Cpcdos OSx

Créez votre propre système d'exploitation

Par Sébastien FAVIER

Manuel du développeur Cpcdos OSx

Pour versions OS2.1 BETA 1.2

Mise à jour le 08 Avril 2020
DOC 2020.1

Accéder au plan

Liens officiels :

- https://cpcdos.net/
- http://www.facebook.com/pages/Kernel-Cpcdos-OSx/479523255400921
- http://forum-cpcdos.fr.nf/
- https://www.youtube.com/user/cpcdososx
- https://discord.gg/3Qm8xDp

Informations du projet

- Cpcdos est un logiciel libre d'utilisation. Vous avez le droit de produire librement vos projets, de le diffuser et partager en public.
- Vous avez la liberté totale de développer des outils dédiés ou partiellement dédiés à Cpcdos pour toutes plateformes. Par exemple : IDE, Compilateur, tout serveurs, tout clients, déploiement, applications, systèmes d'exploitation…
- Aucun code source complet du noyau Cpcdos ne sera dévoilé pour le moment. Seul le code source du wrapper et des modules open source inclus dans le kit de développement Cpcdos SDK et le système d'exploitation CraftyOS sont libres. Le site web de CraftyOS : http://craftyos.fr.nf/ (Ancienne version)
- Les versions de CPCDOS composées de « Béta » sont des versions publiques en phase d'expérimentation, et jugées utilisable.
- Les crashs hasardeux commencent à se réduire de plus en plus. Ils dépendent réellement de la machine.
- Le développement de ce noyau augmente de façon exponentielle, il faudra attendre quelques années pour se trouver au niveau des fonctions de base comme Windows XP (ex : Navigation html/css, JS, Java, Python, Flash, OpenGL, Wave, MP3/MP4, AVI, …)

AVERTISSEMENT

Plan du manuel Cpcdos

Liste des commandes CpcdosC+

Informations syntaxiques **de ce manuel**

[...]	→ Requiert un ou plusieurs paramètres **obligatoires.**
{...}	→ Requiert **ou pas**, un ou plusieurs paramètres **optionnels.**
Commande/	→ Commande CpcdosC+.
/Paramètres	→ Paramètres qui suit la « Commande/ » sur la même ligne.

VERSION : Syntaxes Française

Niveau 1 : Les bases du débutant

SAVOIR UTILISER LA CONSOLE D'INTERPRETATION DU NOYAU CPCDOS ET CREER UN FICHIER EXECUTABLE.

Niveau 2 : Les bases avancés

VIS-A-VIS DE LA BASE APPRISE, IL DEVRA SAVOIR MANIPULER LES VARIABLES ET TABLEAUX, CONDITIONNER, RECUPERER LES DONNEES DU CLAVIER, COPIER, RENOMMER, SUPPRIMER DES FICHIERS & DOSSIERS.

- **Créer des variables, tableaux** - - - - - **fix/** […]
  - + Supprimer variables & tableaux - - - /s […]
  - + Poser une question - - - - /q […]
  - + Récupérer contenu du keyboard-buffer - - /touche […]
  - + Attendre et récupérer la touche pressée - - /atouche […]

- **Configuration du CpcdosC+** - - - - - **ccp/** […]
  - + Changer le niveau de publicité - - /fix.niveau […]
  - + Changer la langue du système (FR ←→ EN) - - /lang […]
  - + Attendre un changement de variable - /change: […]
  - + Mettre en pause un thread pendant x *millisecondes* /Pause: […]
  - + Active la « pré-compilation » des fichiers CCP - /Optimise {…}
  - + Section critique - - - - - /debut_section_critique /fin_section_critique

- **Les conditions** - - - - - - - **si/** […] alors: sinon: fin/ si
  - + Mono-Ligne
  - + Multi-ligne
  - + Multi-conditionnelle

- **Lire dans un fichier** - - - - - - **Ouvrir/** […]
  - + Lecture binaire
- **Ecrire dans un fichier** - - - - - **Ecrire/** […]
  - + Ecriture binaire - - - - /BIN
  - + Ecriture en appending - - - - /APP

- **Configuration paramètres système** - - - - **sys/** […]
  - + Obtenir la mémoire libre, utilisé… - - /mem {…}
  - + Créer un nouveau processus vide - - /processus {…}
  - + Gestion APM (Advanced Power Management)
    - Arrêter le système /arreter
    - Redémarrer le système /redemarrer
    - Mêttre en veille le système /veille
  - + ~~Changer/lister les polices d'écritures - /police […]~~
  - + Activer/désactiver le mode DEBUG - - /debug {…}
  - + Lister/tester les résolutions d'écran - - /Ecran […]
  - + Debugger / logger le système - - /debug {…}
    - Débogage du noyau Cpcdos
    - Débogage de CPinti Core
    - Débogage via port COM RSR232
    - Débogage des connexions TCP/IP
    - Affichage du menu d'information console
  - + Exécuter une fonction wrappé - - /Wrp […]
  - + Gestionnaire de bitmaps - - /bitmap {…}
    - Lister les bitmaps en mémoire - /Liste
    - Calculer la taille de tous les bitmaps /Taille
    - Forcer le rechargement des bitmaps /Recharger
    - Lancer le garbage collector bimaps /GC
  - + Paramètres de(s) Systèmes d'exploitation(s) - /os {…}
    - Liste des OS installés
    - Mettre à jour la liste via OS.LST
    - Afficher le nombre installé

- Switcher de système d'exploitation à chaud

- **Fermer un objet, processus ou un thread** - - - **fermer/** […]

Niveau 3 : Bien plus loin

IL DEVRA SAVOIR EFFECTUER DES CALCULS MATHEMATIQUES, MISES EN FORME LOGIQUE D'UN PROGRAMME, ET UTILISER LES VARIABLES D'ENVIRONNEMENTS.

- **Exécuter le noyau Cpcdos avec 1 ou plusieurs arguments**
- **Exécuter une commande dans nouveau thread/processus** - **cmd/** […]
- **Faire des calculs arithmétiques** - - - - **/C(**[…]**)**
- **Notion et utilisation de fonction, arguments et retours** - **Fonction/** […] Fin/ fonction
  - + Arguments de fonction
  - + Retour de fonction - - - **retour/** {…}
  - + Déclaration d'une fonction externe - - - **declarer/** […]

Niveau 4 : Réseau

IL SEVRA SAVOIR TESTER L'EXISTENCE D'UNE MACHINE SUR UN RESEAU, TELECHARGER UN FICHIER SUR LE WEB EN HTTP, FTP, DEMARRER UN SERVEUR TCP, UDP SE CONNECTER A UN SERVEUR ENVOYER/RECEVOIR DES INFORMATIONS. ET CREER UN PETIT T'CHAT EN TCP.

- **Tester l'existence d'une machine** - - - - **ping/** […]
- **Télécharger un fichier sur le WEB** - - - - **telecharger/** […]
  - + Protocole HTTP - - - - `http://` […]
  - + ~~Protocole sécurisé HTTPS~~ - - - - ~~`https://`~~ […]
  - + ~~Protocole FTP~~ - - - - ~~`ftp://`~~ […]
  - + Informations serveur uniquement - - `/SRVINFO`
  - + Avec informations serveur - - - `/+SRVINFO`

- **Créer un serveur TCP** - - - - - - **serveur/** […]
  - + Mode TELNET - - - - - `/Mode:TELNET`
  - + Mode CCP - - - - - - `/Mode:CCP`
  - + Envoyer - - - - - - `/Envoyer:`[…]
  - + Recevoir - - - - `/Recevoir:`[…]
  - + Attendre la réception - - - - `/Attendre` {…}
  - + Arrêter - - - - - - `/Stop:`[…]

- **Créer un client TCP/se connecter** - - - - **client/** […]
  - + Envoyer - - - - - - `/Envoyer:`[…]
  - + Recevoir - - - - `/Recevoir:`[…]
  - + Attendre la réception - - - - `/Attendre` {…}
  - + Se déconnecter - - - - `/Stop:`[…]

Niveau 5 : Lancement d'un OS

IL SEVRA SAVOIR EXECUTER SON SYSTEME D'EXPLOITATION, GERER L'ENSEMBLE DES TACHES LIER LES FONCTIONS CPCDOSC+, POUVOIR DEVELOPPER DES ROUTINES EN C++

- **Charger un système d'exploitation installé** - - - **demarrer/** {…}
- **Exécuter l'interface graphique** - - - - **iug/** {…}
  - \+ Afficher la GUI d'un OS spécifique - - /OS
  - \+ Afficher la GUI sans OS - - - /SansOS
  - \+ Reduire la GUI pour afficher la console - - /Console ou /LC

Niveau 6 : Créer et gérer son interface graphique

IL SEVRA SAVOIR CREER DES FENETRES, DES BOUTONS, DES PICTUREBOX, GERER EGALEMENT LES POINTEURS GRAPHIQUES, LES EVENEMENTS.

- Afficher un msgbox - - - - - - msgbox/
- Créer une Fenetre - - - - - - **fenetre/** […]
- Créer un Bouton - - - - - - - **Bouton/** […]
- Créer une PictureBox - - - - - - **Picturebox/** […]
- Créer un TexteBloc - - - - - - **TexteBloc/** […]
- Créer un Textebox - - - - - - **TexteBox/** […]
- Créer un CheckBox - - - - - - **CheckBox/** […]
- Créer une barre de progression - - - - - **BarreProgression/** […]
- Modifier un objet ou une fenêtre - - - - - **/Modif:** […]
  - \+ Récupérer les propriétés graphiques - - @#
- Créer un évènement graphique - - - - - *.event & fonctions()*

VOIR AUSSI : FINALISATIONS, AFFINEMENTS, PERSONNALISATIONS ET PETITS TUTOS

- Liste des fonctions CpcdosC+ (CRT interne)
- Variables d'environnements du système complet.

Histoire du CpcdosC+

...

Qu’es ce que Cpcdos ?

Cpcdos est un projet de « co-noyau » modulaire multitâche 32 bits, visant à simplifier le développement de vos projets de système d'exploitation, tout en privilégiant l'interface graphique. Ce projet gratuit inclut principalement le langage de programmation CpcdosC+ pour simplifier la programmation d'OS et d'application, compatible avec les exécutable Windows PE **Win32** (Lua, python, et html/css à venir) pour les amateurs avertis. Le tout dans un style terminal/lignes de commandes et avec ou sans interface graphique, à votre choix !

Interface visuelle

Utilisant des procédés générique VESA, VGA, SVGA, vous pourrez attribuer des résolutions d'écran supportés par votre carte graphique et moniteur.

Les résolutions suivantes

- Standard 4:3 comme 800x600, 1024x768 ...
- Haute définition 16:9, 16:10 comme 1920x1080, 1600x900
- 4K ... à voir ;)

Les profondeurs de couleurs

- 16 bits (65 536 couleurs)
- 24 bits (16 777 216 couleurs)
- 32 bits (4 294 967 296 couleurs)

Vous pourrez créer des fenêtres, des boutons, des picturebox, des blocs de texte, des textbox, des checkbox, des slidebar, des progressbar... créer vos interactions graphiques, des évènements à la volée, des animations... c'est-à-dire tout ce dont un système d'exploitation a besoin pour interagir avec l'utilisateur. Il support nativement le format PNG, JPG/JPEG, JFIF et MJPG (Flux vidéo)
Nous avons en projet de rendre compatible le parsing des fichiers graphique XAML de Microsoft Visual Studio pour pouvoir la créer AUSSI depuis ce dernier.

Mémoire

Etant un système 32 bits, Cpcdos est théoriquement capable de supporter 4 Giga octets de RAM (Cela dépend de si vous avez une mémoire graphique dédiée ou non), ce qui suffit largement pour le moment. Le noyau seul en exécution consomme en moyenne 2 à 30 méga octets de RAM.

Ce noyau gère partiellement tout seule sa mémoire (En expérimentation).
En effet, en vue de la séparation avec le boot FreeDOS, l'allocation mémoire doit être gérée uniquement par Cpcdos. A ce jour, toute la partie interne (Shell, client, serveur tcp,udp, threads, processus, interface graphique, bitmaps RGBA, 3D engines) est soumise à l'allocation mémoire de cpcdos, excepté la LLVM.

Système

De manière approfondi, Cpcdos est capable de gérer une infinité de threads, processus, leur temps d'exécution se compte en microsecondes selon leur priorité d'exécution, mais aussi changer dynamiquement les priorités, par exemple si vous exécutez un serveur TCP MAIS qu'il n'y a aucune activité, alors le système va prioriser les autres tâches. Et vice versa ! Cpcdos Il permet aussi d'exécuter du code natif "externe".

Exécuter du code natif externe voilà d'où vient la notion du « noyau modulaire ». Cpcdos est capable de charger et d'exécuter de manière dynamique une infinité de modules (Bloc de code/d'instruction x86) venant de l'extérieur du noyau et de créer des liaisons. Cela peut être des programmes dans des fichiers compilés tels que des exécutables Windows .EXE, .DLL, LLVM/Clang ou bien des drivers. ATTENTION : La préemption est "désactivée" sur les modules externes. Sous-entendu que les .exe, .bc, et routines C++ sont exécutés en mode "Section critique" pour combler temporairement les crashs liés à une mauvaise gestion de la mémoire virtuelle de Cpcdos et sauvegardes/restaurations des contextes des tâches.

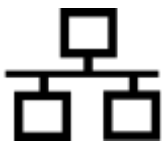
Il gère également son propre langage de programmation "à double syntaxe" sous le nom de CpcdosC+. C'est-à-dire qu'il est possible d'utiliser à la fois la syntaxe (FR) francophone et (EN) anglo-saxon ou bien les deux ! Il a la particularité de simplifier et d'automatiser l'exécution de séquences de code compliqué à gérer pour les débutants.

Petite innovation, Cpcdos est capable d'exécuter plusieurs OS en parallèle (Aucune virtualisation). Les performances restent correctes. Il est également possible de switcher entre les OS exécutés et à l'avenir de fenêtrer l'OS en arrière-plan. Un processeur type PENTIUM 4 minimum et 254Mo de RAM est recommandée.

Cpcdos est actuellement capable de gérer de manière autonome

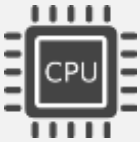
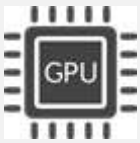
- L'interface visuelle au pixel près
- Multi-OS
- Rendu 3D via GZE Engine/OpenGL
- Curseur graphique
- Lecture des images bmp, jpg, png, gif
- Lecture et écriture des fichiers.
- Caractères UTF-8
- Polices d'écritures
- Ping ICMP
- Client/Serveur UDP/TCP, TELNET, HTTP et téléchargements HTTP.
- Protocoles Client/Serveur HTTP, FTP (en cours)
- Langage CpcdosC+, POO, Fonction(), routines, variables, calculs, exécution de code
- Mémoire RAM
- Garbage collector
- Gestionnaire des tâches
- LLVM (Low-Level-Virtual-Machine)
- Chargeur d'exécutable Win32 PE
- Mémoire virtuelle
- BIOS APM v1.0 v1.1 v1.2+ (Advanced Power Management)
- Multi-threading
- ISR
- etc...

Réseau



Précisément la couche 3 du modèle OSI, le réseau est désormais géré par Cpcdos et non FreeDos/NDIS2. Une avancée majeure, notamment dans la répartition des tâches multi-thread et la possibilité de contrôler son système à distance depuis votre téléphone ou depuis votre PC ou bien depuis un port COM Série RS232. Vous pouvez ouvrir un serveur TCP/UDP en mode Normal, Telnet, Echo. Il est possible de télécharger des fichiers WEB depuis une URL via le protocole HTTP1.1 désormais nativement géré par Cpcdos. Vous pourrez donc faire des requêtes HTML, PHP, des WebServices.. Tout ! Le protocole FTP sera très bientôt mis en place ! *En plus du Telnet, vous pourrez également contrôler/débugger votre système depuis les ports COM Série RS232 avec putty ou un terminal quelconque !*

Configuration minimale/Recommandée

Composant	Configuration minimale (Extrême)	Configuration minimale (recommandée)	Configuration idéale
	Intel x86 (386) Amd K6 (x86 serie) Fréquence 800 Mhz ***APM v1.2 minimum***	Intel Pentium 4 Intel Celeron Fréquence 1 Ghz ***APM v1.2 minimum***	Intel Core i3, i5 Fréquence 2 Ghz ***APM v1.2 minimum***
	256 Mo de RAM DDR1 Fréquence I/O 100 Mhz	512 Mo de RAM DDR2 / DDR3 Fréquence I/O 600 Mhz	4 Go de RAM DDR3 / DDR4 Fréquence I/O 1.6 Ghz
	Tout GPU supportant VGA, SVGA, XGA 50Hz Mémoire VRAM 64 Mo	Tout GPU supportant VGA, SVGA, XGA 50Hz Mémoire VRAM 64/128 Mo	Tout GPU supportant VGA, SVGA, XGA 50Hz Mémoire VRAM 64/128 Mo
	IDE, USB1 Mémoire disque 128 Mo	IDE, SATA1, SATA2, USB1, USB2 Mémoire disque 1 Go	SATA1, SATA2, SATA3, USB2, USB3 Mémoire disque 10 Go
 	Clavier et Souris PS2 Langues prises en charge EN-US **QWERTY** SW-SW **QWERTY** FR-FR **AZERTY**	Clavier et Souris PS2 ou USB Langues prises en charge EN-US **QWERTY** SW-SW **QWERTY** FR-FR **AZERTY**	Clavier et Souris PS2 ou USB Langues prises en charge EN-US **QWERTY** SW-SW **QWERTY** FR-FR **AZERTY**
	Carte réseau compatible NDIS2.0 Les plus populaires : Realtek, 3Com, Atheros, Broadcom, NE2000, NE3000 Vitesse 100 kbits/s	Carte réseau compatible NDIS2.0 Les plus populaires : Realtek, 3Com, Atheros, Broadcom, NE2000, NE3000 Vitesse 100 kbits/s	Carte réseau compatible NDIS2.0 Les plus populaires : Realtek, 3Com, Atheros, Broadcom, NE2000, NE3000 Vitesse 100 kbits/s 1Gbits/s
Exemple visuel de machines	Année 1998 - 2000  Source : Google image	Année 2000 - 2009  Source : Google image	Année 2010 - 2016  Source : Google image

Derniers tests sur les PC x86

…

Installation

Les procédés d'installations sont disponibles sur YouTube, vous trouverez le lien des vidéos ci-dessous. Il concerne VirtualBox et Boot USB

VirtualBox

Tutoriel #1 Installation de Cpcdos OS2.1 sur Virtualbox :
https://www.youtube.com/watch?v=nFkh64QX628

Tutoriel #3 Modifier/Acceder aux fichiers interne d'une VM
https://www.youtube.com/watch?v=3S92FEW9i-8

USB

Tutoriel #2 Installation de Cpcdos OS2.1 sur Virtualbox :
https://www.youtube.com/watch?v=r_5FTDjRvN4

Disque dur / SSD

…

Avant de programmer

A propos des nouvelles syntaxes CpcdosC+, petits mémo sympathiques

- **Majuscules** : Toutes les commandes **ne sont pas** sensibles aux majuscules. Vous pouvez taper TXT/ , txt/ , Txt/ , tXT/ , tXt/ → C'est la même chose !

- **Défilement hors écran :** Afin de pouvoir visualiser des gros textes qui ne tiennent pas sur la console, pour toutes commandes vous pouvez utiliser en fin de ligne le paramètre **\#PAUSE**.
  Ceci va permettre de défiler le texte via une pression de touches.
  → La touche ENTREE annule le défilement au clavier. La touche ECHAP stoppe défilement.

- **Comment ne pas remplacer la séquence « %variable% » par son contenu :** Afin de ne pas chercher/remplacer le contenu de la variable entre les signes '%' et '%', vous pouvez utiliser en fin de ligne, le paramètre **\#NON-VAR**.

- **Ne pas exécuter une fonctions via la séquence « /F:blabla( ) » par son résultat :** Afin de ne pas exécuter la fonction CpcdosC+ et remplacer cette séquence par le résultat, vous pouvez utiliser en fin de ligne le paramètre **\#NON-FN**.

- **RECOMMANDE : Assignation des valeurs** : Toutes assignation de valeurs avec le signe égale « = » dois être composé d'un espace de chaque cotées du signe '=' comme ceci :
  → **Machin** _ESPACE_ **=** _ESPACE_ **Toto**
  ✓COMMANDE/ Machin = Toto *OK*
  ✗ COMMANDE/ Machin=toto *NON*
  ✗ COMMANDE/ Machin =toto *NON*
  ✗ COMMANDE/ Machin= toto *NON*
  Remarque : La commande FIX/ n'a plus cette restriction syntaxique depuis 2017.

- Depuis la console d'interprétation CpcdosC+, vous pouvez utiliser la **complétion automatique des syntaxes CpcdosC+** avec la touche TAB. Oh qu'est-ce que c'est ?

  Par exemple, pour la commande "SERVEUR/", si vous taper "S" puis la touche TAB, vous aurez à l'écran, la liste des commandes qui commencent par "S". Puis si vous continuez, "SER" puis la touche TAB, vous remarquerez que la commande au complet a été écrite.
  Puis si vous pressez TAB sans avoir écrit quelque chose, vous aurez la liste des commandes sans descriptions. Pour avoir les descriptions, utilisez la commande "AIDE/".

- Lorsque vous exécutez un fichier CpcdosC+, ces variables de **niveau 2** sont créées :
  `%_EXE_PATH_%` → Chemin jusqu'à l'exécutable
  `%_EXE_PATH_F_%` → Répertoire Courant où est exécuté le fichier (**F**older)
  `%_EXE_PID_%` → PID du programme exécuté
  `%_EXE_TID_%` → TID du programme exécuté
  `%_EXE_DATE_%` → Date d'exécution du programme

- Il existe deux manières d'afficher le contenu d'une variable
  `%Ma_Variable%` → Utilisé de partout.
  `${Ma_Variable}` → Utilisé dans la propriété `.text` des objets pour les actualisations graphiques lors ce que le paramètre `UPD:1` est utilisé.

A propos du débogage Cpcdos, CPinti-Core et TCP/IP

- Par défaut, le **débogage est désactivé après le démarrage normal du noyau**. Pour l'activer voir ICI S'il est activé, vous aurez à l'écran toutes les opérations arrière plans ainsi que le travail de tous les threads et des fonctions interne de Cpcdos.
- Le debogeur CPinti Core regorge encore plus d'information sur les opérations arrière-plan. Vous verrez le travail colossal que fait le **CPinti Core←→Cpcdos** si vous lui faites faire des opérations ! Très utile si vous hébergez un serveur TCP ou UDP afin de voir les informations détaillées des activités client ←→ Serveur et application ←→ Threads etc…
- Il est possible d'utiliser le port COM et le TELNET pour déboguer le noyau.

C'est parti !

Niveau 1 : Les bases du débutant

Ecrire du texte à l'écran

COMMANDE :

```
TXT/ {/#DEBUG} {Texte} {/#R}
```

TeXTe

FONCTIONNALITE :

Cette commande permet d'écrire des caractères sur l'écran de la console.

Par défaut il s'agit de caractères ASCII. Vous pouvez utiliser cette commande pour écrire des caractères UTF-8 avec une police et une couleur d'écriture personnalisée.

PARAMETRES DISPONIBLE :

- /#R → Permet de **R**ester sur la même ligne pour le prochain affichage.
- /#DEBUG → Permet au texte d'être « logée » tant qu'activité du système.
- \#PAUSE → Permet de défiler l'écran avec une touche. ECHAP pour arrêter. ENTRER pour annuler.
- \% → Affiche le caractère '%' sans être pris en compte tant que variable. Possibilité donc d'être affiché dans le DEBUG.LOG
- \#REFORMAT-VAR **ou** \#VAR-REFORMAT **ou** \#REFORMATER-VAR **ou** \#VAR-REFORMATER
  → Permet de reformater la commande/texte si une variable contient une autre variable à lire.

- \#NON-VAR ou \#NO-VAR
  → Interdit la recherche de variables. *Garde textuellement* ***%variable%.***

- \#NON-FN ou \#NO-FN
  → Interdit l'exécution d'une fonction. Garde textuellement **/F:ma_foncton(…)**

EXEMPLE 1 :

```
Txt/ Hello world !
```

Affiche « Hello world ! » sur votre écran.

EXEMPLE 2 :

```
SYS/ /POLICE /ACTIVER
SYS/ /POLICE ARIAL
TXT/ Je suis écrit en Arial !
SYS/ /POLICE COMIC SANS MS
TXT/ Je suis écrit en Comic sans ms ! :-)
SYS/ /POLICE /DESACTIVER
TXT/ Je suis écrit en ASCII.
```

CORRESPONDANCE ANGLAISE :

```
SYS/ /FONT /ENABLE
SYS/ /FONT ARIAL
TXT/ I'm arial font
SYS/ /FONT COMIC SANS MS
TXT/ I'm Comic sans ms font :-)
SYS/ /FONT /DISABLE
TXT/ I'm ASCII font
```

APERÇU :

EXEMPLE 3 :

```
FIX/ Abc = Hello !
FIX/ Xyz = %Abc%
Txt/ %XYZ%
```

Affiche « **Hello !** »

EXEMPLE 4 :

```
FIX/ Abc = Hello !
FIX/ Xyz = \%%Abc%\%
Txt/ %XYZ%
```

Affiche « **%Abc%** »

EXEMPLE 5 :

```
Txt/ \%Blabla\% 57\%
```

Affiche « **%Blabla% 57%** »

EXEMPLE 6 :

```
FIX/ Abc = Hello !
FIX/ Xyz = \%%Abc%\%
Txt/ %XYZ%
```

Affiche « **%Abc%** »

EXEMPLE 7 :

```
FIX/ Abc = Hello !
FIX/ Xyz = \%%Abc%\%\#REFORMAT-VAR
Txt/ %XYZ%
```

Ou

```
FIX/ Abc = Hello !
FIX/ Xyz = \%%Abc%\%
Txt/ %XYZ%\#REFORMAT-VAR
```

Affiche « **Hello !** »

EXEMPLE 8 (INTERDIT LA RECHERCHE DE VARIABLE) :

```
Fix/ abc = 123
Txt/ Pour afficher une variable il faut ecrire %abc%.\#NON-VAR
```

Affiche « **Pour afficher une variable il faut ecrire %abc%.** »

EXEMPLE 9 (EXECUTER UNE FONCTION)

```
Txt/ Resultat de la fonction : /F:CPC.MID(Salut,3)
```

Affiche « **Resultat de la fonction : ut** »

EXEMPLE 10 (INTERDIT L'EXECUTION D'UNE FONCTION)

```
Txt/ Resultat de la fonction : /F:CPC.MID(Salut,3)\#NON-FN
```

Affiche « **Resultat de la fonction : /F:CPC.MID(Salut,3)** »

AUTRES INFORMATIONS :

- Utilisable uniquement en mode LC.
- Compatible ASCII et UTF-8.
- Compatible avec les polices d'écritures.

VOIR AUSSI :

`Cls/ ; Couleurc/ ; couleurf/`

Effacer l'écran

COMMANDE :

```
CLS/
```

CLear Screen

FONCTIONNALITE :

Cette commande permet d'effacer l'écran de la console uniquement.

PARAMETRES DISPONIBLE :

- Aucuns paramètres disponibles pour cette version.

EXEMPLE :

```
Txt/ Mon texte affiché à l'écran
Txt/ Mon 2eme texte
CLS/
Txt/ Ah ! Le texte est effacé
```

CORRESPONDANCE ANGLAISE :

```
Txt/ My text on displayed on screen
Txt/ My 2nd text
Cls/
Txt/ Ah! Text was deleted
```

APERÇU :

AUTRES INFORMATIONS :

- Utilisable uniquement en mode LC.
- Compatible ASCII et UTF-8
- Compatible avec la police d'écriture.
- Compatible avec le débogueur CPinti Core

VOIR AUSSI :

`Txt/ ; Couleurc/ ; couleurf/ ;`

Changer la couleur des caractères (+Fond)

COMMANDE :

```
CouleurC/ {R:[Valeur 000 à 255]} {V:[000 à 255]} {B:[000 à 255]}
```

CouleurCaractere

```
CouleurF/ {R:[Valeur 000 à 255]} {V:[000 à 255]} {B:[000 à 255]}
```

CouleurFond

FONCTIONNALITE :

Ces commandes permettent de changer la **Couleur** des **C**aractères ou du **F**ond de la console. Vous pouvez définir 1, 2 ou 3 couleurs (R et/ou V et/ou B) en même temps.

