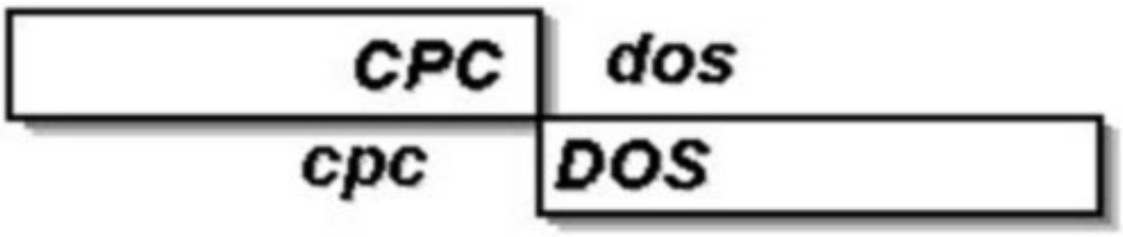
FAVIER Sébastien 01
MICROSF01



Microsf01 CpcdosC+

Manuel doc.1.2013

(En cours de rédaction – 19/05/2013)

Apprendre la programmation CpcdosC+
pour le Kernel Cpcdos OSx

- Version OS2.0.5 Alpha 1.1 32Bit -

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
Bugs&Info Contact : sebastien.ordinateur@hotmail.fr ou (ci-dessus)

Copyright©Microsf01

Avertissement :

Ce logiciel est protégé par la loi relative au droit d'auteur et par les conventions internationales. Toute reproduction ou distribution partielle ou totale de ce logiciel, par quelque moyen que ce soit, est strictement interdite. Toute personne ne respectant pas ces dispositions se rendra coupable du délit de contrefaçon et sera passible des sanctions pénales prévues par la loi.

- - -

Copyright©Microsf01 J8781B5 depuis Mai 2011

Sommaire :

Copyright©Microsf01

- Démarrer l'initialisation de l'OS - - - `demarrer/`
- Lancer l'interface graphique (l'OS) - - `iug/`
- Les événements - - - - - `ev/`
- Afficher un MSGBOX - - - - `msgbox/`
- Fermer une fenêtre ou objet - - - `fermer/`
- Actualiser une ou plusieurs propriétés - - `actualise/`
- Focus sur une fenêtre - - - - `focus/`
- Lancer le mode console - - - - `lc/`
- Lire ou 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**
  3. Status du Cache CPU - **/cpu_cache**
  4. Level du CPU - **/cpu_level**
  5. Compatibilité du VESA - **/testvesa**
  6. Lister modes d'écran&bit - **/testecr**
  7. Initialiser le Kernel&SCI - **/ini_krnl**
  8. Tester la mémoire Etendu (XMS) - **/memtest**
  9. Nettoyer la mémoire - **/memclear**
  10. Tester le CPU - **/test**
  11. Appel interruptions DOS - **/int**
  12. Écrire & lire dans la mémoire - **/poke** & **/peek**
  13. Mémoire buffer - **/buffer**
  14.
  15.

6. Exemples - - - - - - - - - - - - - Chapitre VI
7. Remerciements & liens

Copyright©Microsf01

Chapitre I – Projet Cpcdos ?

(Acronyme CPCDOS : Créé Pour Concevoir Des OS)

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

Tout cela grâce à un système 32Bit tout prêt, la séquence de démarrage , pilotes , paramètres déjà au point, une création d'une interface graphique très avancée et simple qui peux aller jusqu'à **32Bits de couleurs** , une résolution qui peux aller jusqu'à **2048x1536** *(1600x1200 testé > ICI)*

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

Rien viens Linux, aucuns rapport et pas le même système du tout!

\- - - - - - - - -
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 est regroupé en un bloc dans le code source

Multitâche Coopératif : Simple multitâche gestions de plusieurs processus en même temps , d'où les processus doivent permettre à une autre tâche de s'exécuter , son inconvénient c'est que si un des processus plante , le système entier peux bloquer , *voir Section critique*

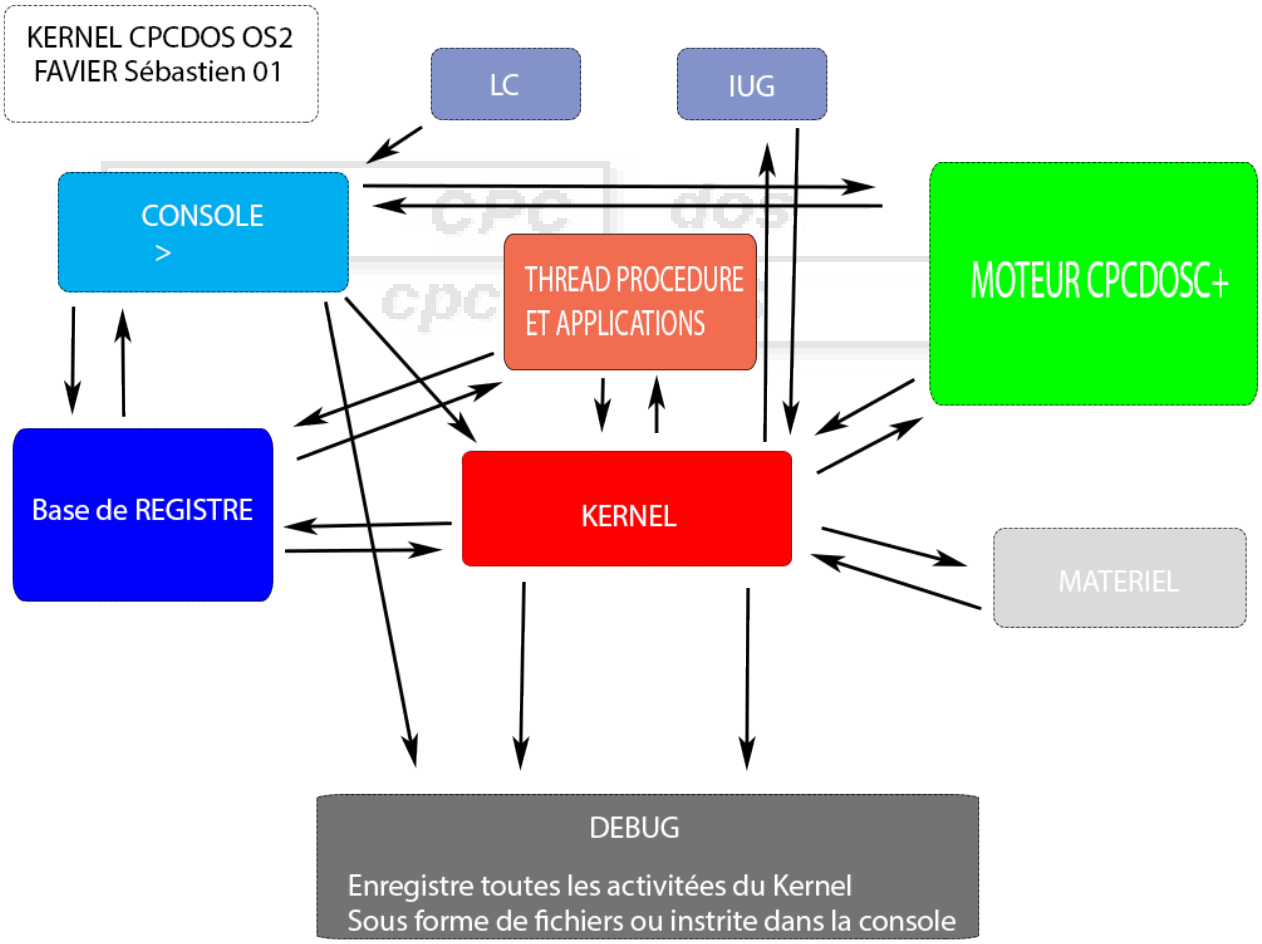
Son principe c'est d'exécuter du code CpcdosC+ en lignes de commandes (en Console) ou via des fichiers de format

