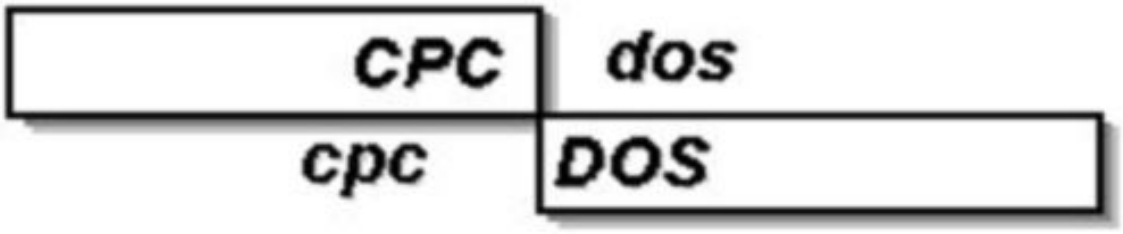


- Crée Pour Concevoir Des OS -

Microsf01 CpcdosC+

Manuel doc.1.2014 Alpha

(Dernière mise à jour – 31/12/2014)

Apprendre la programmation CpcdosC+
pour le Kernel Cpcdos OSx

- Version OS2.0.5 Alpha 3.9 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://cpcdos.fr.nf/ | *http://microsf01.fr.nf*
http://www.facebook.com/pages/Kernel-Cpcdos-OSx/479523255400921
http://forum-cpcdos.fr.nf/
https://www.youtube.com/user/cpcdososx

Avertissement :

Ce logiciel ainsi que le CpcdosC+ sont protégés par la loi relative au droit d'auteur et par les conventions internationales. Toute reproduction ou distribution partielle ou totale de ce logiciel ou son langage sans autorisation, par quelque moyen que ce soit, est strictement interdite. Toute personnes 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

Copyright France

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. Le site web: **http://craftyos.fr.nf/**

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

Cette version de Cpcdos est totalement gratuite

Les versions de CPCDOS composées de « ALPHA » sont des versions non terminées et en tests progressif. Des crashs hasardeux se réduit de plus en plus. Les Crashs dépendent totalement de la machine. Les fonctionnalités sont minimes.

Après 3 années de développement, en réalité si on compare les fonctionnalités (en excluant le graphique) au cœur de CPCDOS avec les fonctionnalités au cœur de Microsoft Windows 98, nous sommes proche de 35%.
Le développement de ce noyau augmente de façon exponentielle, il faudra attendre au moins 4 années pour se trouver au niveau de Windows 98 / NT 2000. En parlant évidement de la navigation internet, décodage MP3, MP4, WAV, AVI, MPEG, SDL, OPENGL… puis le multi-core logique&physique du processeur.

La fonctionnalité finale du projet est de pouvoir exécuter des application graphique type Windows, Linux, Java, etc.

Sommaire

1. Histoire du CpcdosC+ - - - - Chapitre I
2. Projet Cpcdos ? - - - - - Chapitre II
3. Configuration minimale - - - - Chapitre III

+ Derniers tests sur les PC x86

+ Installation DosBox/USB Bootable/Disque dur

4. Commandes de bases - - - - Chapitre IV

- Afficher l'aide des commandes - - - `aide/`
- Effacer l'écran - - - - - `cls/`
- Commentaires - - - - - `rem/`
- Affichage de textes - - - - `txt/`
- Positionner curseur (Console/iug - - - `posx/ & posy/`
  + Couleurs de la console - - - - `couleurf/ couleurp/`
- Variables/Créer un tableau/supprimer - - `fix/`
- Exécuter du code Java *(Natif / non compilé)* - **java/**
- Exécuter un fichier CpcdosC+ - - - `exe/`
  + Exécuteur Multi-Thread
- Arrêter l'exécution d'un fichier CpcdosC+ - `stop/`
  + Arrêter net le Kernel - - - - `stopk/`
- Les conditions - - - - - `si/ sinon/`
- Exécuter un fichier exécutable DOS/WIN32 - `shell/`
  + Exécuter commande MSDOS/FreeDos - `dos/`
- Récupérer les entrées au claviers - - - `touche/`
- Mettre en pause le système ou un processus - `pause/`
- Lister un répertoire - - - - `rep/`
- Prendre un Screenshot (IUG/LC) - - - `src/`
- Exécuter une commande d'entrée - - - `cmd/`
- Interaction Cpcdos - - - - `cpc/`
  1. Arrêter & Redémarrer
  2. Lancer la console graphique
  3. Réduire & Restaurer une ou plusieurs fenêtres

4. Agrandir & Rétrécir une ou plusieurs fenêtres
5. Interaction sur les processus

- Lire & Écrire dans un fichier (+binaire) - - `fichier/`
- Copier un fichier ou répertoire - - - `copier/`
- Supprimer un fichier ou répertoire - - `supprimer/`
- Renommer un fichier - - - - `renommer/`
- Créer un répertoire - - - - `repertoire/`
- Activer/désactiver un service - - - `service/`
- Donner la priorité au système (anti-bloquage) - `doevents/`
- Tester un réseau ou une machine - - - `ping/`
- Télécharger/envoyer un fichier http/ftp - - `telecharger/`
- Créer un dossier de partage - - - `partager/`
- Connecter un lecteur réseau - - - `connecter/`
  + Déconnecter un lecteur réseau - - `deconnecter/`
- **Les fonctionnalités**
  + Fonctions mathématiques et autres
  + Les fonctions graphiques
  + Fonctions CpcdosC+ du noyau
  + Créer son propre format d'exécutable CCP
  & Créer son propre format de fichiers
  + Paramètres d'entêtes des fichiers CpcdosC+

5. Commandes avancées - - - - Chapitre V

- Gestion Multitâche
- Initialiser une interface - - - - `ini/`
  - Transfère de textes entre objets et fichiers - `ini/ /[Objet][F|E..`
  - Créer une Fenêtre - - - - `ini/ fenetre( )`
  - Créer un Bouton - - - - - `ini/ bouton( )`
  - Créer un Label - - - - - `ini/ label( )`
  - Créer une ImageBox - - - - `ini/ imagebox( )`
    + EFFETS GRAPHIQUES
  - Créer un TexteBox - - - - `ini/ textebox( )`
  - Créer un Compteur - - - - `ini/ compteur( )`
  - Créer une barre de progression - - - `ini/ progression( )`

- Créer une interface - - - - `creer/`

+ «Pirater» le SCI / modifier les propriétés

+ Exécution enfant ordonné au processus parent (`INI/ /CCP_THREAD`)

- Démarrer l'initialisation de l'OS - - - `demarrer/`
- Lancer l'interface graphique (l'OS) - - `iug/`
- Les événements - - - - - `ev/`
  + Événements à la volée
- Afficher un MSGBOX/TexteMSGBOX - - `msgbox/`
- Explorer un dossier - - - - `explorer/`
- Fermer une fenêtre ou objet - - - `fermer/`
- Actualiser une ou plusieurs propriétés - - `actualise/`
- Focus sur une fenêtre ou objet & Tester - - `focus/`
- Lancer le mode console - - - - `lc/`
- Lire & éditer le registre - - - - `reg/`
- **Variables d'environnements**
- **Résolutions d'écran**
- **Créer un loading screen**
- **Créer une fenêtre sizable**
- **Menu boot**

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

1. Résumé système (Application externe) - **/quick**
2. Tester le Kernel - - - - **/krnltest**
3. Éteindre l'écran VGA - - - - **/vgaoff**
4. Allumer l'écran VGA - - - - **/vgaon**
5. Fixer l'affichage VGA - - - **/vgaFix**
6. Débloquer l'affichage VGA - - - **/vgaDeFix**
7. Status du Cache CPU - - - - **/cpu_cache**
8. Level du CPU - - - - **/cpu_level**
9. Compatibilité du VESA - - - **/testvesa**
10. Lister modes d'écran&bit - - - **/testecr**
11. Tester la mémoire étendu (XMS) - - **/memtest**
12. Nettoyer la mémoire - - - - **/memclear**
13. Tester le CPU - - - - - **/test**
14. Appel interruptions DOS - - - **/int**
15. Écrire & lire dans la mémoire - - **/poke & /peek**
16. Sortie mémoire buffer - - - **/buffer**
17. Charger un pilote - - - - **/device**
18. Booter sur un serveur ou local - - **/boot**
19. Créer une image boot virtuel RAM - - **/creerimg**
20. Booter sur une image/restaurer - - **/bootimg**

21. Gestionnaire d'extensions fichieres - - **/ext**
22. Ordonner exécution à processus - *voir Exécution ordonné au processus parent*
23. «Beeper» le système - - - - **/beep**
24. Informations lecteur (C/H/S, Syst, mem..) - **/disque_info**
25. Créer/chercher automatiquement un lecteur virtuel **/virtuel**
26.

6. Exemples - - - - - - Chapitre VI
7. Codes d'erreurs et avertissements - - - Chapitre VII
8. Remerciements & liens - - - - - Dernière page

Chapitre I – Projet Cpcdos ?

(Débuté le 15 Juillet 2011)

(Acronyme CPCDOS : Créé Pour Concevoir Des OS)

Le but de ce projet, est de permettre à n'importe qui de créer son propre système exploitation en toute simplicité sans utiliser de langages de programmation complexes comme l'**Assembleur** , le C. Mais seulement le **CpcdosC+** qui se veut **simple d'utilisation** et **en français** se qui permet une **meilleure compréhension du code** par une communauté francophone. Tout cela grâce à un système 32Bit tout prêt, la séquence de démarrage, les pilotes, le réseau et les paramètres déjà prédéfinis, possibilité de créer une interface utilisateur & graphique très personnalisée, avancée et très simple qui peux aller jusqu'à **32Bits de couleurs**, une résolution qui peux aller jusqu'au maximum de votre carte graphique.

Ce système s'utilise qu'avec un seul langage de programmation très simpliste avec des syntaxes, messages et commandes entièrement en **Français**, qui se nomme le **CpcdosC+.**
(Acronyme CPCDOS : Créé Pour Concevoir Des OS)

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

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

Cpcdos est un Co-Kernel Monolithique modulaire Multitâche Coopératif

Source

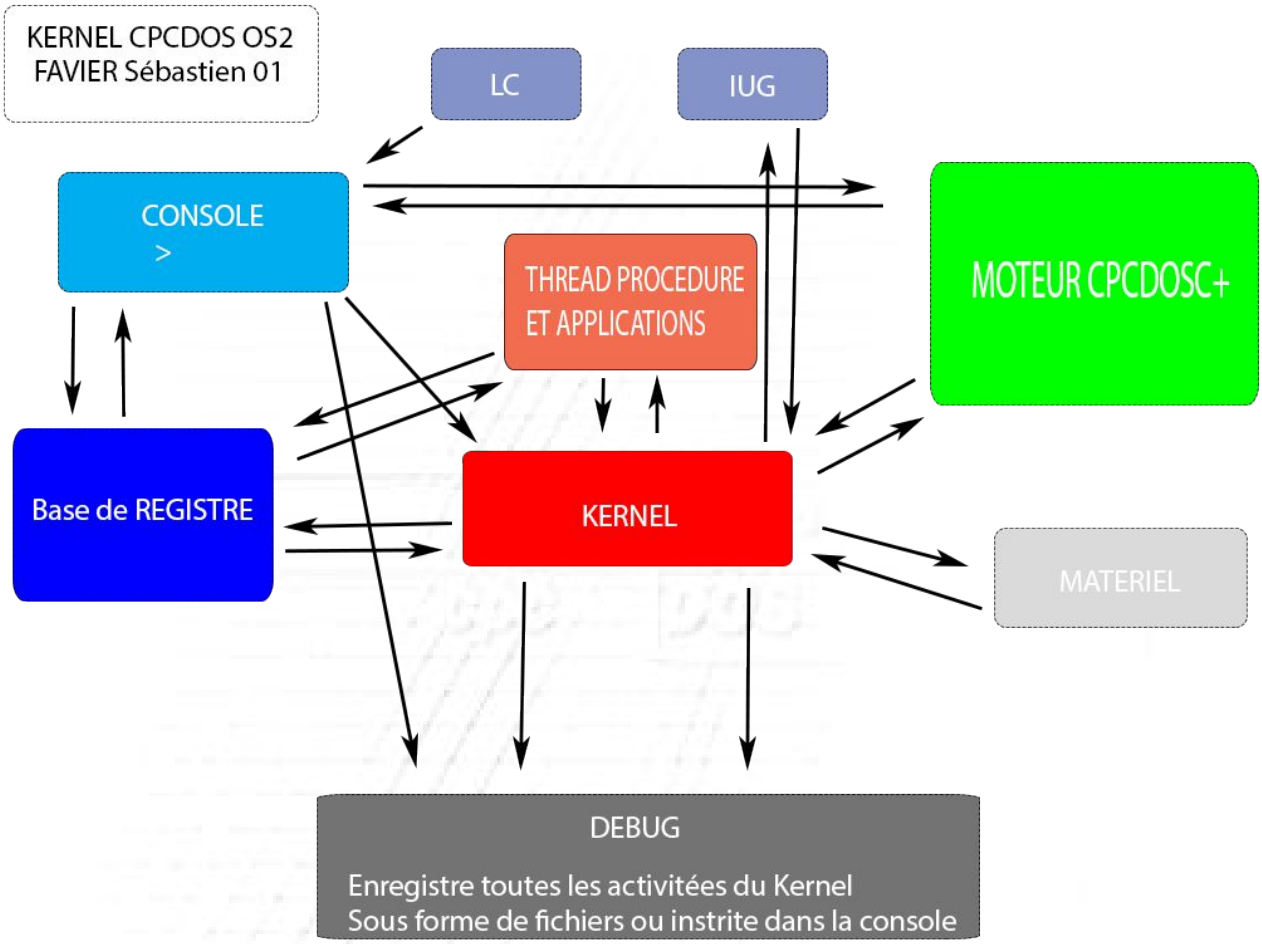
Co-Kernel : C'est un sous-kernel, qui lance l'interface homme-machine principale en se faisant passer pour le noyau principal. Il est exécuté de manière à remplacer le noyau principal ***Main Kernel*** *(Dos)* par celui-ci *(Cpcdos)* tout en rendant possible l'utilisation des api du noyau principal.

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*

MAIS

Ce noyau à une sorte d'un multitâche coopératif/*preemptif* avancé, en effet, les processus sont gérés autrement que ceux des systèmes preemptif actuels. C'est grâce au moteur CpcdosC+, le moteur de traitement de commandes et d'événements et des services qui est multi-tâche et multi-threads. Si un processus Cpcdos plante, ou dans une boucle infinit, le système peut toujours fonctionner! *avec risque d'augmentation de l'utilisation du CPU* mais il est possible de débloquer un processus Cpcdos simplement avec ALT+D.

Schéma du fonctionnement du noyau *(Attention mise à jour → ALPHA 4.0)*



-

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 saisit les commandes CpcdosC+

Base de REGISTRE :

Partie du système , elle fournit et enregistre les informations et paramètres système du Kernel.
Il est aussi liée a la Console car on peut 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 les activités du Kernel !