PARAMETRES DISPONIBLE :

- ~~R:[nombre à 3 chiffres] → Définit la couleur **R**ouge entre 000 et 255.~~
- ~~V:[nombre à 3 chiffres] → Définit la couleur **V**ert entre 000 et 255.~~
- ~~B:[nombre à 3 chiffres] → Définit la couleur **B**leu entre 000 et 255.~~
- [Nombre 0 a 15] → Couleurs EGA de la console 4 bits

EXEMPLE GRAPHIQUE:

```
~~CouleurC/ R:080 V:210 B:120~~
~~Txt/ Mon texte en vert clair.~~
~~CouleurC/ R:255 V:050~~
~~Txt/ Mon texte en rose~~
```

Couleur
0 – noir (#000000)
1 – bleu (#0000AA)
2 – vert (#00AA00)
3 – cyan (#00AAAA)
4 – rouge (#AA0000)
5 – magenta (#AA00AA)
6 – brun (#AA5500)
7 – gris clair (#AAAAAA)
8 – gris foncé (#555555)
9 – bleu brillant (#5555FF)
10 – vert brillant (#55FF55)
11 – cyan brillant (#55FFFF)
12 – rouge brillant (#FF5555)
13 – magenta brillant (#FF55FF)
14 – jaune (#FFFF55)
15 – blanc (#FFFFFF)

CORRESPONDANCE ANGLAISE :

```
~~ColorF/ R:080 G:210 B:120~~
~~Txt/ My text in light green~~
~~ColorF/ R:255 G:050~~
~~Txt/ My text in purple~~
```

APERÇU :

```
Mon texte en vert clair.
Mon texte en rose.
```

EXEMPLE CONSOLE (EGA):

```
CouleurC/ 5
Txt/ Mon texte Magenta
CouleurC/ 11
Txt/ Mon texte Cyan brillant
```

APERÇU :

```
 > couleurC/ 5
 > txt/ Mon texte magenta
Mon texte magenta
 > CouleurC/ 11
 > txt/ Mon texte Cyan brillant
Mon texte Cyan brillant
```

AUTRES INFORMATIONS :

- Utilisable uniquement en mode LC.
- Compatible ASCII, UTF-8
- Compatible avec les polices d'écritures.
- Pour le graphique, impérativement utiliser 3 chiffres ! ex : 005 050 150
- Couleur console EGA des caractères par défaut : **7** et le fond : **0**

VOIR AUSSI :

Txt/

Exécuter un fichier CpcdosC+

COMMANDE :

```
EXE/ {/win32, /llvm, &, &+} [FichierProgramme]{/l:Label, /l:#NoLigne}
```

EXEcuter

FONCTIONNALITE :

Cette commande permet l'exécution d'un fichier exécutable CpcdosC+, Windows 32bits et LLVM.

A propos du CpcdosC+ le fichier source est de format TEXTE, vous pouvez écrire votre code depuis un simple bloc-notes et ne requiert aucune compilation !

Par défaut, cette commande utilise le thread courant pour s'exécuter. Elle peut bien évidement être exécutée dans un autre thread afin de ne pas bloquer le thread courant.
Elle termine également l'exécution du précédent fichier exécutable parent sans reprise. Sauf si vous spécifiez le paramètre « & ». (Compatible uniquement avec des fichiers .CPC)

Vous pouvez aussi exécuter des fichiers Windows et LLVM. Si vous souhaitez exécuter ceci dans un autre thread, vous devez utiliser la commande :

```
CMD/ /Thread exe/ /Win32 program.exe
```

PARAMETRES DISPONIBLE :

- & → Permet d'exécuter un autre fichier CpcdosC+ dans le même thread et SANS mettre fin au thread parent. Elle attend la fin de l'enfant **pour reprendre** son exécution.
- &+ → Permet d'exécuter un fichier CpcdosC+ dans un nouveau thread. *(Exécution asynchrone)*
- /L: → Permet d'exécuter le code à une zone spécifique du fichier via un nom de label.
- /LLVM → Permet d'exécuter un programme IR compilé avec CLANG, depuis une LLVM.
- /PE ou /Win32 → Permet d'exécuter un programme Win32 au format PE.

EXEMPLE 1 :

Exécuter un fichier CpcdosC+ sur le même thread **en fermant** le précédent fichier exécuté.

```
Exe/ MonProgramme.cpc
```

EXEMPLE 2 :

Exécuter un fichier CpcdosC+ sur le même thread **sans fermer** le précédent fichier exécuté.

```
Exe/ & MonProgramme.cpc
```

EXEMPLE 3 :

Exécuter un fichier CpcdosC+ sur un nouveau thread **en parallèle** en continuant l'exécution de précédent

```
Exe/ &+ MonProg.cpc
```

EXEMPLE 4 :

MonProg.CPC

```
Txt/ On execute notre premier programme
Exe/ & MonProg1.cpc

Txt/ On execute notre 2eme programme dans un nouveau thread.
Exe/ &+ MonProg2.cpc

Txt/ On Execute un autre programme
Exe/ MonProg3.cpc

Txt/ Ce texte ne sera jamais affiche.
```

MonProg1.CPC

```
Txt/ Hello, je suis le programme 1, bye!
```

MonProg2.CPC

```
Txt/ Hello, je suis le programme 2
CCP/ /PAUSE 1000
Txt/ Je suis toujours le programme 2 \#R
Txt/ Je viens de faire une pause de 1 seconde. Bye!
```

MonProg3.CPC

```
Txt/ Hello, je suis le programme 3, bye!
```

APERÇU :

```
 > exe/ MonProg.cpc
On execute notre premier programme
Hello, je suis le programme 1, bye!
On execute notre 2eme programme dans un nouveau thread.
On Execute un autre programme
Hello, je suis le programme 3, bye!
Hello, je suis le programme 2
Je suis toujour le programme 2, je viens de faire une pause de 1 seconde. Bye!
```

EXECUTER UNE LLVM VIA CLANG

Ce paramètre permet l'exécution du code IR généré via un compilateur Clang (Windows, linux, mac...) avec les paramètres dédiés à Cpcdos, le tout dans une **LLVM** (Low-Level-Virtual-Machine) inclus nativement dans CPinti Core. Ce qui vous donne la possibilité de coder vos application C/C++ et de l'exécuter ici même ! Les formats acceptés sont : **.BC** et **.LL**

Concernant la compilation, il est fortement recommandé d'utiliser ces arguments suivants :

```
-std=c++14 -m32 -fno-exceptions -target i686-pc-mingw32-elf -S -emit-llvm
```

« **-std=c++11**», «**-std=c++14**» et «**-std=c++17**» sont fonctionnels.

EXEMPLE 1 – EXECUTER UNE LLVM AVEC UN FICHIER .BC :

```
Exe/ /LLVM MonProgramme.bc
```

EXEMPLE 2 – EXECUTER UNE LLVM AVEC UN FICHIER .LL :

```
Exe/ /LLVM MonProgramme.ll
```

EXECUTER UN WIN32 FORMAT PE

Ce paramètre permet l'exécution de code x86 au format PE spécifique à Microsoft Windows. Bien évidemment, il s'agit seulement d'exécutable Windows 32 Bits. Cette fonctionnalité est toujours en phase de développement en vue de sa stabilité et des gourmandes dépendances concernant les fonctions liées aux fichiers .DLL du noyau NT de Microsoft.
Cette fonctionnalité est "*99% compatible*" s'il s'agit de code quasi, voire indépendant des ressources Microsoft. Ce qui veut dire que moins l'exécutable "***import*** des librairies" plus il sera stable et compatible. **Beaucoup de fonction du kernel NT de Microsoft ont été importés et adapté pour supporter quelques dépendances de vos .EXE, ça fonctionne, mais rien n'est parfait, ne soyez pas prétentieux** 😉
Je vous conseille fortement d'utiliser le compilateur **CWC** (Acronyme « **CWave Collection** ») un remplacement du compilateur GCC mis au point par mon ami Mickael BANVILLE. Ce dernier permet de générer des exécutables Windows en simplifiant la compatibilité entre CPCDOS ←→ WINDOWS.
Vous pouvez MIXER du code Windows et Cpcdos dans le MÊME fichier .exe !

Plus d'informations et téléchargement du compilateur :
https://openclassrooms.com/forum/sujet/compilateur-cwc
Ou via les outils de développement sur le site de Cpcdos.net
https://cpcdos.net/fr/download

EXEMPLE :

```
Exe/ /win32 MonProgramme.exe
```

AUTRES INFORMATIONS :

- Compatible ASCII et UTF-8. (Instance host)
- Compatible avec les polices d'écritures. (Instance host)
- Niveau de visibilité des variables, niveau 2 par défaut. Qu'es ce que le niveau de visibilité ?
- Cette commande gère intégralement le multithreading, il est capable d'exécuter en parallèle des centaines fichiers CpcdosC+ en même temps, avec des priorités plus ou moins différentes, avec autant que la mémoire de la machine le permet. Pour cette version, chaque nouvelle instance CpcdosC+ prennent environ ~500Ko de mémoire RAM.

VOIR AUSSI :

```
Aller/ ; stop/ ;
```

Afficher l'aide

COMMANDE :

```
AIDE/ {Commande}
```

FONCTIONNALITE :

Cette commande affiche à l'écran, la liste des commandes disponibles sur le noyau.
Si elle est suivie par une autre commande CpcdosC+, alors elle affiche l'aide de cette dernière.

EXEMPLE 1 :

```
Aide/
```

EXEMPLE 2 (DANS UN FICHIER) :

```
Aide/ txt/
Aide/ Couleurf/
Aide/ Colorb/
Aide/ rem/
…
```

CORRESPONDANCE ANGLAISE :

```
Help/ txt/
Help/ Couleurc/
Help/ Colorf/
Help/ rem/
…
```

APERÇU :

AUTRES INFORMATIONS :

- Compatible ASCII et UTF-8.
- Compatible avec les polices d'écritures.
- Utilisez le paramètre \#PAUSE pour défiler les commandes manuellement avec une touche puis ENTER pour annuler le défilement et ECHAP pour arrêter

VOIR AUSSI :

Positionner le curseur

COMMANDE :

```
POS/ {x/x: y/y:}
```

POSition

FONCTIONNALITE :

Cette commande permet d'afficher ou de changer la position du curseur console.

PARAMETRES DISPONIBLE :

- X : → Changer en X
- Y : → Changer en Y
- X → Afficher en X
- Y → Afficher en Y
- → Afficher X et Y

EXEMPLE 1 :

```
POS/ X:15
Txt/ Je suis la !
POS/ Y:10
Txt/ et aussi la !
```

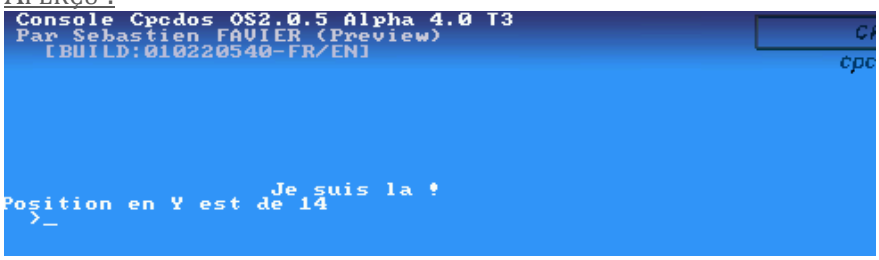
EXEMPLE 2 (RECUPERER POSITION Y):

```
POS/ X:15 Y:5
Txt/ Je suis la !
@#Position_Y POS/ Y
Txt/ Position en Y est de %Position_Y%
```

CORRESPONDANCE ANGLAISE :

```
LOC/ X:15 Y:5
Txt/ I'm here !
@#Y_Location LOC/ Y
Txt/ Y location is in % Y_Location%
```

APERÇU :



AUTRES INFORMATIONS :

- Compatible ASCII et UTF-8.
- Compatible avec les polices d'écritures.

VOIR AUSSI :

Les commentaires

COMMANDE :

```
REM/ {Remarques}
```

***REM**arque*

FONCTIONNALITE :

Cette commande inoffensive, aucune influence sur le système, et permet de placer des commentaires dans votre code afin de laisser des traces et de l'aide pour vous ou pour les développeurs Open source :-).

Vous pouvez également utiliser le caractère « ' » ou « // » pour commenter.

EXEMPLE 1 :

```
REM/ Ceci est un commentaire
```

EXEMPLE 2 (DANS UN FICHIER) :

```
REM/ Nous allons executer le programme qui fait blaba…
EXE/ &+ MonProgramme.cpc

// On affiche a l'ecran que le programme a été execute
Txt/ Programme execute !

'  FIN du code.
```

CORRESPONDANCE ANGLAISE :

```
REM/ We will to launch program
EXE/ &+ MyProgramme.cpc
// We display a text on the screen
Txt/ Programme execute !
' END of code.
```

APERÇU :

AUTRES INFORMATIONS :

Lors de la pré-compilation CpcdosC+ via **CCP/ /OPTIMISATION = 1** Les commentaires sont supprimés de la mémoire lors de l'optimisation. Attention à ce que votre code d'indexe pas les zones d'exécution **via leur numéro de ligne**, ceci pourrait décaler votre numéro de ligne souhaité à un autre, et finir par reproduire Hiroshima… ☹

VOIR AUSSI :

ARRETER LE NOYAU URGEMMENT

COMMANDE :

```
STOPK/
```

STOP Kernel

FONCTIONNALITE :

Cette commande permet de stopper de manière brute le noyau Cpcdos en prenant bien-sûr en compte de manière automatique, la fermeture des descripteurs de fichiers.

Vous avez également la commande

```
SYS/ /STOPK
```

Qui stoppe le noyau **PROPREMENT**, ferme tous les objets graphiques, tous les threads & processus, et attend qu'ils soient tous fermé et décharge le noyau. Ce qui évite des probables crash.

PARAMETRES DISPONIBLE :

EXEMPLE :

```
STOPK/
```

VOIR AUSSI :

Atteindre un label / Saut de code

COMMANDE :

```
ALLER/ [Nom du label]
```

FONCTIONNALITE :

Cette commande à utiliser dans un fichier permet d'atteindre un autre emplacement dans le code spécifié par un 'nom de label'.

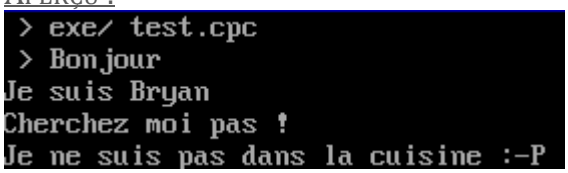
EXEMPLE 1 (DANS UN FICHIER) :

```
Txt/ Bonjour
Aller/ label_1
:label_2:
Txt/ Cherchez moi pas !
Aller/ Label_3
:Label_1:
Txt/ Je suis Bryan
Aller/ label_2
:Label_3:
Txt/ Je ne suis pas dans la cuisine :-P
```

CORRESPONDANCE ANGLAISE :

```
Txt/ Hello
Goto/ label_1
:label_2:
Txt/ don't search me !
Goto/ Label_3
:label_1:
Txt/ I'm Bryan
Goto/ label_2
:label_3:
Txt/ I'm not in the kitchen :-P
```

APERÇU :



AUTRES INFORMATIONS :

- A utiliser avec prudence. Eviter d'utiliser cette commande en sortant du contexte **de manière récurrente** au risque de cracher la stack mémoire du moteur CpcdosC+ et donc le noyau.
  Par exemple, évitez ceci :

```
:BOUCLE:
// ...
SI/ "blabla" = "blabla" alors:
     Aller/ BOUCLE   ici le noyau croit que "boucle" est encore dans le contexte "ALORS:"
     Du coup il accumule en mémoire « ALORS: ALORS: ALORS: etc… » et CRASH
Fin/ si
```

VOIR AUSSI :

Fonction/ ;

STOPPER LA LECTURE D'UN CODE CPCDOSC+

COMMANDE :

```
STOP/
```

FONCTIONNALITE :

Cette commande à utiliser dans un fichier permet d'arrêter l'exécution du code d'un fichier CpcdosC+ dont sont TID est actif. Ceci va également libérer les ressources allouées en mémoire RAM.

EXEMPLE 1 (DANS UN FICHIER) :

```
TXT/ Voulez-vous quitter le programme ? (O/N)
FIX/ /Q Question
SI/ "%Question%" = "O" alors:
     Txt/ Byebye !
     Stop/
     Txt/ Ce texte ne sera jamais exécuté !
Sinon:
     Txt/ Ok, on recommence !
Fin/ si
```

AUTRES INFORMATIONS :

Fonctionne exactement du même principe que si la lecture arrive à la fin du fichier.

VOIR AUSSI :

LISTER le contenu d'un REPERTOIRE

COMMANDE :

`REP/ {chemin} {filtres *|*.*}`	`DIR/ {path} {filters *|*.*}`

FONCTIONNALITE :

Cette commande permet de lister le contenu d'un répertoire (Fichier&dossiers). Il est possible d'appliquer des filtres pour affiner les résultats

PARAMETRES DISPONIBLES :

- *Aucuns paramètres* : liste le contenu du répertoire courant avec le filtre standard « *.* »
- Avec un chemin : Liste le contenu du répertoire précisé.
- Avec un filtre : Liste le contenu du répertoire en appliquant le filtre.
- /B : Lister de manière "**b**rève" avec le minimum d'informations

EXEMPLE 1 :

```
REP/      ou     REP/ .
```

Affiche le contenu du répertoire courant

EXEMPLE 2 :

```
REP/ C:\Dossier
```

Affiche le contenu du répertoire du lecteur C « Dossier »

EXEMPLE 3 :

```
REP/ ..\dossier\toto
```

Affiche uniquement tous les fichiers avec l'extension. CPC du répertoire toto

EXEMPLE 4 :

```
REP/ dossier\toto\*.cpc
```

Affiche uniquement tous les fichiers avec l'extension. CPC du répertoire toto

EXEMPLE 5 :

```
REP/ dossier\toto\b*.*
```

Affiche uniquement tous les fichiers qui commencent par la lettre « b »

EXEMPLE 5 :

```
REP/ dossier\toto\*e*.*p*
```

Affiche uniquement tous les fichiers qui contiennent la lettre « e » et l'extension « p »

AUTRES INFORMATIONS :

- Cpcdos supporte les deux URI "\" "/" vous pouvez les mixer entre eux 😉
- BUG CONNU : Pour filtrer le répertoire courant, vous devez ajouter un « / » ou « \ »
  Exemple (Lister les fichiers .CPC) **`REP/ \*.CPC`**

VOIR AUSSI :

Supprimer/ ; renommer/ ; copier/ ; supprimer/

COPIER UN FICHIER ou REPERTOIRE

COMMANDE :

```
COPIER/ {source},{destination}
```

```
COPY/ {source},{destination}
```

FONCTIONNALITE :

Cette commande permet de copier l'intégralité d'un fichier, ~~ou d'un répertoire.~~

PARAMETRES DISPONIBLES :

- /Progression:*{Variable}* → Stocke dans variable, la progression en %
- /Octets:*{Variable}* → Stocke dans variable, le nombre d'octets écrits.
- /OctetsParSec:*{Var.}* → IDEM, mais par secondes uniquement.

EXEMPLE 1 :

```
Copier/ MonFichier.CPC, MonRepertoire/MaCopie.cpc
```

Copie "`MonFichier.CPC`" vers le répertoire "`MonRepertoire`" sous le nom de "`MaCopie.cpc`"

EXEMPLE 2 :

```
Copier/ ..\Repertoire\fichier.exe, .\fichier.exe
```

EXEMPLE 2 (DANS UN AUTRE THREAD) :

```
CMD/ /THREAD Copier/ MonFichier.ZIP, Dossier/MonFichier.ZIP
```

Utiliser CMD/ /THREAD va permettre d'exécuter la copie dans un nouveau thread sans bloquer le courant. Si vous utilisez une console, ceci va éviter de le "figer" le temp de la copie.

EXEMPLE 2 (DANS UN AUTRE THREAD + SATISTIQUES) :

```
CMD/ /THREAD Copier/ FichierA.exe, FichierB.exe /Progression:toto
Txt/ %toto% \%
```

La variable %toto% contiendra une valeur entre 0 et 100 qui indique la progression en pourcentage.
Le " **\%** " permet d'afficher "**%**" sans qu'il soit confondu avec les "**% %**" des variables.

EXEMPLE 3 (IDEM) :

```
CMD/ /THREAD Copier/ FichierA.exe, FichierB.exe /octets:toto
Txt/ %toto% \%
```

La variable %toto% contiendra le nombre d'octets écrits sur la destination.

EXEMPLE 4 (IDEM) :

```
CMD/ /THREAD Copier/ FichierA.exe, FichierB.exe /OctetsParSecondes:toto
Txt/ %toto% \%
```

La variable %toto% contiendra le nombre d'octets écrits **PAR SECONDES** sur la destination.

Convertir les octets à une unité souhaitée :

```
// Affiche le nombre d'octets écrit par secondes
txt/ Vitesse %toto%

// Affiche le nombre de kilo-octets écrit par secondes
txt/ Vitesse /c(%toto% / 1024)

// Affiche le nombre de mega-octets écrit par secondes
txt/ Vitesse /c(%toto% / (1024*1024))
```

AUTRES INFORMATIONS :

- Vous pouvez utiliser les 3 paramètres /PROGRSSSION, /OCTETS /OCTETSPARSECONDES sur la même ligne
- Cpcdos supporte les deux URI "\" "/" vous pouvez les mixer entre eux 😉
- Pour cette version, n'oubliez pas la virgule entre SOURCE et DESTINATION qui sert d'argument

VOIR AUSSI :

Supprimer/ renommer/

RENOMMER UN FICHIER ou REPERTOIRE

COMMANDE :

`RENOMMER/ {source},{Nouveau nom}`	`RENAME/ {source},{New name}`

FONCTIONNALITE :

Cette commande permet de renommer un fichier ~~*ou un répertoire.*~~

EXEMPLE 1 :

```
Renommer/ MonProg.CPC, NouveauNom.log
```

Copie "`MonFichier.CPC`" vers le répertoire "`MonRepertoire`" sous le nom de "`MaCopie.cpc`"

EXEMPLE 2 :

```
Renommer / ..\Repertoire\fichier.exe, prog.exe
```

AUTRES INFORMATIONS :

- Pour cette version, n'oubliez pas la virgule entre SOURCE et NOM qui sert d'argument
- Cpcdos supporte les deux URI "\" "/" vous pouvez les mixer entre eux 😉

VOIR AUSSI :

Supprimer/ copier/

SUPPRIMER UN FICHIER ou REPERTOIRE

COMMANDE :

`SUPPRIMER/ {source}`	`DELETE/ {source}`

FONCTIONNALITE :

Cette commande permet de supprimer un fichier ~~*ou un répertoire.*~~

PARAMETRES DISPONIBLES :

- ~~*/ZERO:{Nombre de passes} : Effacement "sécurisé".*~~

EXEMPLE 1 :

```
Supprimer/ MonFichier.CPC
```

Copie "`MonFichier.CPC`" vers le répertoire "`MonRepertoire`" sous le nom de "`MaCopie.cpc`"

EXEMPLE 2 :

```
Supprimer/ ..\Repertoire\fichier.exe
```

AUTRES INFORMATIONS :

- Cpcdos supporte les deux URI "\" "/" vous pouvez les mixer entre eux 😉

VOIR AUSSI :

Renommer/ copier/

Créer un dossier ou un arbre de répertoires

COMMANDE :

`Dossier/ [chemin]`	`Folder/ [chemin]`

FONCTIONNALITE :

Cette commande permet de créer un dossier ou un arbre de répertoires.

EXEMPLE SIMPLE :

```
Dossier/ MonDossier
```

Créer un simple dossier nommé "MonDossier" dans le dossier courant.

EXEMPLE AVEC VARIABLE:

```
Fix/ Variable = OS/MonOS/Système/temporaire
dossier/ %Variable%/Toto
```

Créer un dossier (ou l'arbre de répertoires) « `OS/MonOS/Système/temporaire/Toto`»

AUTRES INFORMATIONS :

- Cpcdos supporte les deux URI "\" "/" vous pouvez les mixer entre eux 😉

VOIR AUSSI :

Renommer/ ; copier/ ; Supprimer/ ;

Créer des variables / tableaux

COMMANDE :

```
FIX/ {/s|/q|/touche|/atouche} [NomDeVariable] = [Contenu]
```

FIXer

FONCTIONNALITE :

Cette commande permet de créer en mémoire une variable ou un tableau pouvant contenir des valeurs numériques et caractères ASCII.
Attention à savoir gérer le niveau de publicité des variables. Voir Qu'es ce que le niveau de visibilité ?

PARAMETRES DISPONIBLE :

- **/S** [Nom de variable en mémoire] → Supprime une variable en mémoire.
- **/Q** [Nom de variable] → Pose une question et stocke la réponse utilisateur.
- **/TOUCHE** [Nom de variable] → Récupère la touche en mémoire.
- **/ATOUCHE** [Nom de variable] → Idem, mais attend qu'une touche soit pressée.

- \%
  → Affiche le caractère '%' sans être pris en compte tant que variable.
  Possibilité donc d'être affiché dans le DEBUG.LOG
- \#REFORMAT-VAR **ou** \#VAR-REFORMAT **ou** \#REFORMATER-VAR **ou** \#VAR-REFORMATER
  → Permet de reformater la commande ou le texte si une variable **contient** une autre variable à lire.

- \#NON-VAR ou \#NO-VAR
  → Interdit la recherche de variables. *Garde textuellement **%variable%**.*

- \#NON-FN ou \#NO-FN
  → Interdit l'exécution d'une fonction. Garde textuellement **/F:ma_fonction(…)**

VARIABLE - EXEMPLE :

```
FIX/ Nom = Lussiaud
FIX/ Prenom=Timothee
Txt/ Bonjour %nom% %Prenom% !
FIX/ Prenom = Bernard
Txt/ Bonjour %nom% %Prenom% !
```

CORRESPONDANCE ANGLAISE :

```
FIX/ Name = Lussiaud
FIX/ FirstName=Timothee
Txt/ Hello %Name% %FirstName% !
FIX/ FirstName=Bernard
Txt/ Hello %Name% %FirstName% !
```

APERÇU :

```
 > exe/ test.cpc
Bonjour Lussiaud Timothee !
Bonjour Lussiaud Bernard !
```

VARIABLE QUESTION - EXEMPLE :

```
Txt/ Quel est votre nom ? /#R
Fix/ /Q Nom
Txt/
Txt/ Bonjour %NOM% !
```

/#R : permet de Rester sur la même ligne
Txt/ : Va permettre de passer en dessous de a reponse tapée (Saute une ligne)

APERÇU :

```
 > exe/ test.cpc
Quel est votre nom ? Nathan
Bonjour Nathan !
```

"

REMARQUE : Si vous faîtes ceci :

```
Txt/ Souhaitez-vous quitter ? (O:Oui N:NON) /#R
Fix/ /Q question
Si/ "%question%" == "N" alors:
    …
Fin/ si
```

Vous savez qu'ici si vous souhaitez utiliser « NON » vous devez mettre le « N » en majuscules.
Si vous souhaitez utiliser le minuscule « n » et le « N » en entré de clavier, vous pouvez fixer l'entré au clavier, **tout en majuscules**, comme ça, pas de différenciations.
Le voici :

```
Txt/ Souhaitez-vous quitter ? (O:Oui N:NON) /#R
Fix/ /Q question
Fix/ question = CPC.MAJ(%question%) // Met en majuscules sa propre variable
Si/ "%question%" == "N" alors:
    …
Fin/ si
```

TABLEAU - EXEMPLE :

```
FIX/ Mon_Tableau[0] = Lussiaud
FIX/ Mon_Tableau[1] = Timothee
Txt/ Bonjour %Mon_Tableau[0]% %Mon_Tableau[1]% !
FIX/ Mon_Tableau[1] = Bernard
Txt/ Bonjour %Mon_Tableau[0]% %Mon_Tableau[1]% !
```

1 : Numéro d'index du tableau qui subit une modification
`Mon_Tableau[`*Infinit → Infinit*`]` : Un tableau peut contenir une infinité d'éléments en commençant par quelconque numéro.

APERÇU :

```
 > exe/ test.cpc
Bonjour Lussiaud Timothee !
Bonjour Lussiaud Bernard !
```

GENERER AUTOMATIQUEMENT UN TABLEAU VIDE - EXEMPLE :

```
FIX/ MonTableau[5 a 20]
```

Remarque : Numéro d'index du tableau peut aussi commencer de "0"

GENERER AUTOMATIQUEMENT UN TABLEAU AVEC CONTENU - EXEMPLE :

```
FIX/ MonTableau[2 a 12] = Ceci est repete 10 fois !
```

Vous pouvez également générer un tableau à l'envers

```
FIX/ MonTableau[12 a 2] = Ceci est repete 10 fois !
```

Remarque : Très utile s'il faut stocker des valeurs incré/décré-mentant dans un sens ou l'autre de manière simplifié dans une boucle par exemple.

