M2I

COLLABORATORS			
	TITLE : M2I		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		August 19, 2022	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

M2I

1.1 Index des commandes MCL

```
Les instructions MCL : Index et résumé des commandes

   (pas d'args)
   (pas d'args)
   <Chaîne>
   <Répertoire>
   Var1 [Var2 [... Varx]
   <Secondes>
   <Texte>
   (pas d'args)
   (pas d'args)
   (pas d'args)
   (pas d'args)
   [-OPT] [Pattern] <VAR(s)> <"Texte">
    Options : p pattern, d directory
   Var1 [Var2 [... Varx]
   <Label>
   <Conditions>
    Conditions : > < = #
    Relations : | (ou) & (et)
    Mots-clés : FILE PORT TASK DIR CHK
   <Texte>
   <Label>
   <Menu>
   [Sortie]
   <port> <message>
   [Remarque]
   [<entrée] [>sortie] <programme>
   [-OPT] <VAR> <"Invite"> REQ1|...|REQX
    Options :   c (Cancel par défaut)
        i (disk insert)
        r (no return)
   (pas d'args)
   <VAR> <Valeur>
   <pri> <nom de tâche ou process>
   <Octets>
   <VAR> <Invite>
   <VAR>
```

```
  TBAR (supprimée)
   <var> <fichier>
   <var> <fichier>
   (pas d'args)
   <Conditions>
    Conditions : > < = #
    Relations : | (ou) & (et)
    Mots-clés : FILE PORT TASK DIR CHK
```

1.2 Documentation de M2I 4.25 / MIE 2.23

```
                    Une production du Ringard' 1993
      M2I Version IV
           (c)PIMMEL Thomas

  CONTENU  ( voir  )



Présentation


                Copyright et distribution

                Compatibilité

                Présentation de M2I

                Historique

                La concurrence...

                Installation et exécution
                MIE


                MIE introduction et présentation

                Les menus de MIE

                Divers MIE
                 (Fontes, ED et MIE...)

                Recommandations
                MCL


                Le Menu-Command Langage (MCL)

                Commentaires et tabulations

                Exécution de programmes

                Les Instructions MCL
                M2I
```

1.3 Copyright et distribution

L'appellation M2I comprend M2I/MCL et son éditeur MIE.

```
 M2I (Menu-Item Interpretor) version 4.24
 MCL (Menu Command Langage)  version 4.24
 MIE (Menu Item Editor)      version 2.23

M2I/MCL et MIE sont des productions de :

   Thomas PIMMEL ( voir
               Me contacter
                )

Ces programmes sont 100% Domaine Public, diffusables librement avec les
conditions suivantes :

 L'intégralité de M2I doit être fournie, sans altération.
 Le programme CallIO doit également être diffusé, il est
 utilisé par MIE.

La reqtools.library ne fait pas partie de M2I et est soumise
au copyright de NICO François.

Merci aux inconditionnels de M2I pour leurs encouragements, Fabien,
Jean-Marc et tout particulièrement Brice pour ses conseils et suggestions.

Un coucou à Zarbi et Thierry...
```

1.4 Contactez-moi

Les versions de M2I et MIE semblent être maintenant bien au point (3 mois de tests). Si cependant vous découvrez un bug ou si simplement vous avez une idée pour l'évolution de M2I vous pouvez bien sûr m'écrire ou même me téléphoner.

Ce programme est DP mais si vous l'utilisez dans une production commerciale ou si vous voulez le diffuser dans un réseau DP soyez aimables de me tenir au courant.

Contactez-moi à l'adresse suivante

Thomas PIMMEL
21 rue du repos
68200 BRUNSTATT
Tél. 89 06 05 25 (soir)

1.5 Compatibilité

Compatibilité

M2I est compatible avec les Kickstarts 2.0 et supérieurs.
M2I est compatible A1200 et A4000.
M2I a été écrit sous A500+, Kick 3.0, DD A590, 4 Meg.

1.6 Présentation

Présentation

Le but de M2I est l'exécution de programmes grâce à une interface menu intuition.

Ce menu est attaché à une fenêtre Shell ou à sa fenêtre propre. Il peut comporter des touches de raccourci (Hotkeys), des items et sous-items.

A chaque point de menu se rattache un script qui peut intégrer des instructions du langage .

Le menu est conçu à partir de , utilitaire sous intuition.

M2I est 100% multitâche, il est capable d'exécuter plusieurs scripts simultanément.

Ses caractéristiques et sa fiabilité font de M2I un outil indispensable pour exécuter rapidement un programme caché au fond d'une disquette ou d'un disque dur, en interactivité avec son utilisateur.

1.7 Historique

Historique

M2I est né en 1991. A l'époque, le WorkBench n'était pas un outil franchement pratique. J'utilisais SetKey pour exécuter mes prog favoris. Ce système était lourd, et je n'arrivais pas à obtenir les ^ les é et les è si chers à notre langue française dans la plupart des progs.

Ainsi est né M2I, Menu Item Interpretor, qui se logeait dans la fenêtre de mon Shell et me permettait d'avoir tous mes utilitaires sous la souris.