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

(Débuté le 15 Juillet 2011)

Copyright©Microsf01

Schéma du fonctionnement du noyau



-

LC (Lignes Commandes) :

Partie en lignes de commandes semblabe à la partie console
Il permet 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 aux objets etc...

Console :

Partie où l'utilisateur entre ses commandes CpcdosC+
et peux être aux commandes de la partie Kernel

Base de REGISTRE :

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

THREAD PROCEDURE ET APPLICATION :

Partie où les information des propriétés et objets sont placée , et utilisé afin que la partie Kernel dessine

Copyright©Microsf01

sur l'interface

MOTEUR CPCDOSC+ :

Partie du système où toutes les commandes CpcdosC+ sont analysée et éxécutées par la partie KERNEL

MATERIEL :

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

KERNEL :

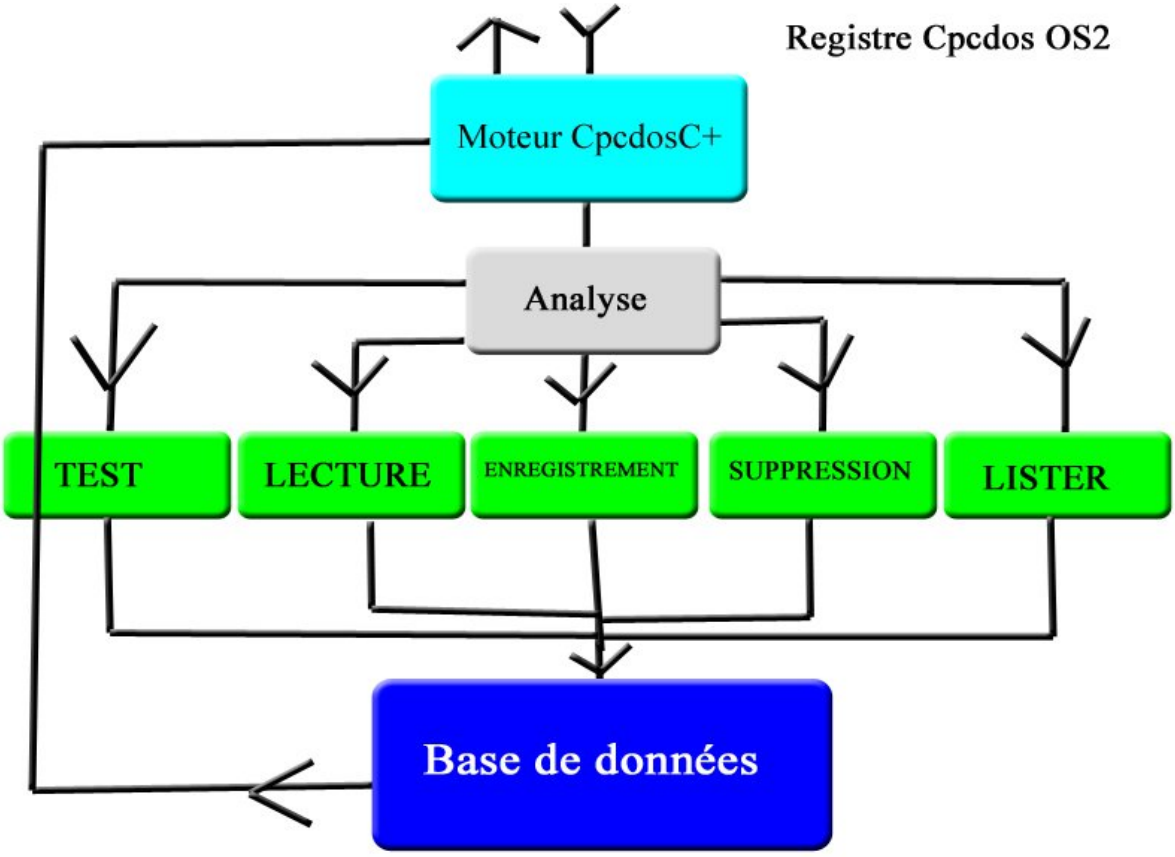
Partie NOYAU .. tout simplement celui "qui gère tout"..