SUPPRIMER UN TABLEAU ENTIER - EXEMPLE :

```
FIX/ /S MonTableau[]
```

~~Supprime explicitement **MonTableau**[ ]~~

SUPPRIMER UN TABLEAU PRECISEMENT - EXEMPLE :

```
FIX/ /S MonTableau[5 a 20]
```

Supprime **MonTableau**[ ] à partir de 5 jusqu'à 20.
Remarque : Vous pouvez aussi le faire de 20 à 5.

1ER EXEMPLE AVEC UN NIVEAU DE PUBLICITE :

```
CECI FONCTIONNE
CCP/ /FIX.NIVEAU = 2 ou 3 ou 4 ou 5
FIX/ Toto = 123
Txt/ %Toto%
CECI NE FONCTIONNE PAS «Variable introuvable»
CCP/ /FIX.NIVEAU = 2 ou 1
FIX/ Toto = 123
CCP/ /FIX.NIVEAU = 3 ou 4 ou 5
Txt/ %Toto%
```

Le 1er fonctionne !
→ On crée une variable au **niveau 2** et on affiche la variable `%Toto%` toujours au **niveau 2**.

Pourquoi la 2eme ne fonctionne pas ?
→ `Txt/ %Toto%` Etant au **niveau 3**, donc supérieur au **niveau 2**, la variable `%Toto%` n'est visible qu'au **niveau 2** ou **1.** Et n'est donc pas visible aux niveaux **3**, **4**, **5**.

2EME EXEMPLE AVEC UN NIVEAU DE PUBLICITE :

```
CECI FONCTIONNE
CCP/ /FIX.NIVEAU = 4 ou 5
FIX/ Toto = 123
CCP/ /FIX.Niveau = 2
Txt/ %Toto%
CECI NE FONCTIONNE PAS «Variable introuvable»
CCP/ /FIX.NIVEAU = 4 ou 3 ou 2 ou 1
FIX/ Toto = 123
CCP/ /FIX.NIVEAU = 5
Txt/ %Toto%
```

Le premier fonctionne

→ On crée une variable au **niveau 4**, et on affiche `%Toto%` au **niveau 2** (qui est inférieur à 4).

Pourquoi la 2eme ligne ne fonctionne pas ?

→ `Txt/ %Toto%` Etant au **niveau 5**, donc supérieur au **niveau 4**, la variable `%Toto%` n'est visible qu'au **niveau 2** ou **1.** Et n'est donc pas visible aux niveaux **3**, **4**, **5**.

AUTRES INFORMATIONS :

- Les caractères des NOMS de variables autorisés :
  - A B C D E F G H I J K L M O P Q R S T U V W X Y Z
  - a b c d e f g h i j k l m o p q r s t u v w x y z
  - 0 1 2 3 4 5 6 7 8 9
  - _ . - $ / \ ( ) [ ] ~ & # ^ @
- Pour une meilleure notion des niveaux de publicité sur Cpcdos, voir : Qu'es ce que le niveau de visibilité ?

VOIR AUSSI :

`Txt/`

COMMANDE :

```
CCP/ [/Lang {= FR, EN}, /FIX.NIVEAU {= 1, 2, 3, 4, 5}] /Change:[…] …
```

FONCTIONNALITE :

Ces commandes permettent d'accéder aux paramètres du moteur CCP.

PARAMETRES DISPONIBLE :

- **/LANG […]**
  Changer la langue du noyau tout en utilisant la syntaxe CpcdosC+ Française ET OU Anglaise
- **/FIX.Niveau = […]**
  Changer le niveau de publicité des variables et fonctions
- **/Change:[…]**
  Met en pause le thread jusqu'à que la variable indiqué change de contenu.
- **/Pause […]**
  Met en pause le thread pendant un temps définit en millisecondes.
- **/Optimisation {…}**
  Optimise et pré-compile les fichiers CpcdosC+.
- **/Debut_section_critique**
  Bloque tous les threads et interruptions.
- **/Fin_section_critique**
  Débloque tous les threads et interruptions.

CHANGER LA LANGUE - EXEMPLE :

- Langue Anglaise :

```
CCP/ /Lang = EN
```

- Langue Française :

```
CCP/ /Lang = FR
```

CHANGER NIVEAU DE PUBLICITE – EXEMPLE : CORRESPONDANCE ANGLAISE :

Variables, fonctions et autre visible par :

- Affiche le niveau de l'instance en cours

```
CCP/ /FIX.NIVEAU
```

```
CCP/ /set.level
```

- Tout le système *Niveau kernel* FR EN

```
CCP/ /Fix.Niveau = 5
```

```
CCP/ /Set.LEVEL = 5
```

- Tout le système *Niveau Système d'exploitation*

```
CCP/ /Fix.Niveau = 4
```

```
CCP/ /SET.LEVEL = 4
```

- Tout le système *Niveau Utilisateur*

```
CCP/ /Fix.Niveau = 3
```

```
CCP/ /SET.Level = 3
```

- Tout le système *Niveau Application*

```
CCP/ /Fix.Niveau = 2
```

```
CCP/ /SET.level = 2
```

- Tout le système *Niveau Fonctions*

```
CCP/ /FIX.NIVEAU = 1
```

```
CCP/ /set.level = 1
```

→ Qu'es ce qu'un niveau de publicité ?

ATTENDRE LE CHANGEMENT D'UNE VARIABLE

Ceci est très utile si vous voulez **espionner** une variable pour avancer dans votre code.

Exemple avec la variable CPC.SYS.CPU.MEMU *(conseil : à exécuter dans un autre thread !)*

```
Txt/ On attend que la memoire change…
CCP/ /CHANGE:CPC.SYS.CPU.MEMU
Txt/ Ah ! la mémoire utilisee a changee de taille !
```

Un autre exemple avec la variable %TOTO% stocké au niveau 5 (KERNEL), il faut le mettre dans un fichier **Test.cpc** a executer dans un autre thread.

```
CCP/ /FIX.NIVEAU = 5
Fix/ Toto = 123
Txt/ On attend que TOTO change de contenu ...
CCP/ /CHANGE:TOTO
Txt/ Ah ! Toto a ete modifie en %TOTO%
```

Une fois ce fichier exécuté dans un autre thread, faîtes le changement du contenu de la variable à la console, vous verrez réagir le programme Test.cpc en arrière-plan 😉

Suivez l'aperçu :

APERÇU

```
 > exe/ &+ test.cpc
On attend que TOTO change de contenu ...
 > ccp/ /fix.niveau = 5
 > Fix/ Toto = ABC
Ah ! Toto a ete modifie en ABC
```

METTRE EN PAUSE PENDANT UN TEMPS EN MILLISECONDES

Met le thread en cours d'exécution en pause pendant un temps donné.
L'unité est en millisecondes, mais vous pouvez utiliser la virgule pour mettre en pause en microsecondes.
Exemples simple :

```
CCP/ /Pause 5000
```

Fait une pause de 5 secondes.

```
CCP/ /Pause 1500
```

Fait une pause de 1 secondes et 500 millisecondes (1.5secondes).

```
CCP/ /Pause 300
```

Fait une pause de 300 millisecondes (0.3 secondes).

```
CCP/ /Pause 0.5
```

Fait une pause de 500 microsecondes.

```
CCP/ /Pause 0.001
```

Fait une pause de 1 microseconde.
Correspondance anglaise :

```
CCP/ /Pause = 0.001
```

ACTIVER LA PRE-COMPILATION / OPTIMISATION

Ceci permet d'optimiser l'exécution de vos programmes CpcdosC+. Une fois cette option activée, Cpcdos analyse l'ensemble du code et tente une optimisation du code pour réduire le code le plus possible ainsi que la charge du processeur durant l'exécution tout en gagnant de +10% à + 30%. Et puis il référence l'ensemble des sauts de codes, l'emplacement des fonctions tant que pointeurs mémoires afin d'optimiser les petites boucles en gagnant +40% à +80% de performances.
Par défaut, cette option est désactivée.

Afficher si l'optimisation est active « 1 »
Ou désactivé « 0 »

```
CCP/ /Optimisation
```

Correspondance anglaise :

```
CCP/ /Optimization
```

Activer l'optimisation

```
CCP/ /Optimisation = 1
```

```
CCP/ /Optimization = 1
```

Désactiver l'optimisation

```
CCP/ /Optimisation = 0
```

```
CCP/ /Optimization = 0
```

SECTION CRITIQUE

Ceci permet d'exécuter une séquence de lignes de code CpcdosC+ de manière "**prioritaire**" et sans être interrompu par **un thread** coopératif lié au processus parent qui héberge le thread critique et sans être interrompu non plus par **une interruption** du type IRQ, excepté le clavier et la souris.

Exemple :

```
FIX/ Compteur = 0
CCP/ /Debut_section_critique
:BOUCLE:
FIX/ Compteur = /C(%Compteur% + 1)
Txt/ %Compteur%
Si/ "%Compteur%" > "10000" alors:
     Aller/ FIN
Fin/ si
Aller/ Boucle
:FIN:
CCP/ /Fin_section_critique
```

```
FIX/ Counter = 0
CCP/ /Begin_critical_section
:LOOP:
FIX/ Counter = /C(%Counter% + 1)
Txt/ %Counter%
If/ "%Counter%" > "10000" Then:
     Goto/ EXIT
End/ if
Goto/ Loop
:EXIT:
CCP/ /Begin_critical_section
```

Chronométrez le temps, et testez ce code AVEC et SANS cette commande. Vous verrez la différence

Mais retenez que cette fonctionnalité sert principalement quand vous ne voulez pas « d'interférences » entre 2 ou plusieurs threads qui utilisent par exemple la même variable publique. Mais je dois admettre que ça apporte des avantages :-)

AUTRES INFORMATIONS :

VOIR AUSSI :

Sys/

Les conditions

COMMANDE :

```
SI/ [[Valeur] [=, !=,>, !>, <, !<, >=, !>=, <=, !<=] [Valeur]] …
```

SYSteme

FONCTIONNALITE :

Cette commande permet d'exécuter du code, ou pas, si la condition de deux valeurs est vraie ou fausse. Vous pouvez comparer des valeurs ou bien des caractères/textes.
Vous pouvez faire une condition en :

- Mono-ligne → Une seul ligne et peut contenir SINON: sur la même ligne.
- Multi-ligne → Plusieurs lignes et peut contenir SINON: sur une autre ligne.
- Multi-conditionnelle → Peut contenir une infinité de conditions dans une autre condition etc…

PARAMETRES DISPONIBLE :

- **= ou ==** → Egale à
- **!= ou N=** → N'est pas égale à
- **>** → Plus grand que
- **!> ou N>** → N'est pas plus grand que
- **<** → Plus petit que
- **!< ou N<** → N'est pas plus petit que
- **>=** → Plus grand ou égale à
- **!>= ou N> =** → N'est pas plus grand ou égale à
- **~=** → Contient
- **!~= ou N~=** → Ne contient pas

EXEMPLE 1 MONO-LIGNE :

```
SI/ "8" = "8" alors:Txt/
```

Résultat : « Vrais !»

EXEMPLE 2 MONO-LIGNE :

```
SI/ "8" < "4" alors:Txt/ Vrai ! Sinon:Txt/
```

Résultat : « Faux..»

EXEMPLE 3 MONO-LIGNE AVEC 2 VARIABLES :

```
Fix/ Valeur1 = 54
Fix/ Valeur2 = 4
SI/ "%Valeur1%" > "%Valeur2%" alors:Txt/ Vrais ! Sinon:Txt/ Faux..
```

Résultat : « Vrais !»

EXEMPLE 4 MULTI-LIGNES :

```
Fix/ Valeur = 14
SI/ "6" !< "%Valeur%" alors:
     Txt/ Vrai !
     Txt/ Les valeurs sont bonnes
Sinon:
     Txt/ Faux...
     Txt/ Ah… Dommage !
Fin/ si
```

Résultat : « Vrai ! Les valeurs sont bonnes »

EXEMPLE 5 MULTI-LIGNES :

```
SI/ "10" >= "5" alors:
     Txt/ Vrai !
Fin/ si
```

Résultat : « Vrai !»

EXEMPLE 6 MULTI-LIGNES / MULTI-CONDITIONNELLE :

```
SI/ "5" = "5" alors:
     Txt/ Vrai !
     Si/ "10" > "5" alors:
          Txt/ Encore vrai !
     Fin/ si
Fin/ si
```

Résultat : « Vrai ! Encore vrais !»

EXEMPLE 7 MULTI-LIGNES / MULTI-CONDITIONNELLE :

```
SI/ "2" >= "8" alors:
     Txt/ Vrai !
     Si/ "10" > "5" alors:
          Txt/ Encore vrai !
     Fin/ si
     Txt/ C'est bien !
Sinon:
     Txt/ Faux..
     Si/ "4" > "5" alors:
          Txt/ Hm.. Vrai !
     Sinon:
          Txt/ Encore faux !
          Txt/ Tu feras mieux !
     Fin/ si
     Txt/ Bye !
Fin/ si
```

Résultat : « Faux.. Encore faux ! Tu feras mieux ! Bye ! »

EXEMPLE 8 CONDITION SUR DES CARACTERES:

```
SI/ "TOTO" = "toto" alors:
     Txt/ Vrai !
Sinon:
     Txt/ Faux..
Fin/ si
```

*Résultat : « Faux.. » Par ce que **TOTO** et **toto** sont différents*

EXEMPLE 9

```
SI/ "Je m'appelle Jean michel" ~= "an" alors:
     Txt/ Vrais !
Sinon:
     Txt/ Faux..
Fin/ si
```

*Résultat : « Vrai.. » Par ce que « **`Je m'appelle Jean michel`** » contient « an »*

Correspondance anglaise

```
IF/ "TOTO" = "toto" then:
      Txt/ True !
Else:
      Txt/ False
End/ if
```

AUTRES INFORMATIONS :

- ATTENTION : La plupart des développeurs oublient les "" **oubliez-les pas !**
- Tester une condition avec des **caractères** n'est possible seulement qu'avec =, ==, !=, N=, ~=, !~= N~=
- Tester une condition avec des **valeurs numériques**… Vous pouvez tout utiliser !

Lire dans un fichier

COMMANDE :

```
OUVRIR/ [Chemin d'accès au fichier source] {/LIGNE:{No} /TEXTE:[TE..
```

FONCTIONNALITE :

Cette commande permet d'ouvrir et de récupérer TOUT le contenu d'un fichier en respectant l'ordonnancement et les priorités des threads selon la charge appliquée sur le disque en lecture.

PARAMETRES DISPONIBLE :

- **/LIGNE:**[No Ligne] Permet de récupérer le texte d'une ligne
- **/TEXTE:**[Texte] Permet de récupérer la ligne qui contient la séquence de texte indiquée.
- ~~/BIN Permet la lecture en mode BINaire~~ NON DISPONIBLE

EXEMPLE 1 – AFFICHAGE DIRECT SUR L'ECRAN :

Affiche tout le contenu de MonFichier.txt sur l'écran

```
OUVRIR/ MonFichier.txt
```

```
OPEN/ MyFile.txt
```

EXEMPLE 2 – AFFICHAGE D'UNE SEULE LIGNE

Affiche le contenu d'une ligne d'un fichier

```
OUVRIR/ MonFichier.txt /Ligne:3
```

```
OPEN/ MyFile.txt /Line:3
```

Affiche le contenu d'une ligne d'un fichier qui contient « Toto »

```
OUVRIR/ MonFichier.txt /Text:Toto
```

```
OPEN/ MyFile.txt /Text:Toto
```

EXEMPLE 3 – STOCKAGE DANS UNE VARIABLE :

Stocke tout le contenu du fichier MonFichier.txt dans la variable **ContenuFichier**

```
Fix/ File = MonFichier.txt
@#ContenuFichier OUVRIR/ %File%
Txt/ Contenu du fichier %File% :
Txt/ %ContenuFichier%
Txt/ FIN !
```

AUTRES INFORMATIONS :

- Pour visualiser le contenu d'un gros fichier qui dépasserai la taille de l'écran, vous pouvez utiliser le paramètre \#PAUSE en fin de ligne afin de faire défiler ligne par ligne en pressant une touche.
  → La touche ENTRER permet d'annuler le défilement.
  → La touche ECHAP permet de stopper le défilement.

VOIR AUSSI :

Ecrire/ ;

Ecrire dans un fichier

COMMANDE :

```
ECRIRE/ {/BIN et ou /APP} [Chemin d'accès au fichier source]
```

FONCTIONNALITE :

Cette commande permet d'écrire directement dans un fichier spécifié.

PARAMETRES DISPONIBLE :

- *Par défaut*→ Ecrire normale avec le retour chariot CRLR en fin de ligne et écrase si existe.
- /BIN → Permet l'écriture en mode BINaire.
- /APP →Permet l'écriture en mode APPending (à la suite, sans écraser le fichier).
- /BINAPP
  ou
  /APPBIN → Permet d'écrire en BINaire **et** en mode APPending.

EXEMPLE 1 – ECRITURE SIMPLE :

Crée ou écrase le fichier MonFichier.txt si existe et écrit directement « Coucou » puis ferme le fichier.

```
ECRIRE/ MonFichier.txt = Coucou
```

```
WRITE/ MyFile.txt = Hello
```

Ajoute automatiquement le CRLF en fin de ligne.

EXEMPLE 2 – ECRITURE EN MODE APPENDING

Crée ou écrit les données à la suite sans écraser le fichier si existe.

```
ECRIRE/ /APP MonFichier.txt = Ca va ?
```

Ajoute automatiquement le CRLF en fin de ligne.

CONTENU DU FICHIER **MONFICHIER.TXT**

Coucou
Ca va ?

EXEMPLE 2 – ECRITURE BINAIRE :

Crée ou écrase le fichier MonFichier.txt si existe et écrit directement « Coucou » puis ferme le fichier.

```
ECRIRE/ /BIN MonFichier.txt = Coucou
```

```
WRITE/ /BIN MyFile.txt = Hello
```

EXEMPLE 2 – ECRITURE EN BINAIRE EN APPENDING

Crée ou écrit les données à la suite sans écraser le fichier si existe.

```
ECRIRE/ /BINAPP MonFichier.txt = Ca va ? %CPC.CRLF%Bye!
```

CONTENU DU FICHIER **MONFICHIER.TXT**

CoucouCa va ?
Bye!

AUTRES INFORMATIONS :

- Par défaut, à la fin de chaque ligne, Cpcdos ajoute les caractères ASCII 13 et 10 (Retour chariot CRLF), utile pour stocker du texte ou des lignes de commandes.
- Le paramètre /BIN fait la même chose mais n'**ajoute pas** les caractères de retour chariot. Utile pour écrire des octets dans un fichier.
- Pour ajouter le CRLF "*la touche entré*", il faut utiliser la variable %CPC.CRLF%, ou bien %CPC.CR%, %CPC.LF% ou même %CPC.LFCR%.

VOIR AUSSI : Lire/ ; Supprimer/ ; Renommer/ ; Copier/ ;

Configurer paramètres systèmes

COMMANDE :

```
SYS/ [/mem{…}, /Police [/List,NomDePolice] /Debug {/CPinticore, …}…]
```

***SYS**teme*

FONCTIONNALITE :

Cette commande permet d'obtenir ou changer et paramétrer des informations du système.

PARAMETRES DISPONIBLE :

- /Mem {…} → Obtenir la mémoire libre
- /Processus {…} → Créer un nouveau processus vide.
- /Arreter → Arrêter le système.
- /debug → Debugger le système
- /Redemarrer → Redémarrer le système.
- /Veille → Mettre en veille le système.
- /MemU {…} → Obtenir la mémoire utilisée
- /Police […] → Lister ou changer la police d'écriture de la console
- /Debug {…} → Afficher l'état ou désactiver/activer le mode débogage
- /Ecran […] → Gérer l'affichage
- /WRP[…] → Exécute une instance du wrapper avec un numéro de fonction.
- /Bitmap {…} → Gérer l'allocation bitmaps de l'interface GUI.
- /OS {…} → Paramètres des systèmes d'exploitation
- /ListVar […] → Lister toutes les variables en mémoire de niveau 1, 2, 3, 4 et 5.

MEMOIRE LIBRE - EXEMPLE : CORRESPONDANCE ANGLAISE :

Affiche la mémoire libre en Octets

```
SYS/ /MEM
```

```
SYS/ /MEM
```

Affiche la mémoire libre en Méga-Octets

```
SYS/ /MEM mo
```

```
SYS/ /Mem mb
```

Affiche la mémoire libre en Giga-Octets

```
SYS/ /MEM Go
```

```
SYS/ /Mem Gb
```

Affiche la mémoire libre en Tera-Octets

```
SYS/ /MEM To
```

```
SYS/ /Mem Tb
```

Affiche la mémoire utilisée en Octets

```
SYS/ /MEMU
```

```
SYS/ /MEMU
```

Affiche la mémoire utilisée en Méga-Octets

```
SYS/ /MEMU mo
```

```
SYS/ /MemU mb
```

Affiche la mémoire utilisée en Giga-Octets

```
SYS/ /MEMU Go
```

```
SYS/ /MemU Gb
```

Affiche la mémoire utilisée en Tera-Octets

```
SYS/ /MEMU To
```

```
SYS/ /MemU Tb
```

NOUVEAU PROCESSUS - EXEMPLE : CORRESPONDANCE ANGLAISE :

Créer un nouveau processus vide nommé "Toto"

```
SYS/ /PROCESSUS Toto
```

```
SYS/ /PROCESS Toto
```

Cette commande génère un numéro de PID à récupérer via "***@#variable***" comme ceci :

`@#NumeroPID SYS/ /PROCESSUS Toto`	`@#PIDNumber SYS/ /PROCESS Toto`

Vous pouvez fermer le processus avec la commande **FERMER/**

`Fermer/ /PID:%NumeroPID%`	`Close/ /PID:%PIDNumber%`

GESTION BIOS APM (ADVANCED POWER MANAGEMENT) CORRESPONDANCE ANGLAISE :

Arrêter le système

`SYS/ /ARRETER`	`SYS/ /SHUTDOWN`

Redémarrer le système

`SYS/ /REDEMARRER`	`SYS/ /REBOOT SYS/ /RESTART`

Arrêter (Proprement) le noyau uniquement

`SYS/ /STOPK`	`SYS/ /STOPK`

POLICE D'ECRITURE - EXEMPLE : CORRESPONDANCE ANGLAISE :

~~Activer la gestion des polices~~

`SYS/ /POLICE /ACTIVER`	`SYS/ /FONT /ENABLE`

~~Désactiver la gestion des polices~~

`SYS/ /Police /Desactiver`	`SYS/ /FONT /Disable`

~~Lister les polices installées~~

`SYS/ /Police /Liste`	`SYS/ /Font /List`

~~Charger et utiliser une police installée (ex :Arial)~~

`SYS/ /Police Arial`	`SYS/ /Font Arial`

~~→ Humour : Si *débogage Cpcdos* actif lors du chargement d'une police, attention aux épileptiques !~~

DEBOGAGE DU SYSTEME - EXEMPLE :

Afficher l'état du débogueur du noyau Cpcdos

```
SYS/ /DEBUG
```

Afficher l'état du débogueur CPinti Core

```
SYS/ /DEBUG /CPintiCore
```

~~Afficher l'état du débogueur TCP/IP~~ ~~Correspondance anglaise~~

`SYS/ /DEBUG /Serveur:TCP`	`SYS/ /Debug /Server:TCP`

Activer le débogueur du noyau Cpcdos

```
SYS/ /DEBUG = 1
```

Activer le débogueur du noyau Cpcdos + **inscription** dans le fichier DEBUG.LOG

```
SYS/ /DEBUG = 2
```

Activer le débogueur CPinti Core

```
SYS/ /DEBUG /CPintiCore = 1
```

Activer le débogueur CPinti Core + **inscription** dans le fichier DEBUG.LOG

```
SYS/ /DEBUG /CPintiCore = 2
```

~~Activer le débogueur Serveur port 23~~ ~~Correspondance anglaise~~

`SYS/ /DEBUG /Serveur:23 = 1`	`SYS/ /Debug /Server:23 = 1`

Afficher le menu d'informations de la console du noyau (1 par défaut)

```
SYS/ /DEBUG /Menu = 1
```

Activer le débogueur via le port com 1 (Lecture seule)

```
SYS/ /DEBUG /COM1 = 1
```

Activer le débogueur via le port com 1 (Lecture et écriture)

```
SYS/ /DEBUG /COM1 = 2
```

Désactiver le débogueur COM

```
SYS/ /DEBUG /COM1 = 0
```

Il y a **COM1**, **COM2**, **COM3** et **COM4** de disponible. Il faut qu'il soit également disponible physiquement.

→ Et puis pour **désactiver** l'un(s) de ces débogueurs, assignez-lui tout simplement la valeur **0**.

GERER L'AFFICHAGE - EXEMPLE :

Afficher la liste des résolutions compatibles en 16 bits

`SYS/ /Ecran /Liste 16`	`SYS/ /Screen /list 16`

Afficher la liste des résolutions compatibles en 24 bits

`SYS/ /Ecran /Liste 24`	`SYS/ /Screen /list 24`

Afficher la liste des résolutions compatibles en 32 bits

`SYS/ /Ecran /Liste 32`	`SYS/ /Screen /list 32`

Tester une résolution Exemple avec 1024x768 et 24 bits

`SYS/ /Ecran /Test 1024x768x24`	`SYS/ /Screen /test 1024x768x24`

Affiche **« 1 » si la résolution est compatible** avec votre carte graphique. Et **« 2 » si elle n'est pas.**

Modifier résolution d'écran GRAPHIQUE Exemple avec 1024x768

`SYS/ /Ecran 1024x768x24`	`SYS/ /Screen 1024x768x24`

TEMPORAIRE : Utilisez **FIX/ SCR_BIT = 24 pour modifier le nombre de couleurs**.

Modifier résolution d'écran CONSOLE Exemple avec 20 (1024x768)

`SYS/ /Ecran 20`	`SYS/ /Screen 20`

Liste des résolution **CONSOLE** disponible :

1	320x200 (CGA)
2	640x200 (CGA)
7	320x200 (EGA)
8	640x200 (EGA)
9	640x350 (EGA)
11	640x480 (VGA)
12	640x480 (VGA)
13	320x200 (MCGA)
14	320x240 (SVGA)
15	400x300 (SVGA)
16	512x384 (SVGA)

17	640x400 (SVGA)
18	640x480 (SVGA)
19	800x600 (SVGA)
20	1024x768 (SVGA)
21	1280x1024 (SVGA)

Récupérer le pointeur vidéo (Adresse mémoire)

Sous forme décimale

```
SYS/ /Ecran /PTR
```

```
SYS/ /Screen /PTR
```

Sous forme hexadécimale

```
SYS/ /Ecran /HEXAPTR
```

```
SYS/ /Screen /HEXAPTR
```

PEUT SERVIR POUR LES MOTEURS/MODULES DE GESTION GRAPHIQUES COMME OPENGL, SDL …

Récupérer le pointeur du bitmap depuis le numéro d'un Handle

```
SYS/ /Bitmaps /Obtenir /PtrByHandle
```

```
SYS/ /Bitmaps /Get /PtrByHandle
```

Récupérer le bitmap ID depuis le numéro d'un handle

```
SYS/ /Bitmaps /Obtenir /IdByHandle
```

```
SYS/ /Bitmaps /Get /IdByHandle
```

EXECUTER UNE FONCTION WRAPPEE - EXEMPLE :

Réservé aux développeurs du SDK Cpcdos.

Ce paramètre permet d'exécuter une fonction C/C++ précédemment **écrite ET compilé** DANS le **CONTRIB** de Cpcdos.

Exemple :

Exécuter votre fonction SDK numéro « exemple : **1024** »

```
SYS/ /WRP 1024
```

Pour votre functionID «**1024**» complétez les arguments :

« const char Arg_1, int Arg_2, void* Arg_3, void * Arg_4, void * Arg_5 »*

Et pour les compléter depuis le CpcdosC+

```
fix/ sys.wrp(5).1 = 123     Char
fix/ sys.wrp(5).2 = 456     Nombre Entier
fix/ sys.wrp(5).3 = Toto    tout format
fix/ sys.wrp(5).4 = Tata    tout format
fix/ sys.wrp(5).5 = 0x123   tout format

SYS/ /WRP 1024              Executer la FunctionID 1024
```

GESTIONNAIRE DE BITMAPS - EXEMPLE : **CORRESPONDANCE ANGLAISE :**

Cette fonctionnalité permet au système d'allouer un bloc de mémoire gigantesque dans une zone isolée du reste pour ne pas interférer les éléments alloués en cas de buffer-overflow. Et permet également de référencer tous les bitmaps à un numéro de handle parent spécifique à une instance. Si cette instance vient à être détruite (Bouton, Fenêtre, icône, Checkbox, picturebox…) et bien les bitmaps associés sont détruites à leur tour. Si le bitmap n'est pas détruit pour X raisons, le garbage collector effectue le nettoyage en supprimant tous les bitmaps ayant un numéro de handle qui n'existe plus.