Jusqu'à la version 4, M2I utilisait un langage script complexe et passablement sensible aux erreurs. La maintenance du disque devenait un problème. Cette nouvelle version rend M2I plus convivial, mais surtout beaucoup plus puissant.

Ce qui change de M2I 3 (tout)

- L'éditeur de menu, bien sûr
- MCL (34 commandes)
- Abandon de la Arp.library pour la reqtools.
- Intégration de Flarg (requester de fichier pour argument programme)
- Support du WorkBench pour MIE (3.0 oblige)
- Optimisation du code
- Support des CHECKMENU
- Adoption du format IFF pour les menus etc...

M2I a été entièrement récrit. Les anciens fichiers menu ne sont pas compatibles avec la nouvelle version. En fait M2I IV n'a plus rien à voir avec M2I III.

1.8 La concurrence...

Et la concurrence?

M2I n'est pas le seul utilitaire du genre. Il existe de très bons DP qui traitent du même sujet.

Son orientation est cependant différente.
Un menu M2I se construit et se pense dans une utilisation quotidienne, et décharge son utilisateur des tâches répétitives. Il utilise un langage de commande qui permet de réaliser de véritables applications, de façon simple. MCL est une sorte de Basic simplifié, et adapté à l'exécution de tâches. Il faut bien sûr construire ces applications. Mais MIE vous y aide en créant la plupart des scripts dont vous aurez besoin.

Consacrez un peu de temps à M2I, il vous le rendra.

Enfin, M2I est écrit directement en Assembleur, il est concis (10ko environ), et rapide. M2I est fiable, Cela fait bientôt 4 ans qu'il tourne sans problème sur mon Amiga et ceux de mes amis. C'est aussi l'assurance d'un upgrade régulier, M2I est mon programme phare.

Exécutez la démonstration est jugez par vous-même.

1.9 Installation

Installation et exécution

Choisissez l'Item "Installer M2I" dans le menu et suivez le guide.
Pour une installation à la main, pas de problème...

Copiez la Reqtools.library dans votre répertoire LIBS:
Copiez M2I et MIE dans le répertoire de votre choix.
Copiez les #?.M2I dans un répertoire de votre choix (S: par exemple).
Copiez CallIO dans le répertoire C (ou autre répertoire du Path).

Exécutez la commande (veillez à préciser le chemin) :

  run M2I <nom du menu>

par exemple :

  run sys:utilitaires/M2I S:Tutorial.m2i

M2I s'installera sur la barre de menu de votre fenêtre active, à
condition qu'aucune autre application ne l'ai déjà fait.

Pour MIE :

  run MIE [EDIT=<nom de votre éditeur>] [MENU=<nom du menu>]

Les arguments sont optionnels. Pour débuter je vous conseille d'exécuter
MIE sans argument.

  !!! MIE se sert par défaut de ED comme éditeur !!!

     !!! Il doit se trouver dans le Path de votre système !!!

1.10 MIE

Introduction et présentation

Notre visite dans M2I commencera par MIE. MIE est l'éditeur de menu.
Une fenêtre (très look 2.0) s'ouvre, divisée en trois colonnes.
La première est réservée au Header de menu, la deuxième aux items et
la troisième aux sous-items.

Le titre de l'écran vous informe sur la version MIE que vous utilisez (2.22
ou plus), ainsi que sur la version M2I compatible avec votre éditeur (4.2x
en principe).

La fenêtre MIE

La barre de fenêtre indique le nom du menu que vous êtes en train de modifier.
En plus des deux gadgets de fenêtre classiques (avant-plan et wclose) le
gadget zoom permet d'icônifier votre fenêtre.
A droite, des carrés de couleur vous indique la couleur par défaut de votre
item, et la couleur du point sélectionné.

Le gadget "Test du menu" permet de visualiser votre menu. Dans ce cas le menu M2I se substitue au menu classique MIE. Pour rétablir l'ancien menu il suffit de cliquer à nouveau sur ce gadget.

Les points de
menu de MIE
agiront soit sur le menu édité en entier
(sauver par exemple), soit sur l'item sélectionné.

La sélection

Avec la souris ou les flèches de déplacement du clavier vous déplacez un curseur de haut en bas et de colonne en colonne. Vous cliquez sur le bouton droit ou sur la touche <Entrée> pour sélectionner un item.

