048-60
```

Récupérer les information d'un lecteur :

```
sys/ /DISQUE_INFO {lecteur A, B, C … Z}
```

Exemple :

```
sys/ /DISQUE_INFO C
```

Les informations du lecteur définit seront inscrites temporairement dans ces variables suivantes :

%SYS_DISQUE_NUM%	Numéro de série du lecteur
%SYS_DISQUE_LABEL%	Nom du lecteur
%SYS_DISQUE_SYS%	Système de fichier
%SYS_DISQUE_OPS%	Octets par secteur
%SYS_DISQUE_SPC%	Secteur par cluster
%SYS_DISQUE_CTOTAL%	Total de clusters
%SYS_DISQUE_CDISPO%	Clusters disponibles

%SYS_DISQUE_OTOTAL% Octets total (Calcul C/H/S automatique)
%SYS_DISQUE_OLIBRE% Octets libre

Comment je fais pour avoir un résultat Mo ? Go ?

Il faut faire un calcul (Taille totale ou libre / 1024 ^ 1(Ko) ou 2(Mo) ou 3(Go)

Exemple pour avoir la taille totale avec le lecteur D : (si vous n'avez pas de D utilisez C)

```
FIX/ DISQUE = D
@SYS/ /DISQUE_INFO %DISQUE%
REM/ Resultat en Ko:1024 en Mo:1024^2 ou en Go:1024^3
REM/ Si vous modifiez, oubliez pas de modifier l'unite en derniere ligne
FIX/ Variable1 = /c 1024 ^ 2
REM/ Convertir en Mo
FIX/ Resultat = /c %SYS_DISQUE_OTOTAL% / %Variable1%
REM/ Arrondir la valeur
FIX/ Resultat = /c INT >%Resultat%
TXT/ Taille du disque %DISQUE%: "%SYS_DISQUE_LABEL%": %Resultat% Mo
```

Exemple pour avoir la taille utilisée avec le lecteur D : (si vous n'avez pas de D utilisez C)

```
FIX/ DISQUE = D
@SYS/ /DISQUE_INFO %DISQUE%
REM/ Resultat en Ko:1024 en Mo:1024^2 ou en Go:1024^3
REM/ Si vous modifiez, oubliez pas de modifier l'unite en derniere ligne
FIX/ Variable1 = /c 1024 ^ 2
REM/ Taille totale - Taille restante
FIX/ Resultat = /c %SYS_DISQUE_OTOTAL% - %SYS_DISQUE_OLIBRE%
REM/ On convertit en Mo
fix/ Resultat = /c %resultat% / %variable1%
REM/ Et arrondir la valeur
fix/ resultat = /c int >%resultat%
TXT/ Taille utilisee du disque %DISQUE%: "%SYS_DISQUE_LABEL%": %Resultat% Mo
```

Créer un lecteur virtuel :

```
sys/ /VIRTUEL
```

Permet de créer un lecteur virtuel (en RAM) ce qui fait un lecteur extrêmement rapide
Si le lecteur virtuel est déjà installé, il est fortement déconseillé de réutiliser cette commande
Mais elle sera bloqué, vous pouvez forcer l'utilisation :

```
sys/ /VIRTUEL /FORCE
```

Enregistrer automatiquement le lecteur virtuel installé dans la variable %SYS_VIRTUEL%

```
sys/ /VIRTUEL /FIX
```

Reinitialiser le lecteur virtuel

```
sys/ /VIRTUEL /RESET
```

/!\ Attention cette commande efface tout ce qu'il y a du dossier CPCDOS\ dans le lecteur indiqué dans %SYS_VIRTUEL%

Si cette variable n'a pas été définit, il sera alors impossible de réinitialiser le lecteur.

Chapitre VI – Exemples de programmes (Copier/coller) :

Ce programme crée un exécutable .com 16Bit avec une petite phrase personnalisée

```
REM/ Pour Ecrire du texte dans un executable .com 16Bit
lc/
fix/ debug = 0
cls/
txt/ Taper une phrase
fix/ /q PHRASE
FIX/ TAILLE_PHRASE = /C LEN >%PHRASE%
fix/ BB = 0
fix/ PHRASE_HEXA = #NUL
TXT/ Conversion en cours...
:boucle1:
FIX/ BB = /c %BB% + 1
REM/ Capturer 1 lettre par lettre et le convertir en HEXA
FIX/ CAPTURE = /c CAP >%PHRASE%;%BB%-1
FIX/ CAPTURE = /c ASC >%CAPTURE%
FIX/ CAPTURE = /c HEX2 >%CAPTURE%
REM/ Assembler les codes HEXA
FIX/ PHRASE_HEXA = %PHRASE_HEXA%%CAPTURE%
SI/ %BB% < %TAILLE_PHRASE% (:aller/ boucle1:)
REM/ Assemblage du STUB .COM et de la phrase convertit
fix/ STUB = B409BA0901CD21CD20%PHRASE_HEXA%0D242000
FICHIER/ /SORTIRB #1;PROG.COM
fix/ POSYO = /C %CURPOSY% + 1
fix/ TAILLE = /C LEN >%STUB%
fix/ TAILLE = /C %TAILLE% - 2
fix/ BB = -1
:boucle2:
Fix/ BB = /c %BB% + 2
REM/ Capture 1 octet par 1 octet et conversion en Caracteres ASCII et enregistrement
FIX/ DONNEES = /c CAP >%STUB%;%BB%-2
fix/ DONNEES = &H%DONNEES%
FIX/ DONNEES = /C VAL >%DONNEES%
fix/ DONNEES = /c CHR >%DONNEES%
FICHIER/ /ECRIREB #1;%DONNEES%
POSY/ 8
txt/ Creation du fichier executable %BB%/%TAILLE%
si/ %BB% < %TAILLE% (:aller/ boucle2:)
fichier/ /fermer #1
txt/ Terminee, execution de PROG.COM
shell/ PROG.COM
txt/
stop/
```

Par FAVIER Sébastien 01 (01/09/2013)

Ce programme à exécuter en console vous pose des question, a vous de répondre !