Voir le nombre de bitmaps alloués

```
SYS/ /Bitmaps
```

```
SYS/ /Bitmaps
```

Liste des bitmaps alloué (Détaillé)

```
SYS/ /Bitmaps /Liste
```

```
SYS/ /Bitmaps /List
```

Taille totale de tous les bitmaps alloués

```
SYS/ /Bitmaps /Taille
```

```
SYS/ /Bitmaps /Size
```

Forcer le rechargement des bitmaps (Utile en cas de changement de bits de couleurs)

```
SYS/ /Bitmaps /Recharger
```

```
SYS/ /Bitmaps /Reload
```

Lancer le ramasse-miette (Garbage collector)

```
SYS/ /Bitmaps /GC
```

```
SYS/ /Bitmaps /GC
```

Lancer le ramasse-miette (Garbage collector)

```
SYS/ /Bitmaps /GarbageCollector
```

```
SYS/ /Bitmaps /GarbageCollector
```

PARAMETRAGES DE L'OS EXEMPLE :

Ce paramètre permet de changer d'OS en exécution, d'afficher des informations, ou de configurer le ou les OS en cours ou installés

Exemples :

Mettre à jour la liste des OS installé via OS.LST

```
SYS/ /OS /UPDATE
```

Afficher la liste des OS installés

```
SYS/ /OS /LISTE
```

```
SYS/ /OS /LIST
```

Afficher le nombre d'OS installés

```
SYS/ /OS /NB
```

Vous pourrez également tout faire en une seule ligne

```
SYS/ /OS /UPDATE /LISTE /NB
```

```
SYS/ /OS /UPDATE /LIST /NB
```

Switcher l'instance courant sur un autre OS en cours d'exécution (Exemple avec CraftyOS)

```
SYS/ /OS:CraftyOS
```

Switcher l'instance courant sur l'interface blank du noyau (Sans OS → par défaut au démarrage)

```
SYS/ /OS:SansOS
```

```
SYS/ /OS:WithoutOS
```

LISTER LES VARIABLES EN MEMOIRE :

Purement réservé aux développeurs, ce paramètre permet de lister toutes les variables en précisant le niveau de publicité d'où il est enregistré.

Exemple :

Lister les variables de niveau 4 (Variables de l'OS)

```
SYS/ /Listvar 4
```

AUTRES INFORMATIONS :

VOIR AUSSI :

Ccp/ ;

Fermer un objet graphique, processus ou un thread

COMMANDE :

```
Fermer/ {/PID:[N°PID]} {/TID:[N°PID} {/Handle:[N°handle} NomObjet
```

FONCTIONNALITE :

Cette commande permet de fermer toutes sortes d'instances !
Vous pouvez fermer

- Processus, et tous ses threads associés !
- Threads
- Handle (associé à un objet graphique ou instance d'objet)
- Objet GUI.

PARAMETRES DISPONIBLE :

- Nomobjet → Ferme l'objet graphique se trouvant dans le même processus que cette commande
- /PID:[...] → Fermer un processus et éventuels threads associés.
- /TID:[...] → Fermer un thread.
- /handle:[...] → Fermer un objet graphique ou instance d'objet via ce numéro généré.
- /CleID[...]
  /KeyID[...] → IDEM mais depuis un numéro de Key ID (ou Clé ID)

EXEMPLE 1 – FERMER UN PROCESSUS (EX: 123) :

```
Fermer/ /PID:123
```

```
Close/ /PID:123
```

EXEMPLE 2 – FERMER UN THREAD (EX: 456) :

```
Fermer/ /TID:456
```

```
Close/ /TID:456
```

EXEMPLE 3 – FERMER OBJET DEPUIS SON HANDLE (EX: 789) :

```
Fermer/ /handle:456
```

```
Close/ /handle:789
```

Pour la partie graphique, le numéro de handle est généré via la commande CREER/. Il est dont impératif de le récupérer via '@#' ou de le mémoriser.

EXEMPLE 4 – FERMER OBJET GUI DEPUIS SON NOM (EX: 789) :

```
Fermer/ MonBouton
```

```
Close/ MyButton
```

IMPORTANT : Il est impératif que cette commande soit exécutée depuis **LE MEME PROCESSUS** où est hébergé "MonBouton" Le nom des objets sont inconnus depuis un autre PID.

Cpcdos sécurise vos objets, les noms d'objets sont "invisibles" depuis UN AUTRE PROCESSUS. Il est donc important d'exécuter cette commande **avec le même PID que l'objet graphique**. Autrement... Prout ! Autrement il faut connaître son numéro de handle. Plus compliqué, mais possible !

AUTRES INFORMATIONS :

-

VOIR AUSSI :

SYS/ ; ccp/

Lancer le noyau Cpcdos avec des arguments spécifiques

Ceci est utile pour exécuter des paramètres, des commandes ou des programmes dès le lancement du noyau Cpcdos.

PARAMETRES DISPONIBLES :

- /DOSBOX → Mode dosbox (Pilote souris interne, pas de support réseau).
- /NONET /SANSRESEAU /NONETWORK → Sans support réseau.
- /NODBG ou /SANSDBG → Masquer le debug de cpcdos ET cpinticore au démarrage.
- /CCP:"*Commande CpcdosC+*" → Exécuter une commande CpcdosC+ **après** l'initialisation.
- /NOTELNET ou /NONTELNET → Ne pas démarrer le serveur TELNET au démarrage.
- /SRVCCP → Démarrer le serveur Shell CpcdosC+ au lieu du TELNET.
- /NOGUI ou /NONGUI → Rester en mode console. (Mode debug)

EXEMPLES SOUS DOS FREEDOS/MS-DOS :

« KRNL32.BAT» ou « DPMILD32 NOYAU.EXE »

Mode dosbox :

```
KRNL32.BAT /DOSBOX
```

Mode debug cpinticore :

```
KRNL32.BAT /NOGUI
```

Exécuter une commande après le chargement de Cpcdos :

```
KRNL32.BAT /CCP:"TXT/ Coucou !"
```

Créer un serveur TCP après le chargement de Cpcdos :

```
KRNL32.BAT /CCP: "Serveur/ /tcp:1234"
```

Ou se connecter avec Client/ ….
Bref, laissez libre votre imagination ! ;)

Vous pouvez également modifier le fichier KRNL32.BAT pour placer vos paramètres ! ☺

Faire des calculs arithmétiques

COMMANDE :

```
/C([chiffre [*,/,+,-, (, )] chiffre [*,/,+,-, (, )] chiffre …  ])
```

Calculer

FONCTIONNALITE :

Ce paramètre compatible avec toutes les commandes CpcdosC+ permet de remplacer ce « /C(…) » par le résultat de votre calcul situé entre les deux parenthèses « (» et «) ».

Vous pouvez écrire des équations imbriquées dans des parenthèses comme sur une calculatrice scientifique. La vitesse de traitement est très optimisée.

PARAMETRES DISPONIBLE :

- * → Effectue une multiplication entre deux chiffres.
- / → Effectue une division entre deux chiffres.
- + → Effectue une addition entre deux chiffres.
- - → Effectue une soustraction entre deux chiffres. Ou indique une valeur négative.
- ( ) → Imbriquer des sous-opérations dans des parenthèses.

EXEMPLE 1:

```
Txt/ /C(5 + 5 * 2)
```

Résultat : « 15 ».

EXEMPLE 2 – EN STOCKANT LE RESULTAT DANS UNE VARIABLE :

```
Fix/ Resultat = /c(5 + (2.35 / 55) * (8*8))
Txt/ Voici le resultat : %Resultat%
```

Résultat : « Voici le résultat : 7.734545454545454 ».

Ou *(en stockant le résultat d'une commande dans une variable)*

```
@#Resultat txt/ /c(5 + (2.35 / 55) * (8*8))
Txt/ Voici le resultat : %Resultat%
```

Résultat : « Voici le résultat : 7.734545454545454 ».

EXEMPLE 3 (EFFECTUE LE CALCUL DANS UN NOUVEAU THREAD DE PRIORITE MAXIMALE):

```
CMD/ /Thread[MAX]FIX/ Resultat = /c(5 + (2.35 / 55) * (8*8))
Txt/ %Resultat%
```

Résultat « 7.734545454545454 »

AUTRES INFORMATIONS :

Un bug est présent dans le noyau, si vous soustrayez un nombre négatif, le résultat est faux.

VOIR AUSSI :

NOTION ET UTILISATION DE FONCTION, ARGUMENTS ET RETOURS.

COMMANDES :

```
FONCTION/ [Nom de la fonction]({Argument, argument, argument….})
```

Fonction

FONCTIONNALITE :

Cette commande permet de créer une fonction du type CpcdosC+.

PARAMETRES DISPONIBLE :

Exécuter une fonction (Possibilité de l'utiliser sans commandes ! Attention au retour)

```
/F:[Nom de la fonction]({Argument, argument, argument….})
```

Exemples à faire dans un fichier .CPC !

EXEMPLE 1 – SIMPLE :

```
/F:Ma_Fonction()

Fonction/ Ma_fonction()
     Txt/ Hello !
Fin/ fonction
```

Résultat : « Hello ! ».

EXEMPLE 2 - AVEC UN ARGUMENTS :

```
/F:Ma_Fonction(Hello)

Fonction/ Ma_fonction(MaVariable)
     Txt/ %MaVariable% !!!
Fin/ fonction
```

Résultat : « Hello !!! ».

EXEMPLE 3 - AVEC PLUSIEURS ARGUMENTS :

```
/F:Ma_Fonction(Hello, 20, Manon)

Fonction/ Ma_fonction(Texte, age, nom)
     Txt/ %Texte%, tu t'appelles %nom% et tu as %age% ans.
Fin/ fonction
```

Résultat « Hello, tu t'appelles Manon et tu as 20 ans. »

RETOUR DE FONCTION - EXEMPLE :

```
Retour/ {Texte et/ou valeur et/ou variable}
```

```
Return/
```

Cette commande à utiliser uniquement dans une fonction CpcdosC+, permet de retourner une valeur et/ou variable et/ou des caractères.

EXEMPLE 4 - AVEC UN RETOUR :

```
Txt/ Ma fonction retourne : /F:Ma_Fonction()

Fonction/ Ma_fonction()
     Retour/ Salut ! :)
Fin/ fonction
```

Résultat « Ma fonction retourne : Salut ! :) »

EXEMPLE 5 – PLUS COMPLEXE AVEC PLUSIEURS ARGUMENTS EN UTILISANT UNE VARIABLE ET UN RETOUR :

```
Fix/ MaValeur = 6
Txt/ Resultat de calcul %MaValeur% + 5 = /F:Calculer(%MaValeur%, 5)

Fonction/ Calculer(Valeur_A, Valeur_B)
     Fix/ Resultat = /C(%Valeur_A% + %Valeur_B%)
     Retour/ %Resultat%
Fin/ fonction
```

Résultat « Resultat de calcul 6 + 5 = 11 »

EXEMPLE 6 – IDEM MAIS EN TRANSMETTANT **DIRECTEMENT** LE RESULTAT DANS LE RETOUR :

```
Fix/ MaValeur = 6
Txt/ Resultat de calcul %MaValeur% + 5 = /F:Calculer(%MaValeur%, 5)

Fonction/ Calculer(Valeur_A, Valeur_B)
     Retour/ /C(%Valeur_A% + %Valeur_B%)
Fin/ fonction
```

Résultat « Resultat de calcul 6 + 5 = 11 »

DECLARER UNE FONCTION EXTERNE :

Cette commande à utiliser uniquement là où se trouve vos fonctions (les définition) permet d'utiliser vos fonctions HORS du fichier .CPC. En gros, ceci permet déclarer de manière « publique / externe » selon un niveau, votre fonction CpcdosC. Votre fonction sera utilisable depuis l'extérieur de cette source.

```
Declarer/ [Nom fonction]({Arguments}) : Niveau([2-5])
```

```
Declare/ [Nom fonction]({Arguments}) : Level([2-5])
```

Il est donc impératif que les lignes **declarer/** soient exécutés, ET exécuté dans le même fichier au moins 1 seule fois.

Exemple 1 – Simple (Niveau 5 : Visibilite niveau KERNEL)

Fichier **Fonction.CPC**

```
Declarer/ Ma_fonction() : Niveau(5)  ← Memorise et associe « Ma_Fonction() » à « Fonction.cpc » au niveau 5.

Fonction/ Ma_fonction()
     Retour/ Salut ! :)
Fin/ fonction
```

Et… depuis un autre fichier ou depuis votre console :

```
EXE/ & Fonction.CPC   ← importer en mémoire, les fonctions déclarées.

/F:Ma_Fonction()      ← Permet d'exécuter «Ma_Fonction()» déclarée en mémoire.
```

Résultat « Salut ! :) »

Exemple 2 – Avec arguments et retour

Fichier **Fonction.CPC**

```
Declarer/ Additionner() : Niveau(4)
Declarer/ Multiplier() : Niveau(4)

Fonction/ multiplier (Valeur1, Valeur2)
     Retour/ /C(%Valeur1% * %valeur2%)
Fin/ fonction

Fonction/ additionner ()
     Retour/ /C(%Valeur1% + %valeur2%)
Fin/ fonction
```

Dans un autre fichier ou depuis votre console :

```
EXE/ & Fonction.CPC     ← Permet d'importer en mémoire, les 2 fonctions déclarées.

TXT/ Addition de 3 + 14 = /F:Additionner(3, 14)

TXT/ Multiplication de 3 x 2 = /F:Multiplier(3, 2)
```

Résultat
« Addition de 3 + 14 = 17
Multiplication de 3 x 2 = 6 »

Autres informations :

- Quand une fonction est appelée, Cpcdos va créer un nouveau thread indépendant afin que toutes les ressources créent depuis une fonction, soient détruites à la fin de son exécution.
- Il faut obligatoirement utiliser la commande **DECLARE/** dans le fichier où se trouve les fonctions source, et puis de d'exécuter ce dernier via **EXE/** au moins 1 fois afin d'importer les fonctions en mémoire et de pouvoir les exécuter ultérieurement.
- **L'ordre** des déclarations, de la **position** des fonctions, **des majuscules ou minuscules** dans le fichier n'a **aucune importance** ! Peut être déclaré au début tant qu'à la fin du fichier.
  Pour que ça soit propre, il est recommandé de declarer au DEBUT du fichier.

Exécuter une commande, ou une commande dans un nouveau thread ou processus

COMMANDE :

```
CMD/ {/Thread{[MAX,STD,MIN]}{/PID:[NumeroPID}]} [Commande CpcdosC+]
```

CoMmanDe

FONCTIONNALITE :

Cette commande permet d'exécuter des commandes CpcdosC+. Il peut aussi servir à exécuter des commandes CpcdosC+ qui sont dans des variables.
L'avantage de ce dernier, c'est que le développeur peut exécuter la commande dans un nouveau thread + choisir sa priorité d'exécution [Maximale, standard, minimale] afin de pas bloquer le thread courant.
Initialement, Cpcdos utilise « ses propres procédés » pour répartir ses tâches (Réseau, I/O, GUI etc…)

PARAMETRES DISPONIBLE :

- /THREAD → Exécuter la commande dans un nouveau thread standard
- /THREAD[MIN] → Exécuter la commande dans un nouveau thread priorité minimale
- /THREAD[MI+] → Exécuter la commande dans un nouveau thread priorité minimale et +
- /THREAD[STD] → Exécuter la commande dans un nouveau thread standard
- /THREAD[ST+] → Exécuter la commande dans un nouveau thread standard et plus
- /THREAD[MAX] → Exécuter la commande dans un nouveau thread priorité maximale
- /PID:{N° PID} → Exécuter la commande dans un nouveau thread DEPUIS un autre processus

EXEMPLES :

Exécuter «txt/ coucou» dans le thread et processus courant.

```
CMD/ txt/ Coucou
  ou
Fix/ commande = txt/ coucou
cmd/ %commande%
```

Exécuter «MonPrograme.cpc» dans un nouveau thread

```
CMD/ /THREAD exe/ MonProgramme.cpc
```

Exécuter «MonPrograme.cpc» dans un nouveau thread via une variable (Priorité MAXIMALE)

```
Fix/ LigneDeCommande = exe/ MonProgramme.cpc
CMD/ /THREAD[MAX] %LigneDeCommande%
```

Exécuter «MonPrograme.cpc» dans un nouveau thread depuis un autre processus existant

```
CMD/ /PID:1234 exe/ MonProgramme.CPC
```

Exemple plus complet :

```
@#NumeroDePID SYS/ /Processus MonPROCESS
CMD/ /PID:%NumeroDePID% exe/ MonProgramme.cpc
```

Bien évidement à la place de "**EXE/ MonProgramme.cpc**" vous pouvez utiliser toutes les commandes !

Mais si à la suite vous faîtes ceci :

```
FERMER/ /PID:%NuméroDePID%
```

Ceci entrainera la fermeture du processus crée et de tous les threads associés dont "MonProgramme.cpc"

VOIR AUSSI :

Ccp/ ; sys/

Tester une machine sur le réseau

COMMANDE :

```
PING/ [Adresse IP ou nom du serveur]
```

FONCTIONNALITE :

Cette commande permet de tester l'existence d'une machine sur le réseau
Il envoie une requête ICMP personnalisable et la machine distante doit la renvoyer !

PARAMETRES DISPONIBLE :

EXEMPLE :

Ping chez Google

```
Ping/ www.google.fr
```

Ping chez facebook.com

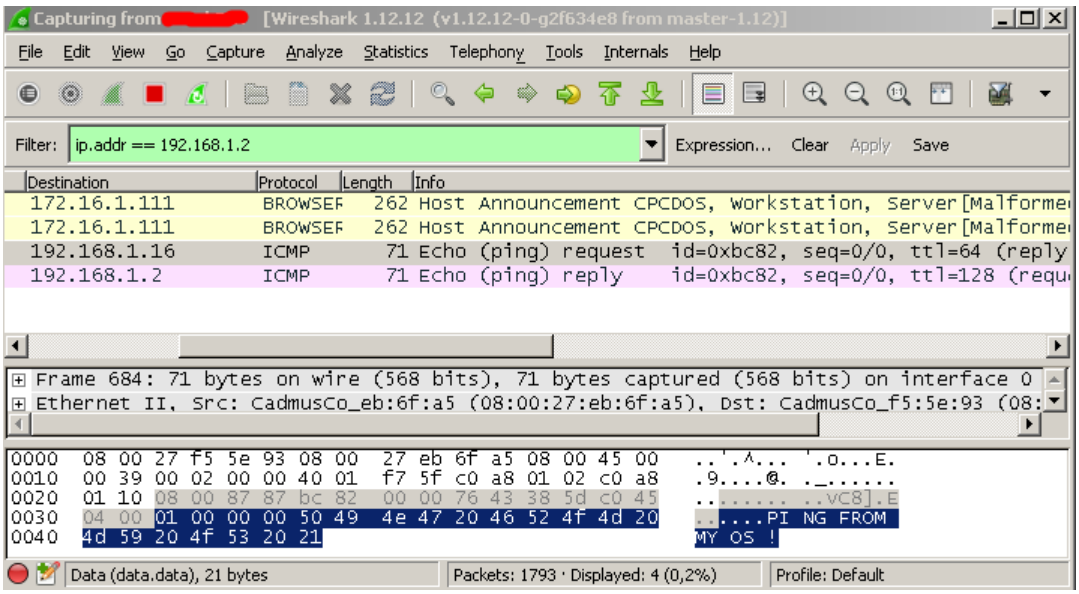
```
Ping/ www.facebook.com
```

Ping son modem/passerelle internet (ex : Livebox orange ou autre)

```
Ping/ 192.168.1.1
```

Modifier sa trame ICMP (Visible depuis Wireshark)

```
Fix/ CPC_SYS.NET.ICMP = Ping from my OS !
Ping/ 192.168.1.1
```



AUTRES INFORMATIONS :

- Requête ICMP personnalisable, modifiez à volonté la variable **%CPC_SYS.NET.ICMP%**
- Afin de ne pas bloquer le thread courant comme votre console, vous pouvez exécuter la requête PING dans un **autre** thread en utilisant :

```
CMD/ /Thread Ping/ www.google.fr
```

VOIR AUSSI :

Telecharger/ , Client/ , Serveur/

Télécharger un fichier sur le WEB

COMMANDE :

```
TELECHARGER/ {/SRVINFO, /+SRVINFO} [http://ServeurReseau]
```

FONCTIONNALITE :

Cette commande permet de télécharger un fichier WEB en utilisant le protocole **TCP/HTTP1.0** via une URL. L'user-agent, le dossier temporaire et destination peuvent être modifiées.

PARAMETRES DISPONIBLE :

- /SRVINFO → Télécharger **uniquement** les informations du serveur/fichier (HEAD)
- /+SRVINFO → Télécharger le fichier **+ information du serveur**
- /TEMP:[path..] → Modifier le dossier de destination du fichier téléchargé
- /TEMP**R**:[path..] → Idem, mais en concevant son chemin **R**elatif par rapport au serveur.
- Sans paramètres → Télécharger **uniquement** le fichier.

EXEMPLE :

La page web de google (Si https non forcé)

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

Voir le contenu de la page html :

```
Ouvrir/ Temp\net\www.google.fr\index.htm
```

ou

```
Ouvrir/ %CPC_Temp.NET%\www.google.fr\index.htm
```

La page web d'un routeur/modem internet local

```
Telecharger/ http://192.168.1.1
```

Voir le contenu de la page html :

```
Ouvrir/ Temp\net\192.168.1.1\index.htm
```

Un fichier WEB comme un fichier texte.

```
Telecharger/ http://deposit.cpcdos.net/test.txt
```

Voir le contenu de la page html :

```
Ouvrir/ Temp\net\deposit.cpcdos.net/test.txt
```

Télécharger uniquement les informations du serveur (HEAD)

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

Voir le contenu :

```
Ouvrir/ Temp\net\www.google.fr\index.htm
```

Télécharger les informations du serveur (HEAD) + le contenu de la page web ciblé

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

Voir le contenu :

```
Ouvrir/ Temp\net\www.google.fr\index.htm
```

Télécharger un fichier et le stocker dans un dossier temporaire personnalisé

```
Telecharger/ /TEMP:C:\MonDossier\toto http://MonServeur/abc/Fichier.txt
```

Voir le contenu :

```
Ouvrir/ C:\MonDossier\toto\Fichier.txt
```

Télécharger un fichier et le stocker dans un dossier temporaire personnalisé en concevant son chemin **R**elatif à l'emplacement du serveur

```
Telecharger/ /TEMPR:C:\MonDossier\toto http://MonServeur/abc/Fichier.txt
```

Voir le contenu :

```
Ouvrir/ C:\MonDossier\toto\MonServeur\abc\Fichier.txt
```

Remarque utile (pour une fois):

A la place de «**Temp\net**» vous pouvez utiliser la variable **%CPC_TEMP.NET%**

```
Ouvrir/ %CPC_TEMP.NET%\192.168.1.1\index.htm
```

Ce qui permet d'adapter votre programme aux répertoires temporaire de l'OS en exécution.

AUTRES INFORMATIONS :

- Si vous spécifiez une URL sans URI, du style http://monserveur.fr sans nom de fichier *(en bout de ligne)* alors Cpcdos enregistrera ce dernier tant que « INDEX.HTM » dans son dossier temporaire.
- Utilisez la commande OUVRIR/ pour ouvrir un fichier afin de voir son contenu en texte.
- La variable `%CPC_Temp.NET%` contient le chemin d'accès au dossier temporaire réseau.
- L'user-agent modifiable est la variable %CPC_SYS.NET.USERAGENT%.
- Pour des raisons presque évidentes, les paramètres CpcdosC+ doivent être spécifiés **avant l'URL**. Autrement il pourrait être confondu par des arguments URI.

VOIR AUSSI :

`Ping/ ; serveur/ ; client/`

Créer un serveur TCP/UDP

COMMANDE :

```
Serveur/ {/TCP:Numéro de port /MODE:CCP, TELNET} {/ENVOYER: | RE…}
```

FONCTIONNALITE :

Cette commande permet de créer un simple serveur TCP.
Par défaut il s'agit d'un serveur en mode VAR. (Plus d'informations en dessous)

PARAMETRES DISPONIBLE :

- /TCP:[…] → Démarrer un **serveur TCP** avec un port spécifié de 1 à 65 536
- ~~/UDP:[…] → Démarrer un **serveur UDP** avec un port spécifié de 1 à 65 536~~
- /STOP:[…] → Stoppe le serveur avec le port spécifié.
- /ENVOYER:[…] → Envoie une requête à un ou plusieurs clients.
- /RECEVOIR:[…] → Recevoir une requête. (Depuis le buffer)
  - /ATTENDRE → Bloquer le thread courant jusqu'à réception d'une donnée. Ou en ms.
- /MODE:[…] → Mode du serveur.
  - CCP → Permet l'accès ADMIN à la console CpcdosC+.
  - TELNET → Stock les données dans un buffer.
  - ECHO → Répète en renvoyant toutes les requêtes reçues à toutes les machines.

RETOUR

- Retourne '1' si le serveur à bien été créé.
- Retourne autre chose si la création a échoué.

EXEMPLES :

Démarrer un simple serveur TCP avec comme port '2316' :

```
Serveur/ /tcp:2316
```

Vous pouvez vous connecter au serveur, les requêtes envoyées seront stockées dans un buffer.

Si un client TCP est connecté, vous pouvez lui envoyer un petit message !

```
Serveur/ /Envoyer:2316 Coucou !
```

```
Server/ /Send:2316 Hello !
```

Si un client TCP à déjà envoyé un message sur le port, réceptionnez sa réponse !

```
Serveur/ /Recevoir:2316
Ou
serveur/ /recevoir:2316 /Attendre
```

```
Server/ /Receive:2316
Or
Server/ /receive:2316 /Wait
```

Cette commande affiche sur votre écran, vous pouvez stocker les données reçues dans une variable en procédant comme ceci :

```
@#Reponse Serveur/ /Recevoir:2316 /Attendre
Txt/ %Reponse%
```

/Attendre permet de mettre en **pause le thread en attendant la réception d'une requête.**
Si ce paramètre est omis, alors cette commande va simplement voir s'il y a une trame reçue dans le buffer, la récupère s'il y en a une, et continue son exécution, même s'il n'y en a pas.

Vous pouvez également spécifier un temps en millisecondes comme ceci :

```
@#Reponse Serveur/ /Recevoir:2316 /Attendre 10000
SI/ "%Reponse%" = "" alors:
     Txt/ N'a pas repondu dans les 10 secondes
Sinon:
     Txt/ Le client a repondu : %Reponse%
Fin/ si
```

EXEMPLE COMPLET :

```
Serveur/ /tcp:2316
   Un ou plusieurs client se connectent
Serveur/ /Envoyer:2316 Bonjour !

   Un client repond envoie "ca va ?"
@#Reception Serveur/ /Recevoir:2316 /Attendre
Txt/ Un client a repondu :%Reception%
```

CORRESPONDANCE ANGLAISE :

```
Server/ /tcp:2316
   Un ou plusieurs client se connectent
Server / /Send:2316 Hello !

   Un client repond envoie "ca va ?"
@#Reception Server / /Receive:2316
Txt/ A client had reply :%Reception%
```

Arrêter un serveur avec comme port '2316' :

```
Serveur/ /Stop:2316
```

DEMARRER UN SERVEUR D'INTERPRETATION CPCDOSC+ (MODE CCP) AVEC COMME PORT '1234 :

```
Serveur/ /tcp:2316 /Mode:CCP
```

Vous pouvez envoyer des commandes CpcdosC+ au SHELL de Cpcdos à distance via un client TCP.

DEMARRER UN SERVEUR TELNET (MODE TELNET OU TEL) AVEC COMME PORT '1234' :

```
Serveur/ /tcp:1234 /Mode:TELNET      ou /Mode:TEL
```

Vous pouvez accéder à la console CpcdosC+ à distance via un client Telnet.
Compatible Android, Windows Phone, Windows, Mac, unix...
A noter qu'en général, le port telnet est le port 23.

DEMARRER UN SERVEUR D'ECHO TCP (MODE ECHO OU ECH) AVEC COMME PORT '1234' :

```
Serveur/ /tcp:2316 /Mode:ECHO
```

Ce qui permet de diffuser à toutes les machines connectées, les requêtes reçues depuis un ou plusieurs clients. Très utile pour les programmes comme les t'chats.
A noter qu'en général, le port echo est le port 7.