1.11 Les menus

Le menu de MIE

PROJET

Nouveau, efface le menu actuellement édité de la mémoire
Ouvrir..., permet de charger un menu MIE
Sauver, permet la sauvegarde de votre menu avec le nom par défaut
Sauver sous..., idem sous un autre nom
Quitter, bouger de là
A propos de..., info sur la version et le programmeur

MENU

Ajouter, permet d'ajouter un item après l'item précédent. Selon la colonne où se trouve le curseur vous ajoutez un header, un item ou un sous-item. Pour ajouter un item en tête de liste il suffit de sélectionner <---Headers---> ou <---Items---> etc...
Détruire, détruit irrémédiablement l'item et son script, ainsi que tout item inférieur. Donc attention...
Couper, coupe et place dans un buffer une fraction de l'arborescence à partir de l'item sélectionné.
Coller, colle un item coupé avec "Couper". Le tandem Couper/coller sert à déplacer un item.
Editer Texte, permet la modification du nom de l'item.
Check Item, permet la création/suppression d'un menu à bascule. Ces items particuliers permettent l'affichage d'un crochet à gauche du menu lorsque celui-ci est sélectionné. La valeur de sélection (oui/non) du menu par défaut peut être forcée lors du "Test de menu". Cette valeur sera ainsi sauvée en même temps que le menu.
Ajouter HotKey, un caractère activant un point de menu en combinaison

avec Left-Amiga. MIE vérifie si ce caractère n'est pas déja
utilisé. Conseil : n'utilisez pas C et V, ils servent à faire
du copier/coller dans le Shell (le menu est prioritaire sur
le gestionnaire du Shell).
Enlever Hotkey, supprimer une hotkey.
Ajouter Fichier(s), la commande la plus puissante et la plus utile.
Elle agit comme "Ajouter", mais elle permet une multi-sélection
de fichiers exécutables via le filerequester de la reqtools
et leur ajoute un script qui effectue un CD et un RUN de la
commande. Pour faire une sélection multiple de fichiers appuyez
simultanément sur la touche Majuscule et le bouton gauche de la
souris.

SCRIPT

Editer, édite un script en langage MCL (voir)
Copier, copie le script dans un buffer
Couper, coupe le script et le place dans le buffer
Coller, colle le script du buffer
Détruire, élimine sauvagement un script, mais que fait la police!

COULEUR Chaque item ou sous-item peut avoir sa couleur propre, de
façon à améliorer la lisibilité du menu.

Changer défaut, par défaut lorsque vous créez un item il prend
la couleur de la "couleur par défaut". Pour changer cette
couleur "Changer défaut" affiche la palette de couleur
du WBench. Il suffit de cliquer sur cette couleur. Le nombre
de couleurs de la palette peut aller de 2 à 16, selon la config
de votre WBench. En principe la couleur de fond de votre menu
est la couleur 1. La prendre comme couleur par défaut n'est
pas la meilleure solution pour la lisibilité immédiate de votre
menu.
Colorier item, colorie l'item ou sous-item de la couleur par défaut.
Colorier tout, colorie tout les items et sous-items avec la couleur
par défaut.

PREFERENCES Les préférences sont :

- la couleur par défaut,
- la position de la fenêtre,
- l'état des flags de préférences

Les préférences sont sauvegardées lors de la sortie de MIE

Sauver, sauve les préférences dans ENVARC:
Utiliser, sauve les préférences dans ENV:

1.12 Divers MIE

Divers

MIE envoie un message à M2I lors de la sauvegarde d'un menu. Si le menu sauvé est celui actuellement utilisé par M2I, ce dernier tiendra compte des modifications. Ceci ne fonctionne que si CallIO est présent dans le répertoire C (ou dans le Path du système).

MIE calcule la disposition des menus par rapport à la fonte définie dans les préférences du WBench. Si vous changez cette fonte il est indispensable de recharger le menu sous MIE et de le resauver. Lors de la sauvegarde en effet la disposition du menu est recalculée.

MIE utilise ED par défaut. Il doit donc être présent dans le path (chemin) du système. Si vous voulez utiliser un autre éditeur vous devez le préciser à MIE dans l'argument de la commande ou dans son icône sous WBench (type d'outils, menu information). Ajoutez simplement
EDIT=nom_de_l'éditeur
avec éventuellement le chemin permettant à MIE de le trouver dans votre arborescence. Votre éditeur doit pouvoir recevoir un nom de fichier en paramètre, que ce fichier existe ou non (il doit pouvoir le créer le cas échéant). Il ne doit pas non plus rendre la main au système (exit CED) car MIE attend que l'éditeur lui rende la main avant de prendre en compte les modifs faites au fichier. Le plus simple est d'essayer votre éditeur favorit avec MIE. MIE a été testé avec l'éditeur de Devpac avec succès.

1.13 Recommandations

Recommandations

Pour les menus, Commodore a établi quelques conventions.
En voici quelques unes :

On trouvera dans votre menu un menu Projet, qui sera
en première place et qui comprendra dans l'ordre :

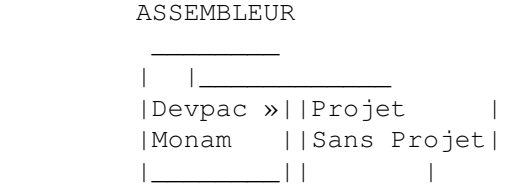
- Nouveau ou ouvrir * Nouveau menu (NEW)
- Quitter * Quitter M2I (EXIT)
- A propos de... * (ABOUT)

Bien sûr MIE ne vous y oblige pas.