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 converti 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 préprogrammée 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 noyau d'OS
- CpcdosC+ pour Jeux (Microsf01 Games 32 bit) (Projet actuellement abandonné)

Chapitre III – Exigences du système

Configuration minimum requis pour le PC :

Processeur : 800 *mhz* Intel ou Amd

RAM : 256 Mo 133Mhz

USB **2.0**

Carte graphique : 64 Mo supportant le VGA , SVGA , VESA

Disque dur : au moins un 1 Go

Configuration recommandée :

Processeur : 2 Ghz Intel i5

RAM : 512 Mo

USB **2.0** ou **3.0**

Carte graphique : 128 Mo supportant le VGA , SVGA , VESA
Nvidia Geforce 410M ou +

Disque dur : au moins un 1 Go

Attention aux carte graphiques des serveurs !

ARM : Ne fonctionne pas hors-mis l'émulation

A votre attention

Si Cpcdos ne fait que de « Crasher » sur votre système **ET** que vous avez un HP ou Compaq

ne cherchez pas plus loin, votre système ne supporte pas Cpcdos, ces fabriquants ont bridé leur BIOS pour rendre compatible UNIQUEMENT LEUR Windows pré-installé dessus et non un autre système...

Si ce n'est pas cette marque, alors il se peut que votre Carte graphique ne soit pas supportée par Cpcdos.

Concernant les drivers réseaux, le système utilise de « vieux » drivers génériques, les nouveau PC tout beau, tout récent risquent de ne pas fonctionner. En effet les constructeur ne communiquent plus de drivers NDIS 2.0..

Attention aux Futjisus Siemens & Lenovo, l'installation du réseau peut bloquer.

Derniers tests sur des PC standards x86

Model	Photo	Résultats	Model	Photo	Résultats
Sony Vaio VPCEJ - Intel i5-2450M 2.50Ghz x2 - 6Go RAM - CG :Nvidia GeForce 410M 1Go		Aucun « crash » Parfait 1600x900x32 max. Note : 9/10	Acer x1700 - Intel Pentium - 4Go RAM DDR2 CG :Nvidia GeForce 9300 GE		Crash : Assez rare Fonctionnelle 1600x1200x16 & 1280x1024x32 max. Note : 7/10
ASUSTeK - AMD Athlon II X2 250 3GHz 4Go RAM CG :RADEON X600/X550	PC monté	Aucun «crash» Très Bon 1280x1024x32 max. Note : 8/10	Lenovo B590 Intel i3 2,40Ghz 4Go RAM CG:Intel HD graphics 4000 1,7Go		Aucun « crash » Parfait PB réseau 1366x768x32 Max Note : 8/10
Acer Aspire V3-771G Intel Core i5-3230M 2,6Ghz 4Go RAM NVIDIA GeForce 710M		Aucun « crash » Très bon 1920x1080x32 Max Note : 9,5/10	Asus A52JC i3 370M 2,13Ghz 4Go RAM AMD HD5500 512Mo		Aucun « Crash » détecté, pas très stable problème 16,24 bits couleurs 1024x768x32 Max Note : 5/10
Dell inspiron 17R5721 Intel Core i5 3337U 1,8Ghz 8Go RAM Intel HD4000 & AMD Radeon 8730M 2Go		Aucun « crash » Très bon 1920x1080x32 Max Note : 9/10			

Faîtes part de votre PC et votre note personnelle à l'adresse suivante

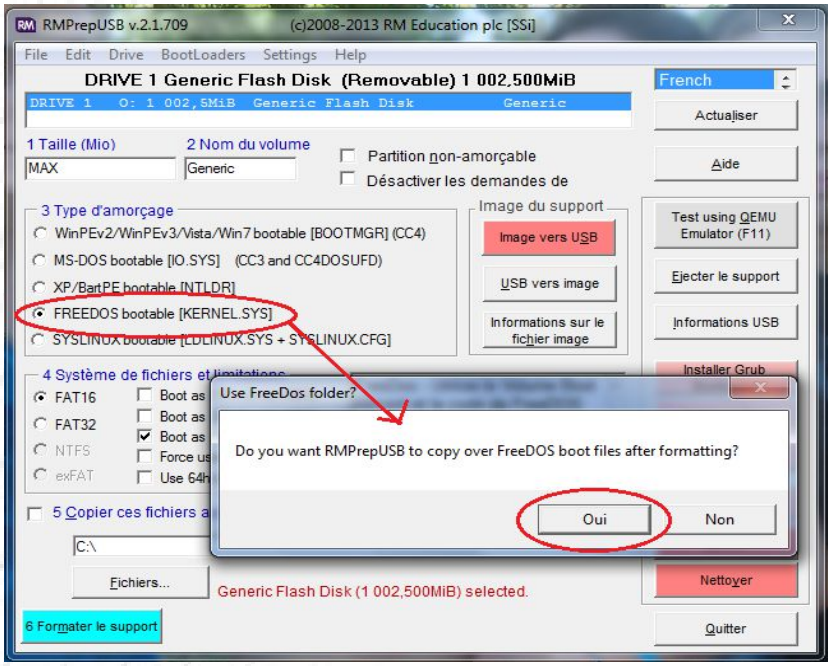
cpcdososx@gmail.com

Installation sur USB :

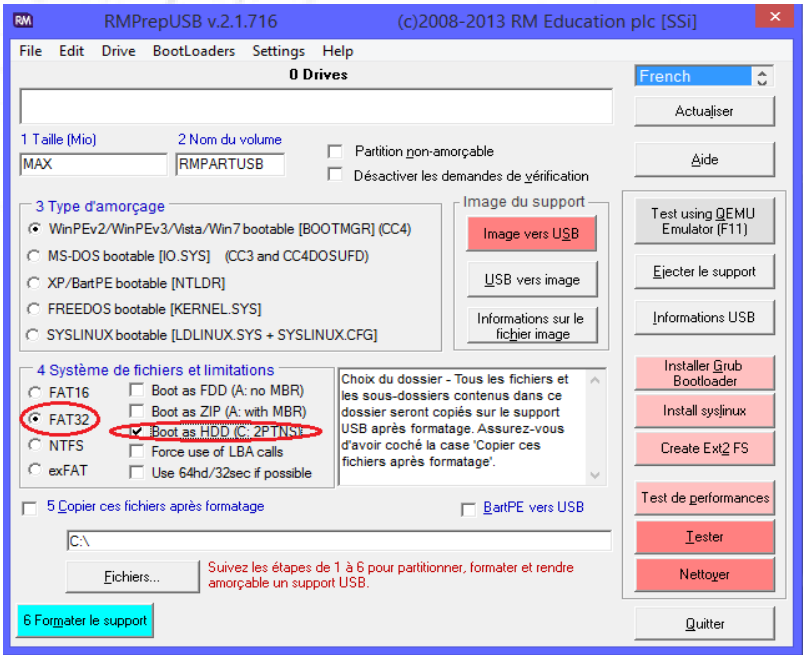
1. Pour commencer, téléchargez le logiciel *RMPrepUSB*
   sur **http://www.rmprepusb.com/documents/release-2-0**
   (en bas de la page)
   Version conseillée «Install_RMPrepUSB_Full..... .zip» (La dernière version)

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

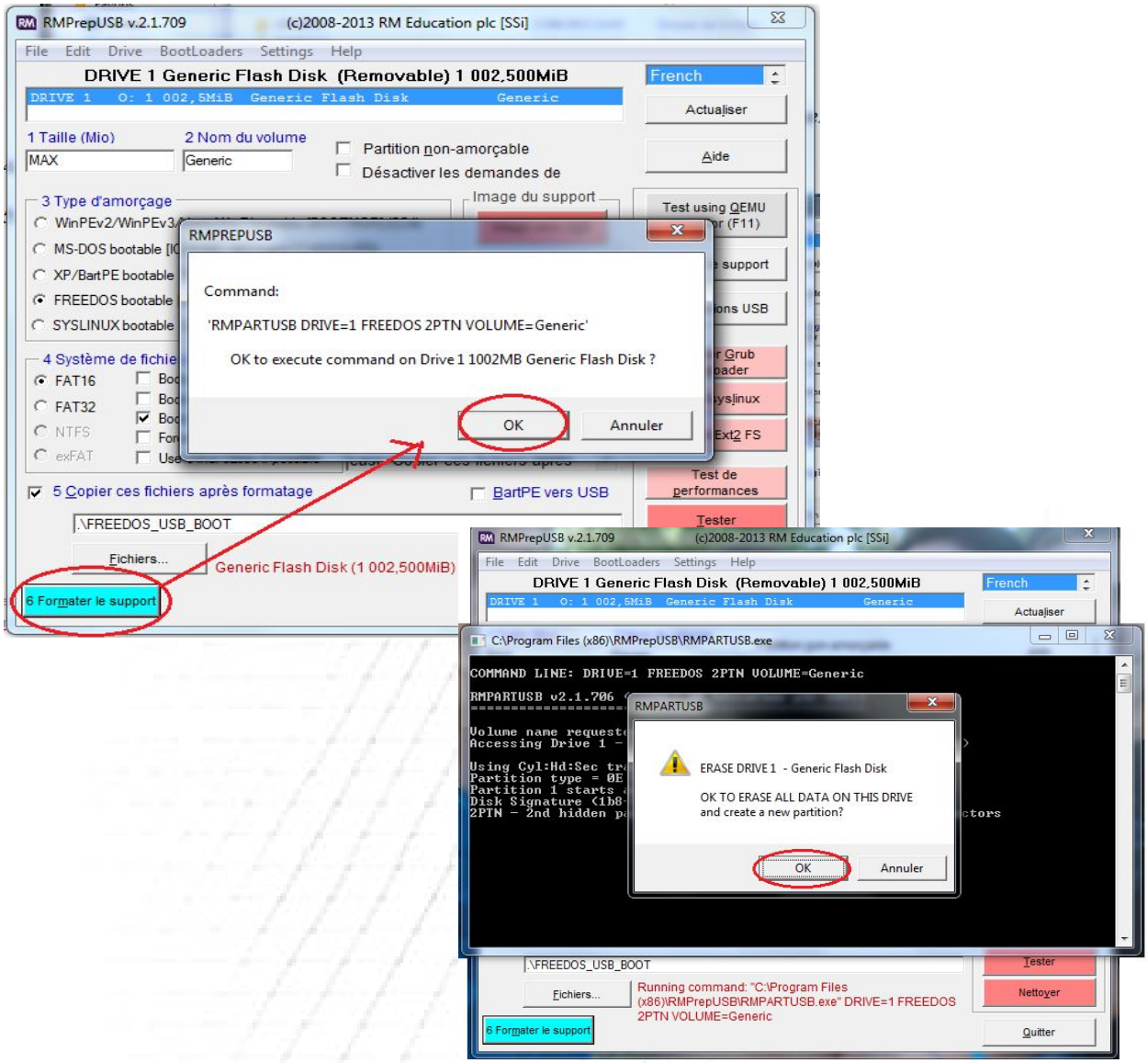
3. Sélectionnez ***FREEDOS Bootable [KERNEL.SYS]***
   Et valider le message par **Oui**



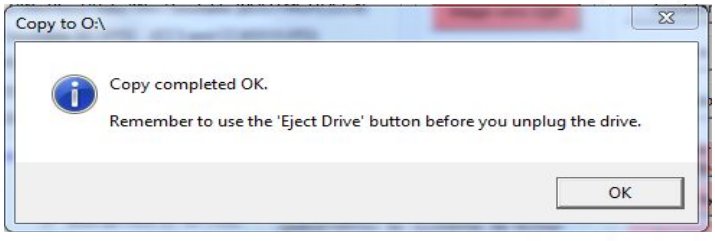
4. Sélectionnez **FAT32** (Si vous avez une clé plus grand que 4Go) et **Boot as HDD**



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

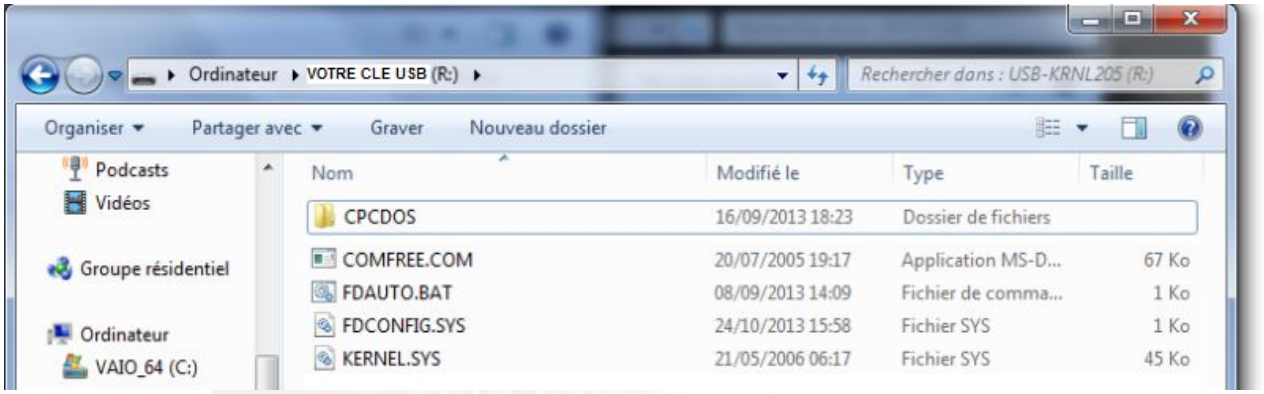


Vous devrez avoir ce message



6. Copiez tout simplement le contenu de votre répertoire CPCDOS

7. Vous devrez avoir ceci :



8. 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 le système démarrer.