Pour tester, connectez-vous simplement à votre serveur Cpcdos tant que client TCP, envoyez une requête bidon, dans les millisecondes qui suivent vous allez recevoir votre propre requête mais accompagné de l'adresse IP.
Répétez cette opération mais avec plusieurs clients TCP ouvert et connecté à votre serveur Cpcdos, toutes les requêtes envoyées seront répétées par le serveur digne d'un écho sonore ;-)

TESTER SON SERVEUR EN LOCALHOST (127.0.0.1) :

```
REM/ Creer un serveur
Serveur/ /tcp:2316

REM/ Se connecter au serveur en LOCALHOST (127.0.0.1)
@#NumeroTID Client/ /tcp:127.0.0.1:2316

REM/ Envoyer un message du SERVEUR ---> CLIENT
Serveur/ /Envoyer:2316 Hello world !

REM/ Recuperer et afficher a l'ecran le message recu
Client/ /Recevoir:%NumeroTID%

REM/ Envoyer un message du CLIENT ---> SERVEUR
Client/ /Envoyer:%NumeroTID% Comment tu vas ?

REM/ Recuperer et afficher a l'ecran le message recu
Serveur/ /Recevoir:2316 /Attendre

REM/ Fermer le serveur (Ceci va deconnecter automatiquement le client)
Serveur/ /Stop:2316
```

AUTRES INFORMATIONS :

- Multi-client → Capable d'accepter et répondre à plusieurs clients connectés en même temps.
- Multithreads → Capable de gérer plusieurs autres serveurs en fonctionnement.
- Attention : Si vous utilisez le paramètre **/ATTENDRE** sur votre console, le thread de la console sera bloqué jusqu'à réception d'une trame. Utilisez **CMD/ /THREAD** pour l'exécuter depuis un nouveau thread 😉

VOIR AUSSI :

`telecharger/ ; ping/ ; client/`

Créer un client TCP/UDP

COMMANDE :

```
Client/ [/TCP:Numéro de port] {/MODE:VAR, GET, CCP}
```

FONCTIONNALITE :

Cette commande permet de se connecter à un serveur TCP ou UDP.

PARAMETRES DISPONIBLE :

- /TCP:[…] → Se connecter à un **serveur TCP**
- ~~/UDP:[…] → Se connecter à un **serveur UDP**~~
- /STOP:[…] → Se déconnecter du serveur actuel.
- /ENVOYER:[…] → Envoyer un message au serveur actuel.
- /RECEVOIR:[…] → Recevoir un message depuis le buffer.
  /ATTENDRE → Bloquer le thread courant jusqu'à réception d'une trame.

RETOURNE :

- Renvoie un numéro supérieur à zéro si le client est connecté au serveur.
  Ce numéro correspond au numéro de TID, ou plus précisément au numéro de Socket.
- Renvoie autre chose si la connexion a échoué.
- Renvoie '#DECO' si à connexion a été interrompue.

SERVEUR EXEMPLE :

Etablir une simple connexion au serveur "192.168.1.5" et port "2316"

```
Client/ /tcp:192.168.1.5:2316
```

Ceci **renvoie un numéro de TID** (Thread ID qui est relié au socket réseau) qui permet d'interagir avec le serveur.

Pour la suite des exemples, nous allons considérer que nous avions reçu le numéro «13».

Si le serveur vous a envoyé un message, il est stocké dans le buffer, pour le récupérer :

```
Serveur/ /Recevoir:13 /Attendre
```

```
Client/ /Receive:13 /Wait
```

*Remarque : Renvoie **#DECO** si vous êtes déconnecté du serveur*

Cette commande affiche sur votre écran, vous pouvez stocker les données reçues dans une variable en procédant comme ceci :

```
@#Reponse Serveur/ /Recevoir:13 /Attendre
Txt/ %Reponse%
```

/Attendre permet de mettre en **pause le thread en attendant la réception d'une requête.**
Si ce paramètre est omis, alors cette commande va simplement voir s'il y a une trame reçue dans le buffer, la récupère s'il y en a une, et continue son exécution, même s'il n'y en a pas.

Si vous voulez répondre au serveur :

```
Client/ /Envoyer:13 Coucou !
```

```
Client/ /Send:13 Hello !
```

*Remarque : Renvoie **#DECO** si vous êtes déconnecté du serveur.*

Arrêter un serveur avec comme port '2316' :

```
Client/ /Stop:13
```

Exemple plus complet :

```
@#MonNumeroTID Client/ /tcp:192.168.1.5:2316

Client/ /Envoyer:%MonNumeroTID% Bonjour !!

   Le serveur repond
@#Reception Client / /Recevoir:%MonNumeroTID% /Attendre
Si/ "%Reception%" = "#DECO" alors:
     Txt/ Le serveur a ferme la connexion...
Sinon:
     Txt/ Le serveur a repondu:%Reception%
Fin / si
```

CORRESPONDANCE ANGLAISE :

```
@#MyTIDNumber Client/ /tcp:192.168.1.5:2316

Client/ /Envoyer:%MyTIDNumber% Hello !!

   Le serveur repond
@#Reception Client / /Recevoir:%MyTIDNumber% /Wait

If/ "%Reception%" = "#DECO" then:
     Txt/ Server has closed connection...
Else:
     Txt/ Server has reply:%Reception%Fin / si
End/ if
```

SERVEUR INTERNET

Il est bien évidement possible de contacter les serveurs internet. Faut savoir lui parler 😉
Exemple avec le port 80 qui est un port http chez Google. Pour obtenir la page web :

```
@#IDClient Client/ /tcp:www.google.fr:80
client/ /envoyer:%IDClient% GET / http/1.0

// Affiche a l'ecran la reponse reçue avec un délai de 3 secondes.
client/ /recevoir:%IDClient% /Attendre 3000
```

AUTRES INFORMATIONS :

- Multithreads → Capable de gérer plusieurs clients en fonctionnement.
- Attention : Si vous utilisez le paramètre **/ATTENDRE** sur votre console, le thread de la console sera bloqué jusqu'à réception d'une trame. Utilisez **CMD/ /THREAD** pour exécuter depuis un nouveau thread 😉
- Les paramètres /ENVOYER et /RECEVOIR (Donc aussi /SEND et /RECEIVE) peuvent renvoyer le message "**#DECO**". C'est que vous êtes ou vous avez été tout simplement déconnecté. Ou que le serveur a fermé la connexion. C'est très utile d'utiliser une condition de ce type :

```
@#Ma_variable Client/ /envoyer:13 Blabla

Si/ "%Ma_Variable%" = "#DECO" alors:
     Txt/ Vous etes deconnecte du serveur...
Sinon:
     // Tout est ok
Fin/ si
```

→ Le procédé est le même pour le paramètre **/recevoir**

VOIR AUSSI :

telecharger/ ; ping/ ; serveur/

Niveau 5 : Lancement d'un OS

Charger un système d'exploitation installé

COMMANDE :

```
demarrer/ {Nom ou Numéro de l'OS }
```

Correspondance anglaise

```
Start/
```

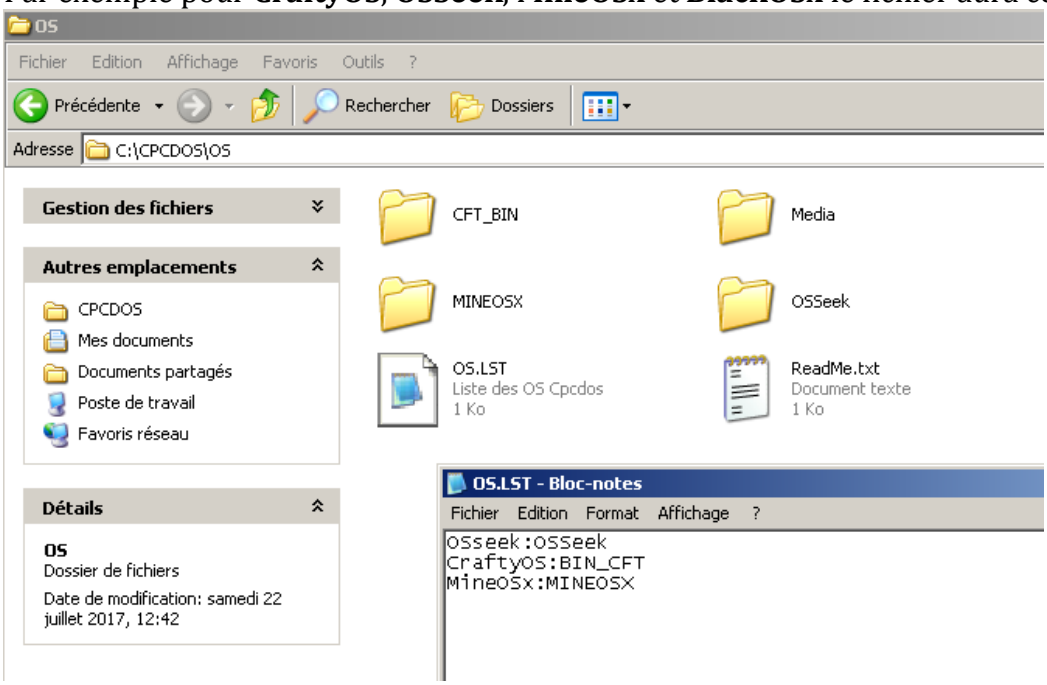
FONCTIONNALITE :

Cette commande permet de lancer la séquence de démarrage du système d'exploitation choisis. En utilisant la commande **SYS/ /OS /LISTE** vous pourrez visualiser les OS présent prêt à démarrer.

Votre OS doit être inscrit dans le fichier OS.LST avec comme syntaxe :

```
{Nom de votre OS}:{Dossier relatif de votre OS}
```

Par exemple pour **CraftyOS**, **OSSeek**, **MineOSx** et **BlackOSx** le fichier aura cette gueule :



Si vous êtes perdu, c'est normal, cela veut dire que vous vous êtes trop précipité ! 😉
Je vous l'ai dit, prenez votre temps pour lire ce manuel, le début est riche d'informations !

PARAMETRES DISPONIBLE :

- Optionnel, vous pouvez préciser le nom de l'OS à démarrer

AUTRES INFORMATIONS :

- Si aucun nom ou numéro d'OS a été précisé, par défaut Cpcdos démarrera **le premier** OS de la liste OS.LST, cet à dire l'index numéro **0**

VOIR AUSSI :

```
Fenetre/ ; demarrer/ ; iug/ ; sys/ /os ;
```

Exécuter l'interface utilisateur graphique

COMMANDE :

```
iug/ {/MULTI-PICTUREBOX} {/OS:NomOS}
```

InterfaceUtilisateurGraphique

Correspondance anglaise

```
gui/ {/MULTI-PICTUREBOX}{/OS:NomOS}
```

GraphicUserInterface

FONCTIONNALITE :

Cette commande permet d'exécuter l'interface graphique en initialisant un nouveau pointeur vidéo et la résolution graphique indiquée dans la variable %SCR_RES% sous la forme de « LLLLxHHHH » et la variable %SCR_BIT% pour indiquer le nombre de couleurs par pixel en bits « 16, 24 ou 32 ». Si aucun système d'exploitation est exécuté, ne soyez pas surpris que l'écran soit vide d'une couleur ou d'un simple fond d'écran. Pour exécuter un OS il faut avant tout utiliser la commande DEMARRER/

Par défaut, il y a un OS exécuté un arrière-plan qui se nomme « BLANK-KERNEL » identifié par son OSid de « 8 ». Si vous tapez donc cette commande sans avoir lancé vous-même un OS, l'OS par défaut sera ce BLANK-KERNEL. Vous verrez en premier le fond d'écran par défaut.

Le principal fichier de configuration graphique se trouve dans le dossier **KRNL\CONFIG\ENV_GUI** sous le nom de **GUI_STD.CPC** NE LE MODIFIEZ PAS. Mais UTILISEZ LES VARIABLES, ELLES SONT MODIFIABLES depuis VOTRE OS

PARAMETRES DISPONIBLE :

- /OS → Afficher la GUI d'un autre OS.
- /SansOS → Afficher la GUI sans OS
- /CONSOLE
  /LC → Réduire la GUI et afficher la Console

EXEMPLES

```
iug/
```

Affiche l'interface graphique de l'OS courant. *(Voir SYS/ /OS pour changer d'OS)*

```
iug/ /OS:MonAutreOS
```

Affiche l'interface graphique de 'MonAutreOS'. → Il faut qu'il soit installé.

```
iug/ /CONSOLE
```

```
iug/ /LC
```

Réduit la GUI et affiche la Console CpcdosC+

AUTRES INFORMATIONS :

- Afin de choisir manuellement votre résolution d'écran, utiliser la commande **SYS/ /ECRAN /LISTE 32** si vous voulez lister les résolutions 32 bits, vous avez 16 et 24 aussi !
- Si votre système est instable, crash, utilisez une petite résolution et pourquoi pas 16 bits de couleurs le temps des mises à jour. Certains PC fonctionnent mieux que d'autres… mystère !

VOIR AUSSI :

```
Fenetre/ ; demarrer/
```

Créer un messagebox

COMMANDE :

```
Message/ [Texte] {/titre:texte /erreur:[0-4] /type:[0-3] /evene…}
```

FONCTIONNALITE :

Cette commande permet de créer une boite de dialogue graphique avec l'utilisateur.
Il existe 5 types de message (Simple, information, question, avertissement, erreur) et 4 modes d'interactions (Ok, Oui/Non, Oui/Non/Annuler)

PARAMETRES DISPONIBLE :		ENGLISH :
➢ **/Titre:***{texte}*	→ Titre du message box	/Title:
➢ **/Erreur:***{0 à 4}*	→ Numéro d'erreur	/Error:
○ 0 : (Par défaut) Simple message		
○ 1 : Information		
○ 2 : Question		
○ 3 : Avertissement		
○ 4 : Erreur		
➢ **/Type:***{0 à 3}*	→Type d'interaction	/Type:
○ 0 : (Par défaut) OK		
○ 1 : Oui / Non		
○ 2 : Oui / Non / Annuler		
➢ **/Evenement:***{Fichier .cpc}*		/Event:
➢ **/Nom:***{Texte}*		/Name:

EXEMPLE SIMPLE :

```
Message/ Coucou !
```

Affiche un simple message box avec « coucou ! »

EXEMPLE COMPLET :

```
Message/ Coucou /Titre:Blabla /Erreur:3 /Type:0
```

Affiche un simple message box avertissement avec « coucou » en texte « Blabla» en titre

Correspondance anglaise :

```
Msgbox/ Hello! /Title:Blabla /Error:3 /Type:0
```

AUTRES INFORMATIONS :

- Vous n'êtes pas obligé d'utiliser tous les arguments /titre /nom etc…
  Utilisez seulement ceux que vous avez besoin et dans n'importe quel ordre ! 😉
- Code source : **KRNL\CONFIG\ENV_GUI\msgbox.cpc**

VOIR AUSSI :

`Picturebox/ ; Bouton/ ; textebloc/ ;`

Créer une fenêtre

COMMANDE :

```
fenetre/ [Nom de fenetre]
Fin/ fenetre
```

FONCTIONNALITE :

Cette commande permet de créer une fenêtre graphique dans un processus existant ou le processus hôte. Si le processus n'est pas défini, il sera hébergé dans le processus courant qui exécute cette commande.

PARAMETRES DISPONIBLE :

- .Titre → Le titre de la fenêtre dans la barre de titre.
- .Parametres → Paramétrages de la fenêtre.
- .PX → Position horizontale de la fenêtre.
- .PY → Position verticale de la fenêtre.
- .TX → Taille horizontale de la fenêtre.
- .TY → Taille verticale de la fenêtre.
- .Opacite → Opacité de la fenêtre (0 → 255 Opaque)
- .CouleurFenetre → Couleur générale de la fenêtre.
- .CouleurTitre → Couleur des caractères du titre.
- .CouleurFond → Couleur du conteneur.
- .ImgTitre → Image de la barre de titre.
- .Icone → Icône de la fenêtre.
- .Evenements →Fichier évènementiel des interactions graphiques.

Voici les arguments possible pour la propriété « .Parametres » :

- **TYPE:***valeur entre 0 et 5*
  valeur Correspond au type de fenêtre graphique
  → **TYPE:0** (par défaut) Fenêtre normale.
  → **TYPE:1** Fenêtre sans conteneur visible.
  → **TYPE:2** Fenêtre sans bitmap de titre.
  → **TYPE:3** Fenêtre sans bitmap de titre et sans conteneur visible.
  → **TYPE:4** Fenêtre sans barre de titre.
  → **TYPE:5** Fenêtre sans barre de titre et sans conteneur visible.

- **CTN:***valeur entre 0 et 1*
  Afficher le conteneur au complet (sans la barre de titre cliquable)
  CTN:1 Oui ou **CTN:0** Non

- **BORD:***valeur entre 0 et 1*
  Afficher le contour graphique d'une fenêtre.
  BORD:1 Oui ou **BORD:0** Non

- **OMBRE:** *valeur entre 0 et 255*
  Opacité de l'ombre derrière la fenêtre une valeur personnalisable entre 0 → 255.
  OMBRE:0 Ombre désactivé pour la fenêtre
  OMBRE:128 semi-transparente
  OMBRE;255 Très opaque.

EXEMPLE SIMPLE : (LES GRANDES ESPACES SONT DES TABULATION, ET ILS NE SONT PAS OBLIGATOIRES)
C'EST JUSTE … PLUS PROPRE COMME ÇA !

```
fenetre/ ma_fenetre_1
     .titre              = "Application test"
     .px                 = "60"
     .py                 = "40"
     .tx                 = "420"
     .ty                 = "263"
     .CouleurFenetre     = "255,255,255"
     .CouleurTitre       = "255,255,255"
     .CouleurFond        = "255,255,255"
     .Icone              = "Icone.png"
     .ImgTitre           = ""
     creer/
Fin/ fenetre
```

La propriété « .Parametre» est omit, il s'agirait donc d'une simple fenêtre par défaut.

Correspondance anglaise :

```
Window/ my_window_1
     .title              = "My first window!"
     .px                 = "60"
     .py                 = "40"
     .sx                 = "420"
     .sy                 = "263"
     .WindowColor        = "255,255,255"
     .TitleColor         = "255,255,255"
     .BackColor          = "255,255,255"
     .Icon               = "Icone.png"
     .TitleImg           = ""
     Create/
End/ Window
```

Récupérer le numéro de handle et le stocker dans une variable
Il faut savoir que la commande qui génère un numéro de handle est la commande **Creer/** ou **Create/**
Et **@#[Variable]** va permet de stocker le résultat d'une commande

Voici un exemple :

```
fenetre/ ma_fenetre_1
     .titre              = "Ma premiere fenetre !"
     .px                 = "60"
     .py                 = "40"
     .tx                 = "800"
     .ty                 = "600"
     .type               = "1"
     .CouleurFenetre     = "255,255,255"
     .CouleurTitre       = "255,255,255"
     .CouleurFond        = "255,255,255"
     .Icone              = "Icone.png"
     .ImgTitre           = ""
     @#Mon_Numero_handle creer/
Fin/ fenetre
```

Que faire avec ce numéro ?
Il suffit simplement de poser cette variable dans la propriété « .Handle » d'un objet graphique-t-elle que votre bouton, Imagebox, textebox… afin que votre objet soit hébergé DANS votre fenêtre précédemment créée.

```
Bouton/ MonBouton
     .PID      = "%Mon_Numero_handle%"
     …
     …
Fin/ Bouton
```

AUTRES INFORMATIONS :

VOIR AUSSI :
Picturebox/ ; Bouton/ ; textebloc/ ;

Créer une Imagebox

COMMANDE :

```
Imagebox/ [Nom du l'imagebox]
Fin/ Imagebox
```

FONCTIONNALITE :

Cette commande permet de créer une ImageBox (Picturebox en anglais). Il s'agit d'un objet graphique capable d'afficher une image résidant sans sa mémoire. Elle peut charger un fichier image de format PNG, JPG, GIF et BMP, ou une couleur, la stocker dans sa mémoire sous forme d'un pointeur.
Elle peut donc également afficher l'image d'une autre Imagebox ou d'une autre ressource graphique d'un format brute du type RGBA (Red Green Blue Alpha) en indiquant l'adresse mémoire de ce dernier. Et afficher également un contenu dynamique (OpenGL, SDL, GZE etc…)
Cette Imagebox est également capable de "sizer" la taille de son image afin qu'elle soit adaptée aux dimensions de l'Imagebox.

Simple fenêtre avec une imagebox qui contient l'image de la terre en format PNG.

PARAMETRES DISPONIBLE :

- .handle → Numéro de handle parent. (Fenêtre)
- .Nom → Le nom de l'objet.
- .Parametres → Le type de picturebox et autres paramètres.
- .PX → Position horizontale.
- .PY → Position verticale.
- .TX → Taille horizontale.
- .TY → Taille verticale.
- .CouleurFond → Couleur de fond RVB.
- .CouleurTexte → Couleur de du texte RVB.
- .Opacite → Transparence de l'image
- .Image → Chemin d'accès au fichier ou adresse mémoire.
- .Evenement → Fichier évènementiel des interactions graphiques.

Voici les arguments possible pour la propriété « .Parametres » :

- **COL:***valeur entre 0 et 1*
  → **COL:0** N'affiche pas de couleur de fond.
  → **COL:1** (Par défaut) Affiche la couleur de fond issue de la propriété « .CouleurFond ».

- **IMGAUTO:***valeur entre 0 et 2*
  valeur Correspond au mode d'affichage de l'image
  → **IMGAUTO:0** (par défaut) affichage brute, l'image est coupée si elle est plus grande.
  → **IMGAUTO:1** Affichage adapté aux dimensions de l'imagebox
  → **IMGAUTO:2** Affichage adapté aux dimensions de l'image source (Affecte la qualité)

- **UPD:***valeur entre 0 et 1*
  Utiliser un thread d'actualisation graphique de la propriété .TEXT

EXEMPLE SIMPLE : (LES GRANDES ESPACES SONT DES TABULATION, ET ILS NE SONT PAS OBLIGATOIRES)

```
Imagebox/ mon_imagebox
     .Handle         = "Numero De handle De La Fenetre Parent"
     .Parametres     = "IMGAUTO:2"
     .PX             = "10"
     .PY             = "10"
     .TX             = "500"
     .TY             = "300"
     .Image          = "Dossier\MonImage.PNG"
     creer/
Fin/ Imagebox
```

Cet exemple affiche "MonImage.PNG". A la présence du paramètre "IMGAUTO:2" elle respecte les dimensions de «500x300»

Correspondance anglaise :

```
Picturebox/ my_picturebox
     .Handle         = "handle number of window parent"
     .Parameters     = "1"
     .PX             = "10"
     .PY             = "10"
     .SX             = "500"
     .SY             = "300"
     .Image          = "Directory\MyPicture.PNG
     Create/
End/ Picturebox
```

EXEMPLE AVEC UNE BITMAP ID

```
Picturebox/ mon_picturebox
     .Handle   = "Numero De Handle De La Fenetre Parent"
     .Type     = "1"
     .PX       = "10"
     .PY       = "10"
     .TX       = "500"
     .TY       = "300"
     .Image    = "#1234"
     creer/
Fin/ Picturebox
```

Il faut bien évidemment utiliser un bitmap ID existant (Voir SYS/ /BITMAP /LIST).

EXEMPLE AVEC ADRESSE MEMOIRE HEXADECIMALE

```
Picturebox/ mon_picturebox
     .Handle   = "Numero De Handle De La Fenetre Parent"
     .Type     = "1"
     .PX       = "10"
     .PY       = "10"
     .TX       = "500"
     .TY       = "300"
     .Image    = "@0x00BC614E"
     creer/
Fin/ Picturebox
```

Pour utiliser une ressource BITMAP en mémoire, il suffit d'utiliser le caractère «@» suivit de l'adresse mémoire en hexadécimale.

AUTRES INFORMATIONS :

- Si aucune image est définit, la couleur de la propriété « .CouleurFond » sera pleinement affiché si le paramètre « COL:1 » est définit. 😉 *(Donc IMGAUTO:2 et 1 ne servira à rien)*
- Si vous avez un rendu non souhaité de type :

C'est que vous avez pointé dans la mauvaise adresse mémoire et que vous voyez une sorte d'une memory map, c'est assez drôle de percevoir ceci, vous pouvez visualiser d'une manière brute, le contenu de votre mémoire RAM. Le pictureBox interprête les séquences octets ARGB en séquence de pixels colorisée. Vous pourrez constater que la partie du haut s'agit des données occupés et que la partie plus foncé s'agit de l'espace « libre » … mais un peut fragmenté. 😉

VOIR AUSSI :

fenetre/ ; bouton/ ; textebloc/ ; textebox/

Créer un bouton

COMMANDE :

```
Bouton/ [Nom du bouton]
Fin/ Bouton
```

FONCTIONNALITE :

Cette commande permet de créer une nouvelle instance d'un bouton graphique avec la possibilité principale de … attention …. Attention ….. de « cliquer ». Elle est dotée d'une petit effet graphique de survole, pression et le relâchement.

PROPRIETES DISPONIBLE :

- .Handle → Numéro de handle parent. (Fenêtre)
- .Nom → Le nom de l'objet.
- .Parametres → Paramètres et modes
- .PX → Position horizontale.
- .PY → Position verticale.
- .TX → Taille horizontale.
- .TY → Taille verticale.
- .CouleurFond → Couleur de fond RVB.
- .CouleurTexte → Couleur de du texte RVB.
- .Opacite → Transparence de l'image
- .Image → Chemin d'accès au fichier ou adresse mémoire.
- .Evenement → Fichier évènementiel des interactions graphiques.

Voici les arguments possible pour la propriété « .Parametres » :

- **COL:***valeur entre 0 et 1*
  → **COL:0** N'affiche pas de couleur de fond.
  → **COL:1** (Par défaut) Affiche la couleur de fond issue de la propriété « .CouleurFond ».

- **IMGAUTO:***valeur entre 0 et 2*
  valeur Correspond au mode d'affichage de l'image
  → **IMGAUTO:0** (par défaut) affichage brute, l'image est coupée si elle est plus grande.
  → **IMGAUTO:1** Image source adapté aux dimensions de l'imagebox
  → **IMGAUTO:2** Imagebox adapté aux dimensions de l'image source (Affecte la qualité)

- **UPD:***valeur entre 0 et 1*
  Utiliser un thread d'actualisation graphique de la propriété .TEXT

EXEMPLE SIMPLE : (LES GRANDES ESPACES SONT DES TABULATION, ET ILS NE SONT PAS OBLIGATOIRES)

```
Bouton/ MonBouton
      .Handle              = "%Handle _Fenetre%"
      .Parametres          = "IMGAUTO:2"
      .PX                  = "80"
      .PY                  = "80"
      .tx                  = "100"
      .ty                  = "30"
      .texte               = "Clique moi!"
      .Image               = "Bouton.PNG"
      .event               = "OS/Evenement_bouton.CPC"
      creer/
Fin/ Bouton
```

Correspondance anglaise :

```
Button/ MyButton
      .handle              = "%handle_Window%"
      .Parametres          = "IMGAUTO:2"
      .PX                  = "80"
      .PY                  = "80"
      .sx                  = "100"
      .sy                  = "30"
      .texte               = "Clique moi!"
      .Image               = "Bouton.PNG"
      .event               = "OS/Evenement_bouton.CPC"
      creer/
Fin/ Bouton
```

AUTRES INFORMATIONS :

- Si aucune image est définit, la couleur de la propriété « .CouleurFond » sera pleinement affiché , seulement si le paramètre « COL:1 » est définit.

VOIR AUSSI :

`Fenetre/ ; Picturebox/ ; textebloc/ ; textebox/`

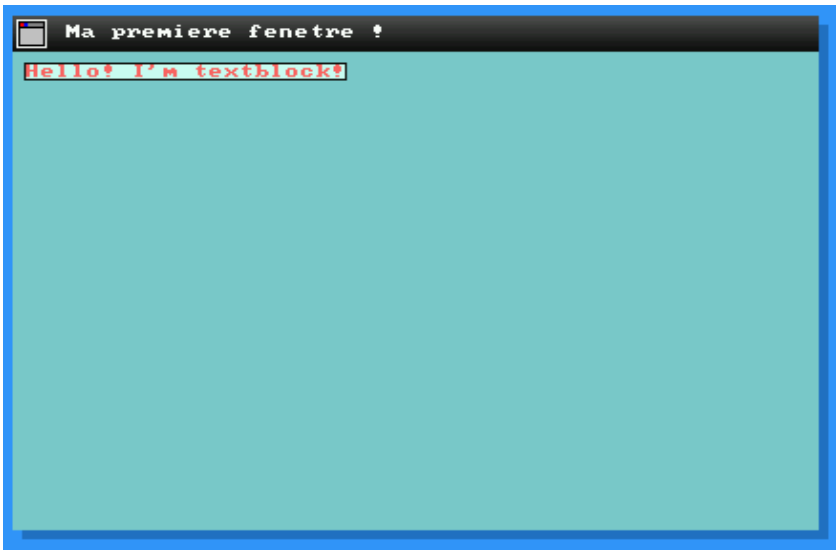
Créer un Textebloc

COMMANDE :

```
Textebloc/ [Nom du Textebloc]
End/ Textebloc
```

FONCTIONNALITE :

Cette commande permet de créer une nouvelle instance d'un bloc de texte graphique. Il est capable d'inclure le retour de ligne, une couleur de fond et de la bordure, ou sans.



