

**Misc**

**COLLABORATORS**

	<i>TITLE :</i> Misc		
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>	<i>SIGNATURE</i>
WRITTEN BY		August 24, 2022	

**REVISION HISTORY**

NUMBER	DATE	DESCRIPTION	NAME

# Contents

<b>1</b>	<b>Misc</b>	<b>1</b>
1.1	Misc V1.20 . . . . .	1
1.2	delay . . . . .	2
1.3	getfilepart . . . . .	2
1.4	getpathpart . . . . .	2
1.5	mousewait . . . . .	2
1.6	peekx . . . . .	3
1.7	pokex . . . . .	3
1.8	printn . . . . .	3
1.9	print . . . . .	3
1.10	printnumber . . . . .	4
1.11	printnumbern . . . . .	4
1.12	vwait . . . . .	4
1.13	programpriority . . . . .	4
1.14	runprogram . . . . .	5

---

# 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'exécution 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'exécution 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 suffixe utilisé (B,W,L ou S) qui se trouve à l'adresse spécifiée.

\*Address: Adresse mémoire d'où les données vont être lues.

## 1.7 pokex

Syntaxe

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

Résumé

Écrit les données (Data) à l'adresse spécifiée (\*Address). 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 chaîne de caractères sur la sortie par défaut (CLI/Shell) et ajoute un retour à la ligne.

String\$: Chaîne de caractères à afficher.

## 1.9 print

Syntaxe

```
Print(String$)
```

Résumé

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

String\$: Chaîne 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'exécution 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 rafraîchissement 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 une priorité inférieure à 0 (-1 de préférence), comme ça le système Amiga n'est pas ralenti par cette tâche qui s'exécutera 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 exécuté.

CommandLine\$: Commande complète à exécuter (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'exécution de votre programme. Sinon, votre programme sera bloqué jusqu'à ce que l'exécution de l'autre programme se termine.

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