Et puis voilà ! :)

Si il vous manque la souris, c'est normal, par défaut le ZIP Cpcdos téléchargé est configuré pour démarrer sur DosBox donc tapez la commande :

```
SYS/ /BOOTIMG 0
```

Pour revenir en mode DOSBOX tapez :

```
SYS/ /BOOTIMG DOSBOX
```

PS : Utilisation de la commande SYS/ voir à partir de la page 87

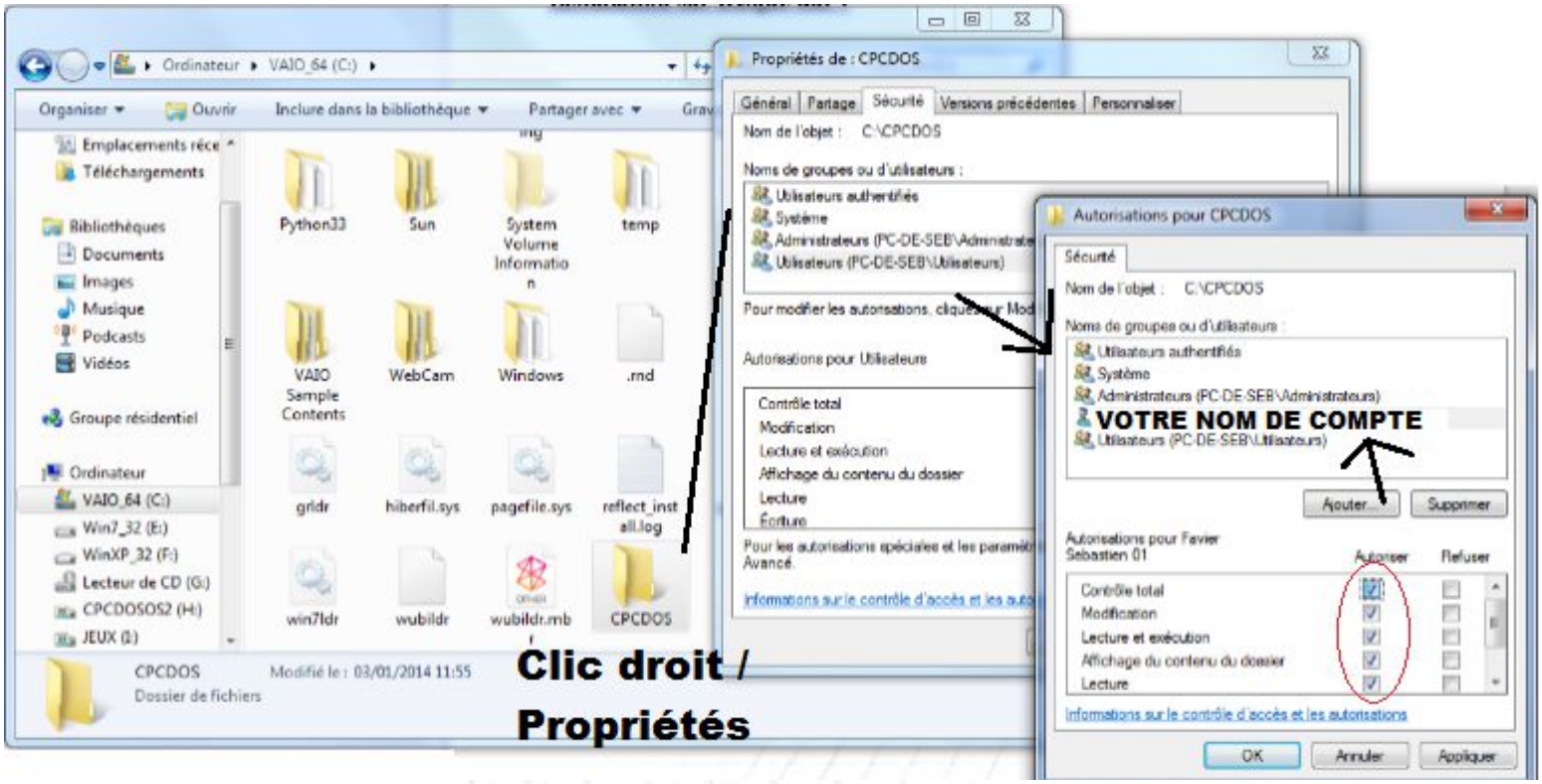
Installation sur Disque dur (IDE/SSD) :

{ Page réservé à l'installation du Kernel sur un disque dur }
Prochainement sur le Doc 2.2015

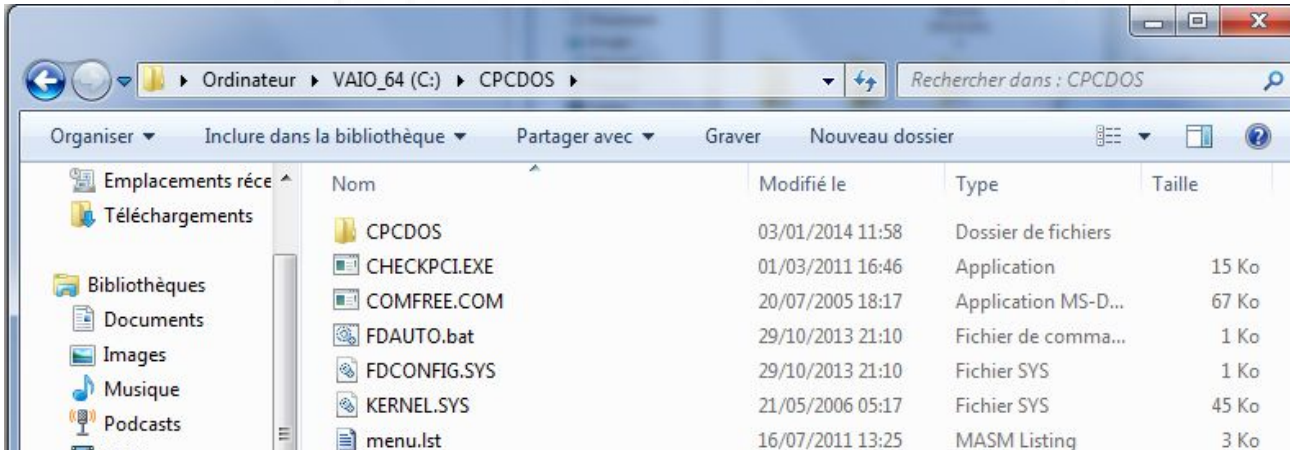
Installation avec DosBox :

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 lecteur)**
Si vous êtes sur un système NTFS (+sécurisé..) donnez lui tous le droits à ce dossier

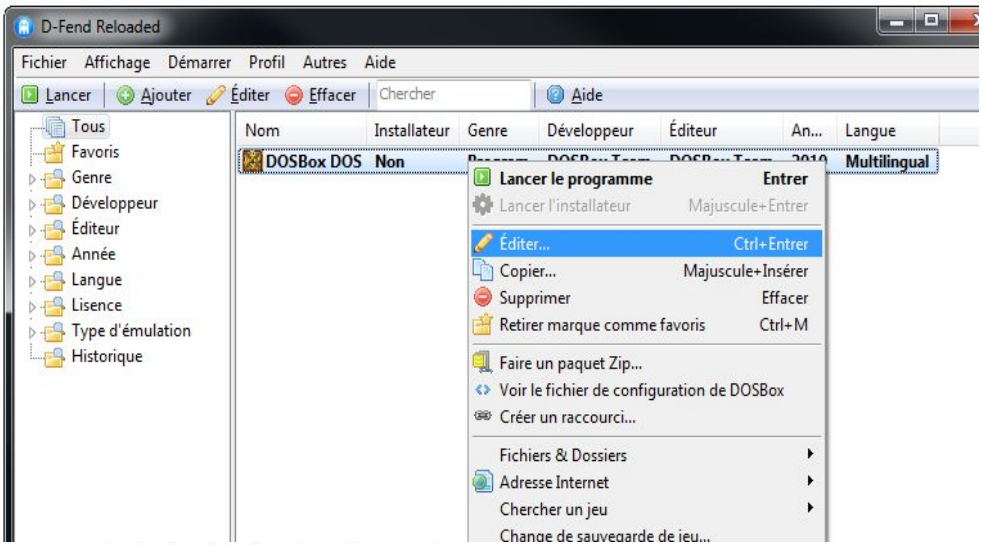


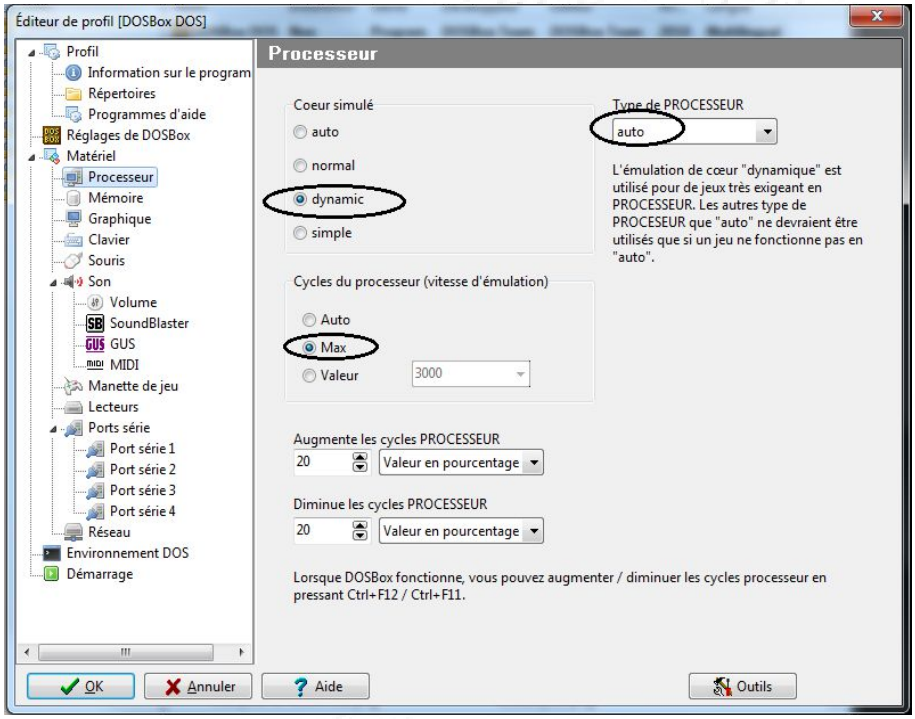
et copiez le contenu du dossier « RACINE A PLACER » dedans.
Vous devrez avoir ceci :



Puis lancez D-Fend

Clique droit sur DosBox, Éditer

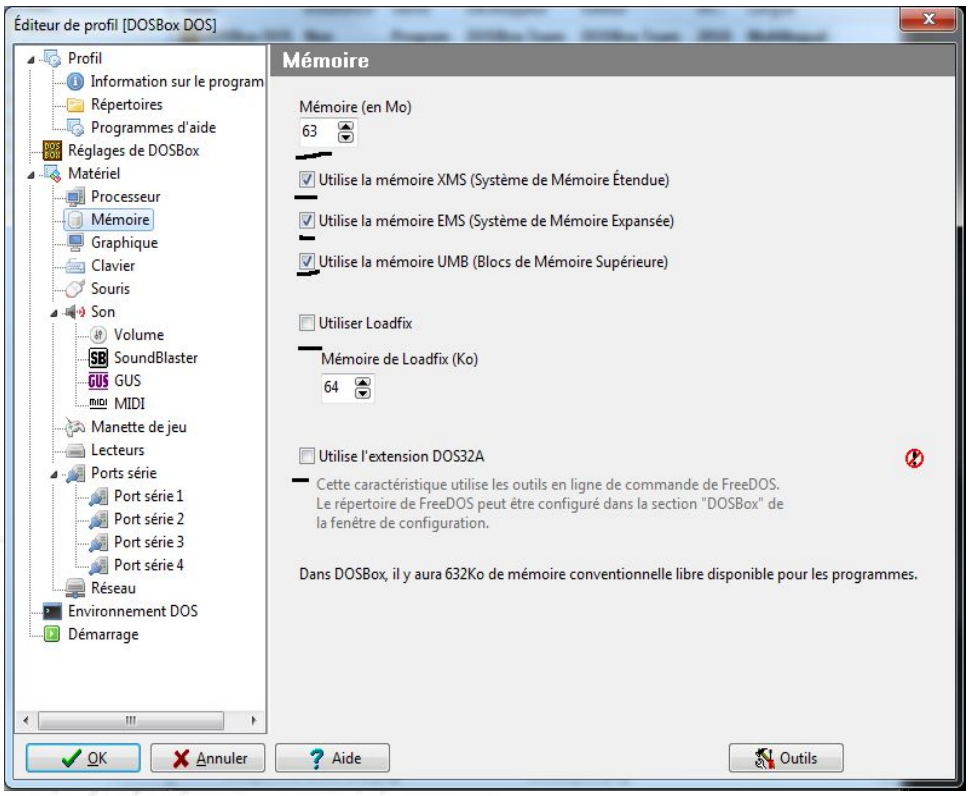




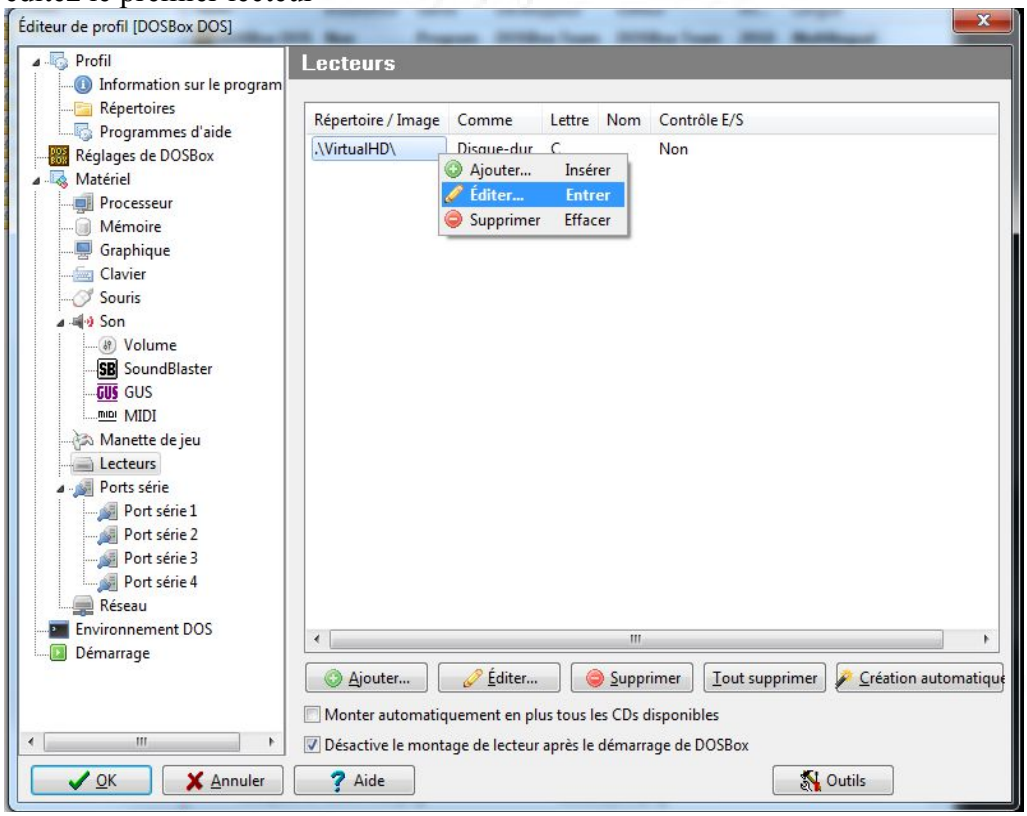
Choisissez et cochez

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

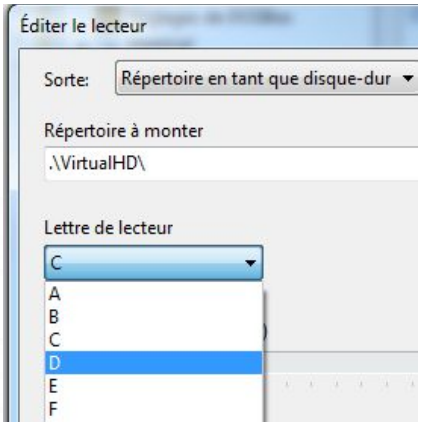
Choisissez la mémoire à 63 mo (maximum) puis cochez toutes les mémoire
XMS
EMS
et UMB
décochez LoadFix
et l'extension DOS32A