Les hotkeys sont standardisés (N nouveau, O ouvrir, Q quitter)
C'est parfois difficile de s'y tenir (regardez les menus de MIE)

Lorsqu'un item dispose d'un sous-item il faut mettre un » après
l'intitulé du item (Alt-à clavier français)

Exemple



```
        |_____________|
```

Ainsi l'utilisateur de votre menu sait où se trouvent les S-items.

1.14 MCL

Le Menu-Command Langage (MCL)

MCL (Menu Command Langage) est un interpréteur de commande orienté Shell. L'utilisation la plus simple de MCL consiste à inscrire une commande OS ou un nom de programme dans le script. Cependant MCL permet de faire bien plus.

MCL est inclus dans le programme , dont il compose la partie principale.

MCL est multitâche, écrit en code réentrant il permet le lancement de plusieurs scripts en même temps. Il exploite la plupart des possibilités de la Reqtools.

Il répond à trois besoins, exécuter des programmes, dialoguer avec l'utilisateur et enfin permettre la programmation d'un script construit.

Un script MCL se tape sous MIE. Le principe est le même que celui de la plupart des langages de programmation. Il exécute les instructions de façon séquentielle, de la première ligne à la dernière. Il connaît la plupart des instructions conditionnelles et de boucle.

1.15 Tabulations

Commentaires et tabulations

Pour rendre votre source clair :

Vous pouvez placer avant chaque instruction une ou plusieurs tabulations, un ou plusieurs espaces.

Vous pouvez placer des commentaires entre deux lignes du script. Pour cela il suffit de commencer votre ligne par un ; ou une *, ou encore de placer le mot-clé REM au début de votre ligne.

1.16 Exécution de programmes

L'exécution de commandes DOS ou de programmes

Vous pouvez placer une instruction DOS ou un programme n'importe où dans le script.

Si votre commande rentre en conflit avec une instruction MCL il suffit de préciser le chemin d'accès de votre commande devant la commande.

N'oubliez pas que la sortie par défaut des programmes appelés est NIL: Si vous voulez une sortie écran vous devez utiliser la commande OUT.

Certaines commandes DOS n'ont aucun effet sur M2I. Par exemple CD et STACK. MIE vous offre un substitut qui porte le même nom... et qui a les mêmes effets (voir
Substitution du DOS
).

1.17 MCL

Les instructions MCL

Rien ne distingue une instruction MCL. Vous pouvez les rentrer en minuscule ou en majuscule. Dans votre menu ces instructions sont codées afin de prendre moins de place (2 octets par instruction). Les chapitres suivants reprennent en détail chaque commande. Il en existe 34, qui produisent une action spécifique.

Gestion des variables

Information et dialogue

Logique et boucle

Gestion de M2I

Substitution du DOS

Autres instructions

Possibilités étendues

Scripts utiles

1.18 Les variables

Gestion des variables

Les variables sont des espaces mémoire qui contiennent une chaîne de texte. Cette chaîne une fois mémorisée peut être rappelée pour être incorporée dans n'importe quelle instruction.

Chaque variable est identifiée sous la forme d'une lettre. Il existe donc 26 possibilités de variables, de A à Z. Vous pouvez les rentrer en majuscule ou en minuscule, MCL ne fait pas la différence.

L'utilisation des variables comporte 4 phases au moins :
la déclaration, l'affectation, l'utilisation proprement dite,
et la libération.

La déclaration de la variable entraîne une allocation mémoire
de 256 octets, nécessaires pour mémoriser une chaîne.

```
Syntaxe   DECLARE A B C D ... Z

Exemple   DECLARE V   MCL reconnaît la variable V
    DECLARE W A C   MCL reconnaît les variables W A et C
```

Cette étape est indispensable.

L'affectation d'une variable permet de donner une valeur à cette
variable. Cette valeur peut être changée à volonté. Une affectation est
le résultat d'une instruction MCL. L'instruction la plus simple
d'affectation est l'affectation directe d'une valeur par la commande SET.

```
Syntaxe   SET <VAR> Chaîne de caractère

Exemple   SET S Salut mec, tu vas bien?
```

La variable S a pour valeur la chaîne 'Salut mec,
tu vas bien?'

Si vous placez des " dans votre ligne, les " feront partie
de la chaîne.

Vous pouvez rediriger le résultat d'une commande DOS
dans une variable avec SET :

```
Syntaxe   SET <VAR> `Commande`

Exemple
    DECLARE D
    SET D `date`
    INFO Date et heure du jour $D
```

Affecte le résultat de la commande DATE à la variable
D. Cela ne marche qu'avec des instructions externes à MCL.

