==========

Guide d'utilisation de la fonction *User_Choose_File()*

==========

Cette fonction a été développée par DarkJura pour le site Planète-Casio.com et ses développeurs.
Ce programme fait l'objet d'une license CC BY-NC-SA.

Page du programme : cliquer ici

Langage de programmation : C
Dépendances : MonochromeLib.h, développée par PierrotLL.
Il est nécessaire de décommenter les fonctions suivantes :

- ⇨ ML_clear_vram()
- ⇨ ML_rectangle()

N'oubliez pas d'inclure "File.h" dans votre fichier .c qui utilise la fonction User_Choose_File().

Cette fonction permet d'ouvrir une fenêtre de sélection d'un fichier dans la mémoire de la calculatrice, un peu comme lorsque vous faites CTRL+o dans un éditeur de texte.

Prototype :

```
int User_Choose_File(const FONTCHARACTER path_name_base[], FONTCHARACTER path_name[],
FONTCHARACTER File[], int path_name_size, opt_tri tri, int show_folder_at_the_top,
int search_directory);
```

Cette fonction prend les paramètres suivants :

- `const FONTCHARACTER path_name_base[]` : le répertoire où vous voulez effectuer la recherche.
  Cet argument est de la forme : `FONTCHARACTER PathName[]={'\\','\\','f','l','s','0','\\','*','.', '*', 0};` On précise bien "fls" pour la mémoire de stockage, "crd" pour la carte SD. L'extension doit bien être choisie, en fonction du type de fichiers que vous recherchez. N'oubliez pas le 0 de fin de chaîne. (source)
  Le caractère `'*'` peut être utilisé comme caractère générique.
  Exemples :
  je veux rechercher tous les fichiers dans la mémoire de stockage :
  - `PathName[]={'\\','\\','f','l','s','0','\\','*','.','*',0};`

  je veux rechercher tous les fichiers .py dans le dossier CASIOPY, lui-même dans la mémoire de stockage :
  - `PathName[]={'\\','\\','f','l','s','0','\\','C','A','S','I','O','P','Y','\\','*','.','p','y',0};`

  je veux rechercher dans la carte SD les fichiers commancant par un 'M', quelle que soit leur extension :
  - `PathName[]={'\\','\\','c','r','d','0','\\','M','*','.','*',0};`

- `FONTCHARACTER path_name[]` : prévoir une taille entre 10 et 80.
  Il n'y a pas besoin de l'initialiser au début. Il stocke le chemin d'accès du fichier choisi sans son nom.
  En fin de fonction, il sera par exemple égal à `'\\','\\','f','l','s','0','\\','C','A','S','I','O','P','Y','\\'`.

- `FONTCHARACTER File[]` : prévoir une taille de 13.
  Il n'y a pas besoin de l'initialiser au début. Il stocke le nom du fichier choisi.
  En fin de fonction, il sera par exemple égal à `'S','T','R','A','T','E','G','O','.','g','1','a'`.
- `int path_name_size` : vous devez l'initialiser avec la taille que vous avez choisie pour le deuxième paramètre : `path_name[]`.

- `opt_tri tri` : énumération, représente les options de tri dans l'affichage des fichiers.
  Peut-être égal à :
  - `AUCUN_TRI` : Ne trie pas les fichiers trouvés, ils apparaissent dans le désordre.
  - `DOSSIERS_DEVANT` : Place les dossiers devant, sans tenir compte de l'ordre alphabétique.
  - `DOSSIERS_DEVANT_ORDRE_ALPHABETIQUE` : Place les dossiers devant, trie les fichiers et les dossiers par ordre alphabétique.
  - `TAILLE_CROISSANT` : Place les dossiers devant, trie les fichiers par taille (du plus léger au plus lourd).
  - `TAILLE_DECROISSANT` : Place les dossiers devant, trie les fichiers par taille (du plus lourd au plus léger).

- `int show_folder_at_the_top` : booléen.
  Affiche (1) ou non (0) le chemin d'accès en haut de l'écran.

- `int search_directory` : booléen.
  Recherche (1) ou non (0) les dossiers, quel que soit le chemin d'accès original.

La fonction renvoie 0 si tout s'est déroulé comme prévu, et 1 si l'utilisateur a appuyé sur [EXIT] dans le dossier source. Il est alors inutile de continuer. Les deux paramètres qui stockent le chemin d'accès du fichier n'auront pas été modifié. Si vous continuez, le programme plantera !

Vous pouvez également changer un paramètre important : dans `File.h`, la constante de préprocessor `NB_FILES_MAX` indique le nombre maximal de fichiers et dossiers à charger.

Si, par la suite, vous voulez ouvrir (avec Bfile_OpenFile) le fichier choisi, il vous sera nécessaire de concaténer les chaînes `path_name` et `File`, afin d'obtenir par exemple `FONTCHARACTER File_to_open[]={'\\','\\','f','l','s','0','\\','C','A','S','I','O','P','Y','\\','p','r','o','g','.','p','y',0};`
À cette fin, vous pouvez utiliser la fonction `FONTCHARACTER_cat()` qui prend en argument les deux chaînes à concaténer et les écrit dans le troisième argument. Faites attention aux dépassement de mémoire ! Prévoyez assez de place pour la chaîne qui contiendra les deux autres.

Voici un exemple de code si tout n'a pas été très clair :

```
int AddIn_main(int isAppli, unsigned short OptionNum){
  unsigned int key;
  int handle, j;
  const FONTCHARACTER Path_Name[]={'\\','\\','f','l','s','0','\\','*','.','g','*',0};
  FONTCHARACTER File[13], Path_To_File[50], File_To_Open[63];
  opt_tri mon_tri = DOSSIERS_DEVANT_ORDRE_ALPHABETIQUE;

  if(User_Choose_File(Path_Name, Path_To_File, File, 50, mon_tri, 1, 1) == 1)
      return 0;

  FONTCHARACTER_cat(Path_To_File, File, File_To_Open);
  handle = Bfile_OpenFile(File_To_Open, _OPENMODE_READ);
  if(handle >= 0) {
      /*    code ...    */
  }

  Bfile_CloseFile(handle);
  PrintXY(0, 0, "Fini !", 0);
  GetKey(&key);
  return 1;
}
```