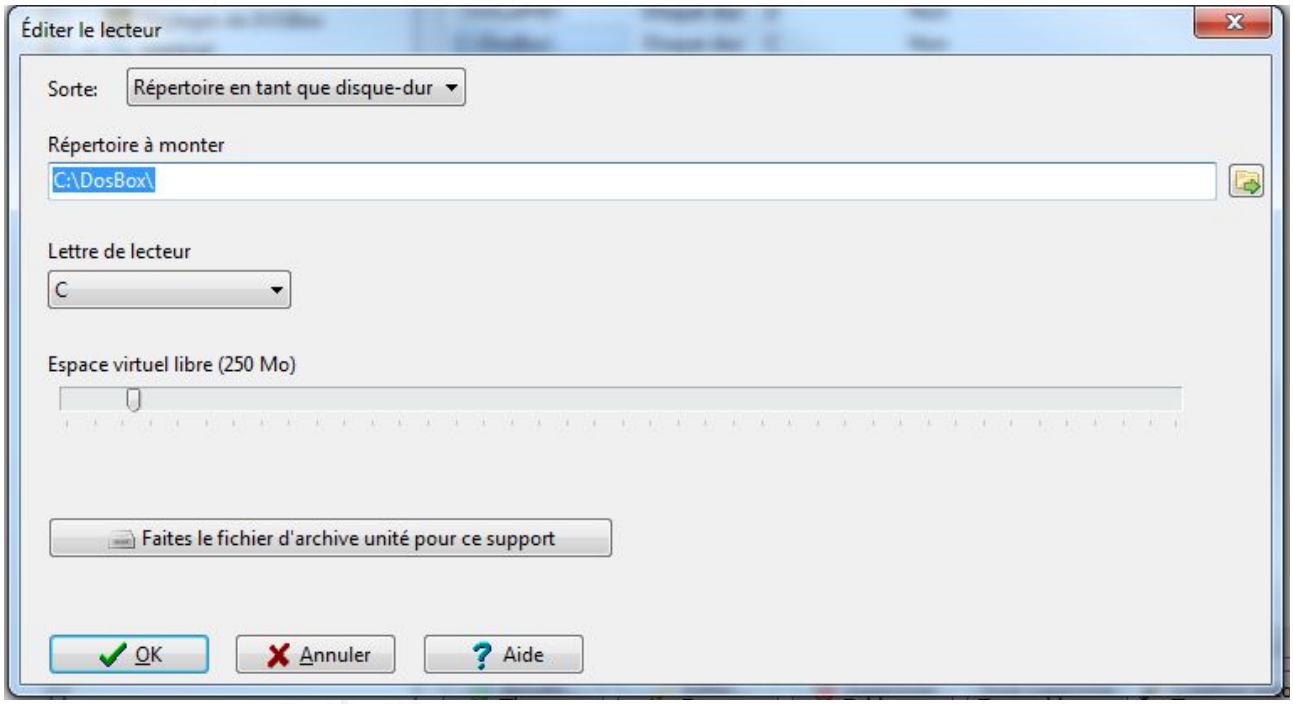
Puis dans le menu Lecteurs, éditez le premier lecteur



Choisissez tant que lecteur D
puis valider



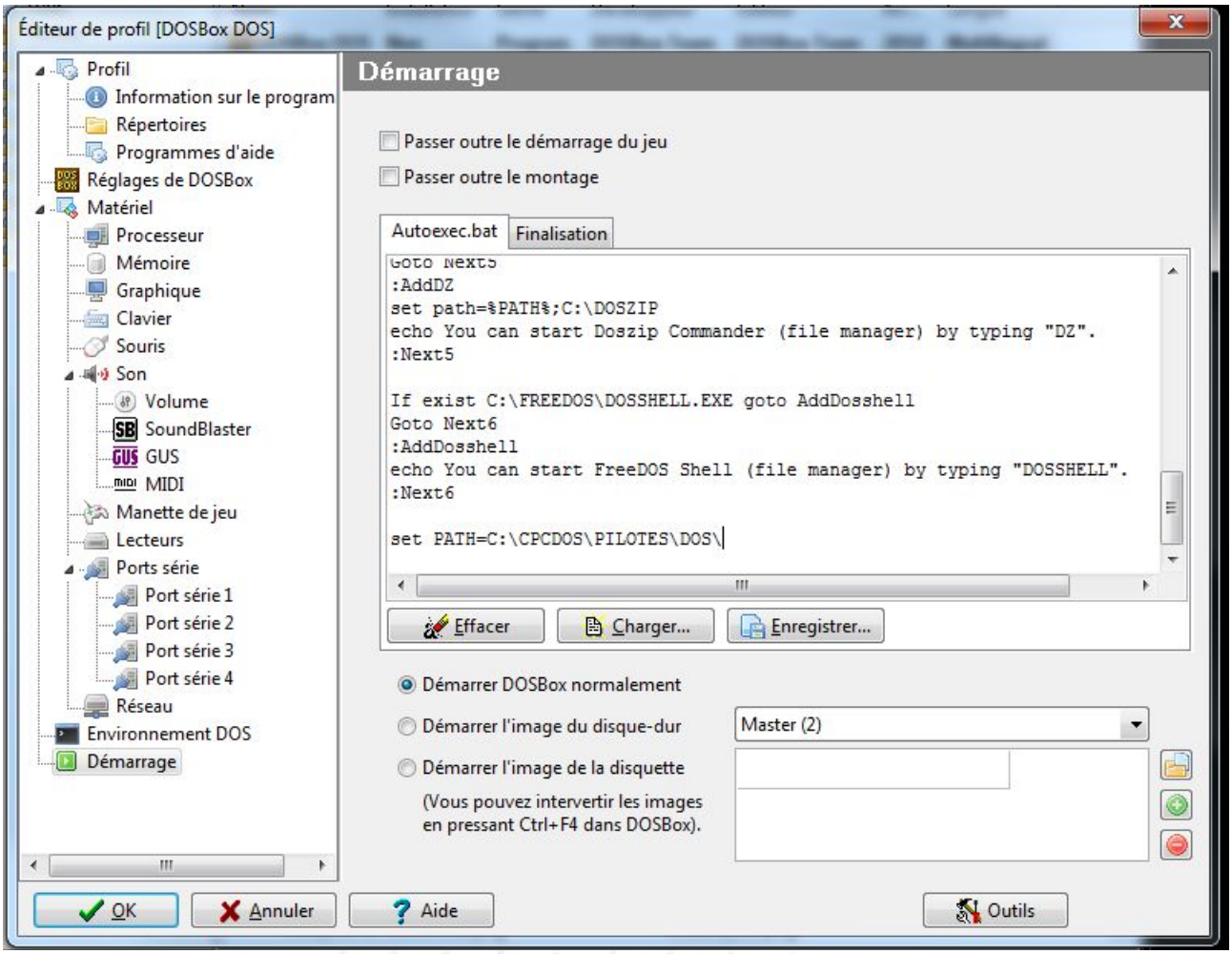
Puis Cliquez sur Ajouter.
Créez un lecteur dont « répertoire tant que disque dur »
Ciblez le dossier que vous avez précédemment crée « DOSBOX »
et déclarez le comme un lecteur **C** puis validez



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

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

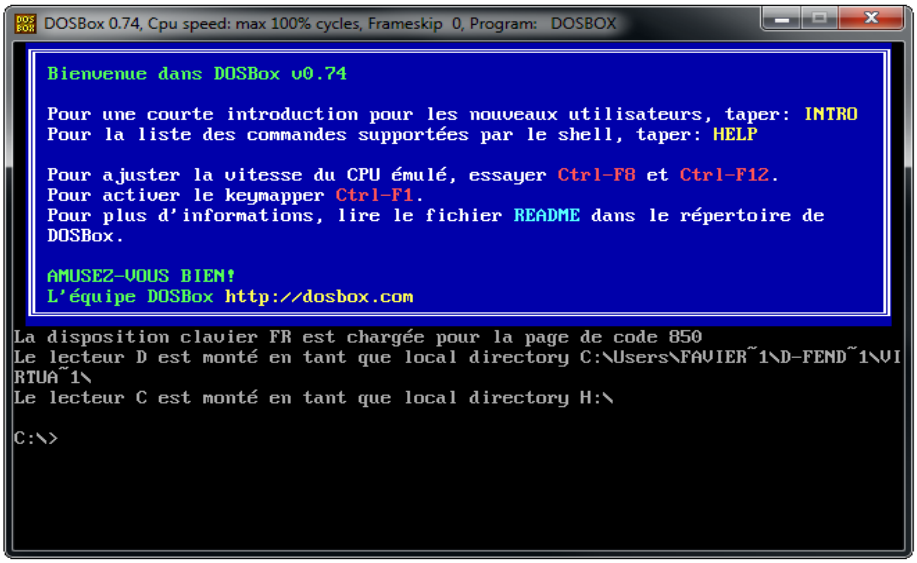
Afin de permettre d'utiliser les commandes et programmes DOS où que vous soyez



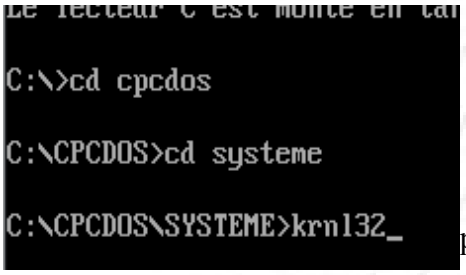
Puis validez le tout

Lancez DosBox (double clic)

Vous devrez avoir ceci :

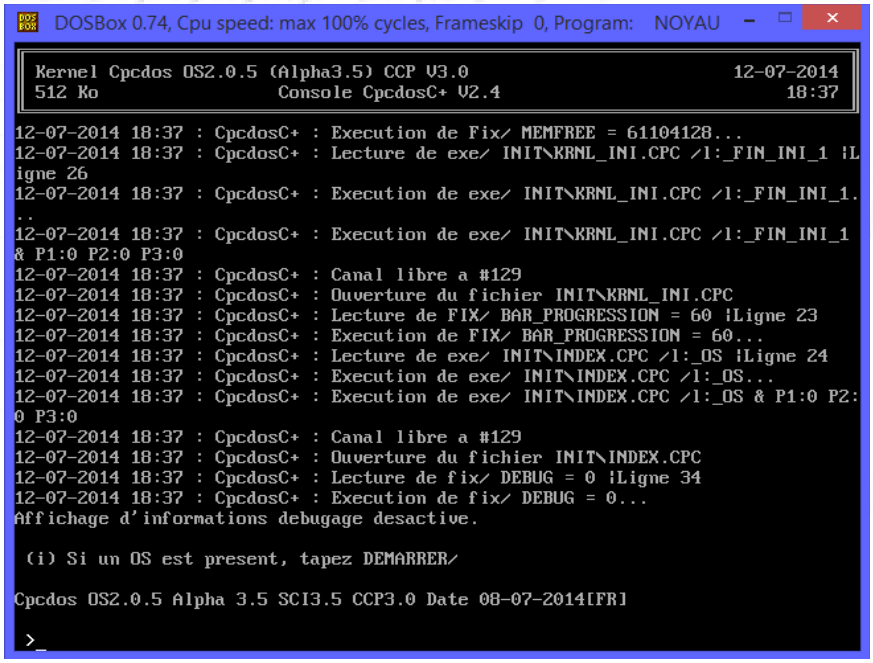


Puis tapez ces commandes



(ces commandes peuvent être ajoutées dans l'interface de paramétrage du démarrage de DosBox!)

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

Maintenant APPRENNEZ A VOUS SERVIR DU LANGAGE DE PROGRAMMATION

LE CPCDOSC+

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 lors de l'exécution des commandes*
- *Ajoutez «* **fix/ LOG = 2** *» pour permettre l'enregistrement des informations de débogage dans SYSTEME\DEBUG.LOG (ralenti un peut 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 type DOS*
- *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/IUG)

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
```

Pour positionner le curseur graphique de votre interface graphique il suffit d'utiliser ces paramètres

La commande qui permet ceci est :

```
posx/ /IUG {valeur pixel}
```

et

```
posy/ /IUG {valeur pixel}
```

Pour retourner la position en cours il suffit de pas ajouter de valeur après /IUG

Pour savoir si l'utilisateur clique sur le bouton gauche, droit ou les deux

```
posx/ /CLIC
```

Pour récupérer la valeur observez la page 89 du manuellement

Petit exemple pour récupérer les valeurs x, y et si l'utilisateur clique sur un bouton

```
@#VALEUR_X posx/ /IUG
@#VALEUR_Y posy/ /IUG
@#CLIC posy/ /CLIC    ou  @#CLIC posx/ /CLIC
MSGBOX/ Position X: %VALEUR_X% Y: %VALEUR_X% Clic: %CLIC%
```

POSY/ /CLIC retourne

1 : Clic gauche

2 : Clic droit

3 : Clic gauche et droit

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 supprimer un tableau (ex : MON_TABLEAU) de la mémoire :

```
fix/ /s MON_TABLEAU()
```

(s : Supprimer)

Pour lister les variables de la mémoire :

```
fix/ /liste
```

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

Pour connaître la taille de votre tableau :

```
fix/ /taille MON_TABLEAU()
```

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)%
```

Créer automatiquement un tableau (ex : de 1 a 10) :

```
fix/ MON_TABLEAU(1 a 10)
```

EXECUTER DU CODE JAVA

La commande qui le permet est :

```
Java/ {fichier}
```

Cette commande permet d'exécuter du code Java natif compatible Windows et autres ;-)
Les fonctionnalités sont très limitées pour les premières versions de Cpcdos

Pour exécuter du code CpcdosC+ en interne, il suffit par exemple de faire ceci :

```
//CCP fix/ variable = Jean-HUD
//CCP Msgbox/ Hello %nom%
```

Pour utiliser les variables de Cpcdos DANS votre code Java, il suffit juste d'enlever les %% exemple avec la variable *%OS%* :

```
string Ma_Variable = "Le nom de mon OS est " + OS
```

Actuellement vous pouvez :

- Créer & modifier des variables et des tableaux (String, int)
- Faire des calculs (Simples)
- Récupérer les entrées au clavier (type input)
- Graphiquement :
    - Créer des fenêtres
    - Créer des objets graphiques
    - Créer des événements (Clic, double clics, cycles, rezisement, fermer...)

Les tutoriels de la programmation Java ne se trouvent pas ici, voici des liens :

http://java.developpez.com/cours/

http://openclassrooms.com/courses/apprenez-a-programmer-en-java

Vous avez des exemples exécutables dans le dossier CPCDOS\SYSTEME\OS\PROG\JAVA

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 **Multi-**Threads

Il existe un paramètre permettant d'exécuter un autre fichier tout en laissant en pause le premier et puis reprendre son exécution à la fin de ce dernier. 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. Qui peut être utilisé en 1 seule ligne ou bien étendu.

Exemples :

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

```
txt/ Continuer tapez 1
txt/ Quitter tapez 2
FIX/ /Q VARIABLE

Rem/ Si l'utilisateur choisis autre chose que 1 alors il saute la condition
Rem/ Si l'utilisateur choisis 1 alors
SI/ %VARIABLE% = 1 (:
      Txt/ Vous avez choisis de continuer
      FIX/ CALCUL = /C %RND% * 5
      ALLER/ CONTINUER
```

```
FIN/ SI
Txt/ Aurevoir!
STOP/

:CONTINUER:
si/ %CALCUL% > 2 (:
      txt/ Valeur plus grande
sinon/
      txt/ Valeur plus petite
fin/ si
```

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 « < »

Cette version de Cpcdos ne vous permet pas encore de sur-conditionner une condition.

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

Pour mettre en pause un processus, ou plus ou moins, l'exécution du code en attendant la fermeture d'un autre processus

Par exemple, on lance un messagebox puis on bloque le code et attendre la fermeture du messagebox définit. Tout en rendant le reste du système utilisable.