Une variable peut s'utiliser dans n'importe quelle instruction ou
appel de programme. Pour être reconnue comme une variable elle doit être
précédée du signe $. Un exemple vaut mieux qu'un long discours.

```
Exemple   DECLARE D
    SET D Dessin
    RUN dh0:dpaint ram:$D
```

MCL va exécuter la commande "RUN dh0:DPaint ram:Dessin"

Si vous voulez utiliser le sigle $ dans un cas différent il suffit
de le répéter deux fois

```
Exemple   echo Je veux des $$!
```

```
    Donne à l'écran: Je veux des $!

  La libération, est l'opération inverse de la déclaration.
La mémoire allouée pour la variable est libérée.

Syntaxe FREE A B C ... Z

Exemple FREE a f

  La mémoire est libérée automatiquement à la fin du script (v4.2x).

  Un dernier exemple de script pour la route :

Exemple   DECLARE A B C D
    SET A Il fait
    SET B chaud
    SET C beau
    SET D et
    INFO Contrepèterie belge : $A $B $D $C.
    INFO Solution : $A $C $D $B.
    FREE A B C D
    END

  Je vous laisse tester. Si vous êtes belge, remplacez belge par
suisse, ça marche aussi.


  Voir également
               VREAD et VWRITE
```

1.19 Dialogue

```
Les instructions d'information et de dialogue


  Ces instructions résultent de l'utilisation de la REQTOOLS.
```

1.20 ABOUT

```
                  ABOUT affiche la version de M2I et mes coordonnées.

Syntaxe ABOUT
```

```
  Voir
               Contactez-moi
```

1.21 INFO

```
  INFO permet d'afficher un message à l'écran

Syntaxe INFO Texte de message

Exemple

INFO Bière qui coule, n'amasse pas mousse.
END

  Affiche un message d'une utilité douteuse. Vous pressez
le gadget OK ou ENTREE pour continuer.
```

1.22 REQUEST

```
  REQUEST est une boite de requête à choix multiple.

Syntaxe REQUEST [-OPT] <VAR> "Texte de la requête" REQ1|REQ2 ... REQX

  -OPT est facultatif, ce sont des options :
    C : la dernière requête est en gras et l'utilisateur
la sélectionne en pressant ENTREE (par défaut c'est la première)
    R : la touche ENTREE n'agit pas (utile en cas de décision
    grave comme une confirmation de formatage)
    I : l'insère d'une disquette agit comme la touche ENTREE.

  <VAR> est une variable déclarée qui contiendra la réponse de
l'utilisateur. La réponse est un chiffre qui a la signification suivante :
  1, première requête
  2, deuxième requête
  3, .....
  0, dernière requête


  "Texte de la requête" est le texte qui apparaîtra dans la boite
de dialogue.

  REQ1|REQ2 ... REQX sont les propositions de réponse de la requête.
Elle sont séparées par le signe |.

Exemple
  REQUEST -IR A "Votre programme?" Manger|Boire|Dormir|"Ne Rien faire"

  Pour exploiter une requête l'instruction idéale est
```

1.23 STRING

```
  STRING permet la saisie d'une chaîne dans une variable

Syntaxe STRING <VAR> <Texte d'invite>

Exemple STRING S Quel est ton nom?

  S prend la valeur de la saisie.
```

1.24 FILE

```
  FILE permet la sélection d'un fichier ou d'un répertoire.

Syntaxe

  FILE [-OPT] [Pattern] <VAR(S)> "Titre de fenêtre"

  -OPT, options.

    D, choix d'un répertoire (par défaut fichier)
    P, pattern (motif de nom de fichier)

  <VAR>,  Variable. Dans le cas du choix d'un répertoire, une
    variable renvoyant ce répertoire suffit. Dans le cas
    contraire il en faut trois, une pour le répertoire, une pour
    le nom du fichier, et une pour le nom complet.

  "Titre de fenêtre", le titre de la boite de requête fichier.

Exemple 1 (répertoire seul)

  FILE -D S "Répertoire de sauvegarde"

Exemple 2 (fichier)

  FILE D F T "Sauver sous"

Exemple 3 (pattern)

  FILE -P "#?.BAK" D F T "Détruire fichier"

Dans ce cas le motif suit l'option. La combinaison -PD ou -DP est
impossible.

Exemple de script :

  REM         Boite à musique
  DECLARE D F T
  REM         Répertoire par défaut
  SET D ST-00:Modules
  REM         Musique par défaut
  SET F mod.klisje paa klisje
  REM       GO!
  FILE -P "mod.#?" D F T "Jouer musique"
  C:PROP $t
```

```
  FREE D F T
  END
```

1.25 OUT

```
  OUT

  La sortie par défaut d'un programme exécuté par MCL est NIL:

  OUT vous permet de rediriger cette sortie sur n'importe quel
  canal DOS valide (*, fichier, CON:, SER: etc...)

Syntaxe OUT [sortie]

  Lorsque la sortie est précisée, M2I ouvre cette sortie, sinon
  elle la ferme.
  Toute sortie est fermée à la fin d'un script.

Exemple de script

  OUT CON:0/0/640/200/Répertoire
  List
  DELAY 2
  OUT

  Attention : Vous ne pouvez ouvrir plus d'une sortie à la fois.
```