Exemple avec couleur de fond, et bordure, oui c'est moche… C'est pour vous montrer !

PROPRIETES DISPONIBLE :

- .handle → Numéro de handle parent. (Fenêtre)
- .Nom → Le nom de l'objet.
- .Parametres → Paramètres et modes
- .PX → Position horizontale.
- .PY → Position verticale.
- .TX → Taille horizontale.
- .TY → Taille verticale.
- .CouleurFond → Couleur de fond RVB.
- .CouleurTexte → Couleur de du texte RVB.
- .Evenement → Fichier évènementiel des interactions graphiques.

Voici les arguments possible pour la propriété « .Parametres » :

- **COL:***valeur entre 0 et 1*
  → **COL:0** N'affiche pas de couleur de fond.
  → **COL:1** (Par défaut) Affiche la couleur de fond issue de la propriété « .CouleurFond ».

- **IMGAUTO:***valeur entre 0 et 2*
  valeur Correspond au mode d'affichage
  → **IMGAUTO:0** (par défaut) affichage brute, le texte est coupé s'il est plus grand.
  → **IMGAUTO:1** Affichage adapté aux dimensions du Texte
  → **IMGAUTO:2** *Non applicable.*

- **MULTILINES:***valeur entre 0 et 1*
  valeur Correspond au mode d'affichage du texte
  → **MULTILINES:0** (par défaut) affichage du texte sur 1 seule ligne
  → **MULTILINES:1** Prend en compte le retour chariot (%CPC.CRLF%)

- **UPD:***valeur entre 0 et 1*
  Utiliser un thread d'actualisation graphique de la propriété .TEXT

EXEMPLE SIMPLE : (LES GRANDES ESPACES SONT DES TABULATION, ET ILS NE SONT PAS OBLIGATOIRES)

```
Textebloc/ Mon_Textebloc
     .handle              = "%Handle_Fenetre%"
     .Parametres          = " IMGAUTO:1 COL:1 MULTILINES:0"
     .Texte               = "Hello! I'm textblock!"
     .PX                  = "5"
     .PY                  = "5"
     .tx                  = "100"
     .ty                  = "20"
     .CouleurFond         = "200,255,240"
     .CouleurTexte        = "250,100,100"
     creer/
Fin/ Textebloc
```

Correspondance anglaise :

```
Textblock/ My_Textblock
     .handle              = "%handle_Window%"
     .Parameters          = "IMGAUTO:1 COL:1 MULTILINES:0"
     .PX                  = "5"
     .PY                  = "5"
     .Sx                  = "100"
     .Sy                  = "20"
     .BackColor           = "200,255,240"
     .TextColor           = "250,100,100"
     .text                = "Hello! I'm textblock!"
     create/
End/ Textblock
```

EXEMPLE 2: AFFICHER L'HEURE ET LE CPU ACTUALISE AUTOMATIQUEMENT

```
Textebloc/ Mon_Textebloc
     .handle              = "%Handle_Fenetre%"
     .Parametres          = "IMGAUTO:1 COL:1 MULTILINES:0 UPD:1"
     .Texte               = "Heure: ${CPC.HOUR} CPU: ${CPC.SYS.CPU.ACT}
     .PX                  = "5"
     .PY                  = "5"
     .tx                  = "100"
     .ty                  = "20"
     .CouleurFond         = "200,255,240"
     .CouleurTexte        = "250,100,100"
     creer/
Fin/ Textebloc
```

AUTRES INFORMATIONS :

VOIR AUSSI :

Fenetre/ ; Picturebox/ ; bouton/ ; Textebox/ ;

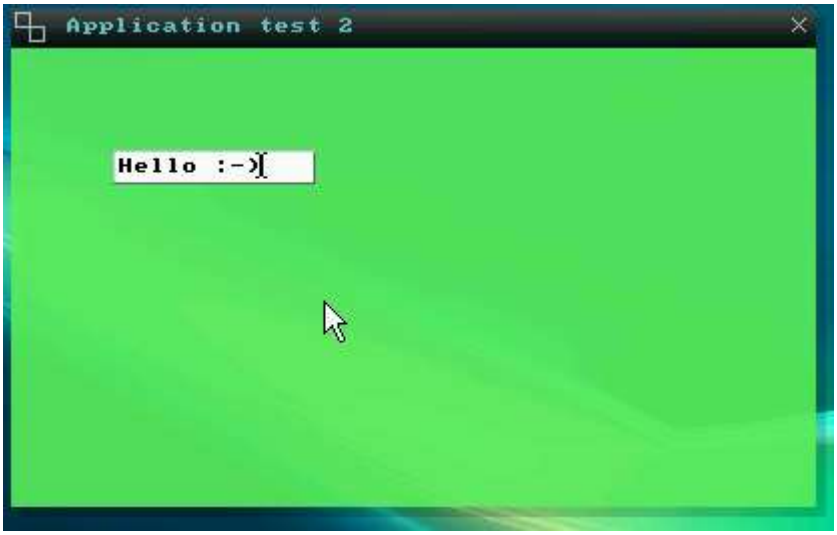
Créer un TexteBox

COMMANDE :

```
Textebox/ [Nom du Textebox]
End/ Textebox
```

FONCTIONNALITE :

Cette commande permet de créer une nouvelle instance d'une zone de texte éditable. Vous pouvez écrire au clavier dans cette zone uniquement s'il a le focus (Si vous l'avez préalablement sélectionné)



PROPRIETES DISPONIBLE :

- .handle → Numéro de handle parent. (Fenêtre)
- .Nom → Le nom de l'objet.
- .Parametres → Paramètres et modes
- .PX → Position horizontale de la fenêtre.
- .PY → Position verticale de la fenêtre.
- .TX → Taille horizontale de la fenêtre.
- .TY → Taille verticale de la fenêtre.
- .CouleurFond → Couleur de fond RVB.
- .CouleurTexte → Couleur de du texte RVB.
- .Evenement → Fichier évènementiel des interactions graphiques.

Voici les arguments possible pour la propriété « .Parametres » :

- **COL:***valeur entre 0 et 1*
  → **COL:0** N'affiche pas de couleur de fond.
  → **COL:1** (Par défaut) Affiche la couleur de fond issue de la propriété « .CouleurFond ».

- **MODIF:** *valeur entre 0 et 1* *ou **EDIT***
  → **MODIF:0** Interdit à l'utilisateur la modification du texte.
  → **MODIF:1** (Par défaut) Permet la modification du texte.

- **MULTILINES:** *valeur entre 0 et 1* *ou **MULTILIGNES***
  → **MULTILINES:0** (Par défaut) Interdit les caractères ASCII CRLF et le retour de ligne.
  → **MULTILINES:1** Autorise .. blabla vous avez compris que c'est l'inverse !

- **IMGAUTO:***valeur entre 0 et 1*
  valeur Correspond au mode d'affichage de l'image
  → **IMGAUTO:0** (par défaut) affichage brute, l'image est coupée si elle est plus grande.
  → **IMGAUTO:1** Affichage adapté aux dimensions de l'imagebox

- **UPD:***valeur entre 0 et 1*
  Utiliser un thread d'actualisation graphique de la propriété .TEXT

EXEMPLE SIMPLE : (LES GRANDES ESPACES SONT DES TABULATION, ET ILS NE SONT PAS OBLIGATOIRES)

```
Textebox/ Mon_Textebox
     .handle              = "%handle_Fenetre%"
     .Parametres          = "MODIF:1 MULTILIGNES:0"
     .Texte               = "Hello! I'm textebox!"
     .PX                  = "5"
     .PY                  = "5"
     .tx                  = "100"
     .ty                  = "20"
     .CouleurFond         = "200,255,240"
     .CouleurTexte        = "250,100,100"
     creer/
Fin/ Textebox
```

Correspondance anglaise :

```
Textbox/ My_Textbox
     .handle              = "%handle_Window%"
     .Parameters          = "EDIT:1 MULTILINES:0"
     .PX                  = "5"
     .PY                  = "5"
     .Sx                  = "100"
     .Sy                  = "20"
     .BackColor           = "200,255,240"
     .TextColor           = "250,100,100"
     .text                = "Hello! I'm textbox!"
     create/
End/ Textbox
```

AUTRES INFORMATIONS :

VOIR AUSSI :

Fenetre/ ; Picturebox/ ; bouton/ ;

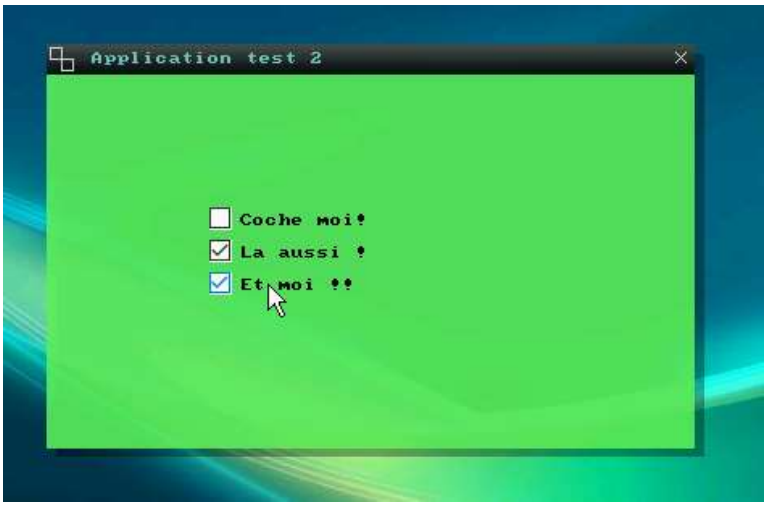
Créer un CheckBox

COMMANDE :

```
CheckBox/ [Nom du checkbox]
End/ Checkbox
```

FONCTIONNALITE :

Cette commande permet de créer une nouvelle instance d'une boite de sélection.
Dès que vous entrez le curseur, l'objet « s'illumine » … tadadaaaa !



PROPRIETES DISPONIBLE :

- .handle → Numéro de handle parent. (Fenêtre)
- .Nom → Le nom de l'objet.
- .Parametres → Paramètres et modes (**IMGAUTO:2 recommandé**)
- .PX → Position horizontale de la fenêtre.
- .PY → Position verticale de la fenêtre.
- .TX → Taille horizontale de la fenêtre.
- .TY → Taille verticale de la fenêtre.
- .Valeur → **1** : Coché / **0** : décoché
- .CouleurFond → Couleur de fond RVB.
- .CouleurTexte → Couleur de du texte RVB.
- .Evenement → Fichier évènementiel des interactions graphiques.

Voici les arguments possible pour la propriété « .Parametres » :

- **IMGAUTO:***valeur entre 0 et 1*
  valeur Correspond au mode d'affichage de l'image
  → **IMGAUTO:0** Objet adapté aux dimensions TX et TY
  → **IMGAUTO:1** Objet adapté aux dimensions du texte

- **COL:***valeur entre 0 et 1*
  valeur Correspond au mode d'affichage de l'image
  → **COL:0** Transparent
  → **COL:1** (Par défaut) Affiche la couleur de fond

- **UPD:***valeur entre 0 et 1*
  Utiliser un thread d'actualisation graphique de la propriété .TEXT

Pour savoir si la case est cochée, il suffit de récupérer la propriété « .valeur » ou « .value »
Si elle est égale à 0 c'est qu'elle est décochée, si elle est égale à 1, alors elle est cochée

EXEMPLE SIMPLE : (LES GRANDES ESPACES SONT DES TABULATION, ET ILS NE SONT PAS OBLIGATOIRES)

```
Checkbox/ Mon_Checkbox
     .handle              = "%handle_Fenetre%"
     .Texte               = "Coche moi !"
     .Valeur              = "0"        → Décoché par défaut
     .PX                  = "5"
     .PY                  = "5"
     .tx                  = "100"
     .ty                  = "20"
     .CouleurFond         = "255,255,240"
     .CouleurTexte        = "000,000,000"
     creer/
Fin/ Checkbox
```

Correspondance anglaise :

```
Checkbox / My_bar
     .handle              = "%handle_Window%"
     .PX                  = "5"
     .PY                  = "5"
     .Sx                  = "100"
     .Sy                  = "20"
     .BackColor           = "255,255,240"
     .TextColor           = "000,000,000"
     .Value               = "0"          → No checked by default
     .text                = "Check me!"
     create/
End/ Checkbox
```

Un autre exemple complet dans la partie exemple, ICI

AUTRES INFORMATIONS :

VOIR AUSSI :

Fenetre/ ; Picturebox/ ; bouton/ ;

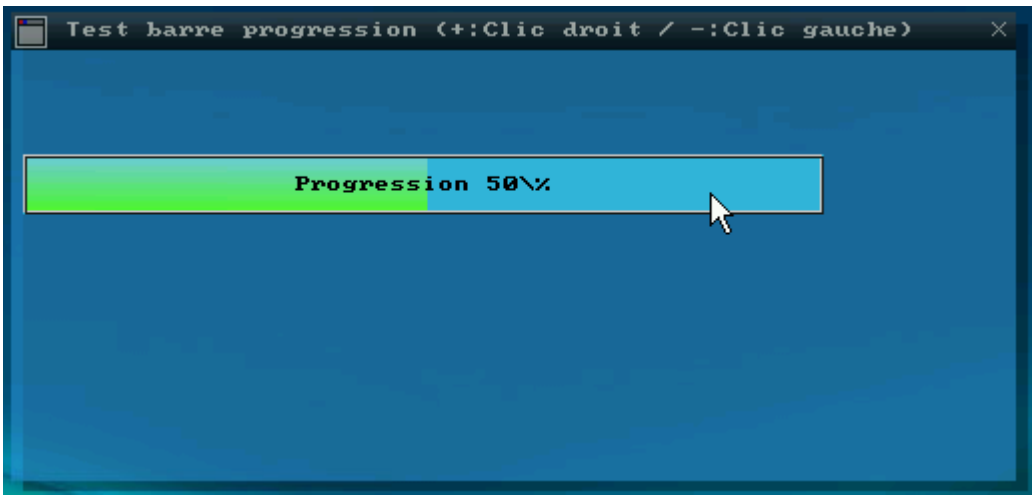
Créer une barre de progression

COMMANDE :

```
BarreProgression/ [Nom de la barre]
Fin/ BarreProgression
```

FONCTIONNALITE :

Cette commande permet de créer une nouvelle instance d'une barre de progression



PROPRIETES DISPONIBLE :

- .handle → Numéro de handle parent. (Fenêtre)
- .Nom → Le nom de l'objet.
- .Parametres → Paramètres et modes (**IMGAUTO:2 recommandé**)
- .PX → Position horizontale de la fenêtre.
- .PY → Position verticale de la fenêtre.
- .TX → Taille horizontale de la fenêtre.
- .TY → Taille verticale de la fenêtre.
- .Valeur → Valeur en pourcentage de 0% à 100%
- .CouleurFond → Couleur de fond RVB.
- .CouleurTexte → Couleur de du texte RVB.
- .Evenement → Fichier évènementiel des interactions graphiques.

Voici les arguments possible pour la propriété « .Parametres » :

- **IMGAUTO:***valeur entre 0 et 1*
  valeur Correspond au mode d'affichage de l'image
  → **IMGAUTO:0** affichage brute, l'image est coupée si elle est plus grande.
  → **IMGAUTO:1** Objet adapté aux dimensions de l'image source
  → **IMGAUTO:2** (par défaut) Affichage adapté aux dimensions de l'objet (Etire l'image a l'horizontale)

- **UPD:***valeur entre 0 et 1*
  Utiliser un thread d'actualisation graphique de la propriété .TEXT

EXEMPLE SIMPLE : (LES GRANDES ESPACES SONT DES TABULATION, ET ILS NE SONT PAS OBLIGATOIRES)

```
Fix/ Valeur = 50

BarreProgression/ Ma_barre
     .handle              = "%handle_Fenetre%"
     .Texte               = "Je suis a %valeur%"
     .Valeur              = "%Valeur%"
     .PX                  = "5"
     .PY                  = "5"
     .tx                  = "100"
     .ty                  = "20"
     .CouleurFond         = "200,255,240"
     .CouleurTexte        = "250,100,100"
     creer/
Fin/ ProgressBar
```

Correspondance anglaise :

```
Set/ Value = 50

ProgressBar/ My_bar
     .handle              = "%handle_Window%"
     .PX                  = "5"
     .PY                  = "5"
     .Sx                  = "100"
     .Sy                  = "20"
     .BackColor           = "200,255,240"
     .TextColor           = "250,100,100"
     .text                = "I'm at %value%"
     create/
End/ ProgressBar
```

Un autre exemple complet dans la partie exemple, ICI

AUTRES INFORMATIONS :

VOIR AUSSI :

Fenetre/ ; Picturebox/ ; bouton/ ;

Modifier un objet ou une fenêtre

COMMANDE :

```
[fenetre/, bouton/ textebox/…] /MODIF:[Nom ou handle de l'objet]
```

FONCTIONNALITE :

Ce paramètre permet de modifier les propriétés d'un objet ou d'une fenêtre. Grâce à ceci vous pouvez par exemple modifier en direct le **nom d'une fenêtre**, le **contenu d'un bouton**, l'**emplacement**, la **taille**, **couleurs**, **image** d'un objet et bien plus… !

EXEMPLE SIMPLE AVEC UNE FENETRE : (LES GRANDES ESPACES SONT DES TABULATION, ET ILS NE SONT PAS OBLIGATOIRES)

Après la création d'une fenêtre sous le nom de "Ma_Fenetre", il suffit de réutiliser la commande **FENETRE/** mais avec l'argument **/MODIF:** et de préciser le nom de la fenêtre "Ma_Fenetre" ou bien son numéro de handle, d'utiliser les propriétés dont vous avez besoin de modifier, et finir par **CREER/** pour regénérer la fenêtre.

Dans cet exemple, on va juste modifier le titre de la fenêtre et sa position en X.

```
Fenetre/ /Modif:Ma_Fenetre
     .Titre    = "Voici le nouveau titre"
     .PX       = "100"
     creer/
Fin/ Fenetre
```

Correspondance anglaise :

```
Window/ /Modif:Ma_Fenetre
     .Title    = "This is the new title"
     .PX       = "100"
     Create/
End/ Window
```

L'opération est exactement la même pour les picturebox, les textebloc, bouton etc…
IMPORTANT : Il est important d'utiliser cette commande **DANS LE MËME PROCESSUS** (PID).
Dans le cas où vous exécuter cette commande depuis un autre PID, il faudra indiquer l'objet via son numéro de handle.

Ah et petite information intéressante, les objets graphiques comme les boutons, les pictureBox, TexteBox, progressBar et bien plus, génère eux aussi un numéro de handle comme la fenêtre via la commande CREER/ 😉

AUTRES INFORMATIONS :

VOIR AUSSI :

Fenetre/ ; Picturebox/ ; bouton/ ;

Créer un évènement graphique

COMMANDE :

Il s'agit de la propriété " .event " ou " .evenement " présent lors de vos création graphiques.
Ainsi qu'une fonction de destination personnalisée. Je vous explique en dessous !

FONCTIONNALITE :

Ceci permet de pouvoir interagir avec l'interface graphique ← → le code.
Par exemple

- Si vous cliquez sur un bouton, il faut exécuter tel action.
- Si vous survoler une image avec la souris il faut changer l'image
- Si vous pressez la touche ECHAP sur une fenêtre, il faut la fermer
- Etc…

La propriété **.event** ou **.evenement** doit contenir le chemin d'un fichier qui contient une fonction à son nom. Cette fonction peut se trouver le même fichier actuellement exécuté, dans le même répertoire, ou bien ailleurs.

LISTE DES EVENEMENTS DISPONIBLE SUR CETTE VERSION :

- `.CLICK( )` → Pression sur n'importe quel bouton de la souris
  `.CLICK(`*`Arg1`*`)` → %Arg1%: numéro du bouton (1:Gauche 2:Droit 3:Les deux)

- `.MouseClick()` → Pression + Relâchement de sur n'importe quel bouton.
  `.MouseClick(Arg1)` → %Arg1% numéro du bouton (1:Gauche 2:Droit 3:Les deux)

- `.MouseEnter()` → Quand que la souris ENTRE dans l'objet graphique.

- `.MouseMove()` → Quand la souris survole/bouge DANS l'objet graphique.

- `.MouseLeave()` → Quand la souris SORT de l'objet graphique.

- `.KeyPress()` → Pression sur n'importe quel touche du clavier.
  `.KeyPress(Arg1)` → %Arg1%: Touche pressée.
- `.WindowClosing()` → Durant la fermeture de la fenêtre.
- `.WindowsClosed()` → Après la fermeture de la fenêtre.

Bien évidement vous pouvez utiliser un autre nom que "**Arg1**" 😉

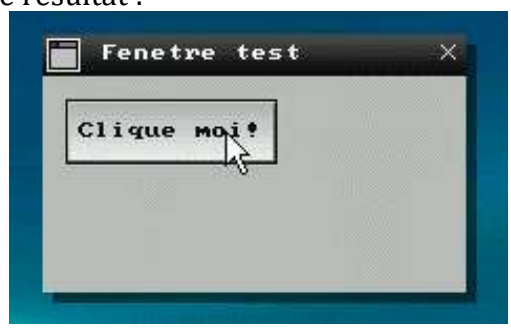
EXEMPLE COMPLET AVEC UN BOUTON:

Test.cpc *(à exécuter)*

```
Fenetre/ MaFenetre

     .Titre              = "Fenetre test"
     .CouleurTitre       = "255,255,255"
     .PX                 = "80"
     .PY                 = "80"
     .tx                 = "400"
     .ty                 = "300"
     @#Handle_Fenetre creer/
Fin/ Fenetre

Bouton/ MonBouton
     .Handle             = "%Handle_Fenetre%"
     .Parametres         = "IMGAUTO:2"
     .PX                 = "10"
     .PY                 = "10"
     .tx                 = "100"
     .ty                 = "30"
     .texte              = "Clique moi!"
     .Image              = "Bouton.PNG"
     .evenement          = "MonEvenement.cpc"
     creer/
Fin/ Bouton
```

La variable %Handle_Fenetre% contient un numéro de handle de la fenêtre "MaFenetre"

Voici le résultat :



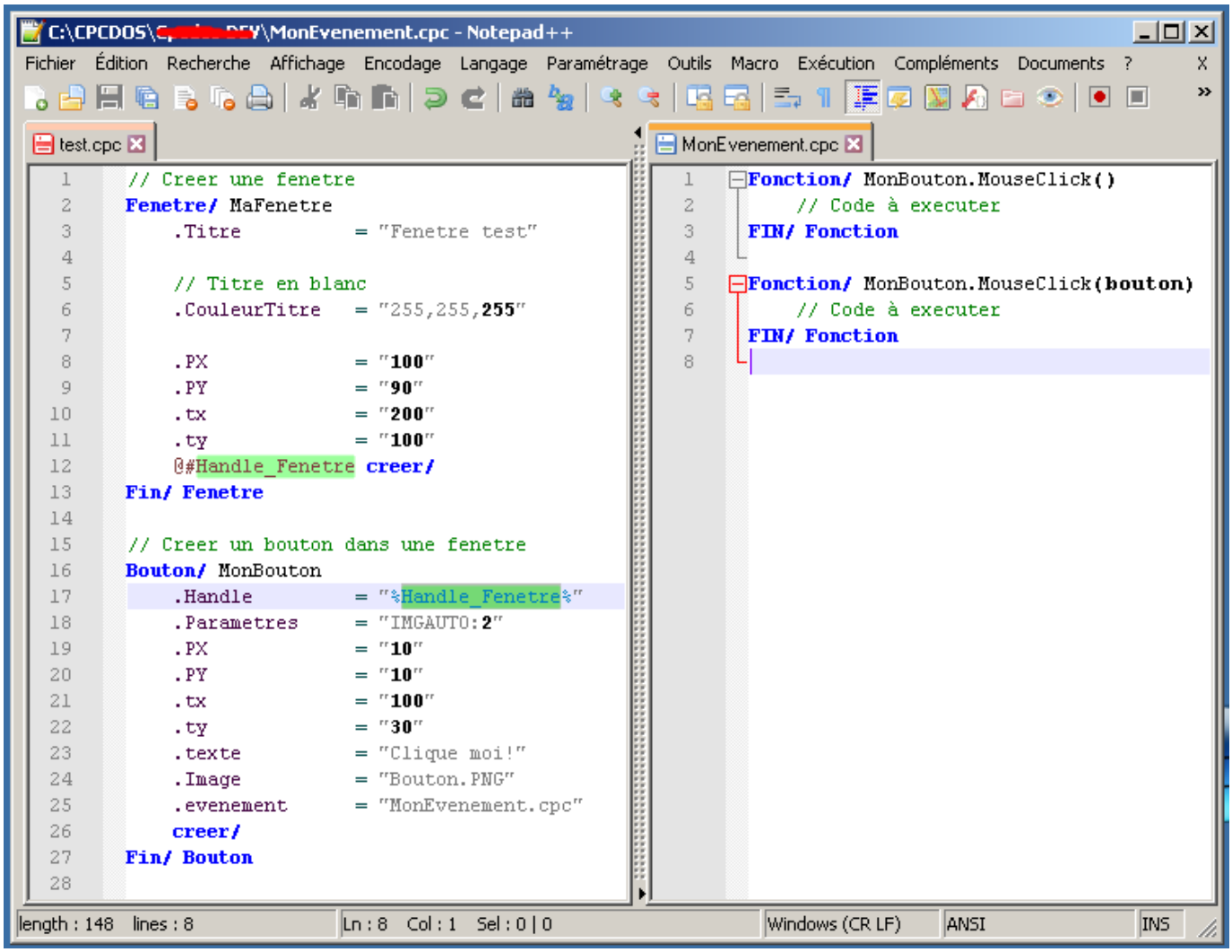
MonEvenement.cpc

```
Fonction/ MonBouton.MouseClick()
     // Code à executer
FIN/ Fonction

Fonction/ MonBouton.MouseClick(bouton)
     // Code à executer
FIN/ Fonction
```

La variable %bouton% contient le numéro de bouton de la souris

En bref, ici, nous avons précisé le fichier évènement du bouton sur "**MonEvenement.cpc**". Ce qui veut dire que si vous survolez, bougez la souris dedans, cliquer, double cliquer … Cpcdos va aller dans ce fichier MonEvenement.CPC.

Voici le résultat écrit :




```
// Creer une fenetre
Fenetre/ MaFenetre
    .Titre          = "Fenetre test"

    // Titre en blanc
    .CouleurTitre   = "255,255,255"

    .PX             = "100"
    .PY             = "90"
    .tx             = "200"
    .ty             = "100"
    @#Handle_Fenetre creer/
Fin/ Fenetre

// Creer un bouton dans une fenetre
Bouton/ MonBouton
    .Handle         = "%Handle_Fenetre%"
    .Parametres     = "IMGAUTO:2"
    .PX             = "10"
    .PY             = "10"
    .tx             = "100"
    .ty             = "30"
    .texte          = "Clique moi!"
    .Image          = "Bouton.PNG"
    .evenement      = "MonEvenement.cpc"
    creer/
Fin/ Bouton
```

```
Fonction/ MonBouton.MouseClick()
    // Code à executer
FIN/ Fonction

Fonction/ MonBouton.MouseClick(bouton)
    // Code à executer
FIN/ Fonction
```

DONC, dans le cas où vous **CLIQUEZ** (Pression + relâchement) Cpcdos va chercher dans le fichier "MonEvenement.cpc" la ligne suivante :

```
Fonction/ MonBouton.MouseClick()
```

Et / Ou

```
Fonction/ MonBouton.MouseClick(argument1)
```

Si l'une de ces lignes sont manquantes, Cpcdos ne vas pas gueuler, mais l'erreur sera indiquée dans le DEBUG.
Pour ce coup-là, les deux lignes sont présentes, Cpcdos va donc exécuter les deux.

« argument1 » : Correspond au bouton cliqué, **1 : Gauche 2 : Droit 3 : Les deux.**
Bien évidement vous pouvez mettre un autre nom que "argument1" !