Le nom du processus du messagebox est « MA_FENETRE ».

```
MSGBOX/ /TEXTE=Mon processus /TITRE=Test /MODE=1 /ALERTE = 0 /NOM=MA_FENETRE
pause/ /FERMER:MA_FENETRE
MSGBOX/ /TEXTE=Reprise de l'execution  /TITRE=Test2 /MODE=1 /ALERTE = 0
```

Vous pouvez bien-sure faire la même chose avec une fenêtre crée via INI/ FENETRE( ) ;-)

LISTER UN REPERTOIRE

La commande qui le permet est :

```
rep/
```

Cette commande permet de lister un répertoire

Exemple :

Lister le répertoire courant

```
rep/
```

Lister le répertoire os\media

```
rep/ os\media
```

Lister en mettant en pause toutes les 16 lignes dans le dossier OS\PROG

```
rep/ /PAUSE OS\PROG
```

Placer fichiers, dossier, taille et attributs dans un tableau

```
rep/ {Repertoire} /RED {variable}
```

En sortie vous aurez MA_VARIABLE en tableau qui contiendra

Par exemple :

- Si c'est un fichier nommé « Texte.txt »
  MA_VARIABLE(0) contiendra F;N=TEXTE.TXT;T=512;A=A

  F = Type Fichier
  N = Nom du Fichier
  T = Taille du Fichier
  A = Attribut (A:Archive R:Lecture Seule S:Système H:Caché)

- Si le prochain c'est un dossier nommé « Perso »
  MA_VARIABLE(1) = D;N=PERSO

  D = Type Dossier
  N = Nom Dossier

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/ /AGRANDIR {/TOUT | nom de fenetre}
```

Pour agrandir une fenêtre
le paramètre /TOUT permet d'agrandir toutes les fenêtres ayant pas le paramètre R0
Vous pouvez aussi préciser 1 seule fenêtre, pour l'agrandir.

```
cpc/ /RETRECIR {/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.

```
cpc/ /TIMERS
```

Liste les Timers présent sur le système

```
cpc/ /PROCESSUS
```

Liste tous les processus en cours d'exécution

```
cpc/ /PROCESSUS /FERMER {Nom processus}
```

Ferme le processus via le nom

```
cpc/ /PROCESSUS /FERMER /PID {Numero PID}
```

Ferme le processus via le numéro **P**rocessus **Id**entification

```
cpc/ /PROCESSUS /BLOQUER {Nom processus}
```

Bloque le processus via le nom

```
cpc/ /PROCESSUS /BLOQUER /PID {Numero PID}
```

Débloque le processus via le numéro **P**rocessus **Id**entification


```
cpc/ /PROCESSUS /BLOQUER {Nom processus}
```

Bloque le processus via le nom

```
cpc/ /PROCESSUS /BLOQUER /PID {Numero PID}
```

Débloque le processus via le numéro **P**rocessus **Id**entification


```
cpc/ /PROCESSUS /DEBLOQUER {Nom processus}
```

Bloque le processus via le nom

```
cpc/ /PROCESSUS /DEBLOQUER /PID {Numero PID}
```

Débloque le processus via le numéro **P**rocessus **ID**entification

```
cpc/ /PROCESSUS /KERNEL
```

Relance le processus KERNEL (en cas de problèmes)

```
cpc/ /PROCESSUS /SCI
```

Relance le processus Service Création Initialisation (IUG) (en cas de problèmes)

```
cpc/ /PROCESSUS /CCP
```

Relance le processus CpcdosC+

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/ /SORTIR #{N°canal};{Fichier}
```

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

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

Ouvrir un canal pour écriture binaire dans un fichier

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

Ouvrir un canal pour lire dans un fichier

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

Ouvrir un canal pour lecture binaire dans un fichier

```
fichier/ /ENTRERB #{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%
```

COPIER UN FICHIER OU UN REPERTOIRE

La commande qui le permet est :

```
copier/
```

Si vous êtes en mode LC, vous aurez par défaut une copie avec une progression en octets et en %
Si vous êtes en mode IUG, vous aurez par défaut une copie avec une fenêtre + barre de progression.
(Uniquement si vous ne copiez que des fichiers pour cette version de Cpcdos)

Dans cette version vous pourrez faire autre chose pendant la copie, mais si vous exécutez une autre copie, le noyau suspendra la copie en cours jusqu'à la fin de la dernière.

Le paramètre */RAPIDE* Permet de suspendre le système et donner priorité à 100 % de la copie du fichier, ce qui améliore considérablement la vitesse de copie !

Exemples :

Pour copier un fichier

```
copier/ fichier.JPG OS\MEDIA\FOND\fichier.cpc
```

(En destination, vous pouvez spécifier un autre nom de fichier **ou simplement le dossier**)

Attention cette commande remplace le fichier sans confirmation

Pour copier un fichier en arrière plan

```
copier/ /CACHE fichier.JPG OS\MEDIA\FOND\fichier.cpc
```

Ceci permet une copie totalement discrète.

Pour copier un répertoire (+ son contenu)

```
copier/ /REPERTOIRE OS\MEDIA C:\DOSSIER
```

(En destination, vous pouvez spécifier un autre nom de fichier)

Attention cette commande remplace les fichiers de même nom

SUPPRIMER UN FICHIER OU UN REPERTOIRE

La commande qui le permet est :

```
supprimer/
```

Attention supprime sans confirmation

Exemples :

Pour supprimer un fichier

```
supprimer/ os\fichier.cpc
```

Pour supprimer un répertoire

```
supprimer/ /repertoire c:\dossier
```

RENOMMER UN FICHIER

La commande qui le permet est :

```
renommer/
```

Exemple :

La commande qui le permet est :

```
renommer/ Fichier.cpc program.tmp
```

CREER UN REPERTOIRE

La commande qui le permet est :

```
repertoire/
```

Exemple :

La commande qui le permet est :

```
repertoire/ os\prog2
```

Ne crée pas l'arbre, mais uniquement 1 dossier

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*

Vous avez à disposition un fichier *\KRNL\SERVICES\Exemple.cpc* qui vous donne un exemple de la structure d'un fichier de service

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:
```

Bref regarde comment un service est foutu, si t'a un blèm va sur le forum !

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 système, services et timers du noyau afin d'éviter les «bloquages» lors ce que le code effectue par exemple une boucle un peux trop longue … **ce qui permet de gagner en fluidité pour votre OS !!**

Enfin il est pas très utile si vous avez la variable %MULTI_TACHE% est déjà définit
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
Vous devrez spécifier le serveur DNS afin de résoudre le nom

Exemple sans DNS sur google.fr:

```
ping/ 74.125.24.94
```

Exemple avec DNS sur google.fr:

```
ping/ www.google.fr
```

TELECHARGER/ENVOYER UN FICHIER HTTP/FTP

(Version cpcdos basée sur CURL) → En développement pour le rendre indépendant !! ;-)
Aucun message d'erreur n'a été « fait » pour le moment, si vous avez des doutes après l'exécution de votre commande, consultez le fichier *CPCDOS\SYSTEME\TEMP\DOWN.LOG (sortie de CURL)*

La commande qui le permet est :

```
telecharger/ {http://[adresse serveur]{/fichier}}/{ftp://[adresse....}
```

Cette commande permet de télécharger un fichier trouvant sur le WEB en http.
Étant *basé pour le moment sur CURL, ce module n'est pas multitâche et se base sur le shell du DOS, donc pendant le téléchargement, le noyau est suspendu..*

Exemple téléchargement HTTP :

```
Telecharger/ http://www.google.fr/
```

ou

```
Telecharger/ http://74.125.24.94/
```

ou

```
Telecharger/ http://74.125.24.94/index.html
```

Télécharger une image jpg

```
Telecharger/ http://www.joomlaworks.net/images/demos/galleries/abstract/7.jpg
```

Exemple téléchargement FTP:

Sans compte utilisateur

```
Telecharger/ ftp://monserveurFTP//monfichier.cpc
```

ou

```
Telecharger/ ftp://74.125.24.94//monfichier.cpc
```

ou

```
Telecharger/ ftp://74.125.24.94/unDossier/maSauvegarde.zip
```

Avec compte utilisateur

```
Telecharger/ -u MonCompte:MotDePasse ftp://monserveurFTP//monfichier.cpc
```

ou

```
Telecharger/ -u MonCompte:MotDePasse ftp://74.125.24.94//monfichier.cpc
```

ou

```
Telecharger/ -u MonCompte:MotDePasse ftp://74.125.24.94/unDossier/Save.zip
```

Exemple d'envoi FTP :

Sans compte utilisateur

```
Telecharger/ -T MonFichier.cpc ftp://monserveurFTP//monfichier.cpc
```

ou

```
Telecharger/ -T MonFichier.cpc ftp://192.168.1.2//monfichier.cpc
```

ou

```
Telecharger/ -T maSauve.zip ftp://192.168.1.2/unDossier/maSauve.zip
```

Avec compte utilisateur

```
Telecharger/ -u MonCompte:MotDePasse -T maSauve.zip ftp://monFTP//prog.cpc
```

ou

```
Telecharger/ -u MonCompte:MotDePasse -T maSauve.zip ftp://74.125.24.94//prog.cpc
```

ou

```
Telecharger/ -u MonCompte:MotDePasse -T Save.zip ftp://74.125.24.94/dossier/Save.zip
```

Télécharger une image, texte, ZIP, CPC... Fonctionne pour tous les formats (en correction)
Tous vos fichiers téléchargés sont enregistré dans le dossier *TEMP/*

Si le format est inconnu il sera enregistré tant que TELECH.xxx → En développement
Attention au JPG « web » qui a tendance à pas bien fonctionner..,

CREER UN DOSSIER DE PARTAGE

La commande qui le permet est :

```
partager/
```

Utilisant le protocole SMB cette commande permet de diffuser un dossier sur le réseau

Exemple :

```
partager/ MONPARTAGE=C:\CPCDOS\SYSTEME\OS
```

Ceci partage le dossier OS sur le réseau en lecture seule pour tous les utilisateurs

```
partager/ /LECTURE:abc123 MONPARTAGE=C:\CPCDOS\SYSTEME\OS
```

Ceci place un mot de passe pour la lecture

```
partager/ /TOTAL: MONPARTAGE=C:\CPCDOS\SYSTEME\OS
```

Ceci donne droit à tous le monde l'accès lecture et écriture

```
partager/ /TOTAL:abc123 MONPARTAGE=C:\CPCDOS\SYSTEME\OS
```

Ceci place un mot de passe pour l'accès en écriture

```
partager/ /LECTURE:abc123 /TOTAL:azerty MONPARTAGE=C:\CPCDOS\SYSTEME\OS
```

Ceci place un mot de passe différent pour la lecture ET écriture

```
partager/ /TEXTE:"Partage pour Jean" MONPARTAGE=C:\CPCDOS\SYSTEME\OS
```

Ceci place une description du dossier de partage

```
partager/ /SUPPRIMER:MONPARTAGE
```

Ceci supprime la diffusion du dossier de partage sur le réseau

Les performance d'héberger le dossier de partage sur la machine fonctionnement sur Cpcdos ne sont pas top top ...

Cette version de Cpcdos ne contrôle pas encore les domaines, les mot de passe ne risqueront que de bloquer l'accès au dossier partagé

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

Il se peut que vous ne puissiez pas accéder au contenu des fichiers distant.. Cela est causé par FreeDos

LES FONCTIONNALITES

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, .JPG, ou .GIF)
- DIMY (Dimension Y d'un fichier .BMP, .JPG, ou .GIF)
- FEX (Test si un fichier existe ou pas)
- LENF (Taille d'un fichier en octets. Convertir en Ko : *résultat* *1024)
- CTN (Afficher le contenu d'un fichier)

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 termine 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 serait si vous utilisez dosbox Windows prend le relais)

Pour réduire toute les fenêtres, pressez ALT + B ***(Fonction non finalisée bugs)***

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

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

Pour afficher l'explorateur de fichiers **ALT + E**

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

Pour déplacer une fenêtre presser **CLIC GAUCHE** 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 afficher l'icône du statut réseau sur un imagebox ajoutez à la fin du nom « **STATUT_RESEAU*** »

Pour afficher le debug (les opérations arrière plan) sur un label ajoutez à la fin du nom « **DEBUG*** »

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, GIF 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*
CP_F01		
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	Création + Événement du textebox de l'explorateur	Utilisable qu'à partir d'un événement Variable à communiquer : EXP_CIBLE
EXP_F04	Création + événement du bouton Lecteur	Variable à communiquer : EXP_FENETRE
EXP_F05	Création + événements de la barre d'outils (Créer, renommer, supprimer dossier/fichier)	Variable à communiquer : EXP_FENETRE
NUL_0	DOS.CPC Permet d'exécuter vos application DOS ou Win32 en mode console	
RED_F01	Fonction de réduction et restauration des	Compléter les variables :

	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 (Si il s'agit d'une création « 1 » ou suppression « 2 »)	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 : *EXE/ KRNL\EXP_F01\SCROL.CPC*
SERVICES	Contient la liste des services. Actuellement : - VEILLE : Veille/extinction du moniteur - RESEAU : Détection statut réseau	En passant par le fichier source, Pour activer un service EXE/ KRNL\SERVICE*.CPC /L:ACTIVER Pour désactiver un service EXE/ KRNL\SERVICE*.CPC /L:DEACTIVER Ou sinon il y a plus simple que de se faire chier à écrire la cible, va chercher bonheur avec la commande SERVICE/ La plus pars des services ont besoin de variables, ouvrez le fichier .cpc et lisez le début, il indique les « Variables à communiquer »
SRV_SVC.CPC	Serveur d'hébergement de services local Ce programme permet d'héberger jusqu'à 512 services en 1 seul processus qui reste insivible Très utile si vous creez un service pour qu'il reste constamment en marche en arrière plan sans fermer par erreur une fenêtre qui hébergeait ce service	Démarrage manuel EXE/ KRNL\SRV_SVC.CPC ou ajouter au démarrage de votre OS EXE/ & KRNL\SRV_SVC.CPC
IPCONFIG	INIT.CPC Permet de convertir/initialiser les adresse réseau ayant un espace entre chaque octets par des points «.» et générer un fichier WATTCP. IPCONFIG.CPC est généré automatiquement au démarrage vous ne pourrez pas changer l'adresse ip ici	Simple démarrage : EXE/ KRNL\IPCONFIG\REFORME.CPC

LC_IUG	Fonction ou plutôt nommé « Routine LC » permet à vos programmes en ligne de commandes d'être exécuté automatiquement en mode LC et retour automatique en mode IUG si le vôtre programme a été exécuté en IUG.
	
	Par exemple, en mode IUG, exécutez
	`EXE/ %PROG%\QUESTION.CPC`
	La routine va automatiquement mettre votre programme en mode LC ; puis à la fin de son exécution il re-bascule en mode IUG.
		Début de votre programme :
		
		`SI/ %LC_IUG_SOURCE% = OK (:`
		`  FIX/ LC_IUG_SOURCE = 0`
		`SINON/`
		`  FIX/ LC_IUG_SOURCE = %EXE_EN_COURS%`
		`  EXE/ KRNL\LC_IUG\LC_IUG.CPC`
		`FIN/ SI`
		
		Suite de votre programme...

Créer son propre format d'exécutable :

Pour ceci, il suffit tout bêtement définir 2 variables EXE_PERSO et EXE_PERSO_ICONE

Exemple :

Choisir le format de fichier, par exemple celui-ci permettra d'exécuter du code CCP avec un fichier de format .ABC

```
FIX/ EXE_PERSO = ABC
```

Choisir l'icône qui représente votre fichier exécutable

```
FIX/ EXE_PERSO_ICONE = %MEDIA%\ICONES\MONICONE.JPG
```

A retenir que les fichier au format .CPC serons toujours exécutables !

Vous ne pouvez choisir qu'un seul format.

Créer son propre format de fichiers :

Pour cela, accédez au fichier *CPCDOS\SYSTEME\KRNL\EXT.CFG* avec un éditeur de textes.