DEBUG :

Partie assez importante qui enregistre ou affiche à la console , toutes le activitées 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)

Copyright©Microsf01

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)

Copyright©Microsf01

Chapitre III - Requirements minimum

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

Configuration minimum requis pour le PC :

Processeur : 800 *mhz* Intel *80x86* ou Amd (AM286)

Ram : 192 mo

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

Disque dur : au moins un 5 Go

Un processeur dans les allentours de

INTEL

Intel (P5) : Pentium

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

Intel (NetBurst) : Pentium 4

Intel (P6) Core 2 Duo

et plus

AMD

Amd Am386

Amd K5

et plus

Systèmes BIOS

Phoenix , Award Software

(Energy)

ARM : Ne fonctionne pas

Copyright©Microsf01

Installation sur Disque dur :

{ Page réservé à l'installation du Kernel sur un disque dur }
Prochainement sur le Doc 2.2012

Copyright©Microsf01

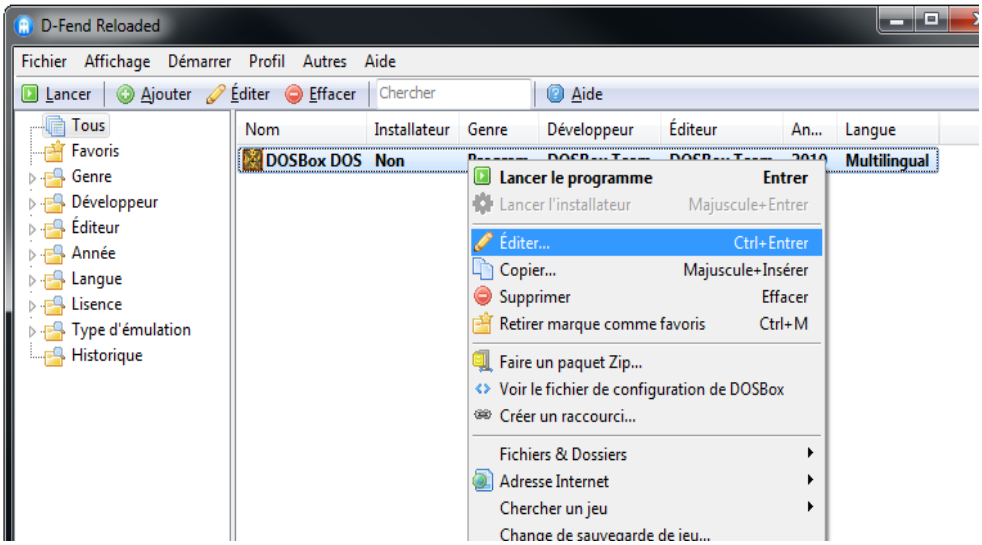
Installation avec DosBox :

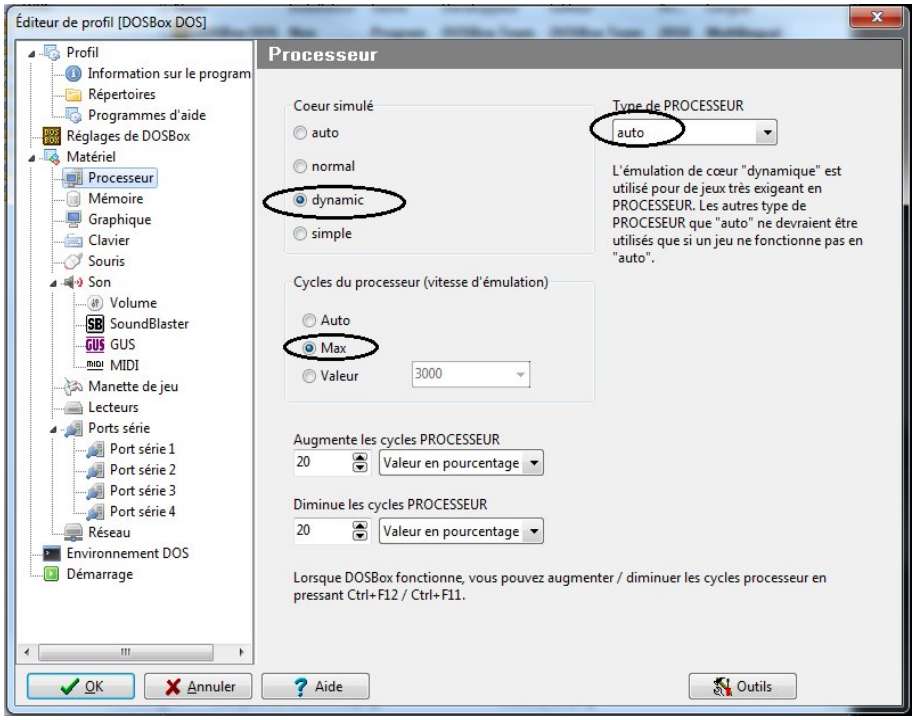
Pour commencer, une fois que vous avez le Kernel en main
veuillez télécharger et installer D-Fend
ou à l'adresse suivante : **http://dfendreloaded.sourceforge.net/Download.html**

Une fois téléchargé, créez le répertoire nommé **DosBox** dans **C:**
et copiez le contenu du dossier « RACINE A PLACER » dedans.

