Misc

COLLABORATORS

	TITLE : Misc	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		January 23, 2025	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

Misc

1.1 Misc V1.20

```
PureBasic Misc library V1.20

  'Misc' signifie 'Divers' en anglais... On va donc trouver dans cette
  bibliothèque toutes les commandes inclassifiables mais très utiles.

Commandes disponibles:

   Delay
   GetFilePart
   GetPathPart
   MouseWait
   PeekX
   PokeX
   Print
   PrintN
   PrintNumber
   PrintNumberN
   ProgramPriority
   RunProgram
   VWait
```

1.2 delay

```
  Syntaxe
Delay(Time)

  Résumé
Arrête l'execution du programme pendant une durée déterminée.

Time: Durée du temps à attendre (en 1/50 de secondes). Donc pour
      attendre une seconde, il faut mettre une durée de 50.
```

1.3 getfilepart

```
  Syntaxe
FileName$ = GetFilePart(String$)

  Résumé
Retourne la partie 'fichier' d'un chemin AmigaDOS.

Exemple:

  FileName$ = GetFilePart("Dh0:Games/SuperFrog/SuperFrog.exe")

Le contenu de 'FileName$' sera 'SuperFrog.exe'.
```

1.4 getpathpart

```
  Syntaxe
PathName$ = GetPathPart(String$)

  Résumé
Retourne la partie 'chemin' d'un chemin AmigaDOS.

Exemple:

  PathName$ = GetPathPart("Dh0:Games/SuperFrog/SuperFrog.exe")

Le contenu de 'PathName$' sera 'Dh0:Games/SuperFrog/'.
```

1.5 mousewait

```
  Syntaxe
MouseWait()

  Résumé
Arrête l'éxecution du programme et attend que l'utilisateur appuie sur
le bouton gauche de la souris.
```

1.6 peekx

```
  Syntaxe
Result = PeekB/W/L/S(*Address)

  Résumé
Retourne un byte, un word, un long ou une chaîne de caractères, en fonction du
suffix utilisé (B,W,L ou S) qui se trouve à l'adresse spécifiée.


*Address: Adresse mémoire d'où les données vont êtres lues.
```

1.7 pokex

Syntaxe
PokeB/W/L/S(*Address, Data)

Résumé
Ecrit les données (Data) à l'adresse spécifiée (*Adress). Cette peut écrire un byte, word, long ou une chaîne de caractère en fonction du suffixe employé.

*Address: Adresse mémoire où les données vont être écrites.

Data: Données à écrire.

1.8 printn

Syntaxe
PrintN(String$)

Résumé
Affiche la chaine de caractères sur la sortie par défaut (CLI/Shell) et ajoute un retour à la ligne.

String$: Chaine de caratères à afficher.

1.9 print

Syntaxe
Print(String$)

Résumé
Affiche la chaine de caractères sur la sortie par défaut (CLI/Shell)

String$: Chaine de caractères à afficher.

1.10 printnumber

Syntaxe
PrintNumber(Number)

Résumé
Affiche un nombre sur la sortie par défaut (CLI/Shell)

Number: Nombre à afficher.

1.11 printnumbern

```
  Syntaxe
PrintNumberN(Number)

  Résumé
Affiche un nombre sur la sortie par défaut (CLI/Shell)
et ajoute un retour à la ligne.


Number: Nombre à afficher.
```

1.12 vwait

```
  Syntaxe
VWait()

  Résumé
Arrête l'execution du programme jusqu'à ce que le spot de l'écran retourne
en haut à gauche de l'écran (l'image complète a été retracée). Cette commande
permet de synchroniser l'affichage avec l'écran pour des animations très
fluides. Cette commande est aussi connue sous le nom de 'Vertical Blank'.
La durée de cette instruction est dépendante du taux de rafraichissement
de l'écran (plus rapide en NTSC 60 hz qu'en PAL 50 hz par exemple).
```

1.13 programpriority

```
  Syntaxe
OldPriority.b = ProgramPriority(NewPriority)

  Résumé
Permet au développeur de changer la valeur de la priorité de son programme.
L'AmigaOS étant un système totalement multitâche, cette notion de priorité
est très importante. Les valeurs possibles vont de -128 à +127, en sachant
que -128 est la valeur la plus faible, c'est à dire que le programme
ne marchera que quand rien d'autre sur l'AmigaOS ne marche. Dans la
pratique, on utilise la plage -50 à +50 ce qui est amplement suffisant.

Par exemple, si vous programmez un logiciel qui demande des temps de calculs
importants (logiciel 3D, fractale, compression...) il serait de bon ton de
lui attribuer un priorité inferieure à 0 (-1 de préférence), comme ca le systeme
Amiga n'est pas ralenti par cette tache qui s'éxecutera en fond sans même
que l'utilisateur ne s'en rende compte. Par contre si vous attribuez une ↩
   priorité
supérieure à 0, alors le système considèrera que votre programme est très
important et lui allouera beaucoup de ressources processeur au détriment
des autres tâches.

Au contraire, pour un jeu il est préférable de mettre une priorité supérieure
à 10 pour être sur que toutes les ressources disponibles sont utilisées
par votre jeu.
```

NewPriority: Valeur de la nouvelle priorité à affecter à votre programme.

1.14 runprogram

Syntaxe
RunProgram(DefaultPath$, CommandLine$, ASynchrone, Stack)

Résumé
Execute un programme externe à partir de votre programme.

DefaultPath$: Chemin qui sera utilisé par défaut par le programme executé.

CommandLine$: Commande complète à executer (ex: 'Lha -a -2 -r BackUP.lha Dh0:')

ASynchrone: Si vous mettez ce paramètre à 1, alors le programme sera lancé de manière asynchrone, c'est à dire qu'il ne bloquera pas l'execution de votre programme. Sinon, votre programme sera bloqué jusqu'à ce que l'execution de l'autre programme se termine.

Stack: Valeur de la pile à utiliser par le programme executé. Si vous ne savez pas quelle valeur mettre, optez pour '4096'. C'est la valeur par défaut sur AmigaOS.