Comment c'est foutu ?

```
EXT [EXTENSION à 3 lettres](
PRG:[CIBLE de votre programme]
VAR:[Variable à communiquer a votre programme]
ICO:[ICONE du format]
)
```

Par exemple on va créer le format .ABC avec votre propre programme .CPC *(inspirez-vous des autres déjà écrits)*

```
EXT ABC(
PRG:%PROG%\MON_PROG.CPC
VAR:MA_VARIABLE
ICO:%MEDIA%\ICONE\MONICONE.BMP   (Créez une icône BMP24Bit 18x18 dans ce chemin)
)
```

Ensuite on va créer notre petit programme dans *SYSTEME\PROG\MON_PROG.CPC* qui affiche le nom de fichier cliqué

```
MSGBOX/ /TEXTE=Clique: %MA_VARIABLE% /TITRE=123 /MODE=1 /ALERTE=0
```

Puis dans la console, tapez la commande suivante pour mettre a jour les formats sur le noyau

```
SYS/ /EXT
```

Et puis voilà ! Vous pouvez créer jusqu'à 64 format de fichiers ! *(Augmentation sur demande)*

Paramètres d'entêtes des fichiers CpcdosC+ :

Optimisation :

Indicateur d'événements présents

Syntaxe :

```
#EV.DISPONIBLE(Evenement1, Evenement2, Evenement3, …)
```

Ce système permet à Cpcdos, de prévenir les événements présent dans le fichier CpcdosC+. Afin d'optimiser les recherche et éviter les « lags » lors de la lecture.
(Autant d'événement possible et utilisation de « , ; / » espacés ou pas, possible)

Qu'es ce qu'il y a d'avantageux ?

L'avantage de ce système, évite a cpcdos de chercher les événements qui ne sont pas codés dans le fichier CpcdosC+ et ceci permet d'obtenir de meilleures performances.
En effet, la lecture et écriture est le point faible de cpcdos, c'est ce qui le fait le plus ralentir, et si on indique dés le début qu'il y a **par exemple** les événements *CLIC* et *CYCLE* uniquement dans le fichier CpcdosC+, alors cpcdos ne va pas s'amuser à chercher l'événement *FOCUS*, *FERMER*, *CLICD etc.*. qui eux n'existent pas.

Voici un exemple :

Dans l'exemple, il y a que ***proc/ … CLIC*** *et* ***proc/ … CYCLE***
Eh bien on indique dés le début qu'il y a CLIC et CYCLE

```
#EV.DISPONIBLE(CLIC, CYCLE)

PROC/ Mon_Bouton(CLIC)
      Mon code...
FIN/ PROC
PROC/ Mon_Compteur(CYCLE)
      Mon code...
FIN/ PROC
```

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 Co-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
```

Pareil, mais en revoyant une valeur booléen (**1***[vrais]* ou **0***[faux]*) :

```
ini/ /testb MA_FENETRE
```

```
ini/ /testb MON_LABEL
```

TRANSFERE TEXTE ENTRE OBJETS ET FICHIERS

La commande qui le permet est :

```
ini/ /[Objet][F|E];Fichier.txt
```

Pour cette version de Cpcdos, seul le transfère de texte est disponible pour le Textebox uniquement.

F : Fichier (Obligatoire)

E : Écriture

Exemples

Transférer le texte du fichier FICHIER.TXT au textebox MON_TXTBOX:

```
INI/ /TEXTEBOXF MON_TXTBOX;FICHIER.TXT
```

Transférer le texte du textebox MON_TXTBOX au fichier FICHIER.TXT:

```
INI/ /TEXTEBOXFE MON_TXTBOX;FICHIER.TXT
```

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;MxAxRxFxTxFPx...etc"
   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 (Format supportées : BMP, GIF, JPG et PNG)

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

Type de la fenêtre

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


Il y a aussi d'autres paramètres attribuables. Placez 0, 1, 2 ou 3 après ces lettres. *(Par défaut)*

M ovable? (Déplaçable ?) [0|1]
A tive? (Interaction possible ?) [0|1]
V isible (Visible ?) [0|1]
R eductable?(Reduire) [0|1]
F ermable? [0|1]
T ache?(Affiché dans la barre des tâches ?) [0|1]
FP Fenetre**P**rioritaire ? (+entourée d'un fond gris) [0|1]
C ouleur généralisé + translucidité [0|1|2|3] ---------->
O mbre ? (Affichage de l'ombre sous la fenêtre ou pas) [0|1]
AGR ansissable ? (Si la fenêtre peut changer de taille ou pas) [0|1]
SIZ able ? (Si la fenêtre peut changer de taille manuellement via le bord bas/droite) [0|1]
FPP Fenetre**P**remier**P**lan ? (Si la fenêtre reste au premier plan ou pas) [0|1]
B ordure ? (Si on trace les contours de la fenêtre ou pas ?) [0|1]
BC ouleur (Si la couleur de fond (ini;couleur) est dessinée ou pas) [0|1]
FTC Centrer le titre **FTD** Placer le texte sur la Droite [FTG|FTC|FTD]

Exemple avec tous 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 généralisée + 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 :

L'image de coloration de l'effet de « survole » du bouton à la souris, se trouve dans :
SYSTEME\OS\MEDIA\IUG\BSURVOLE.bmp

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;V1"
```

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 = "1"**

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

Permet de définir la taille sur l'axe Y uniquement si **ini;type = "1"**

```
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

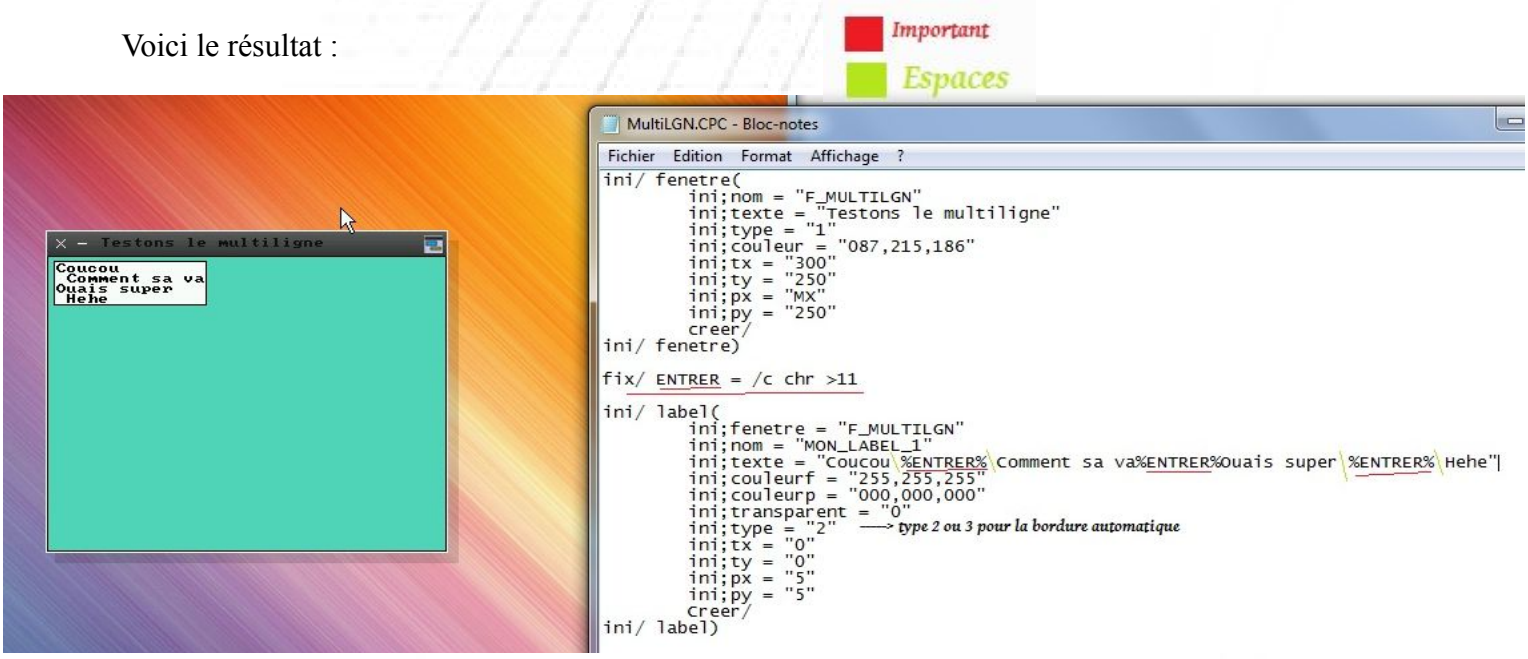
```
ini/ label)
```

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 :



A noter que si le texte est plus grand que la taille du label alors le retour de ligne sera appliqué automatiquement.

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):

Formats supportés :

- BMP 16, 24, 32 bits
- JPG 16, 24, 32 bits
- GIF (Fixe, non alpha)
- PNG (16, 24, 32(+alpha) bits)

```
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"
```


Favier Sébastien 01 Copyright©Microsf01

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 » et TYPE est à 0.
Si TYPE est à 2, 4 ou 5 alors la valeur BLEU sera la translucidité de l'image
000 > 255 (Plus la valeur est grande, plus l'image est opaque)
Si ini;type = 6 alors les valeurs seront les couleurs de fond qui se trouvent derrière l'image transparente.

```
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**
Si ini;type = 6 alors **la 3 eme** partie sera l'opacité de l'image de 000 > 255

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

0 : Image normale 1 : Cadre autour de l'image 2 : Transparence 3 : Resize
4 : Transparence + Cadre 5 :Transparence + Resize 6:Couleur fond + transparence image

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

Définir l'image (Formats supportés : BMP, JPG, PNG et GIF)

```
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

LES EFFETS GRAPHIQUES

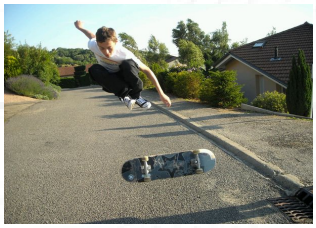
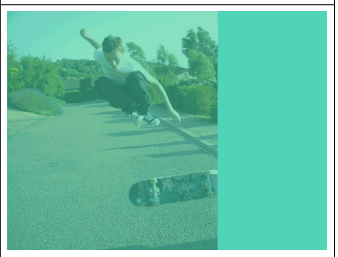
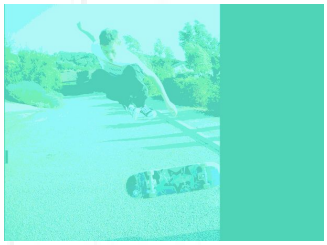
Pour ajouter des effets graphiques à votre imagebox, il faut le paramétrer comme ceci :

```
INI;TYPE = "2"
```
ou `"5"` si vous voulez resizer l'image

Ensuite vous pouvez choisir un mode graphique, et opacité si autorisé

INI;COULEUR = "000,**Mode**,**Opacité**"

Voici à quoi correspond **Mode** et **Opacité**

Mode	Résulta	Mode	Résulta
(Originale)		COULEUR = "000,**005**,000" Affiche l'image originale	
COULEUR = "000,**001**,000" Couleurs négatives		COULEUR = "000,**006**,**030**" Image translucide Le paramètre **030** peut varier entre **001** et **255** Plus la valeur est grande, plus l'image est opaque. (Bleu/vert clair)	
COULEUR = "000,**002**,000" Couleurs qui se mélangent avec les couleurs de fond (Bleu/vert clair)		COULEUR = "000,**007**,000" Affiche l'image originale	
COULEUR = "000,**002**,000" Couleurs qui se mélangent avec les couleurs de fond mais en changeant la température de la couleur (Bleu/vert clair)		COULEUR = "000,**007**,000" Transparence accentuée et mélangé avec les couleurs du fond (Bleu/vert clair)	
COULEUR = "000,**003**,000" Ne fonctionne pas			

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 « * »
Y:{Nombre} : Nombre de caractères maximum

```
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 :

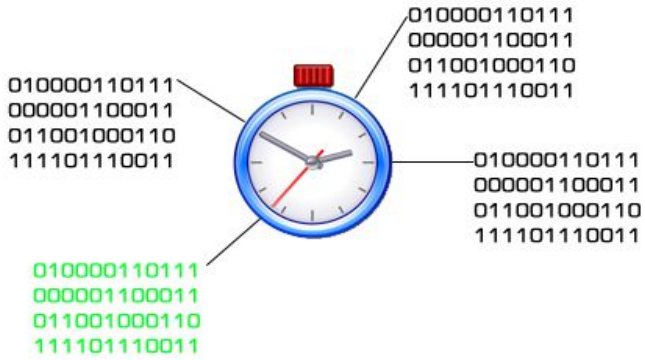
```
ini/ textebox(
      ini;nom = "MON_TEXTEBOX"
      ini;texte = "#%MA_VARIABLE"
ini/ textebox)
txt/ %MA_VARIABLE%
```

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 (compteur).

```
ini;nom = "COMPTEUR_1"
```

Permet de nommer l'objet

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

Indique au Kernel , sur quelle fenêtre allons-nous créer un compteur.
Là , il va le créer sur **FENETRE_1**

```
ini;intervalle = "3.500"
```

Indique l'intervalle en secondes (Exécuter l'événement toutes les 3,5 secondes)
Valeur maximum : 1 401 294 s Valeur minimum : 0.100 s

```
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)
```

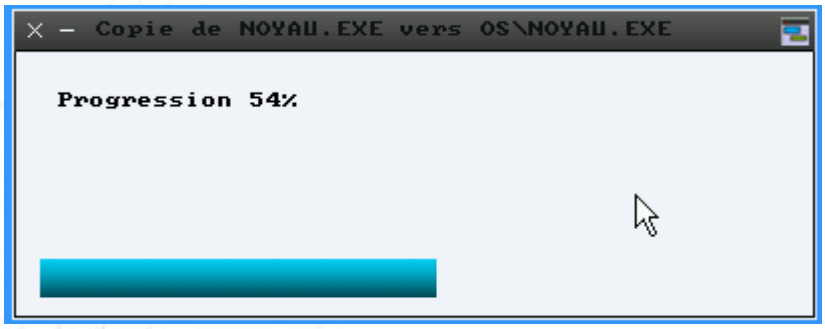
Fin de la boucle...

CREER UNE BARRE DE PROGRESSION

La commande qui le permet est :

```
ini/ Progression( )
```

Cette commande, accompagné de paramètres obligatoires , permet de créer une barre de progression avec une couleur de fond et couleur ou image qui défile en progression (exemple en image)




```
INI/ PROGRESSION(
      INI;FENETRE = "FENETRE_1"
      INI;NOM = "BAR_PROGRESS"
      INI;STATUT  = "54"
      INI;TYPE = "1"
      INI;IMG = "%MEDIA%\IUG\IMAGE.BMP"
      INI;COULEUR = "220,220,220"
      INI;COULEURP = "255,050,050"
      INI;PX = "10"
      INI;PY = "100"
      INI;TX = "370"
      INI;TY = "20"
      CREER/
INI/ PROGRESSION)
```

Explications !

```
ini/ PROGRESSION(
```

Permet d'informer au Kernel , la création d'un objet (barre de progression).

```
INI;NOM = "BAR_PROGRESS"
```

Nomme l'objet

```
INI;FENETRE = "FENETRE_1"
```

Indique au Kernel , sur quelle fenêtre allons-nous créer cette barre
Il va créer sur **FENETRE_1**

```
INI;STATUT  = "54"
```

Indique le statut en pourcentage

```
INI;TYPE = "1"
```

Pour cette version laisser sur 1

```
INI;IMG = "%MEDIA%\IUG\IMAGE.BMP"
```

Permet d'utiliser une image pour la progression
Si ce paramètre est égale à 0 alors il utilisera la couleur de COULEURP

```
INI;COULEUR = "220,220,220"
```

Place la couleur de fond, dans cet exemple, il s'agit d'une couleur gris clair

```
INI;COULEURP = "255,050,050"
```

Place la couleur de défilement si **INI;IMG = "0"**

```
INI;PX = "10"
```

Position X de la barre

```
INI;PY = "10"
```

Position Y de la barre

```
INI;TX = "10"
```

Taille X de la barre

```
INI;TY = "10"
```

Taille Y de la barre

Pour cette version, pas d'événements possible

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 !