OPTIMISATION :

Afin d'améliorer les performances du noyau, aux fichiers non précompilés, vous pouvez préciser les noms d'événements à chercher dans le fichier source.
Reprenons l'exemple du bouton ci-dessus, cette fois-ci dans la ligne «**.evenement**» après le nom de fichier, vous pouvez spécifier les noms d'événements à chercher comme CLIC, KeyPress etc...
Dans ce cas-là, pour le bouton, nous pouvons nous contenter d'un simple "**MouseClick**" et pourquoi pas d'un "**KeyPress**" si la touche entrée est pressée.

```
Bouton/ MonBouton
      .Handle              = "%Handle_Fenetre%"
      .Parametres          = "IMGAUTO:2"
      .PX                  = "10"
      .PY                  = "10"
      .tx                  = "100"
      .ty                  = "30"
      .texte               = "Clique moi!"
      .Image               = "Bouton.PNG"
      .evenement           = "MonEvenement.cpc=MouseClick,KeyPress"
      creer/
Fin/ Bouton
```

AUTRES INFORMATIONS :

VOIR AUSSI :

Fenetre/ ; Picturebox/ ; bouton/ ;

Liste des fonctions CpcdosC+

Voici ici la liste des fonctions CpcdosC+ inclus nativement dans le CRT de Cpcdos (SHELL).
Vous trouverez également la liste dans le fichier **KRNL\INIT\INIT_CRT.CPC**

Nom de fonction	Description	Retour
Manipulation de chaines de caractères		
CPC.INSTR(Texte, Compare)	Recherche la position d'un caractère ou d'une chaine de caractère depuis le ***Texte*** source. *Semblable au INSTR() de FreeBasic/VB/QB.* Retourne :	**«0»** : Introuvable **Supérieur à «0»** : Position de la trouvaille.
CPC.INSTR(Debut, Texte, Compare)	IDEM, mais vous pouvez choisir une position de départ.	*idem*
CPC.INSTRREV(Debut, Texte, Compare)	IDEM, mais REVersé, on cherche la position du caractère depuis la fin!	*idem*
CPC.MID(Texte, Debut)	Découpe une chaine de caractères depuis la position de ***Debut*** jusqu'à la fin, depuis le ***Texte*** source.	Retourne une chaine de caractère découpé ou non.
CPC.MID(Texte, Debut, Fin)	IDEM, mais avec une position de ***FIN*** relatif à la position de ***Debut***.	*idem*
CPC.MIDA(Texte, Debut, Fin)	IDEM, mais avec une position de ***FIN*** absolue. *(à partir du début)*	*idem*
CPC.LEN(Texte) CPC.TAILLE(Texte)	Compte le nombre de caractère dans le ***Texte***.	**«0»** : Chaine vide. **Supérieur à «0»** : Nombre de caractères.
CONVERSIONS		
CPC.MAJ(Texte)	Convertit tous les caractères en caractères MAJuscules.	Retourne une chaîne de caractères en majuscules.
CPC.MIN(Texte)	Convertit tous les caractères en caractères MINuscules.	Retourne une chaîne de caractères en minuscules.
CPC.SIGNE(Texte) CPC.SIGN(Texte)	Connaître le signe d'un nombre.	«1» : Signe positif «0» : Signe négatif
CPC.ENTIER(Valeur) CPC.INTEGER(Valeur) CPC.INT(Valeur)	Convertir un nombre quelconque en ENTIER. *Supprime la virgule*	Retourne un nombre entier.
CPC.FRAC(Valeur)	FRACtionne le nombre pour récupérer uniquement les chiffres APRES la virgule.	Retourne un nombre à virgule.
CPC.VAL(Valeur)	Convertit une unité, en décimale. Exemple : Binaire en décimale **CPC.VAL(&B11100)** Hexadécimal en décimale **CPC.VAL(&H1C)** Octal en décimale **CPC.VAL(&34)** Retournera **28**	Retourne un nombre.
CPC.CHR(Valeur) CPC.CAR(Valeur) CPC.CARACTERE(Valeur)	Convertit un nombre ASCII en caractère ASCII. La valeur doit être située entre 1 et 255.	Retourne un caractère ASCII.
CPC.ASC(Valeur) CPC.ASCII(Valeur)	Convertit un caractère ASCII en une valeur ASCII. 1 seul caractère est pris en compte. Ou bien le premier caractère d'une chaine.	Retourne un nombre entre 0 et 255
CPC.HEX(Valeur) CPC.HEXA(Valeur) CPC.HEXADECIMAL(Valeur) CPC.HEXADECIMALE(Valeur)	Convertit un nombre (entier) en une chaine de caractère hexadécimale.	Retourne une chaine de caractères HEXA.
CPC.ABS(Valeur)	Convertit un nombre en une valeur absolue.	Retourne une valeur TOUJOURS positive.
Fonctions mathématiques		

CPC.LOG(Valeur)	Logarithme	
CPC.EXP(Valeur)	Exponentiel	
CPC.SQR(Valeur) CPC.RAC(Valeur)	Racine carré	
CPC.COS(Valeur)	Cosinus	
CPC.SIN(Valeur)	Sinus	
CPC.TAN(Valeur)	Tangente	
CPC.ACOS(Valeur)	Arc Cosinus	
CPC.ASIN(Valeur)	Arc Sinus	
CPC.ATAN(Valeur)	Arc Tangente	
CPC.ATANR(ValeurY, ValeurX)	Tangente ratio	
Fonctions réseaux		
CPC.NET.PING(AdresseIP)	Tester l'existence d'une machine sur le réseau	**«0»** ou **+** : temps en millisecondes «-1» : Pas de réponse «-2» : Err. config socket «-3» : Impssbl. Créer socket «-4» : Err. binding «-5» : Ecoute impssbl. «-6» : Err. FD «-7» : Err. Lecture FD «-8» : Err. DNS «-9» : Err. Mémoire
CPC.NET.		
CPC.NET.		
Autres fonctions internes		
CPC.FICHIER_EXISTE(Chemin) CPC.FILE_EXIST(Chemin)	Teste l'existence d'un fichier indiqué dans le ***Chemin***.	**«1»** : Existe. **«0»** : N'existe pas.
CPC.TAILLE_FICHIER(Chemin) CPC.FILE_SIZE(Chemin)	Calcule la taille d'un fichier indiqué dans le ***Chemin***.	Le nombre d'octets présent dans le fichier.
CPC.LIRE_FICHIER(Chemin) CPC.READ_FILE(Chemin)	Récupère le contenu complet d'un fichier indiqué dans le ***Chemin***. Conseillé d'utiliser : OUVRIR/ ou OPEN/	Les données du fichier. (Texte, binaire...)

Variables d'environnements

Voici la liste des variables interne à Cpcdos.
Vous pouvez utiliser la commande txt/ afin de voir le contenu de ces variables.

Pour utiliser les variables de manière dynamiques dans les propriétés .TEXT des objets il suffit d'entourer vos variables à la Linux, par `${ }`
Exemple : `.text = "Heure : ${CPC.HEURE}"`
Ce qui permet par exemple d'afficher l'heure, date, CPU, RAM dans un label sur vos barre des tâches.

Exemple pour afficher la mémoire **U**tilisée :
TXT/ %CPC.SYS.MEM**U**%

→ S'il s'agit de variables :

- **D** : Dynamiques *et donc **non modifiable** par l'utilisateur.*
  → Ce sont des variables qui changent constamment de contenu.
  Commencent par "CPC."

- **DM** : Dynamiques et Modifiable par l'utilisateur.
  → Ce sont des variables changent constamment de contenu, et qui peuvent être modifié par l'utilisateur.
  Commencent par "CPC_"

- **F** : Fixes *et donc **non modifiable** par l'utilisateur.*
  → Ce sont des variables qui ont un contenu fixe depuis le démarrage du système.
  Commencent par "CPC."

- **FM** : Fixes et Modifiable par l'utilisateur.
  → Ce sont des variables qui ont un contenu fixe depuis le démarrage du système et qui peuvent être modifié par l'utilisateur.
  Commencent par "CPC_"

#	Anglais	Français	Description
	Interface graphique		
FM	SCR_MODE	SCR_MODE	Résolution CONSOLE compatible Voir ici
FM	SCR_RES	SCR_RES	Résolution d'écran supportée par votre carte graphique pour connaître, voir SYS/ /ECRAN
FM	SCR_BIT	SCR_BIT	Nombre de couleurs **16**, **24** et **32** bits
FM	SCR_COLOR	SCR_COLOR	Couleur d'arrière-plan du bureau. Ex: 050,150,250
FM	SCR_IMG	SCR_IMG	Fond d'écran du bureau (JPG,PNG,GIF)
FM	SCR_IMG_MODE	SCR_IMG_MODE	Mode d'affichage du fond d'écran

			0 : "**FILL**" (Par défaut) Remplir au mieux l'écran sans déformer l'image mais autorise le débordement. 1 : "**AJUSTED**" IDEM mais n'autorise pas le débordement. Ceci peut faire des zones vides de couleurs. 2 : "**ADAPTED**" Adapté à la taille de l'écran, mais peut déformer l'image. 3 : "**MOZAIC**" Répète l'image sans la modifier en replissant l'écran. 4 : "**CENTER RAW**" Affiche l'image au centre et sans modifications. 4 : "**RAW**" Affiche l'image à l'origine 0,0 et sans modifications.
FM	**CPC_GUI.WINDOW.TITLE**	**CPC_GUI.WINDOW.TITLE**	Titre de la fenêtre sans titre
FM	**CPC_GUI.WINDOW.TITLE_PX**	**CPC_GUI.WINDOW.TITLE_PX**	Position X du texte des fenêtres
FM	**CPC_GUI.WINDOW.TITLE_PY**	**CPC_GUI.WINDOW.TITLE_PY**	Position Y du texte des fenêtres
FM	**CPC_GUI.WINDOW.TITLE_TY**	**CPC_GUI.WINDOW.TITLE_TY**	Taille Y de la barre de titre des fenêtres
FM	**CPC_GUI.WINDOW.TITLE_RGB**	**CPC_GUI.WINDOW.TITLE_RGB**	Couleur du titre des fenêtres
FM	**CPC_GUI.WINDOW.ICO**	**CPC_GUI.WINDOW.ICO**	Icône par défaut des fenêtres
FM	**CPC_GUI.WINDOW.ICO_PX**	**CPC_GUI.WINDOW.ICO_PX**	Position X de l'icône des fenêtres
FM	**CPC_GUI.WINDOW.ICO_PY**	**CPC_GUI.WINDOW.ICO_PY**	Position Y de l'icône des fenêtres
FM	**CPC_GUI.WINDOW.ICO_S**	**CPC_GUI.WINDOW.ICO_S**	Changer la taille de l'icône ou pas. 0 : Ne pas changer la taille 1 : Changer la taille
FM	**CPC_GUI.WINDOW.ICO_SX**	**CPC_GUI.WINDOW.ICO_SX**	Nouvelle taille en X Seulement si **ICO_S** = 1
FM	**CPC_GUI.WINDOW.ICO_SY**	**CPC_GUI.WINDOW.ICO_SY**	Nouvelle taille en Y Seulement si **ICO_S** = 1
FM	**CPC_GUI.WINDOX.BORDER**	**CPC_GUI.WINDOX.BORDER**	Bordure des fenêtres 0 : Non 1 : Oui
~~**FM**~~	~~**CPC_GUI.WINDOX.BORDER_RGB**~~	~~**CPC_GUI.WINDOX.BORDER_RGB**~~	~~Couleur de la bordure des fenêtres~~ ~~Seulement si **BORDER** = 1~~ ~~ex :~~ 020,020,020
FM	**CPC_GUI.WINDOW.ALPHA**	**CPC_GUI.WINDOW.ALPHA**	Transparence par défaut des fenêtres
FM	**CPC_GUI.WINDOW.SHADOW_PX**	**CPC_GUI.WINDOW.SHADOW_PX**	Position X relative de la position de l'ombre sous les fenêtres

FM	CPC_GUI.WINDOW.SHADOW_PY	CPC_GUI.WINDOW.SHADOW_PY	Position Y relative de la position de l'ombre sous les fenêtres
FM	CPC_GUI.WINDOW.SHADOW	CPC_GUI.WINDOW.SHADOW	Opacité de l'ombre sous les fenêtres 0 : Aucune ombre 0 → 254 255 : Opaque
FM	CPC_GUI.WINDOW.SHADOW_RGB	CPC_GUI.WINDOW.SHADOW_RGB	Couleur de l'ombre ex: 050,090,150
FM	CPC_GUI.WINDOW.RGB	CPC_GUI.WINDOW.RGB	Couleur des fenêtres
FM	CPC_GUI.WINDOW.BACKGOUND_RGB	CPC_GUI.WINDOW.BACKGOUND_RGB	Couleur de fond des fenêtres
FM	CPC_GUI.TEXT_RGB	CPC_GUI.TEXT_RGB	Couleur des textes en général
FM	CPC_GUI.RGB	CPC_GUI.WINDOW.RGB	Couleur en général
		Contextes	
F	CPC.CRLF	CPC.CRLF	Retour chariot Touche entré. ASCII 13 + ASCII 10 ***« \r\n Windows »***
F	CPC.LFCR	CPC.LFCR	Retour chariot spécial. *Non utilisé* ASCII 10 + ASCII 13
F	CPC.CR	CPC.CR	Retour chariot. Touche entré ASCII 13 ***« \r APPLE »***
F	CPC.LF	CPC.LF	Retour chariot. Touche entré ASCII 10 ***« \n UNIX »***
D	CPC.RND	CPC.RND	Nombre décimal aléatoire
F	CPC.DIR	CPC.REP	Répertoire courant du noyau
F	CPC.DIR.KRNL	CPC.REP.KRNL	Répertoire configuration Kernel
F	CPC.DIR.KRNL.INIT	CPC.REP.KRNL.DIR	Répertoire d'initialisation kernel
FM	CPC_TEMP	CPC_TEMP	Répertoire temporaire du système
FM	CPC_TEMP.NET	CPC_TEMP.NET	Répertoire temporaire des interactions réseaux (Site web etc…)
		Temps date et heure	
D	CPC.HOUR	CPC.HEURE	Temps heure uniquement
D	CPC.MIN	CPC.MIN	Temps minutes uniquement
D	CPC.MINUTS	CPC.MINUTES	Temps minutes uniquement
D	CPC.SEC	CPC.SEC	Temps secondes uniquement
D	CPC.SECONDS	CPC.SECONDES	Temps secondes uniquement
D	CPC.DAY	CPC.DAY	Jour actuel
D	CPC.MONTH	CPC.MOIS	Mois actuel
D	CPC.YEAR	CPC.ANNEE	Année actuel
D	CPC.CENTURY	CPC.SIECLE	Siècle actuel

D	CPC.DATE	CPC.DATE	Date actuel sous la forme **JJ/MM/AAAA**
FM	CPC_DATE.FORMAT	CPC_DATE.FORMAT	Format de la date (ex: **JJ/MM/AAAA**)
D	CPC.TIME	CPC.TIME	Temps Heure, minutes, secondes sous la forme **HH :MM :SS**
FM	CPC_TIME.FORMAT	CPC_TIME.FORMAT	Format de l'heure (ex: **HH:MM:SS**)
D	CPC.TIMER	CPC.TEMPS	Temps en millisecondes écoulés depuis 1970
Système : Mémoire			
D	CPC.SYS.MEMU.P	CPC.SYS.MEMU.P	Mémoire utilisée en %
D	CPC.SYS.MEMU	CPC.SYS.MEMU	Mémoire utilisée en octets
D	CPC.SYS.MEMU.B	CPC.SYS.MEMU.O	Mémoire utilisée en octets
D	CPC.SYS.MEMU.K	CPC.SYS.MEMU.K	Mémoire utilisée en Ko
D	CPC.SYS.MEMU.M	CPC.SYS.MEMU.M	Mémoire utilisée en Mo
D	CPC.SYS.MEMU.G	CPC.SYS.MEMU.G	Mémoire utilisée en Go
D	CPC.SYS.MEMU.T	CPC.SYS.MEMU.T	Mémoire utilisée en To
D	CPC.SYS.MEM.P	CPC.SYS.MEM.P	Mémoire libre en %
D	CPC.SYS.MEM	CPC.SYS.MEM	Mémoire libre en octets
D	CPC.SYS.MEM.B	CPC.SYS.MEM.O	Mémoire libre en octets
D	CPC.SYS.MEM.K	CPC.SYS.MEM.K	Mémoire libre en Ko
D	CPC.SYS.MEM.M	CPC.SYS.MEM.M	Mémoire libre en Mo
D	CPC.SYS.MEM.G	CPC.SYS.MEM.G	Mémoire libre en Go
D	CPC.SYS.MEM.T	CPC.SYS.MEM.T	Mémoire libre en To
Système : CPU			
D	CPC.SYS.CPU.ACT	CPC.SYS.CPU.ACT	Activité en % de(s) CPU(s)
F	CPC.SYS.CPU.NB	CPC.SYS.CPU.NB	Nombre de processeurs détectés (APIC)
D	CPC.SYS.CPU.NB.ACT	CPC.SYS.CPU.NB.ACT	Nombre de processeurs en activité
F	CPC.SYS.CPU.CPUID	CPC.SYS.CPU.CPUID	Si CPUID supporté
F	CPC.SYS.CPU.FPU	CPC.SYS.CPU.FPU	Si coprocesseur mathématique supporté
F	CPC.SYS.CPU.VME	CPC.SYS.CPU.VME	Si Virtual Machine Extension supportée
F	CPC.SYS.CPU.SSE	CPC.SYS.CPU.SSE	Si SSE supporté
F	CPC.SYS.CPU.SSE2	CPC.SYS.CPU.SSE2	Si SSE2 supporté
F	CPC.SYS.CPU.SSE3	CPC.SYS.CPU.SSE3	Si SSE3 supporté
F	CPC.SYS.CPU.SSE4	CPC.SYS.CPU.SSE4	Si SSE4 supporté
F	CPC.SYS.CPU.SSE4A	CPC.SYS.CPU.SSE4A	Si SSE4A supporté
F	CPC.SYS.CPU.3DNOW	CPC.SYS.CPU.3DNOW	Si 3DNow supporté
F	CPC.SYS.CPU.3DNOW2	CPC.SYS.CPU.3DNOW2	Si 3DNow2 supporté
F	CPC.SYS.CPU.MMX	CPC.SYS.CPU.MMX	Si l'extension Multimédia supportée
F	CPC.SYS.CPU.RDTSCP	CPC.SYS.CPU.RDTSCP	Si Read Time Stamp Counter supporté
F	CPC.SYS.CPU.PAE	CPC.SYS.CPU.PAE	Si l'extension d'adresse physique supportée
F	CPC.SYS.CPU.HYPERTHREADING	CPC.SYS.CPU.HYPERTHREADING	Si l'hyper threading supporté
F	CPC.SYS.CPU.MULTIPROCESSOR	CPC.SYS.CPU.MULTIPROCESSEUR	Si multi processeur supporté

F	CPC.SYS.CPU.MULTICORE	CPC.SYS.CPU.MULTICOEUR	Si multi cœur supporté
F	CPC.SYS.CPU.X64	CPC.SYS.CPU.X64	Si 64bits supporté
F	CPC.SYS.CPU.FAMILLY	CPC.SYS.CPU.FAMILLE	Famille du processeur
F	CPC.SYS.CPU.MODEL	CPC.SYS.CPU.MODELE	Modèle du processeur
F	CPC.SYS.CPU.ID	CPC.SYS.CPU.ID	ID du processeur
F	CPC.SYS.CPU.VENDOR	CPC.SYS.CPU.VENDEUR	Vendeur (Intel, amd…)
	CPC.SYS.CPU.NAME	CPC.SYS.CPU.NOM	Nom complet du processeur
F	CPC.SYS.CPU.APIC	CPC.SYS.CPU.APIC	Si APIC supporté
	CPC.SYS.CPU.APIC.OK	CPC.SYS.CPU.APIC.OK	Si driver APIC installé
F	CPC.SYS.CPU.APIC.VER	CPC.SYS.CPU.APIC.VER	Version de l'APIC + pilote
D	CPC.SYS.CPU.APIC.BAT	CPC.SYS.CPU.BAT	Etat en % de la batterie du système
Système : BIOS Advanced Power Management			
D	CPC.SYS.APM	CPC.SYS.APM	Si APM supporté
D	CPC.SYS.APM.OK	CPC.SYS.APM.OK	Si driver APM installé
D	CPC.SYS.APM.RM	CPC.SYS.APM.RM	Si Real mode activé
D	CPC.SYS.APM.PM16	CPC.SYS.APM.PM16	Si protect mode 16b activé
D	CPC.SYS.APM.PM32	CPC.SYS.APM.PM32	Si protect mode 32b activé
D	CPC.SYS.APM.CPU_IDLE	CPC.SYS.APM.CPU_IDLE	0 : Interruption timer 1 : Timer lent
D	CPC.SYS.APM.CPU_BUSY	CPC.SYS.APM.CPU_BUSY	Si driver restore vitesse CPU
D	CPC.SYS.APM.STATE	CPC.SYS.APM.STATE	Etat APM : 1 : Veille / 2 : Suspension 3 : En attêt / 4 : Reboot
D	CPC.SYS.APM.POWER_MANAGEMENT	CPC.SYS.APM.POWER_MANAGEMENT	Si fonction activé
D	CPC.SYS.APM.POWER_MANAGEMENT_DEVICE	CPC.SYS.APM.POWER_MANAGEMENT_DEVICE	Si fonction activé
D	CPC.SYS.APM.SIGNATURE	CPC.SYS.APM.SIGNATURE	Signature APM
D	CPC.SYS.APM.VERSION	CPC.SYS.APM.VERSION	Version APM
Système : ISR			
F	CPC.SYS.ISR.INST	CPC.SYS.ISR.INST	'**0x0027**' si installé
D	CPC.SYS.ISR.DEM	CPC.SYS.ISR.DEM	'**1**' si démarré sinon '**0**'
DM	CPC_SYS.ISR	CPC.SYS.ISR	'**1**' si activé sinon '**0**' ou '**-1**'
Système : Threads			
D	CPC.SYS.THREAD.NB	CPC.SYS.THREAD.NB	Nombre de threads en cours
D	CPC.SYS.THREAD.NAME	CPC.SYS.THREAD.NOM	Nom du thread en cours
D	CPC.SYS.THREAD.ID	CPC.SYS.THREAD.ID	ID du thread en cours
D	CPC.SYS.PROCESS.ID	CPC.SYS.PROCESSUS.ID	ID du processus en cours
	CPC.SYS.PROCESS.NAME	CPC.SYS.PROCESSUS.NOM	Nom du processus en cours
D	CPC.SYS.USER.ID	CPC.SYS.UTILISATEUR.ID	ID de l'utilisateur en cours
	CPC.SYS.USER.NAME	CPC.SYS.USER.NOM	Nom de l'utilisateur en cours
D	CPC.SYS.OS.ID	CPC.SYS.OS.ID	ID de l'OS en cours
	CPC.SYS.OS.NAME	CPC.SYS.OS.NOM	Nom de l'OS en cours
D	CPC.SYS.KERNEL.ID	CPC.SYS.NOYAU.ID	ID du noyau en cours
D	CPC.SYS.KERNEL.NAME	CPC.SYS.NOYAU.NOM	Nom du noyau en cours
D			
D	CPC.SYS.THREAD.	CPC.SYS.THREAD.	
Système : Réseau			

F	CPC.SYS.NET.INST	CPC.SYS.NET.INST	'**1**' si installé sinon '**0**'
F	CPC.SYS.NET.DRV	CPC.SYS.NET.DRV	Nom du pilote utilisé + Carte
D	CPC.SYS.NET.ACT	CPC.SYS.NET.ACT	Activité en % du réseau.
D	CPC.SYS.NET.THREAD	CPC.SYS.NET.THREAD	Thread utilisant le réseau.
D	CPC.SYS.NET.REC	CPC.SYS.NET.REC	Nombre de paquets reçus par sec.
D	CPC.SYS.NET.SND	CPC.SYS.NET.ENV	Nombre de paquets envoyés par sec.
D	CPC.SYS.NET.CLT	CPC.SYS.NET.CLT	Nombre de clients/machines connectés
FM	CPC_SYS.NET.ICMP	CPC_SYS.NET.ICMP	Trame ICMP personnalisée.
FM	CPC_SYS.NET.USERAGENT	CPC_SYS.NET.USERAGENT	User agent pour requêtes HTTP
FM	CPC_SYS.NET.HTTP	CPC_SYS.NET.HTTP	Version http (HTTP/1.0 , HTTP1.1 ...)
	CPC.SYS.NET.	CPC.SYS.NET.	
Système : Affichage			
D	CPC.SCR.X	CPC.SCR.X	Nombre de pixels en X
D	CPC.SCR.Y	CPC.SCR.Y	Nombre de pixels en Y
D	CPC.SCR.XY	CPC.SCR.XY	Résolution du type hauteur **X** largeur
D	CPC.SCR.YX	CPC.SCR.YX	Résolution du type largeur **X** hauteur
D	CPC.SCR.BITS	CPC.SCR.BITS	Nombre de bits par pixels (16, 24, 32)
D	CPC.SCR.MEM	CPC.SCR.MEM	Taille en octets du buffer vidéo
D	CPC.SCR.HZ	CPC.SCR.HZ	Fréquence en Hertz
D	CPC.SCR.DRV	CPC.SCR.DRV	Nom du pilote utilisé
D	CPC.SCR.PTR	CPC.SCR.PTR	Adresse buffer vidéo
D	CPC.SCR.MODE	CPC.SCR.MODE	Mode vidéo (Console)
Système : Version			
F	CPC.VER	CPC.VER	Version actuelle de Cpcdos
F	CPC.VER.BUILD	CPC.VER.BUILD	Numéro de BUILD
F	CPC.VER.MAJ	CPC.VER.MAJ	Numéro de version Majeure
F	CPC.VER.MIN	CPC.VER.MIN	Numéro de version Mineure
F	CPC.VER.DATE	CPC.VER.DATE	Date de déploiement officielle
F	CPC.VER.CCP	CPC.VER.CCP	Version du moteur CpcdosC+
F	CPC.VER.SCI	CPC.VER.SCI	Version du service graphique
F	CPC.VER.CONSOLE	CPC.VER.CONSOLE	Version de la console d'interprétation
F	CPC.ABOUT	CPC.ABOUT	A propos de Cpcdos

Qu'es ce que le niveau de visibilité ?

C'est un point fort du noyau Cpcdos, c'est de pouvoir gérer les variables autrement que les versions antérieures à la OS2.0.5 Alpha 3.9.5, d'une manière fiable, en se rapprochant à la programmation POO. Concrètement, les variables peuvent désormais avoir une vie privée, ou jouer les stars !

Explications des niveaux 1, 2, 3, 4, 5 :

Exemple avec la population/popularité humaine

1. Je suis dans ma chambre, personne ne connaît mon existence.
2. Il y a seulement les occupants de ma maison et donc toutes les autres chambres qui connaissent mon existence.
3. Il y a seulement ma ville, les autres maisons et leur chambre qui connaissent mon existence, je ne suis que peu populaire.
4. Mon pays seulement, donc toutes les villes, toutes les maisons et leur chambre qui connaît mon existence, je ne suis pas mal populaire là non ?
5. Toute la terre entière, tous les pays, toutes les maisons et leurs chambres me connaissent, je suis une star ! Je suis Jules... Ok je sors.

Exemple avec un roi / richesse

1. Je suis un pauvre, je n'ai donc pas le droit de voir le roi. Ni voir d'autres pauvres.
2. Je suis normal, je peux voir les pauvres mais toujours pas le roi. Ni voir d'autres normaux.
3. Je suis assez riche, je peux voir les normaux, et le pauvre, mais pas le roi.
4. Je suis très riche, je peux voir les autres riches, les normaux et les pauvres mais pas le roi.
5. Je suis le roi, je peux voir tout le monde !

Non mais … plus sérieusement

1. **NIVEAU FONCTION** :
   Vos variables sont enregistrées ET visibles seulement dans vos fonctions (Que l'application ne peut pas voir).

2. **NIVEAU APPLICATION** :
   Vos variables sont enregistrées ET visibles seulement dans l'application (processus) **et ses** fonctions.

3. **NIVEAU UTILISATEUR** :
   Vos variables sont enregistrées ET visibles seulement dans le compte utilisateur courant. Donc toutes les applications et leurs fonctions peuvent les vois.

4. **NIVEAU OS** :
   Vos variables sont enregistrées ET visibles par tout votre système d'exploitation. Donc tous les utilisateurs, leurs applications et leurs fonctions peuvent les vois.