1.26 ECHO

```
                  ECHO Permet d'écrire une phrase dans la sortie courante.

Syntaxe ECHO <Texte>

  Par défaut ECHO ne saute pas de ligne après avoir écrit. Si
  vous voulez sauter une ligne il est nécessaire d'utiliser les { }
  ( Voir
               Possibilités étendues
                )

Exemple de script

REM   Ce script écrit dans un fichier mes bonnes résolutions
REM   pour la journée de demain.
OUT ram:test
ECHO {Demain :
 - je me lève tôt
 - je fais le ménage
 - je termine la doc de M2I
}
OUT

  Voir
```

1.27 Logique et boucle

```
                Les instructions de logique et de boucle

  Ces instructions gèrent les décisions et les répétitions.


                GOSUB / LABEL / RETURN

                IF / ELSE / ENDIF

                WHILE / WEND
```

1.28 END

```
  END permet de mettre un terme au script. END peut se trouver
  à n'importe quel endroit du script. END est facultatif.

Syntaxe END

Exemple de script

DECLARE D
STRING D "Quel est votre situation de famille?"
IF $D=marié | $D=fiancé
  INFO Je ne veux pas mettre en péril ton mariage!
  FREE D
  END
ENDIF
show Les_Fesses_de_Vanessa
FREE D
```

1.29 GOSUB

```
  GOSUB / LABEL / RETURN permettent d'exécuter un Sous-programme

Syntaxe GOSUB <Nom du label>
  LABEL <Nom du label>
  RETURN

Exemple de script

DECLARE A
REM   Détruire monbeauprog.asm
SET A monbeauprog.asm
GOSUB Détruire
REM   Détruire monbeauprog.asm.bak
SET A $a.bak
GOSUB Détruire
END
```

```
LABEL Détruire
  delete sys:asm/$a
RETURN
```

1.30 SWITCH

```
  SWITCH / CASE / BREAK permettent la gestion des choix multiples.

Syntaxe SWITCH <VAR>
  CASE <Texte>
  BREAK

Exemples de script

Exemple 1

DECLARE N

STRING N "Ton nom?"
SWITCH N
  CASE Thomas
    INFO "Salut Patron!"
  CASE Germaine
    INFO "Salut patronne!"
  CASE Aldebert
    INFO "Casse-toi!"
BREAK
FREE N

Exemple 2

DECLARE C
REQUEST C "Votre choix?" Quitter|"Dormir 10 secondes"|Abandon
SWITCH C
  CASE 1
    EXIT
  CASE 2
    DELAY 10
  CASE 3
    INFO "Abandon!"
BREAK
FREE C

BREAK est obligatoire et unique à la fin d'un SWITCH
```

1.31 IF

```
                  IF / ELSE / ENDIF  permettent de conditionner un passage du ↩
                     script

Syntaxe IF <condition>
  ELSE
```

```
  ENDIF
```

Condition est une comparaison ou un test d'une chaîne, ou même plusieurs comparaisons séparées par | (or) ou & (and)
MCL reconnaît = (égalité) # (différent) > (supérieur) et < (inférieur)
Condition peut comporter également des mot-clés :

```
  PORT <Nom du port>  vrai si port de message
  TASK <Tâche>    vrai si tâche
  FILE <Nom du fichier> vrai si fichier existe
  DIR <Nom du répertoire> vrai si répertoire existe
  CHK     vrai si item sélectionné ( voir
                les Menus
                 )
```

ELSE est optionnel, les lignes après ELSE sont exécutées si la condition est fausse.

ENDIF termine une condition IF

Exemples de scripts

Script 1

```
DECLARE C
REQUEST C "Formater DF0:" Oui|Non
IF $C=1
  OUT CON:0/0/640/200/Format
  SYS:SYSTEM/FORMAT DRIVE DF0: NAME VIDE FFS QUICK NOICONS
  OUT
ENDIF
FREE C
```

Script 2

```
DECLARE A B
STRING A "Nom du joueur 1?"
STRING B "Nom du joueur 2?"
REM       Teste si A et B non vides et A<>B
IF $a & $b & $a#$b
  RUN game $a $b
ELSE
  INFO "Nom des joueurs invalide!"
ENDIF
FREE A B
```

Script 3

```
DECLARE D F T
FILE D F T "Détruire fichier"
REM     Si un nom est rentré et si le fichier existe
IF $T & FILE $T
  delete $t
ELSE
  INFO "Nom vide ou fichier inexistant!"
ENDIF
FREE D F T
```

```
 !!! Respectez bien les espaces : 1 espace avant et après & et |
          pas d'espace avant et après <, >, #, =
```

1.32 WHILE

```
  WHILE / WEND permettent la répétition d'une action

Syntaxe WHILE <Condition>
  WEND

Condition est identique à celle de

Exemple de script

Exemple 1

DECLARE C
SET $C 1
WHILE $C=1
  INFO "Je suis une larve!"
  REQUEST C "Encore cette insulte?" Oui|Non
WEND
FREE C

Exemple 2

WHILE PORT BlackIOPort
  CallIO BlackIOPort KILL
WEND
```