Puis lancez D-Fend

Clique droit sur
DosBox, Editer



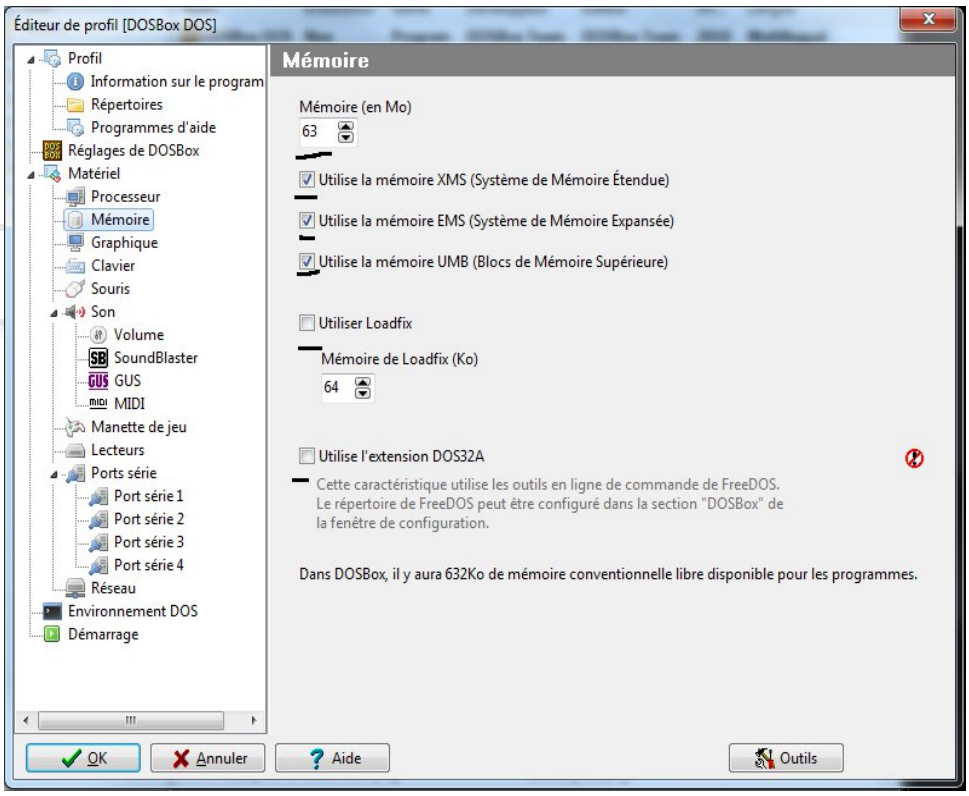


Choisissez et cochez

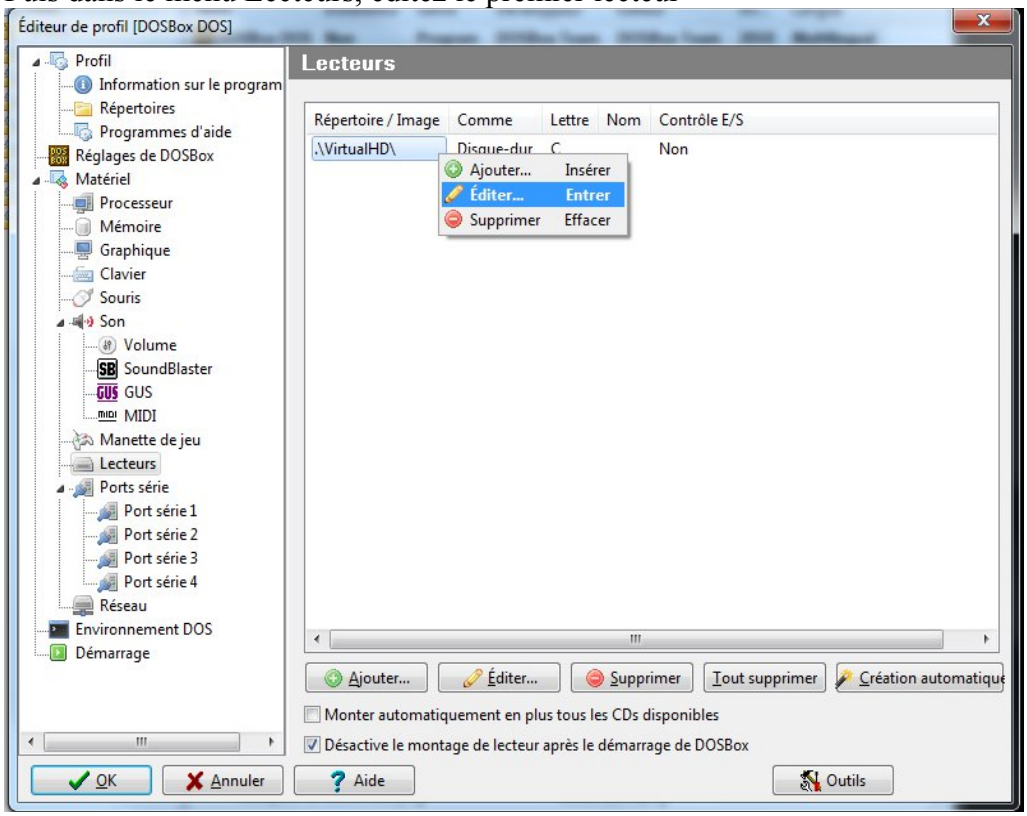
pour le cœur simulé, un **DYNAMIC** , les cycles du processeur à **MAX** et le type de processeur, un **AUTO**