5. **NIVEAU KERNEL** :
   Vos variables n'ont plus aucune vie privée, elles sont enregistrées dans le KERNEL, ET visibles par tous les systèmes d'exploitation en cours d'exécution, leurs utilisateurs, leurs applications et leurs fonctions.

…et exemple brève avec le niveau 2. (Une simple application)

Dans votre programme MonProg.cpc, vous créez une variable nommée %Toto% qui contient « 123 ». Par défaut %Toto% est de niveau 2, donc seulement votre application MonProg.cpc et les fonctions à l'intérieur du programme peuvent voir votre variable %Toto%.

Jusqu'à la ok ?

Et si par hasard, en même temps vous exécutez un autre programme nommé Tata.cpc qui crée aussi une variable de niveau 2, nommé de même nom %Toto% qui contient « abc ».

Du coup Tata.cpc modifie la variable que MonProg.cpc à crée ?!

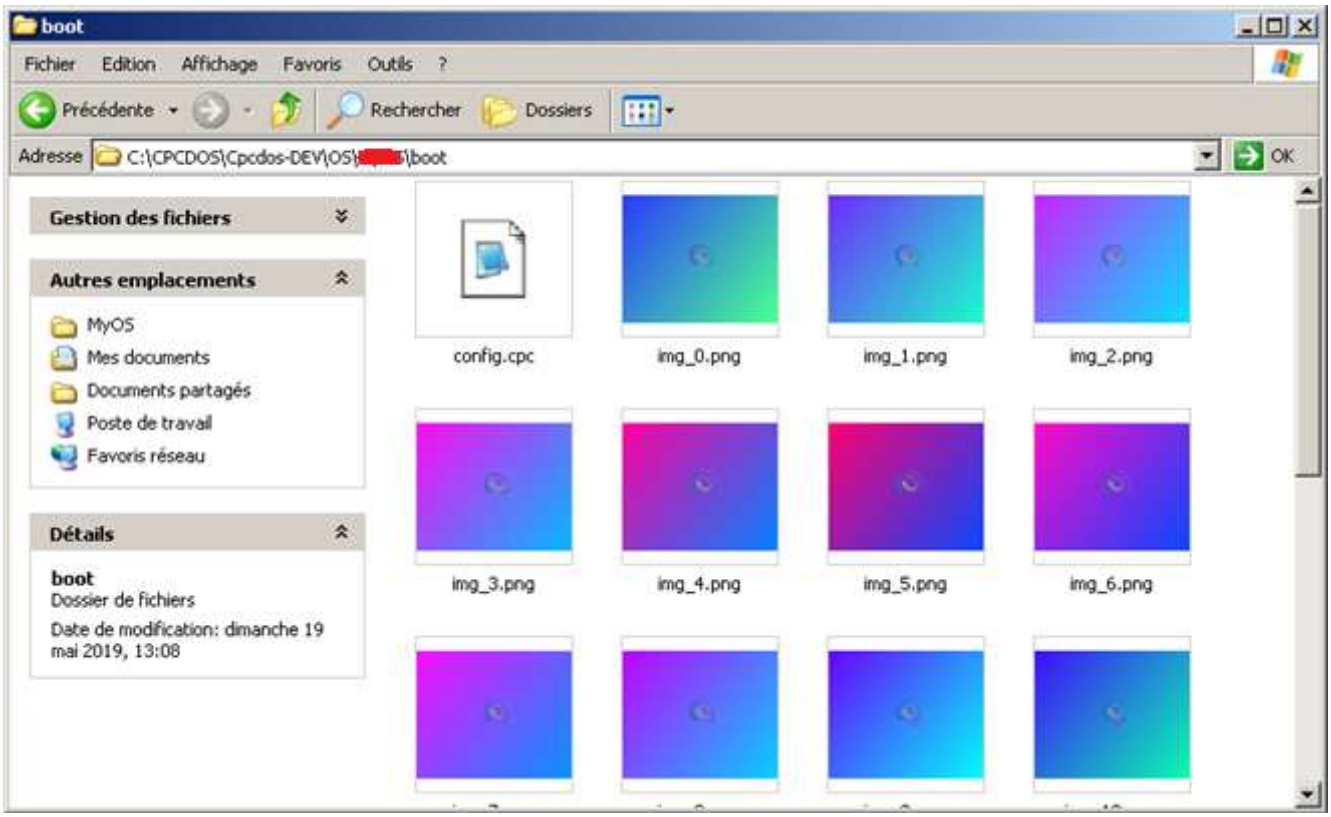
Et bien non ! %Toto% **de** MonProg.cpc et puis %Toto% **de** Tata.cpc ont **CHACUN LEUR PROPRE ESPACE DE MEMOIRE** donc chacun leur %Toto%, impossible donc que Tata.cpc modifie les variables de MonProg.cpc ! Ils sont dans une application différente, ils sont privés.

Ok ! Mais si je relance MonProg.cpc, es ce que ça va modifier l'ancien %Toto% ?

La réponse est : **NON** ! Lancer ou relancer la même application re-**génère toujours** un PID *(Processus ID)* et TID *(Thread ID)* **différents** et donc une emprunte numérique d'identifications des données **différente (Un nouvel espace mémoire)**, ne permet pas d'accéder aux variables de niveau 1 et 2 d'une autre application. Même s'il a le même emplacement et nom d'exécutable et nom de variables.
Car Cpcdos permet l'exécution de plusieurs instances de mêmes programmes sans faire d'interférences entre les variables de niveau 1 et 2. Exactement comme nos systèmes actuels ☺

Créer son Boot Screen

Rendez-vous dans le dossier « boot » de votre OS



Ce dossier contient un exemple de base, soit le nécessaire pour afficher un écran de démarrage animé.

Extrait du fichier config.cpc

```
4
5   CCP/ /SET.LEVEL = 5
6
7
8   // [FR] 0:Desactive / 1:Active  le screen boot
9   // [EN] 0:Disable   / 1:Enable  screen boot
10  set/ cpc_sys.boot.bootscreen.gui = 0
11
12
13  // [FR] Resolution d'ecran
14  // [EN] Screen resolution
15  set/ cpc_sys.boot.bootscreen.scr_res = 800x600
16
17
18  // [FR] Nombre d'images par secondes (IPS)
19  // [EN] Number frames per seconds (FPS)
20  set/ cpc_sys.boot.bootscreen.fps = 10
21
22
23  // [FR] Nombre d'images max
24  // [EN] Max number frames
25  set/ cpc_sys.boot.bootscreen.size = 21
26
27
28  // [FR] Numero d'image de commencement
29  // [EN] Starting image number
30  set/ cpc_sys.boot.bootscreen.begin = 0
31
32
33  // [FR] Numero d'image de bouclage
34  // [EN] Looping image number
35  set/ cpc_sys.boot.bootscreen.loop = 0
```

Explications !

```
CCP/ /SET.LEVEL
```

Toutes les variables seront stockées dans le niveau 5 (Kernel) IMPORTANT !

```
Set/ cpc_sys_bootscreen.gui = 0
```

Permet d'activer ou non le bootscreen
1 : Activé / 0 : Désactivé

```
Set/ cpc_sys_bootscreen.scr_res = 800x600
```

Résolution graphique du bootscreen

```
Set/ cpc_sys_bootscreen.fps = 10
```

Nombre d'images par secondes « FramePerSeconds »

```
Set/ cpc_sys_bootscreen.size = 21
```

Nombre d'images au total

```
Set/ cpc_sys_bootscreen.begin = 0
```

Numéro image de départ

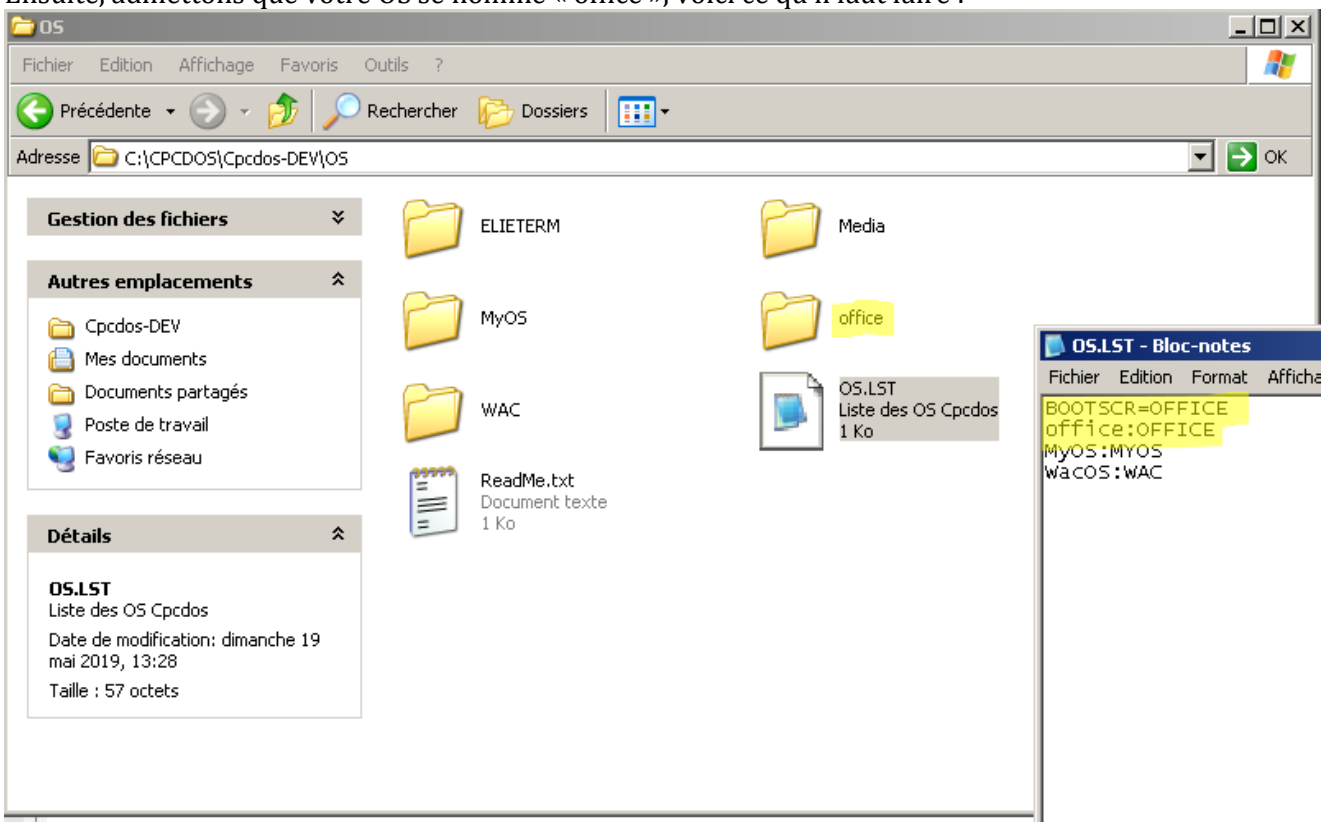
```
Set/ cpc_sys_bootscreen.begin = 0
```

Numéro image de bouclage (Retour au départ)

Mise en place

Il faut que votre dossier « boot » de votre OS contient bien évidemment ce fichier config.cpc, mais aussi des images nommées sous la forme de « img_0.png » « img_1.png » « img_2.png » « img_3.png » « img_x.png » … etc… une infinité ! Les formats PNG ou JPG sont supportés.
Il est idéal que les images fassent une suite d'image animés assez fluide et épuré, digne d'un dessin animé. Inspirez-vous des boot screen des Android !

Ensuite, admettons que votre OS se nomme « office », voici ce qu'il faut faire :



Dans le fichier OS.LST, il faut préciser au noyau, le nom de votre OS à afficher en boot screen.
Dans notre cas, nous avons précisé « BOOTSCR=OFFICE » Ce qui veut dire que le noyau va utiliser les fichiers « OS/OFFICE/BOOT/ » pour afficher le boot screen.

Bien évidemment il est possible d'afficher le boot screen d'un autre OS.

Le boot screen va s'afficher dans une coloration du plus sombre au plus clair. Et va disparaître aux colorations du plus clair au plus sombre, dès que la commande « GUI/ » est utilisée.
Soit, l'utilisation de ces commandes suivantes :

```
GUI/
GUI/ /OS:{Nom OS}
GUI/ /WITHOUTOS
GUI/ /SANSOS
GUI/ /CONSOLE
```

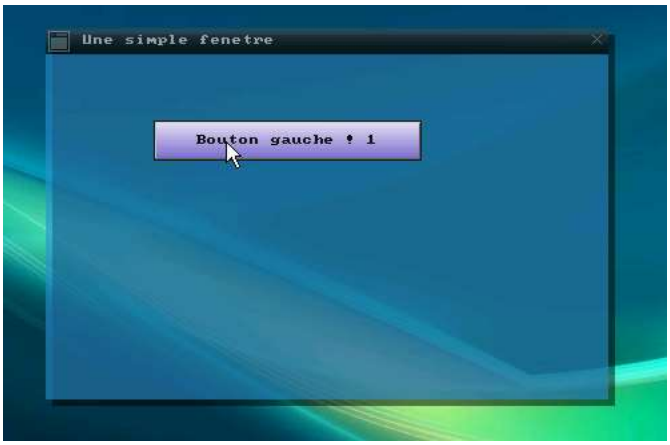
Qui font d'office de fermeture du boot screen.

Quelques détails

Le boot screen est pour le moment affiché en 16 ou 24 bits (Choix automatisée selon la carte graphique).
Il s'agit d'un thread exécuté si le boot screen est autorisé. L'ors de l'apparition et la disparition progressive de l'image animé, le système est « figé ». Ce qui sous-entend que le système perd seulement 2 à 3 secondes au démarrage de votre OS.
Lors ce que la commande GUI/ est employé, la fermeture du thread va enchainer à la disparition progressive de l'image. Une fois le thread fermé, le système poursuit le démarrage.
Vous pouvez utiliser l'argument **CPCLDR /NODBG** pour un démarrage épuré sans ligne de code !

Exemples

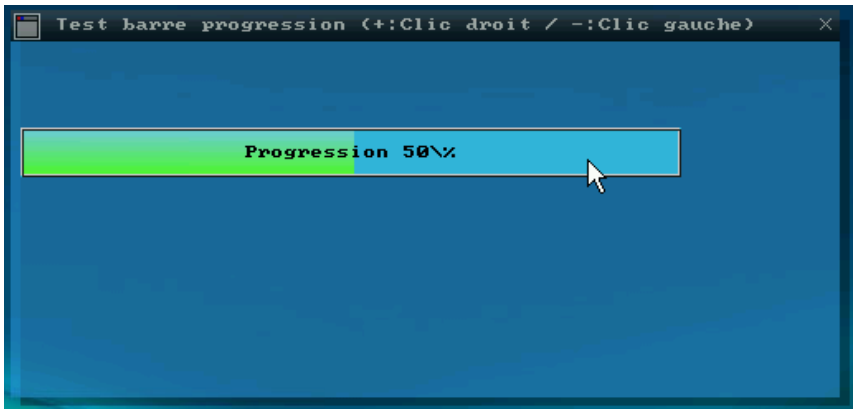
1. Une simple fenêtre transparente, avec un bouton et un évènement dans le même fichier.
   Si l'utilisateur clique sur le bouton, le texte du bouton change et affiche quel bouton on a pressé.




```
fenetre/ FenetreTest
        .titre                  = "Une simple fenetre"
        .px                     = "60"
        .py                     = "40"
        .tx                     = "420"
        .ty                     = "263"
        .opacite                = "150"
        .CouleurFenetre         = "050,150,220"
        .CouleurTitre           = "255,255,255"
        .CouleurFond            = "050,150,220"
        @#FenetreTest creer/
Fin/ fenetre

Bouton/ MonBouton
        .handle                 = "%FenetreTest%"
        .texte                  = "Clique-moi !"
        .Parametres             = "IMGAUTO:2"
        .px                     = "80"
        .py                     = "50"
        .tx                     = "200"
        .ty                     = "30"
        .CouleurTexte           = "255,255,255"
        .image                  = "%os_GUI%/bouton2.png"

        // La variable _EXE_PATH_ contient le chemin + fichier actuel
        .evenement                      = "%_EXE_PATH_%"
        creer/
Fin/ Bouton

Fonction/ MonBouton.MouseClick(bouton)
        ccp/ /fix.niveau = 1

        // Si l'utilisateur clique sur le bouton Gauche
        Si/ "%bouton%" == "1" alors:
                Bouton/ /MODIF:MonBouton
                        .texte                  = "Bouton gauche ! %bouton%"
                        create/
                Fin/ Bouton
        Fin/ si

        // Si l'utilisateur clique sur le bouton Droit
        Si/ "%bouton%" == "2" alors:
                Bouton/ /MODIF:MonBouton
                        .texte                  = "Bouton droit ! %bouton%"
                        creer/
                Fin/ Bouton
        Fin/ si

        // Si l'utilisateur clique sur les deux boutons
        Si/ "%bouton%" == "3" alors:
                Bouton/ /MODIF:MonBouton
                        .texte                  = "Les 2 boutons! %bouton%"
                        creer/
                Fin/ Bouton
        Fin/ si
Fin/ Fonction
```

2. Une simple fenêtre avec une barre de progression qui change de taille

Si l'utilisateur clique sur le bouton droit de la souris, elle diminue, si le bouton gauche est pressé, elle augmente.




```
fenetre/ FenetreTest
        .titre                  = "Test barre progression (+:Clic droit / -:Clic gauche)"
        .px                     = "60"
        .py                     = "40"
        .tx                     = "500"
        .ty                     = "210"
        .opacite                = "150"
        .CouleurFenetre         = "050,150,220"
        .CouleurTitre           = "255,255,255"
        .CouleurFond            = "050,150,220"
        @#FenetreTest creer/
Fin/ fenetre

BarreProgression/ MaBarreProgression
        .handle                 = "%FenetreTest%"
        .texte                  = "Progression 50\%"
        .Parametres             = "IMGAUTO:2"
        .Valeur                 = "50"
        .px                     = "5"
        .py                     = "50"
        .tx                     = "400"
        .ty                     = "30"
        .CouleurTexte           = "000,000,000"
        .CouleurFond            = "050,180,220"
        .image                  = "OS\Media\GUI\Buttons\BTN_GRBL.png"

        // La variable _EXE_PATH_ contient le chemin + fichier actuel
        //  ce qui permet d'exécuter un évenement dans CE fichier actuel.
        .evenement              = "%_EXE_PATH_%"
        creer/
Fin/ BarreProgression

Fonction/ MaBarreProgression.MouseClick(bouton)

        ccp/ /fix.niveau = 1

        // Bouton 1 : Clic gauche
        si/ "%bouton%" == "1" Alors:
                BarreProgression/ /MODIF:MaBarreProgression
                        @#prog .Valeur
                        set/ prog = /C(%prog% + 10)
                        .Valeur = "%prog%"
                        .texte                  = "Clic gauche/droit %prog%\%"
                        Creer/
                Fin/ BarreProgression
        Sinon:
                // Sinon clic droit ou autre boutons
                BarreProgression/ /MODIF:MaBarreProgression
                        @#prog .Valeur
                        set/ prog = /C(%prog% - 10)
                        .Valeur = "%prog%"
                        .texte                  = "Clic gauche/droit %prog%\%"
                        Creer/
                Fin/ BarreProgression
        Fin/ si

Fin/ Fonction
```

N'hésitez pas à nous poser une question si le code ne vous semble pas clair. ! 😉

1. Une simple fenêtre avec un check box

Si l'utilisateur coche le checkbox alors un message s'affichera


```
fenetre/ FenetreTest
        .titre                  = "Test Checkbox
        .px                     = "60"
        .py                     = "40"
        .tx                     = "200"
        .ty                     = "150"
        .opacite                = "150"
        .CouleurFenetre         = "050,150,220"
        .CouleurTitre           = "255,255,255"
        .CouleurFond            = "050,150,220"
        @#FenetreTest creer/
Fin/ fenetre

Checkbox/ Mon_Checkbox
        .handle                 = "%FenetreTest%"
        .Texte                  = "Coche moi !"
        .Valeur                 = "0"
        .PX                     = "5"
        .PY                     = "5"
        .tx                     = "100"
        .ty                     = "20"
        .CouleurFond            = "255,255,240"
        .CouleurTexte           = "000,000,000"
        .Evenement              = "%_exe_path_%=MouseClick"
        creer/
Fin/ Checkbox

Fonction/ Mon_Checkbox.MouseClick()

        // Recuperer l'etat du checkbox
        CheckBox/ /modif:Mon_Checkbox
                @#Etat .Valeur
        Fin/ CheckBox

        // Tester s'il est coché ou non
        Si/ "%Etat%" == "1" Alors:
                Msgbox/ Case coché! :)
        Fin/ si

Fin/ Fonction
```

N’hésitez pas à nous poser une question si le code ne vous semble pas clair. ! 😉

Codes d'erreurs et avertissements

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 manquant	**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 être mis en œuvre	**ERR_031**	Clé non spécifiée
AVT_032	JPG : espace de couleur YCCK ne peux être mis en œuvre	**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**	ISR : impossible de verrouiller la routine ISR
AVT_040	Aucun service en cours	**ERR_040**	ISR : impossible de verrouiller les données
AVT_041	Nom barre de progression existant	**ERR_041**	ISR : La routine ISR n'a pas démarrée.
AVT_042	Structure d'événement incorrecte	**ERR_042**	Impossible d'initialiser la vidéo, vérifiez la compatibilité VGA / SVGA de votre carte graphique
AVT_043	Impossible de lire l'entête	**ERR_043**	Impossible d'arrêter la routine
AVT_044	png_sig_cmp() non disponible	**ERR_044**	Problème de fermeture du processus
AVT_045	png_create_read_struct() non disponible	**ERR_045**	Impossible de charger la police
AVT_046	png_create_info_struct() non disponible	**ERR_046**	Impossible de charger l'image PNG
AVT_047	Des threads ne répondent pas	**ERR_047**	Adresse non disponible
AVT_048	Vous devez espacer '=' ou votre variable de ce type: fix/ variable = contenu	**ERR_048**	Erreur fatale, impossible d'instancier de nouveau CPinti Core.
AVT_049	Vous devez espacer '/S' et votre nom variable de ce type: fix/ /s variable	**ERR_049**	Impossible d'initialiser l'IUG
AVT_050	Vous ne pouvez pas créer une variable en la supprimant, STOP DRINKING :-P	**ERR_050**	Fonction CpcdosC+ introuvable !
AVT_051	Vous devez uniquement spécifier un nombre comme 1, 2, 3, 4, ou 5	**ERR_051**	Le numéro de port est déjà utilisé par un processus
AVT_052	Vous devez espacer '=' de ce type: ccp/ /fix.niveau = {nombre 1-5}	**ERR_052**	Oups problème ! La résolution d'écran n'a pas pu être appliquée.
AVT_053	Vous devez espacer '&' de ce type: exe/ & MonProgramme.cpc	**ERR_053**	Problème lors de l'écriture du fichier
AVT_054	Gestionnaire des polices désactivé ,. Exécutez: SYS/ /POLICE /ACTIVER	**ERR_054**	Erreurs lors de la configuration du socket réseau
AVT_055	Vous êtes en mode DosBox	**ERR_055**	Erreur, lors du binding du socket réseau
AVT_056	Vous devez espacer '/S' et votre nom variable de ce type: fix/ /s variable	**ERR_056**	Ecoute réseau impossible
AVT_057	Caractères non autorise	**ERR_057**	Problème descripteur de fichier
AVT_058	Vous devez spécifier un numéro de couleur entre 16, 24 ou 32	**ERR_058**	Canal cassé (ERRPIPE), redémarrez le serveur
AVT_059	Vous devez spécifier une résolution de ce type '1024x768'	**ERR_059**	Mémoire insuffisante
AVT_060	Vous devez spécifier une résolution de ce type '1024x768x16' (x16 s'il s'agit de 16 bits de couleurs)	**ERR_060**	Pilote souris PS/2 ou USB (INTERRUPT 0x33) introuvable ou non compatible.
AVT_061	Vous devez spécifier un nom de protocole	**ERR_061**	Impossible de copier le fichier source
AVT_062	Vous devez spécifier la FunctionID supérieur à 0	**ERR_062**	Impossible de copier sur le fichier destination
AVT_063	FunctionID non disponible	**ERR_063**	Impossible de copier sur le fichier destination
AVT_064	Vous ne pouvez pas déclarer une fonction inferieur au niveau 3	**ERR_064**	Impossible de lister le contenu du répertoire
AVT_065	L'inscription des propriétés graphiques doivent respecter la syntaxe du type : .PROPRIETE = "valeur"	**ERR_065**	
AVT_066	Vous devez d'abord déclarer un objet graphique avant d'initialiser une propriété graphique	**ERR_066**	
AVT_067	Vous ne pouvez pas créer un serveur d'administration en UDP. Puis c'est risqué !	**ERR_067**	
AVT_068	Le numéro de port est introuvable	**ERR_068**	
AVT_069	Le numéro client TID est introuvable	**ERR_069**	

AVT_070	Impossible de créer un autre serveur	**ERR_070**	
AVT_071	(DNS) impossible de résoudre le nom	**ERR_071**	
AVT_072	Mode avion activé	**ERR_072**	
AVT_073	Le nom d'un tableau doit être identifié avec un '(' avant un ')'	**ERR_073**	
AVT_074	Le port COM est déjà ouvert	**ERR_074**	
AVT_075	Le port COM est déjà fermé	**ERR_075**	
AVT_076	Impossible d'ouvrir le port COM	**ERR_076**	
AVT_077	Aucun numéro d'identification définit	**ERR_077**	
AVT_078	Impossible d'identifier un objet ou une instance	**ERR_078**	
AVT_079	Fichier code en UTF-8. Convertissez-le en ANSI pour une meilleure compatibilité.	**ERR_079**	
AVT_080	Fichier code en UTF-16 BE. Convertissez-le en ANSI pour une meilleure compatibilité.	**ERR_080**	
AVT_081	Fichier code en UTF-16 LE. Convertissez-le en ANSI pour une meilleure compatibilité."	**ERR_081**	
AVT_082	Mémoire bitmap null	**ERR_082**	
AVT_083		**ERR_083**	
AVT_084		**ERR_084**	

Remerciements

Contributeurs actifs	Contributeur du SDK	Ayant contribué
Michael BANVILLE *3D GZE & OpenGL engine, Win32, CWC C++*	Michael BANVILLE ***3D GZE & OpenGL engine, Win32, CWC C++***	Léo VACHET ***CraftyOS system.***
Timothée LUSSIAUD *CraftyOS design et plus…*		Marie-Charlène GAY--BELLATON ***Corrections des rédactions textuelles.***
Esteban CADIC *Discord, Principal ultima tester et plus…*		Johann GRAF ***Correction modules débuts CPinti Core.***
Léo BEUTEL		Steve PRUDHOMME
Simon MICHENAUD *Créateur/administrateur site web. (New logo)*		Dorian WILHELM
Charline GOYET *Correctrice & Traductrice site web &+*		Thomas GROS
Mathieu BELEK Support développement expérimental WEB		Thomas FROMONT
Nadir ZIANI		Mathieu RIBEIRO
Guillaume DEROY		Charles PROVENT
		Kévin CAPOSIENA
		Joris REMANDET
		Gabriel LABROSSE
		Sviat SMOLINET
		Mathieu MARI
		Cyril COELHO
		Jomtek
		Corentin HTROISC
		Alexis BELMONTE
		Moi75ts
		CI Diderot
		Aura FRANCE
		ELX Free
		Adrien HERMAN
		Maxime GASPARINI
		Jordan DALCQ
		Tornade8912
		Fabrice GAUGLIN
		Roni CARTA
		Rida LALI

Si j'ai oublié quelqu'un, manifestez-vous !

Et bien entendu le reste de la communauté et hors-communauté souhaitant rester anonyme.

Liens

Liens principaux	Infos
www.facebook.com/pages/Kernel-Cpcdos-OSx/479523255400921	La page Facebook qui regroupe toutes les nouveautés, les informations des dernières mises à jour, des screenshots et bien autre !
cpcdos.net cpcdos.fr.nf cpcdososx.wixsite.com/cpcdos	Le nouveau site officiel de Cpcdos (+ le nouveau logo) redésigné par Simon (Fukotaku) Et les ancien sites
forum-cpcdos.fr.nf	Le forum officiel de Cpcdos hébergé chez Developpez.net. Consultez, posez vos questions ici Toute la communauté peuvent vous répondre.
www.youtube.com/user/cpcdososx	Tutoriels du développement et nouveautés en vidéo. Et de temps à autres, des lives !
https://discord.gg/3Qm8xDp	Notre serveur de chat officiel ouvert par Esteban CADIC. En plus du forum, ici vous pourrez discuter en direct, textuellement et vocalement avec toute la communauté, ainsi qu'aux développeurs du noyau.