1.33 gestion de M2I

```
Gestion de M2I
```

1.34 EXIT

```
  EXIT Permet de quitter M2I (pas de confirmation)

Syntaxe EXIT

Exemple de script

DECLARE C
REQUEST C "Quitter M2I?" Oui|Peut-être|Non
SWITCH C
  CASE 1
    EXIT
```

```
  CASE 2
    INFO "Quand tu veux tu te décides!"
BREAK
FREE C
```

1.35 NEW

```
  NEW Permet de changer de menu (le script est interrompu)

Syntaxe NEW <Nom du menu>

Exemple de script

DECLARE D F T
FILE -P #?.M2I D F T "Nouveau Menu"
IF $T
  NEW $T
ENDIF
FREE D F T
```

1.36 Substitution

```
Instructions de substitution du DOS
```

1.37 STACK

```
  STACK permet d'augmenter la valeur de la pile par défaut

Syntaxe STACK <octets>

Exemple de script

STACK 100000
RUN dh0:SuperRécursifIEE
STACK 5000
END
```

1.38 CD

```
  CD change le répertoire courant

Syntaxe CD <répertoire>
```

```
Exemple de script

CD SYS:ASM/DEVPACVI
RUN Devpac
```

1.39 RUN

```
  RUN (nouveau v4.23) exécute un programme en multitâche.
  Le script se poursuit en laissant courir le programme.

Syntaxe RUN [<entrée] [>sortie] <prog>

Exemple de script

RUN >ram:toto dir dh0:
```

1.40 csetpri

```
  SETPRI change la priorité d'une tâche ou d'un process


Syntaxe SETPRI <pri> <nom de tâche ou process>

Exemple

CD Synthèse:
STACK 40000
RUN SceneryAnimator
STACK
DELAY 2
SETPRI -4 SceneryAnimator
END

Le "DELAY 2" attend que le programme soit chargé avant de lui affecter
une priorité.
```

1.41 Autres instructions

```
               Autres instructions



               VREAD / VWRITE
                    (ARexx)
```

1.42 DELAY

```
  DELAY permet de mettre le script en sommeil.

Syntaxe DELAY <Secondes>

Exemple DELAY 10
```

1.43 VREAD et VWRITE

```
  VREAD et VWRITE permettent de lire et d'écrire dans un fichier à
  partir de variables. Les variables étant de type chaîne, il n'est
  possible de lire ou d'écrire qu'un texte n'excédant pas 255 octets.


Syntaxes

  VREAD <var> <fichier>
  VWRITE <var> <fichier>

Exemple

DECLARE A
STRING A Texte à sauver
VWRITE A ram:texte
REM vérification
VREAD A ram:texte
INFO $a sauvé!
END
```

1.44 PUTMSG

```
  PUTMSG permet d'envoyer un message AREXX à un programme.

Syntaxe

PUTMSG <port> <message>

Exemple

DECLARE p
REM   on n'aura plus besoin de réécrire le nom du port
SET p RPlayerIOPort
REM   si le port est présent
IF PORT $p
   DECLARE d f t
REM     nom du répertoire et filerequest
   SET d Music:modules
   FILE -p mod.#? d f t "Play Me!"
REM   si utilisateur a cliqué ok
   IF $t
REM   charger le module
      PUTMSG $p "LOAD $t"
```

```
REM   le jouer
      PUTMSG $p PLAY
   ENDIF
   FREE d f t
REM erreur....
ELSE
   INFO RPlayer ne fonctionne pas.
ENDIF
FREE p
```

1.45 Extension

```
Possibilités étendues.

  MCL est récursif.

  MCL gère les instructions imbriquées


Exemple de script

DECLARE M F
STRING M "Entrez le nom du joueur 1"
STRING F "Entrez le nom de sa femme ou rien si célibataire"

IF $M
  IF $M & $F
    INFO "Ah, $M est marié avec $F!"
  ELSE
    INFO "C'est sûr avec son caractère..."
  ENDIF
ELSE
  INFO "Il faut rentrer le nom du joueur!"
ENDIF
FREE M F


  MCL peut avoir des arguments sur plusieurs lignes grâce aux { }
Dans ce cas les caractères de fin de ligne sont incluses dans l'argument.


Exemple 1

INFO {Mon chien disait souvent :
"Mon pauvre Thomas, en fait tu m'envies..."
Le pire est qu'il avait raison.}

Exemple 2

REQUEST C {"Tu ne pense qu'à toi!
Tu est égoïste!" Vrai|Faux}

  Le { peut se trouver n'importe où sur la première ligne.
  Le } doit se trouver à la fin de l'instruction.
```

```
Exemple 3

REM cette instruction saute une ligne dans la fenêtre de sortie
ECHO {
}
END
```

  MCL gère les erreurs des programmes externes grâce au Label ERR.

```
Exemple de script

** Programme inexistant
sys:Programme_inexistant
END

LABEL ERR
  INFO Ce programme n'existe pas!
RETURN
```

Je vous laisse méditer ces exemples sur les possibilités du préprocesseur de MCL :

```
Exemple 1

REM exécution....
DECLARE d f t
FILE d f t "Exécuter"
IF FILE $t
  $t
ENDIF
FREE d f t

Exemple 2

REM absurde...

DECLARE A C

SET A C
SET $A Salut mon pote
INFO $C

FREE A C
```