Copyright©Microsf01

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

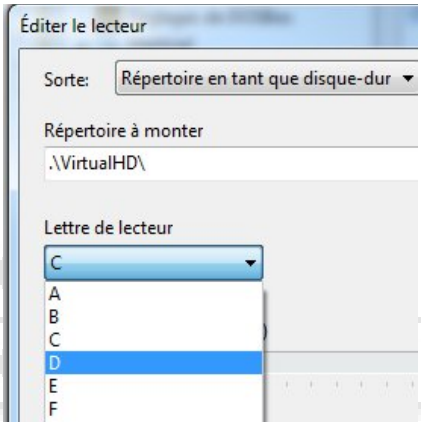


Puis dans le menu Lecteurs, éditez le premier lecteur

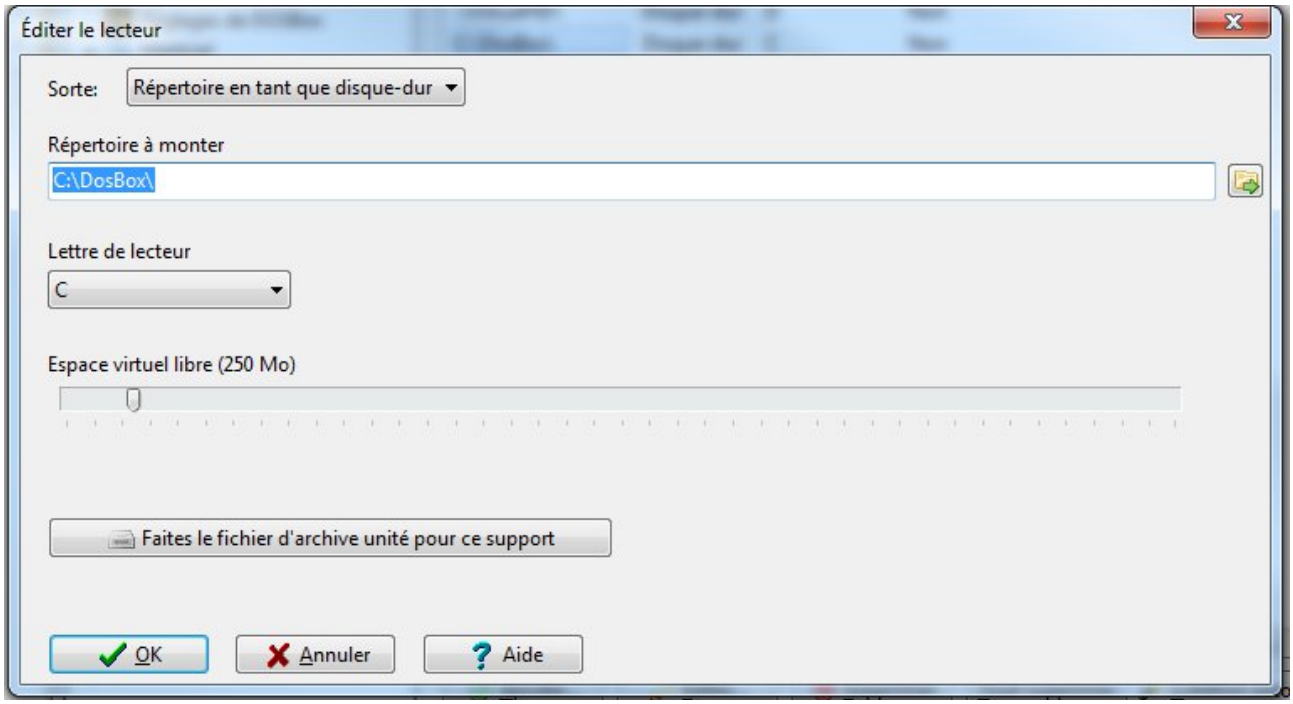


Copyright©Microsf01

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
et déclarez le comme un lecteur **C** puis validez