EXECUTION ORDONNE AU PROCESSUS PARENT

La commande qui le permet est :

```
sys/ /ccp_thread {fichier.cpc}:{processus}
```

Ceci permettrais à votre application de couper la lecture de **fichier.cpc** si le **processus** se ferme ou autre.. (evite les crashs)

Voir page 93 du manuel partie **SYS/**

93

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/ /exe {fichier.cpc}
```

Ce qui permet d'exécuter un fichier CpcdosC+ de votre choix dès que l'IUG à correctement été lancé.

```
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

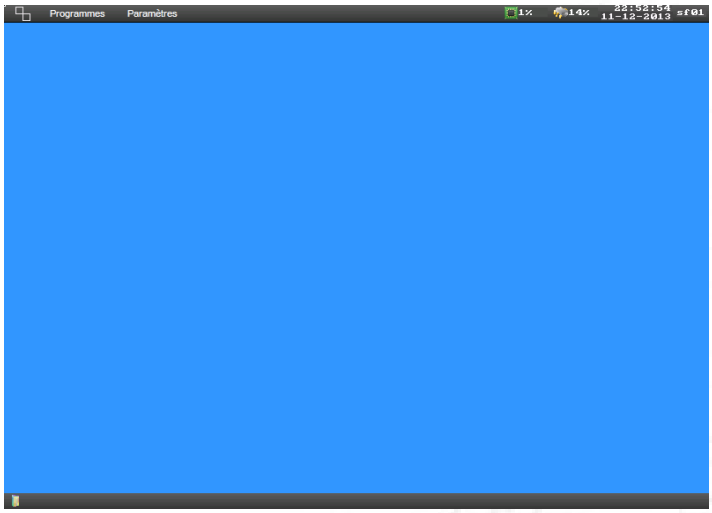
Une tactique assez utile peut être utilisée sur votre OS, par exemple si vous changer de résolution d'écran et que votre bureau, vos barres, vos menus etc.. qui eux se règle du chargement par rapport à la taille de l'écran ; si ils sont tous décalés vous pouvez utiliser IUG/ /RESET en plaçant dans le 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 !

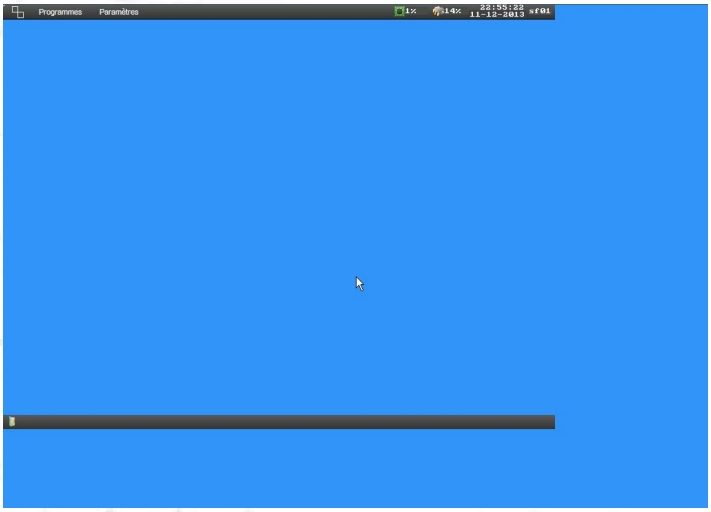
Comment faire ?

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 de coder la procédure, assurez-vous d'avoir inséré 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 s'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\PROG\MON_EV.CPC
```

Après, créez et ouvrez ce fichier MON_EV.CPC avec un éditeur de textes
puis ajoutez ce bout code :

```
Proc/ BOUTON_1(CLIC)
      msgbox/ Hello !
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}
```

Pour supprimer un événements en mémoire, il suffit de taper cette commande :

```
ev/ /s {objet/fenetre}
```

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 ..)
- PERDFOCUS (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) (en correction..)
- 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)
- AGRANDIR (Si l'utilisateur agrandir une fenêtre)
- REDUIRE (Si l'utilisateur réduit une fenêtre)
- SIZE (Si l'utilisateur change la taille de la fenêtre via le bord bas/droit)
- SURVOLE (Si l'utilisateur passe sa souris sur un objet)
- NONSURVOLE (Si l'utilisateur sors sa souris d'un objet)

Événement à la volée

Il s'agit d'une petite chose assez sympa, c'est de pouvoir stocker le nom d'objet dans une variable ou tableau et l'utiliser dans une PROC/

Exemple avec une simple variable :

```
FIX/ MA_VARIABLE = BOUTON_1
```

Plus loin.. dans le fichier événement :

```
Proc/ %MA_VARIABLE%(CLIC)
      msgbox/ Coucou, vous avez clique sur %MA_VARIABLE%
Fin/ Proc
```

Exemple avec un tableau :

```
FIX/ MON_TABLEAU(1) = MON_BOUTON_1
FIX/ MON_TABLEAU(2) = MON_BOUTON_2
FIX/ MON_TABLEAU(3) = MON_LABEL_1
FIX/ MON_TABLEAU(4) = MA_FENETRE
```

Plus loin.. dans le fichier événement :

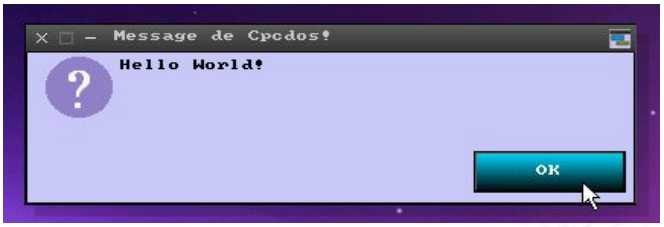
```
Proc/ %MON_TABLEAU[X]%(CLIC)
      msgbox/ Coucou, vous avez clique sur %MON_TABLEAU[X]%
Fin/ Proc
```

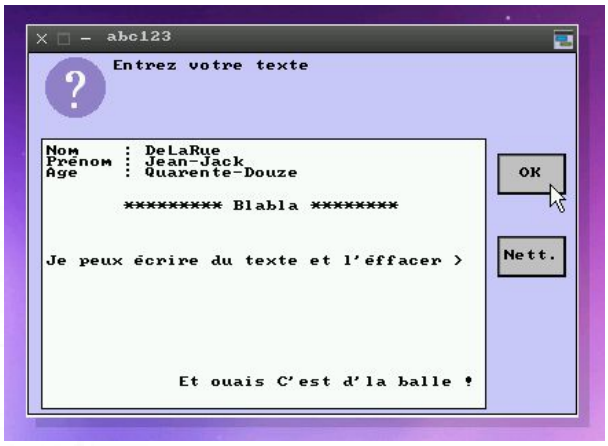
Le moteur CCP va chercher automatiquement dans tous les tableaux de MON_TABLEAU() l'objet correspondant à l'interaction.

Attention à bien mettre des crochets [ ] et non des parenthèses ( ) avec un X entre
Ce qui donne [X]

Ce qui est très utile si vous créez plusieurs objets qui ont la même fonction et finalement pouvoir exécuter le même code juste en changeant le contenu de la variable qui contient le nom de l'objet.

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 ! »

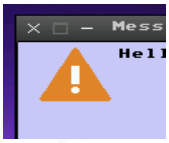
Le paramètre **/titre** indique le titre de la fenêtre, ci-dessus c'est « J'ai un message pour toi »

Le paramètre **/mode** permet de choisir entre un message avec 1 le bouton « OK » , 2 avec les boutons « Oui » et « Non » , 3 avec une zone de saisie au clavier monoligne, 4 idem mais multi-ligne

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 

4:Message ! 

Le paramètre **/nom** est optionnel s'il est omit, par défaut il se nomme « MSGCONSOLE», et la fenêtre se ferme toute seule après un clic sur OK.

Si **/nom** est utilisé, alors il faudra coder l'événement des boutons. *Exemple à la page suivante*

Exemple :

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

Nom de fenêtre par défaut : MSGCONSOLE

Impossible d'omettre un paramètre ou le changer d'ordre

Message rapide

Si par exemple vous voulez vite tester le contenu d'une variable et éviter d'écrire tous ces paramètres qui vous fait perdre du temps, taper simplement :

```
msgbox/ Mon message a ecrire et %MA_VARIABLE%
```

Nom de fenêtre par défaut : MSGRAPIDE

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/ Tu as choisis OUI !
FIN/ PROC
PROC/ BNON.MSG1 (CLIC)
      Fermer/ MSG1
      msgbox/ Tu as choisis NON..
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/ Ok tu as compris !
FIN/ PROC
```

Récapitulation des noms des objets : {Nom} est le nom du msgbox définit par /Nom={Nom}

/Mode=1 :Bouton «OK» : BOK.{Nom}
/Mode=2 :Bouton «Oui» : BOUI.{Nom} et Bouton «Non» : BNON.{Nom}

/Mode=3 :Bouton OK : BOKTXT.{Nom} et TexteBox : TXTB.{Nom}
/Mode=4 :Bouton OK : BOKTXTML.{Nom} et TexteBox : TXTBML.{Nom}

Validation par la touche ENTRE

Les msgbox par défaut se ferme par la touche ENTRE, pas besoin de configurer quoi de ce sois, mais si vous créer votre propre msgbox ou textemsgbox avec le paramètre /NOM=.. aah bizarrement la touche ENTRE ne fonctionne plus...

Pour remédier à ceci, il suffit tout simplement que le nom de votre msgbox commence par uns de ces noms :

- MSGBOX.*toto*
- TXTMSGBOX.*tata*
- TEXTMSGBOX.*toto*
- TEXTEMSGBOX.*tata*
- TEXTEMSG.*toto*
- TEXTMSG.*tata*
- TXTMGS.*toto*
- MSG.*tata*
- MSG_*toto*
- MSGBOX_*tata*

Par exemple :

```
msgbox/ /texte=Erreur ICI /titre=test /mode=1 /alerte=1 /nom=MSG.ERREUR
```

```
msgbox/ /texte=Erreur ICI /titre=test /mode=1 /alerte=1 /nom=MSGBOX.ERREUR
```

```
msgbox/ /texte=Erreur ICI /titre=test /mode=1 /alerte=1 /nom=MSGBOX_ERREUR
```

EXPLORER UN DOSSIER

La commande qui le permet est :

```
explorer/ {cible} {retour:variable/retournf:variable}
```

(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 qui sont configurés dans *SYSTEME\KRNL\EXT.CFG*

Exemple pour aller au dossier PROG :

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

OUVRIR directement un fichier dont l'extension est connu (Exemple avec .JPG) :

```
explorer/ %MEDIA%\FOND\ESPACE.JPG
```

Explorer les partition, lecteurs réseaux et lecteurs du PC :

```
explorer/ :
```

FAT12, FAT16, FAT32 uniquement pour cette version

Retourner le nom de fichier que l'utilisateur a cliqué et fermer l'explorateur, uniquement :
(Utile pour ouvrir un fichier dans un éditeur de textes, on utilise l'explorateur pour ceci)

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

La cible du fichier sélectionné dans l'explorateur par l'utilisateur sera transmis dans %MA_VARIABLE%

OU

Retourner le nom de fichier que l'utilisateur a cliqué et **NE PAS FERMER** l'explorateur :

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

PS : Pour mettre en pause le code/attendre la fermeture de l'explorateur pour continuer, jetez un coup d'œil sur la commande : *PAUSE/ /FERMER:{Fenetre}*

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, (à utiliser avec précautions), qui permet de fermer TOUS les objets et fenêtres.

La commande qui le permet est :

```
fermer/ /tout
```

Si elle est exécutée , 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/OBJETS

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/ /Objet Mon_Bouton
```

```
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 OU OBJET

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
```

Focus sur un objet comme le textebox

```
focus/ MON_TEXTEBOX
```

Ceci très utile si par exemple, vous créez un login screen, ce qui permet d'avoir l'édition au clavier automatiquement à l'ouverture du programme pour tapez directement vos logins et/ou mot de passe

Tester si la fenêtre est sélectionnée ou pas

```
focus/ /TEST MA_FENETRE_1
```

LANCER LA CONSOLE

La commande qui le permet est :

```
lc/
```

(**L**igne 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 ! ;-)

Petit cadeau : Pour lancer la console graphique pressez ALT+F10
ou exécutez la commande CPC/ /CONSOLE

LES VARIABLES D'ENVIRONNEMENTS

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]
  - CURSEUR_CHARGEMENT [Affichage du sablier *1 (défaut) sinon 0*]
  - 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 coté gauche]
  - ANTI_DEB_Y [ANTI DEBordement verticales des fenêtres en haut]
  - ANTI_DEB_XX [ANTI DEBordement horizontale des fenêtres coté droit]
  - ANTI_DEB_YY [ANTI DEBordement verticales des fenêtres en bas]
  - EXP_SCROL [Valeur du scrolling de l'explorateur *(Valeur 0 à ..)*]
  - EXP_DIM_X
  - EXP_DIM_Y [Taille x et y de l'explorateur taille Y minimum = 300]
  - EXP_POS_X
  - EXP_POS_Y [Position x et y de l'explorateur]
  - EXP_COULEUR_F_R
  - EXP_COULEUR_F_V
  - EXP_COULEUR_F_B [Couleur R,V,B du fond de l'explorateur]
  - EXP_COULEUR_C_R
  - EXP_COULEUR_C_V
  - EXP_COULEUR_C_B [Couleur R,V,B des caractères de l'explorateur]
  - 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 [Niveau de transparence de la barre de fenêtre *(valeurs 0 à 255) par défaut 100*]
- FENETRE_DEPLACE_TRANSPARENCE [Niveau de transparence de la fenêtre lors de son déplacement *(0 à 255) par défaut 200. 0:désactivé]*
-FENETRE_PLACE_HAUT_TRANSPARENCE [Niveau de transparence de la « flèche » quand l'utilisateur pointe la fenêtre vers le haut *(0 a 255) par défaut 200. 0:désactivé]*
- FENETRE_OMBRE [Niveau 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]
  - KRNL_THREAD_NB [Nombre de threads en cours]
  - MEMFREE [Mémoire disponible en octets]

- EXE_EN_COURS [Fichier .CPC en cours d'exécution]
- AEXE_EN_COURS [Ancien fichier .CPC exécuté précédemment]
- MEEFREE [Mémoire libre en octets (4Go max.)
- MEM_KRNL [Mémoire utilisée par le Kernel en ko]
- MEM_OS_DEM [Mémoire utilisée lors du démarrage de l'OS]
- MEM_OS_UTIL [Mémoire utilisée par l'utilisateur/Pendant l'utilisation de l'OS]
- SCREENSHOT [Format de fichier à enregistrer BMP , PNG *(Par défaut PNG)*]
- SCREENSHOT_NOM [Nom du fichier image *(Par défaut « SCR_ »)]*
- CLK [CLOCK – vitesse de traitement des commandes en pourcentage 1-100]
- SPEED [Cadence du processeurx86. 1 – 8 (Défaut:8=*max*)]
- 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 (ralenti plus votre PC)
Si LOG = 1 alors enregistre debug.log à la fin de l'exécution
Si LOG = 2 alors enregistre directement dans 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]
- NON_DEMARRER [1:empêche l'utilisation de la commande DEMARRER/ 2:empêche seulement une seule fois]
- 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]
- SERVICES [1 : Autorise le lancement des services 0:N'autorise pas]
- *NOM_PC* *[Nom du PC sur le réseau (Non au point)]*
- ADRESSE_IP [Adresse IP de la machine]
- MASQUE_SR [Masque sous réseau]
- PASSERELLE [Passerelle du routeur / modem / proxy / serveur]
- DNS [Adresse de résolution de noms]

- 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, à voir si votre carte graphique peux les supporter.

320x200, 320x240, 320x400, 320x480, 400x300, 512x384, 640x350, 640x400, 640x480, 640x640, 848x480, 720,480, **800x600**, 1024x576, **1024x768**, 1080x1050, 1152x864, 1280x854, 1280x720, 1280x960, 1280x1024, **1366x768**, 1440x900, 1440x960, 1440x1050, 1152x864, **1600x900**, 1680x1050 1600x1200, **1920x1080**, 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 affichée, 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
```