```
fix/ debug = 0
lc/
:recommencer:
txt/ Bienvenue sur le programme exemple d'Aly !!
txt/ Comment vous appelez vous ?
fix/ /q NOM
txt/ Aaaah donc, vous vous appelez %NOM% !
txt/ Pourriez vous me donner votre age ?
fix/ /q AGE
txt/ D'accord %NOM%, vous avez %AGE%ans
si/ %AGE% > 18 (:txt/ Vous etes majeur:)
si/ %AGE% < 18 (:txt/ Vous etes mineur:)
txt/ Nous allons faire un petit test d'intelligence
fix/ ESSAIS = 0
:boucle1:
txt/ Que fait 98 x 2 + 54 ?
fix/ REP = 0
fix/ ESSAIS = /c %ESSAIS% + 1
fix/ RES = 250
fix/ /q REP
si/ %REP% N %RES% (:aller/ boucle1:)
si/ %REP% = %RES% (:txt/ Bien joue ! 98*2 = 196    196+54 = 250 !:)
:boucle2:
txt/ Que fait 25 x 13 + 111 - 36 ?
fix/ REP = 0
fix/ RES = 400
fix/ ESSAIS = /c %ESSAIS% + 1
fix/ /q REP
si/ %REP% N %RES% (:aller/ boucle2:)
si/ %REP% = %RES% (:txt/ Bien joue ! Le resultat est bien 400:)
si/ %ESSAIS% < 3 (:txt/ Vous avez reussi les 2 tests du premier coup !:)
si/ %ESSAIS% > 2 (:txt/ Vous avez reussi le test en %ESSAIS% essais !:)
txt/ Voulez vous rejouer le programme ? ( 1 = Oui 0 = Non )
fix/ REJOUER = 5
touche/ /p REJOUER
si/ %REJOUER% = 1 (:aller/ recommencer:)
si/ %REJOUER% = 0 (:txt/ Au-revoir et bonne journée !:)
lc/ console
stop/
```

Par Léo VACHET (12 Mai 2013)

Ce programme à exécuter en console vous aussi pose des question ;-)

```
fix/ debug = 0
lc/
rem/ == Variables ==
fix/ AGE1 = 18
fix/ AGE2 = 17
fix/ nom = Steve
fix/ nom2 = Sebastien

rem/ == Reinitialiser les variables ==
fix/ GRAND = 0
fix/ PETIT = 0

rem/ == Chercher qui est plus grand ou petit ==
si/ %AGE1% > %AGE2% (:fix/ GRAND = 1:)
si/ %AGE1% < %AGE2% (:fix/ PETIT = 1:)

rem/ == Affichage texte ==
si/ %GRAND% = 1 (:txt/ %nom% est plus grand que %nom2%:)

si/ %PETIT% = 1 (:txt/ %nom% est plus petit que %nom2%:)

rem/ == Effacer les variables / liberer la memoire ==
@fix/ /s age1
@fix/ /s age2
@fix/ /s nom
@fix/ /s nom2
fix/ debug = 1
stop/
```

Par Steve Prudhomme (20 Janvier 2013)

Ce programme crée une fenêtre et un bouton cliquable et dit combien de fois vous avez cliqué dessus (a enregistrer dans le dossier *OS\PROG*)