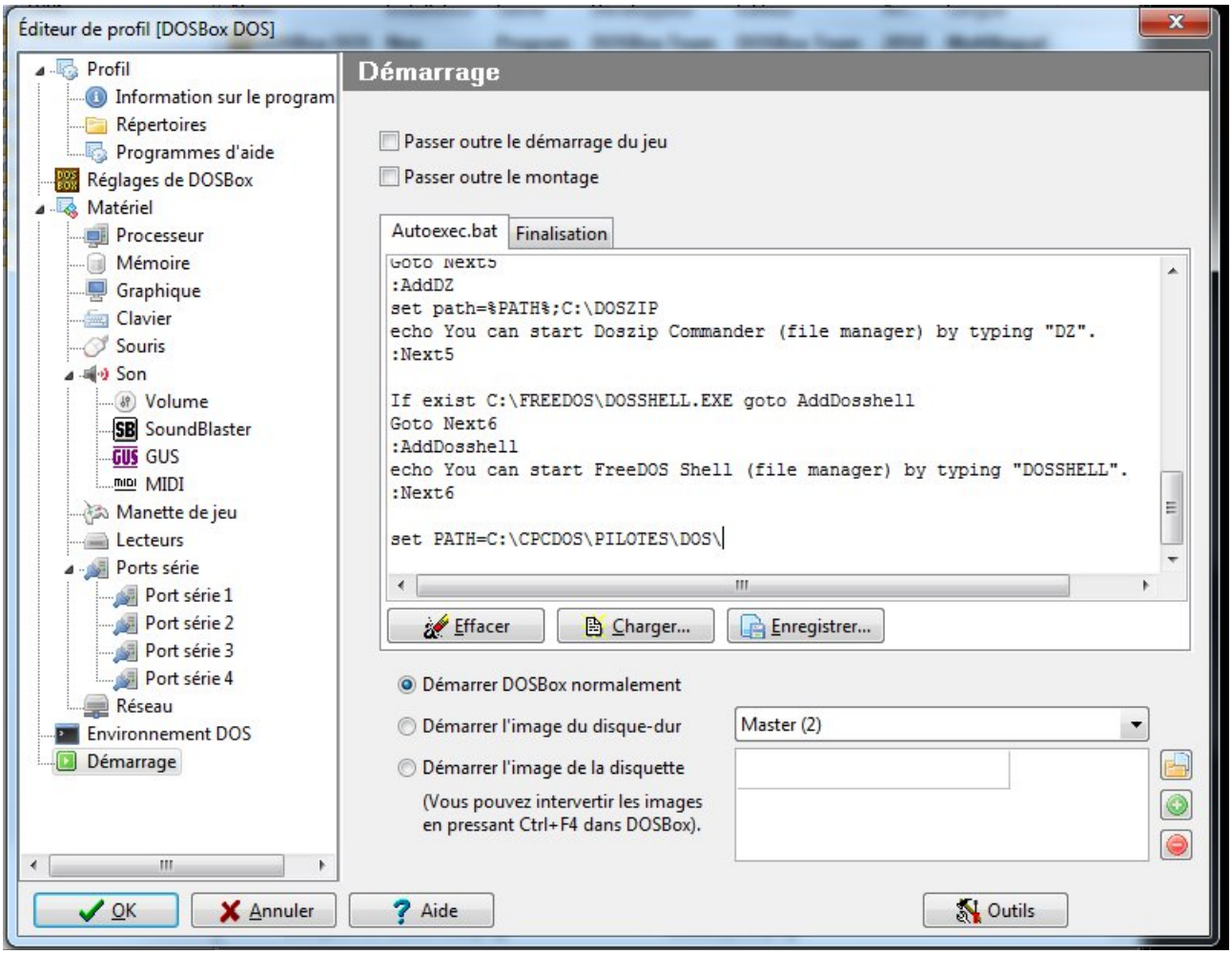


Puis dans le menu **Demarrage** ajoutez la dernière ligne

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


Copyright©Microsf01

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

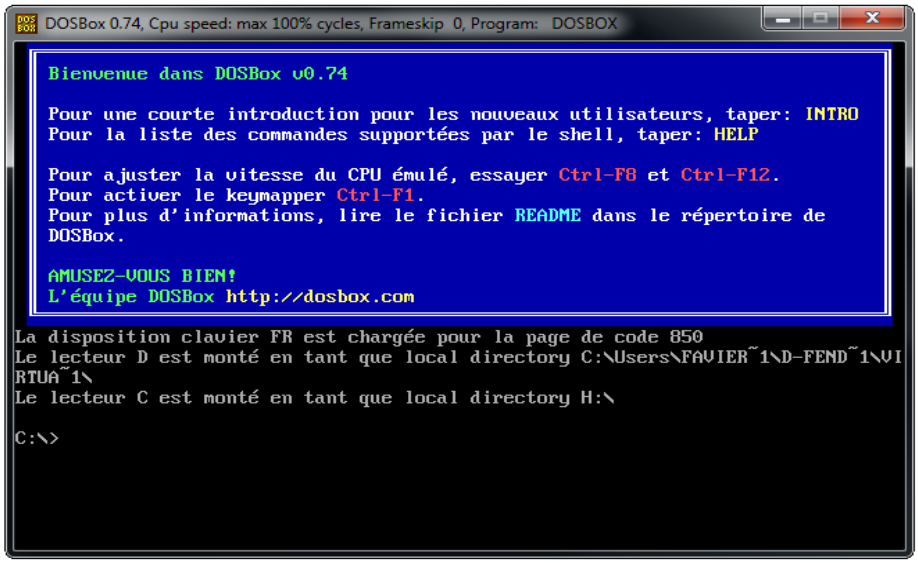


Puis validez le tout

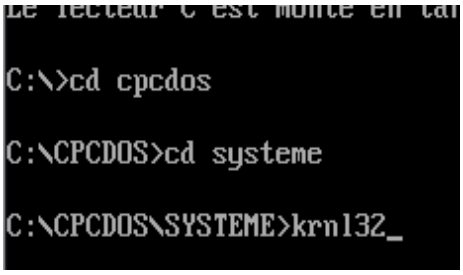
Lancez DosBox (double clique)

Copyright©Microsf01

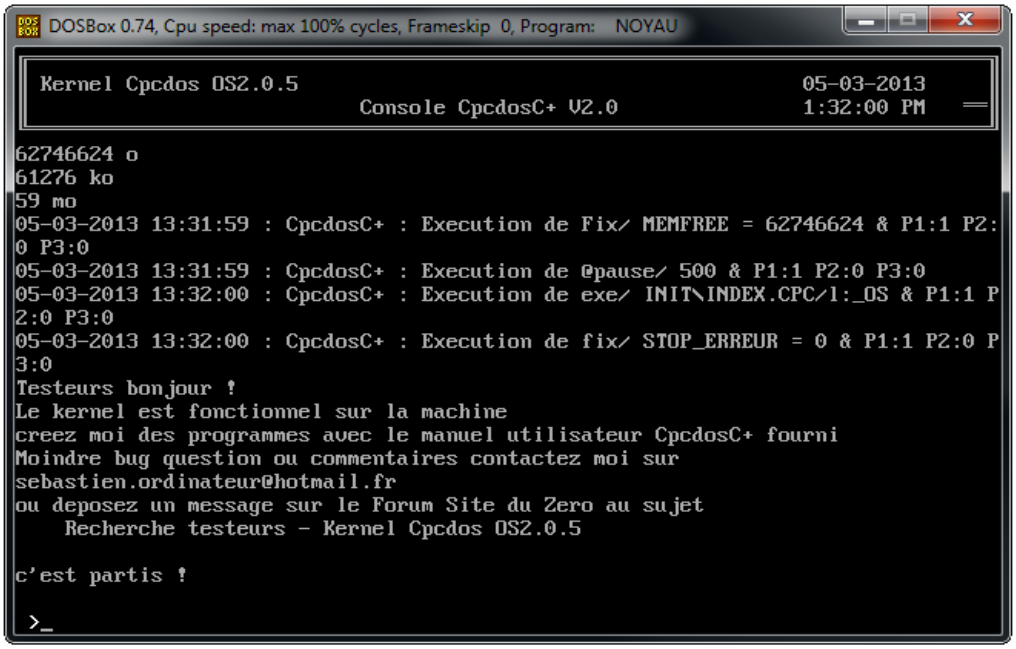
Vous devrez avoir ceci :



Puis tapez ces commandes



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



Copyright©Microsf01

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 appuiez sur CTRL+F4 pour mettre à jour les fichiers et dossiers et lancez le Kernel (si c'est pas déjà fait)
Puis taper à la console

```
exe/ PROG.CPC
```

Puis voilà

Suivez le manuel, moindre questions ou beugues, contactez moi par E-mail :
sebastien.ordinateur@hotmail.fr
ou sur le Forum du Site Du Zéro au sujet **Recherche testeurs - Kernel Cpcdos OS2.0.5**

Copyright©Microsf01

Chapitre IV - Commandes de bases

J'ai ce 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.
Ou joindre nous aider en rejoignant le groupe de développeurs sur Facebook

https://www.facebook.com/groups/microsf01

Mais avant de joindre ce groupe il faut postuler
rendez vous donc à cette page :

http://microsf01.e-monsite.com/pages/nous-aider.html

Aller , commençons la programmation !