1.46 Scripts

  Scripts utiles

```
** Gestion des CHECKITEMS
** Ici exécution de AREXX
IF CHK
   IF PORT AREXX
      INFO ARexx est déjà en fonction.
   ELSE
```

```
      OUT con:100/100/440/100/AREXX
      RexxMast
      DELAY 2
      OUT
   ENDIF
ELSE
   IF PORT AREXX
      rxc
   ELSE
      INFO ARexx n'était pas en fonction.
   ENDIF
ENDIF

** Utilisation de MCL pour ne pas exécuter un utilitaire s'il
** est déjà en route
** Exécution de diskmaster

IF TASK DiskMaster
   DECLARE r
   REQUEST -c r "DiskMaster est déjà en fonction."{
"Faut-il l'exécuter à nouveau?" Exécute|Abandon}
   IF $r=1
      GOSUB exedisk
   ENDIF
   FREE r
ELSE
   GOSUB exedisk
ENDIF
END

LABEL exedisk
   sys:dos/Disk/DiskmasterII
RETURN


** Les deux scripts suivants permettent de définir un projet par défaut
** pour devpac
** Dans le premier on mémorise un fichier
** Dans le deuxième on exécute devpac

** Script 1 (changer projet)

DECLARE d f t
SET d sys:asm
FILE -p #?.ASM d f t "Changer projet"
IF $t
   VWRITE t ram:lastproject
ENDIF
FREE d f t

** Script 2 (charger projet)

DECLARE p
   VREAD t ram:lastproject
CD sys:asm
RUN devpac/Devpac3.02 $p
FREE p
```

```
** Proposition de script pour menu Projet/Nouveau
DECLARE d f t
SET d S:
SET f Menu.M2I
FILE -p #?.M2I d f t "Choix du menu"
IF $t
   NEW $t
ENDIF
FREE d f t

** Proposition de script pour menu Projet/Quitter
DECLARE s
REQUEST s "Vous voulez vraiment quitter?" Oui|NON!
IF $s=1
   EXIT
ENDIF
```

1.47 M2I

```
               Menu-Item Interpretor (M2I)


  Le dernier volet. Une fois que vous avez sauvé votre menu
(l'extension .M2I n'est pas obligatoire mais conseillée), il est prêt à
être utilisé sous M2I.

Syntaxe   [RUN] M2I Nom du menu.M2I

Exemple   run M2I S:Menu.M2I

  M2I s'installe dans la fenêtre active, en principe votre Shell.
Si votre fenêtre comporte un gadget de fermeture, vous pouvez cliquer
dessus pour sortir de M2I (cela ne fermera pas votre Shell).

  Si votre fenêtre Shell est déjà occupée, M2I ouvrira sa propre
fenêtre.

  En cas d'erreur dans votre script, M2I affichera la ligne où se
trouve l'erreur, et vous proposera de passer à la ligne suivante, ou
d'interrompre le script.

  Voir
               Configuration de M2I
```

1.48 Configuration

```
Le fichier de configuration de M2I


  M2I peut afficher sur votre barre de fenêtre un certain nombre
```

d'informations pratiques : La présence de tâches, de port de message,
et de vecteurs Execbase. Cela vous permet d'avoir une idée des tâches
présentes dans votre système, et également de transformer M2I en anti-virus.

  Pour cela vous devez éditer un fichier texte du nom de
  M2I.conf dans le répertoire S: (idem version 3.x)

  Une ligne de configuration comprend trois parties

  Le flag, v p ou t si l'on veut tester un vecteur execbase,
  un port ou une tâche.

  un argument qui doit être l'offset execbase (3 chiffres),
  le port ou la tâche (respectez les maj/minuscules)

  un message à afficher dans la barre de fenêtre (faites court)

Exemple de fichier conf.

```
v 042 Cold!
v 046 Cool!
v 050 Warm!
v 546 MemPtr!
v 550 TagPtr!
p BlackIOPort Blk
p DeluxePaint DPaint
p POWERPACKER PowerP
p AREXX ARexx
t DiskMaster Disk
t DF1 DF1
t CygnusEd CED
```

Vecteurs execbase. Lorsque le vecteur est non-nul le message s'inscrira
dans la barre de fenêtre. Ainsi les vecteurs Coldcapture (042) et
Warmcapture (046) sont souvent occupés par des virus. Les offsets
MemPtr et TagPtr (resp. 546 et 550) servent aux structures résidentes
comme la RAD, mais peuvent abriter un virus.

Tâches. Une tâche est à distinguer d'un processus Shell. Si vous lancez
une tâche par le Shell, vous ne pourrez pas la détecter avec M2I, sauf
les processus qui se détachent du Shell (Diskmaster, powerpacker ...)

Port. Tout port de message qui possède un nom peut-être détecté par M2I.
Les ports ARexx sont très pratiques pour détecter la présence d'une tâche.

1.49 Fin de M2I.guide