```
ini/ fenetre(
        ini;nom = "FENETRE_3"
        ini;texte = "Ma petite fenetre !"
        ini;type = "1"
        ini;couleur = "087,215,186"
        ini;tx = "300"
        ini;ty = "250"
        ini;px = "MX"
        ini;py = "250"        creer/
ini/ fenetre)
ini/ bouton(
        ini;nom = "BOUTON1_F3"
        ini;fenetre = "FENETRE_3"
        ini;texte = "Clique moi !"
        ini;img = "7"
        ini;couleurf = "255,255,255"
        ini;couleurp = "255,000,000"
        ini;tx = "180"
        ini;ty = "30"
        ini;px = "10"
        ini;py = "100"
        creer/
       ev/ OS\PROG\EV.CPC
ini/ bouton)
ini/ bouton(
        ini;nom = "BOUTON2_F3"
        ini;fenetre = "FENETRE_3"
        ini;texte = "Plus"
        ini;img = "6"
        ini;couleurf = "255,155,155"
        ini;couleurp = "255,000,000"
        ini;tx = "60"
        ini;ty = "30"
        ini;px = "20"
        ini;py = "10"
        creer/
       ev/ OS\PROG\EV.CPC
ini/ bouton)
ini/ bouton(
        ini;nom = "BOUTON3_F3"
        ini;fenetre = "FENETRE_3"
        ini;texte = "Moins"
        ini;img = "0"
        ini;couleurf = "155,155,255"
        ini;couleurp = "255,000,000"
        ini;tx = "60"
```

```
        ini;ty = "30"
        ini;px = "20"
        ini;py = "50"
        creer/
       ev/ OS\PROG\EV.CPC
ini/ bouton)
fix/ NBPM = 5
ini/ label(
       ini;fenetre = "FENETRE_3"
       ini;nom = "LABEL_1"
       ini;texte = "5"
       ini;couleurf = "200,200,200"
       ini;couleurp = "000,000,000"
       ini;transparent = "0"
       ini;type = "2"
       ini;tx = "200"
       ini;ty = "20"
       ini;px = "100"
       ini;py = "10"
       Creer/
ini/ label)
rem/ un bouton
ini/ bouton(
        ini;nom = "BOUTON4_F3"
        ini;fenetre = "FENETRE_3"
        ini;texte = "Autre"
        ini;img = "4"
        ini;couleurf = "255,255,255"
        ini;couleurp = "255,000,000"
        ini;tx = "110"
        ini;ty = "30"
        ini;px = "100"
        ini;py = "20"
        creer/
       ev/ OS\PROG\EV.CPC
ini/ bouton)
Fix/ NBIMG = 0
```

Fichier EV.CPC

```
rem/ Fichier d'evenements CpcdosC+ de la fenetre TESTE :)
PROC/ Bouton1_F3(CLIC)
       Fix/ NBIMG = /c %NBIMG% + 1
       ini/ bouton(
              ini;nom = "Bouton1_F3"
              ini;fenetre = "FENETRE_3"
              ini;texte = "T'a clique %NBIMG% fois"
              creer/
       ini/ bouton)
```

```
FIN/ PROC
PROC/ Bouton2_F3(CLIC)
       rem/ Plus
       fix/ NBPM = /c %NBPM% + 1
       ini/ label(
              ini;fenetre = "FENETRE_3"
              ini;nom = "LABEL_1"
              ini;texte = "%NBPM%"
              Creer/
       ini/ label)
FIN/ PROC
PROC/ Bouton3_F3(CLIC)
       rem/ Moins
       fix/ NBPM = /c %NBPM% - 1
       ini/ label(
              ini;fenetre = "FENETRE_3"
              ini;nom = "LABEL_1"
              ini;texte = "%NBPM%"
              Creer/
       ini/ label)
FIN/ PROC
PROC/ BOUTON4_F3(CLIC)
       exe/ os\PROG\FENETRE4.CPC
FIN/ PROC
```

Par FAVIER Sébastien et FROMONT Thomas (Février&Septembre 2013)

REMERCIEMENTS AUX TESTEURS CONTRIBUTEURS ET COMMENTATEURS DE CPCDOS & MANUEL UTILISATEUR

- Timothée LUSSIAUD
- Léo VACHET
- Mathieu RIBEIRO
- Thomas FROMONT
- Steve PRUDHOMME
- Gabriel LABROSSE
- Charles PROVENT
- Sviat SMOLINET

LIENS

Site internet principal de Cpcdos : http://cpcdos.fr.nf/ ou http://cpcdos.e-monsite.com/
Site d'autres projets Microsf01 : http://microsf01.fr.nf/ ou http://microsf01.e-monsite.com/
Forum : http://forum-cpcdos.fr.nf/ ou http://data-serveur.verygames.net/cpcdos/forum/
Nouveautés : http://cpcdos.e-monsite.com/pages/news.html
Au projet (à finir) : http://cpcdos.e-monsite.com/pages/au-projet-a-faire.html
Liste des OS : http://cpcdos.e-monsite.com/pages/systemes-d-exploitation-base-cpcdos.html
Programmes téléchargeables : http://cpcdos.e-monsite.com/pages/programmes-cpcdosc.html
« Brouillon » à la page : http://microsf01.e-monsite.com/pages/cpcdos-os2-1.html
Chaîne YouTube : https://www.youtube.com/channel/UCShOH7zxE4f-r_KU-PlNdNg/videos
Et la page Facebook : https://www.facebook.com/pages/Kernel-Cpcdos-OSx/479523255400921

Microsf01 FAVIER Sébastien 01

sebastien.ordinateur@hotmail.fr
http://microsf01.fr.nf/