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

Pour commencer les commandes de bases

NB (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*
- *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*
- *Les caractères suivants : |» veulent dire que la ligne continue avec celle de en-dessous (manque de largeur ..)*
- *Pour les caractères comme « é è â ä »... utiliser le format ASCII, sinon il n'est pas pris en charge...*
- *Évitez d'utiliser le caractère « : » dans vos commandes excepté les labels*
- *Pour utiliser une commande fantôme, il suffit de placer « @ » juste au début de votre commande ex :* **@txt/ Ma commande fantome**

Copyright©Microsf01

EFFACER L'ECRAN

La commande qui le permet est :

```
cls/
```

(Clear Screen)

Permet d'effacer l'écran,

Exemple :

```
txt/ Appuiez sur une touche pour effacer l'écran
Pause/
cls/
txt/ Ecran effacé !
```

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

Copyright©Microsf01

AFFICHAGE TEXTE

La commande qui permet ceci est :

```
txt/ {texte}
```

(**T**e**XT**e)

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)

Copyright©Microsf01

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:%CURPOS%
```

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

```
fix/ CONSMENU = 0
```


Copyright©Microsf01

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 128 octets)

Puis pour « poser une question »

Il faut donc 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

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

Copyright©Microsf01

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 au Kernel a partir de quelle label le code s'exécute

Exemple :

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

Et dans Fichier.cpc :

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

Sortie :

texte2

Copyright©Microsf01

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 « > »
- **Supérieure à**, de signe « < »

Copyright©Microsf01

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 interpreteur 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++ , VB etc..

Une commande voisine existe et remplit la même fonction :

```
dos/
```


Copyright©Microsf01

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.

Exemple :

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

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

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

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

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

Copyright©Microsf01

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

LES FONCTIONS

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 (Tangeante)
- ATAN (ArcTengeante [TAN^-1])
- INT (Arrondir les valeurs)
- LEN (Obtenir le nombre de caractères dans une chaine ou variable)
- LOG (Logarithme)
- MAJ (Mettre une chaine de caractères ou variables en MAJuscules)
- MIN (Mettre une chaine de caractères ou variables en MINuscules)
- HEX (Convertir une valeur en Hexadécimale)
- EXP (Exponentielle)
- FRE (Mémoire disponible / pile)
- CHR (Convertir valeur ASCII en caractère ASCII)
- ASC (Convertir des caractères ASCII en valeur ASCII)

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 ...**

Copyright©Microsf01

Programme exemple :

```
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 sa donne /c MAJ >%Phrase%
txt/ En minuscules sa 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 !

Copyright©Microsf01

Chapitre V - Commandes avancées

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

Copyright©Microsf01

INITIALISER UNE INTERFACE

La commande qui le permet est :

```
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 :

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

Pour créer un :

```
ini/ (
ini/
```


Copyright©Microsf01

Puis la commande ***ini;*** doit être utilisé uniquement dans une boucle d' INItialisation (vu ci-dessu)

il permet de remplir des valeurs dans un tableau mémoire du Kernel permettant la création d'une interface

```
ini;nom = "NOM"
ini;texte = "TEXTE
ini;type = "0 ou 1"
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"

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***

Copyright©Microsf01

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 = "1"
   ini;couleur = "087,215,186"
   ini;tx = "300"
   ini;ty = "250"
   ini;px = "MX"
   ini;py = "MY"
   Creer/
ini/ fenetre)
```


Copyright©Microsf01

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 mémé nom , la fenêtre sera remplacé

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

Titre de la fenêtre

```
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

Movable? (Déplaçable)

Ative? (Interaction possible)

Reductable?(Reduire)

Fermable?

Tache?(Affiché dans la barre des tâches)

FP FenetrePrioritaire ?

Couleur généralisé + translucidité ---------- >


```
ini;type = "1;M0A0R0F0T0FP0"
```

La lettre + une valeur booleen (1 ou 0) si un(s) des paramètres sont omit il est activé par défaut

Copyright©Microsf01

```
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 la couleur sera généralisé 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 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.

Copyright©Microsf01

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 = "5"
        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***

Copyright©Microsf01

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 = "Mon Bouton"
```

Permet d'écrire le texte du bouton.

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

(Pas d'effets sur la version Cpcdos OS2.0.5 , Laissez sur **"1"**)

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

Permet de définir une image de fond sur un bouton

numéro entre 0-7

vous avez par défaut dans le dossier « MEDIA/IUG :

ini;img = "1"

Copyright©Microsf01

```
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 ;)

```
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..

Copyright©Microsf01

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 = "Il est bien hein ? :)"
  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/
```

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.**

Copyright©Microsf01

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

Permet de nommer la propriété. (l'objet)

```
ini;texte = "Il est bien hein ? :)"
```

Permet de définir du texte graphiquement dans le label.

```
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'arrières plan « entre les caractères »
la transparence arrière plan. (1 = activé / 0 = opaque)
(assurez-vous que ini;type est égale sois à 0 ou 1 uniquement)

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

Permet de définir si la taille du label est réglé automatiquement par rapport a son contenue **"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.**

```
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..

Copyright©Microsf01

CREER UNE IMAGEBOX

La commande qui le permet est :

```
ini/ imagebox( )
```

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


```
ini/ imagebox(
      ini;nom = "IMG_OEUFS"
      ini;fenetre = "FENETRE_1"
      ini;couleur = "000,000,000"
      ini;couleurf = "001,001,001"
      ini;type = "0"
      ini;image = "os\prog\TomJerry.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 cet image.
Là , il va créer cet image sur **FENETRE_1.**

Copyright©Microsf01

```
ini;nom = "IMG_1"
```

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 définir si la taille du label est réglé automatiquement par rapport a son contenue `"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.`

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

Copyright©Microsf01

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)

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 !

Copyright©Microsf01

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.

Paramètres disponibles :

```
demarrer/ /safe
```

Permet de charger l'OS avec des limitations de certains paramètres si l'OS ne démarre pas correctement..

Aucuns pont ou relais se fait avec la commande **IUG/**

Le Kernel n'exécutera pas **IUG/** si elle n'est pas accompagné de son paramètre « **/safe** »

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 ..

Paramètres disponibles :

```
iug/ /reset
```

Peut être tapé à la console avant ou même après l'exécution de l'OS

Permet tout simplement de recharger le code/fichier qui se trouve dans la partie IUG dans l'index principal. (après le label «:IUG : » dans le fichier INDEX.CPC

```
iug/ /safe
```

Permet d'exécuter l'interface graphique de l'OS avec des limitations si l'OS ne démarre pas correctement ou des bug se produit

Il limite l'affichage à 800x600x16

Commande à exécuter si **demarrer/ /safe** à été lancée.