A noter que sur dosbox la résolution 1024x768x32b ne fonctionne pas utilisez le 16bit

CREER UN LOADING SCREEN

Ceci permet d'avoir un écran de démarrage avec une barre de progression, une barre de défilement ou les deux ;-)

Pour faire ceci créez tout d'abord votre image d'écran de démarrage avec les caractéristiques suivante :

1 Image BMP ou JPG 16 ou 24bits 800x600 nommé « FOND800.BMP » « FOND800.JPG »

1 Image BMP ou JPG 16 ou 24bits 1024x768 nommé « FOND1024.BMP » « FOND1024.JPG »

Une fois crée, copiez les dans le répertoire CPCDOS\SYSTEME\INIT

Puis accédez au fichier KRNL_DEM.CPC

```
fix/ ANNIM_COULEUR_F_R = 255
fix/ ANNIM_COULEUR_F_V = 255
fix/ ANNIM_COULEUR_F_B = 255
```

Ces variables permettent de définir la couleur de Fond de la barre de progression

```
fix/ ANNIM_COULEUR_P_R = 200
fix/ ANNIM_COULEUR_P_V = 200
fix/ ANNIM_COULEUR_P_B = 200
```

Ces variables permettent de définir la couleur Premier plan de la barre de progression

ANNIM_TYPE_BARRE Permet de choisir entre 0 et 3

- 0 : Rien
- 1 : Barre de progression (basé sur la variable %BAR_PROGRESSION% en pourcentage /100
- 2 : Barre de défilement ressemblant à celui de Windows XP
- 3 : Rassemble 1 et 2

```
FIX/ ANNIM_BAS = AUTO N16
```

Permet de chercher automatiquement la meilleure résolution d'écran selon votre système
Le paramètre N16 permet d'éviter le plus possible l'affichage 16bits.

Vous pouvez aussi choisir ces paramètres pour passer au manuel

800x16	résolution 800x600 16bits
800x24	résolution 800x600 24bits
800x32	résolution 800x600 32bits
1024x16	résolution 1024x768 16bits
1024x24	résolution 1024x768 24bits
1024x32	résolution 1024x768 32bits

La variable BAR_PROGRESSION permet de donner manuellement un pourcentage de la progression du chargement au noyau afin que le la barre type 1 puisse s'animer.

Tutoriel #5 Création d'un Loading Screen sur YouTube :

https://www.youtube.com/watch?v=0Mxs7MltOXM

CREER UNE FENETRE SIZABLE

Ceci vous permettra de changer de taille horizontale et verticale votre fenêtre manuellement en utilisant le bord bas/droit de votre fenêtre OU utiliser les boutons d'agrandissement et rétrécissement.

Pour ceci, il suffit tout simplement d'ajouter ces paramètres suivants :
AGR1 = Agrandissable et retrécissable via un bouton de la fenêtre intégré
SIZ1 = Sizable manuellement grâce au bord bas/droite de la fenêtre

Dans la procédure ini de votre fenêtre, ajoutez ces paramètres, par exemple :

```
ini/ fenetre(
      ini;nom = "MA_FENETRE_1"
      ini;texte = "Ma petite fenetre !"
      ini;type = "1;AGR1,SIZ1"
      ini;couleur = "087,215,186"
      ini;tx = "300"
      ini;ty = "250"
      ini;px = "MX"
      ini;py = "250"
     creer/
      EV/ MON_EV.CPC
ini/ fenetre)
```

Puis dans le fichier d'événement MON_EV.CPC vous pouvez interagir sur votre programme après que l'utilisateur a cliqué sur le bouton d'agrandissement, rétrécissement ou sizement manuel

```
PROC/ MA_FENETRE_1(AGRANDIR)
      Votre code si la fenêtre a été agrandie
FIN/ PROC
PROC/ MA_FENETRE_1(REDUIRE)
      Votre code si la fenêtre a été rétrécie
FIN/ PROC
PROC/ MA_FENETRE_1(SIZE)
      Votre code si la fenêtre a été sizée manuellement
FIN/ PROC
```

MENU BOOT

Pour utiliser le menu boot, au démarrage du noyau vous pouvez presser la touche F8 quand c'est demandé.

Vous avez ces choix suivant :

1. Booter l'OS en SafeMode
2. Désactiver le multi-tâche et utilise le minimum de services
3. Option 1 et 2
4. Restaurer les fichiers Cpcdos d'origine *(Non disponible pour cette version)*
5. Accéder à la console (Finit l'initialisation)
6. Accéder à la console sans initialiser
7. Arrêter la machine
8. Continuer le boot normalement

CONFIGURER & TESTER LE SYSTEME

La commande qui le permet est :

```
sys/
```

Elle ne peut 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
```

- Éteindre l'écran VGA:

```
sys/ /VGAOFF
```

Si vous avez accidentellement exécuté ceci, à la console tapez la commande ci-dessous.
Si cette commande à été exécuté « automatiquement » par votre OS appuyez sur F10 ou F12 puis tapez la commande ci-dessous.

- Allumer l'écran VGA:

```
sys/ /VGAON
```

- Fixer l'écran VGA:

```
sys/ /VGAFIX
```

- Débeloquer l'écran VGA:

```
sys/ /VGADEFIX
```

- 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:}*
Vous pouvez exécuter le programme OS\PROG\CREVIRTU.CPC *(OS requis)*

(Conseillé d'utiliser FreeDos)

- Booter sur une image virtuel :

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

Le paramètre */R* Permet de charger les pilotes réseau par défaut
Si cette option est omise vous aurez juste à choisir l'option R au démarrage.
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 l'image virtuel sur le **serveur** distant
Il faut que la cible du SERVEUR sois placé dans le dossier racine où se trouve le dossier CPCDOS
Exemples :

- Booter sur une image virtuel :

```
sys/ /BOOTIMG d:
```

Modifie le boot pour permet le boot virtuel sur le lecteur D:

- Booter sur une image virtuel + Réseau par défaut :

```
sys/ /BOOTIMG /R d:
```

Idem, mais tout ayant l'installation par défaut du réseau

> 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 x20 plus rapide théoriquement!!

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

```
KRNL32 NOVIRT
```

- Gestionnaire d'extensions de fichiers (Mettre à jour la liste) :

```
sys/ /ext
```

- 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 C 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 **/1024 ^ 2**(Mo) ou **/1024 ^ 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
```

Ce qui fait un lecteur extrêmement rapide puisque qu'il se trouve en RAM
Permet de chercher si un lecteur virtuel est déjà installé (méthode de recherche banale, il cherche de z : à d : si un lecteur est disponible en sachant que généralement un lecteur virtuel est toujours la dernière lettre. Si le lecteur virtuel n'existe pas, il va alors créer un lecteur virtuel. (D par défaut selon le PC)

Si le lecteur est trouvé, ou crée, le système d'échange virtuel sera automatiquement utilisé puisque le Cpcdos définira automatiquement la variable %SYS_VIRTUEL% sur le lecteur virtuel.

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
```

- Créer + Enregistrer le lecteur virtuel dans la variable %SYS_VIRTUEL%

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

- Créer + sans enregistrer le lecteur virtuel dans la variable %SYS_VIRTUEL%
mais dans%SYS_VIRTUEL_NONDEF% (Pour créer un lecteur sans activer l'échange virtuel)

```
sys/ /VIRTUEL /NONDEF
```

- Réinitialiser le lecteur virtuel (Utile pour faire un nettoyage des fichiers temporaire!)

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

/!\ Attention cette commande efface tout ce qu'il y a du dossier CPCDOS\ du lecteur indiqué dans %SYS_VIRTUEL%
Si la variable %SYS_VIRTUEL% n'a pas été définit, il sera alors impossible de réinitialiser le lecteur.

Attention à ce que cette variable contienne pas le lecteur où se trouve votre dossier système principal !

Favier Sébastien 01 Copyright©Microsf01

(Disque dur, USB etc..) TOUT LE CONTENU SERA EFFACE /!\

- Mettre à jour la liste de format de fichiers pour faire correspondre les icônes et/ou programmes

```
SYS/ /EXT
```

Ce système utilise le fichier *CPCDOS\SYSTEME\KRNL\EXT.CFG*
Voir partie CREER SON FORMAT DE FICHIER pour le modifier et/ou ajouter des extensions

- Exécution CCP enfant ordonné au processus parent

```
sys/ /CCP_THREAD {Fichier.cpc}:{Processus}
```

Ce qui permet de stopper l'exécution du code CpcdosC+ si t-elle fenêtre s'est fermée
Par exemple

```
sys/ /CCP_THREAD PROG.CPC:MA_FENETRE
```

Si **MA_FENETRE** se ferme alors Cpcdos stoppe la lecture de **PROG.CPC**, ce qui évite les crashs et messages d'erreurs si l'utilisateur ferme la fenêtre même si elle charge du code ou n'a pas finis de se charger graphiquement **(Fonction très utile pour améliorer la stabilitée de votre OS!)**

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)
```


Favier Sébastien 01 Copyright©Microsf01

```
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)

Pleins d'autres exemples dans le dossier PROG !

Chapitre VII – Les codes d'erreurs et d'avertissements

Les messages peuvent être affichés en console, graphiquement dans un msgbox ou dans un Debug.

Code d'avertissement	Avertissement	Code d'erreur	Erreur
AVT_000	Avertissement inconnu	ERR_000	Erreur inconnue
AVT_001	*Non définit*	ERR_001	Mémoire pleine
AVT_002	Avertissement, mémoire insuffisante	ERR_002	Erreur mémoire
AVT_003	*Non définit*	ERR_003	Erreur d'application
AVT_004	Limite de longueur chaîne du nom de variable dépasse 32 octets	ERR_004	Conflit logiciel / noyau
AVT_005	Limite de longueur variable dépassée 255ko	ERR_005	Conflit matériel
AVT_006	Création d'objet non déclaré	ERR_006	Impossible d'exécuter cette action
AVT_007	Création d'objet non fermé	ERR_007	Paramètre invalide
AVT_008	Variable introuvable	ERR_008	Fenêtre introuvable
AVT_009	Mode IUG requis	ERR_009	Commande inconnue
AVT_010	Mode LC requis	ERR_010	Erreur de syntaxe
AVT_011	Système d'exploitation non déclaré	ERR_011	Erreur fatale
AVT_012	Nom de fenêtre existante	ERR_012	Le Thread est introuvable !
AVT_013	Nom de bouton existant	ERR_013	Label spécifié introuvable
AVT_014	Nom de label existant	ERR_014	Tableau d'initialisation vide
AVT_015	Nom d'imagebox existant	ERR_015	Fichier non disponible
AVT_016	Nom de textebox existant	ERR_016	Objet introuvable
AVT_017	SI/ l'interrogé manquant	ERR_017	Erreur d'accès au registre
AVT_018	SI/ la condition est manquante	ERR_018	Chemin d'accès registre introuvable
AVT_019	SI/ l'opérateur est manquante	ERR_019	Clé de registre introuvable
AVT_020	FIN/ SI sans SI/	ERR_020	Événement introuvable
AVT_021	Nom compteur existant	ERR_021	Objet ou fenêtre introuvable
AVT_022	Le nom n'a pas pu être résolu, vérifier votre DNS	ERR_022	Format de fichier inconnu
AVT_023	Pilote de votre carte réseau non installé	ERR_023	Adresse invalide
AVT_024	Ping : Pas de réponse(s)	ERR_024	La machine distante ne répond pas
AVT_025	Répertoire réseau non disponible	ERR_025	Le serveur DNS ne répond pas
AVT_026	Lecteur réseau non disponible	ERR_026	Service indisponible
AVT_027	Lecteur non spécifié	ERR_027	Machine introuvable sur le réseau
AVT_028	Dépassement des limites fixées	ERR_028	Connexion au système distant impossible
AVT_029	*Non définit*	ERR_029	OS distant introuvable
AVT_030	JPG : espace de couleur YCbCr ne peux être mis en œuvre	ERR_030	Machine distante introuvable

AVT_031	JPG : espace de couleur CMYK ne peux etre mis en oeuvre	ERR_031	Clé non spécifiée
AVT_032	JPG : espace de couleur YCCK ne peux etre mis en oeuvre	ERR_032	Impossible de créer le lecteur virtuel
AVT_033	JPG : espace de couleur inconnue	ERR_033	Impossible d'écrire
AVT_034	Bloc de mémoire DOS échec	ERR_034	La ressource n'est pas disponible
AVT_035	Impossible d'obtenir les ID du media	ERR_035	Partage déjà en diffusion
AVT_036	Erreur de simulation d'interruption RM	ERR_036	Aucun pilote réseau TCP/IP installé
AVT_037	Allocation mémoire échec	ERR_037	Numéro de lignes hors limites
AVT_038	Lecteur non disponible	ERR_038	Impossible de fermer le processus, vérifiez qu'un PID n'est pas dupliqué
AVT_039	Service introuvable	ERR_039	*Non définit*
AVT_040	Aucun service en cours	ERR_040	*Non définit*
AVT_041	Nom barre de progression existant	ERR_041	*Non définit*
AVT_042	Structure d'événement incorrecte	ERR_042	*Non définit*
AVT_043	Impossible de lire l'entête	ERR_043	*Non définit*
AVT_044	png_sig_cmp() non disponible	ERR_044	*Non définit*
AVT_045	png_create_read_struct() non disponible	ERR_045	*Non définit*
AVT_046	png_create_info_struct()	ERR_046	*Non définit*
AVT_047	*Non définit*	ERR_047	*Non définit*
AVT_0..	..	ERR_0..	..

REMERCIEMENTS AUX TESTEURS CONTRIBUTEURS ET COMMENTATEURS DE CPCDOS ET DU MANUEL UTILISATEUR

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

GRAS : Personnes physiques

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 Developpez.com → Forum → Autres systèmes → Cpcdos

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
Copyright©Microsf01 MAI 2011 -J8781B5-
sebastien.ordinateur@hotmail.fr
http://microsf01.fr.nf/