Copyright©Microsf01

LES EVENEMENTS

Dans cette partie nous allons voir comment créer un événement

Que doit faire le Kernel si l'utilisateur clique sur un 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 ?

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

« ev/ {cible du fichier} »

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\mon_ev.cpc
ini/ bouton)
```

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

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

Copyright©Microsf01

Ajoutez dans la boucle ini de votre bouton, juste après la commande « **creer/** »

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

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

Rappel : Les caractères suivants : |» veulent dire que la ligne continue avec celle de en-dessous (manque de largeur ..) , donc pas de retour à la ligne ou ENTRER (à ne pas écrire)

```
Proc/ BOUTON1(CLIC)
      msgbox/ /texte=Tu m'a cliqué dessu ! /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 cette version de Cpcdos, vous n'avez que CLIC et FOCUS comme événements

Copyright©Microsf01

AFFICHER UN MSGBOX

La commande qui le permet est :

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

Cette commande permet de créer un message graphique

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

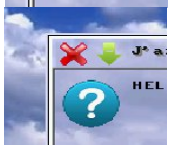
Le paramètre **/titre** indique le titre de la fenêtre, ci dessus c'est « J'ai un message pour toi »

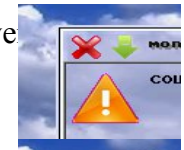
Le paramètre **/mode** permet de choisir entre un message avec 1 choix comme le bouton « OK »

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

Le paramètre **/alerte** définit le niveau d'alerte

0 : message de validation 

1 : Question 

2: Ave 

rreur

Exemple :

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

OS2.0.5 :Cette commande n'est pas tout à fait au point

Les Boutons OK et OUI/NON ne peux pas intervenir sur la continuité de la lecture du code CpcdosC+

Il faut créer l'événement du bouton avec ces nom suivants :(voir partie événements)

« BOK.MSGCONSOLE »:Bouton OK

« BOUI.MSGCONSOLE »:Bouton OUI et « BNON.MSGCONSOLE »:Bouton NON

Copyright©Microsf01

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.

Nb : nom en majuscules !

Exemples :

```
fermer/ MONBOUTON
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 une fenêtre

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

Copyright©Microsf01

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

Mais pour **UTILISER** 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 ! ;-)

Copyright©Microsf01

LES VARIABLES D'ENVIRONEMENTS

Chapitre assez intéressant qui vous permet de savoir les variables utilisé par le Kernel
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 pour les bugs, informations]
  - BOOTOS [Cible du fichier « BOOT » de l'OS]

- 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 pas d'image]
  - SCR_BAS [Résolution d'écran – Base 800x600 & 1024x768]
  - SCR_BIT [Bit de couleur – 8 16 et 32 bits]
  - ECRX [Taille x de l'écran initialisé]
  - ECRY [Taille y de l'écran initialisé]
  - CURPOSX [Position du curseur console X]
  - CURPOSY [Position 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]

Copyright©Microsf01

- Kernel :
  - KRNL_PROC_NB [Nombre de processus en cours]
  - KRNL_PROC_LST [Liste des processus en cours]
  - MEMFREE [Mémoire disponible en octets]
  - 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]
  - SYS_STACK [Mémoire STACK – 4096 ou 8128 par défaut]

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

Par exemple :

Si vous voulez modifier le fond d'écran *en direct*, 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 !

Modifier la résolution d'écran :

```
fix/ SCR_RES = 800x600
```

Voir la liste des résolutions d'écran ci dessous

etc …

Copyright©Microsf01

Rappel :

Pour afficher la liste des variables en mémoire :

```
fix/ /liste
```

Pour afficher une variable en mémoire :

```
txt/ %VARIABLE%
```

Mettez bien sûre autre chose que VARIABLE ,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**, 1152x864, 1280x960, 1280x1024, **1600x1200**, 1920x1200, 1920x1440, 2048x1536

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

Pour tester des résolutions tapez :

```
sys/ /TESTECR 16
```

« 16 » indiquant 16Bits, il y à aussi 8 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 »)

Copyright©Microsf01

CONFIGURER & TESTER LE SYSTEME

La commande qui le permet est :

```
sys/
```

Elle ne peux être exécuté toute seul, il lui faut des paramètres, comme

Tester le CPU :

```
sys/ /TEST
```

Tester le VESA :

```
sys/ /TESTVESA
```

Tester le Kernel :

```
sys/ /KRNLTEST
```

Lister les modes graphiques :

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

Mémoire disponible:

```
sys/ /MEM
```

Tester le CPU :

```
sys/ /TEST
```


Copyright©Microsf01

Chapitre VI – Exemples de programmes :

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)

Copyright©Microsf01

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)

Copyright©Microsf01

REMERCIEMENTS AUX TESTEURS CONTRIBUTEURS ET COMMENTATEURS DE CPCDOS & MANUEL UTILISATEUR

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

LIENS

Site internet principal de Cpcdos : http://microsf01.fr.nf/ ou http://cpcdos.e-monsite.com/

Site d'autres projets Microsf01 : http://microsf01.fr.nf/ ou http://microsf01.e-monsite.com/

« Brouillon » à la page : http://microsf01.e-monsite.com/pages/cpcdos-os2-1.html

Nouveautés : http://cpcdos.e-monsite.com/pages/news.html

Au projet (à faire sur cpcdos) : http://cpcdos.e-monsite.com/pages/au-projet-a-faire.html

Liste des OS basé Cpcdos : http://cpcdos.e-monsite.com/pages/systemes-d-exploitation-base-cpcdos.html

Programmes téléchargeables : http://cpcdos.e-monsite.com/pages/programmes-cpcdosc.html

Chaîne YouTube : https://www.youtube.com/user/sebamstrad

Et page Facebook : https://www.facebook.com/pages/Kernel-Cpcdos-OSx/479523255400921

Microsf01 FAVIER Sébastien 01
Copyright©Microsf01 MAI 2011
sebastien.ordinateur@hotmail.fr
http://microsf01.fr.nf/