PlistGuide

NasGûl

COLLABORATORS

	TITLE : PlistGuide	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	NasGûl	April 16, 2022	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

PlistGuide

1.1 Plist Documentation

```
p_InitList()

p_WriteFList()

p_CleanList()

p_GetAdrNode()

p_GetNumNode()

p_EnleveNode()

p_AjouteNode()

p_EmptyList()

p_CountNodes()

p_DoUpNode()

p_DoDownNode()

p_SortList()
```

1.2 p_initlist()

```
Fonction     : p_InitList()
Para         : NONE.
Return       : Address of the new list if ok,else NIL.
Description  : Initialise a list.

   just initialise the list and return pointer (lh) .
```

1.3 p_writeflist()

```
Function     : p_WriteFList(list:PTR TO lh)
Para         : Address of a list
Return       : NONE.
Description  : Write in stdout the list data and nodes.

   like this:

       Adr:<address of list>  Head:<lh.head> TailPred:<lh.tailpred>

   and the nodes:

       Adr:<address of node> Succ:<address of succ> Pred:<address of pred> Name:< ↩
          name node>
```

1.4 p_cleanlist()

```
Fonction     : p_CleanList(list:PTR TO lh,doit,dat:PTR TO LONG,mode)
Para         : Address of a List,if doit<>0 free data,the data,just clean or  ↩
   clean and remove.
Return       : Address of clean list.
Description  : Remove all nodes in the list.

       This fonction clean the nodes of a list  and the objects.

       if the list content just nodes (ln) you can free it like this:

       DEF mylist:PTR TO lh
       mylist:=p_InitList()
       .... Add nodes to list.

       just clean the list:

               mylist:=p_CleanList(mylist,FALSE,0,LIST_CLEAN)  /* LIST_CLEAN=0 */

       remove it:

           p_CleanList(mylist,FALSE,0,LIST_REMOVE)   /* LIST_REMOVE=1 */

       if your list content nodes and objects attached to it like:

       OBJECT  obj
           node:ln
           mystring:LONG    /* suppose String() allocation */
           mydata:LONG      /* suppose New() allocation */
       ENDOBJECT

   to clean this list do:

       mylist:=p_CleanList(mylist,TRUE,[14,DISL,18,DISP,DISE],LIST_CLEAN)

               14 is the offset of obj.mystring (SIZEOF ln=14) ,DISL make a  ↩
                  DisposeLink(),
```

```
                18 is the offset of obj.mydata ,DISP make a Dispose().
                DISE is the end of data.

        you can remove by the same way (but with LIST_REMOVE).
```

1.5 p_getadrnode()

```
Fonction     : p_GetAdrNode(list:PTR TO lh,numnode)
Para         : Address of a list,number of a node.
Return       : Address of node or -1.
Description  : Find the address of a node.

   like:

       adr_curnode:=p_GetAdrNode(mylist,4)
```

1.6 p_getnumnode()

```
                 Fonction     : p_GetNumNode(list:PTR TO lh,adrnode)
Para         : Address of a list,address of a node.
Return       : The number of the node,else -1.
Description  : Find the num of a node.

   like:

       num_curnode:p_GetNumNode(mylist,adr_curnode)

   (the inverse of
               p_GetAdrNode()
               )
```

1.7 p_enlevenode()

```
                 Fonction     : p_EnleveNode(list:PTR TO lh,numnode,doit,dat:PTR  ↩
                    TO LONG)
Para         : Address of a list,number of a node,if doit<>0 free data,the data.
Return       : The number of the new selected node in the list.
Description  : Remove a node.

   like:
           currentnode:=p_EnleveNode(mylist,numnode,FALSE,0) /* free a node */

       or:

           currentnode:=p_EnleveNode(mylist,numnode,TRUE,[14,DISL,18,DISP,DISE])  ↩
              /* free a OBJECT obj */

               (see
               p_CleanList()
                for more infos).
```

1.8 p_ajoutenode()

```
                Fonction     : p_AjouteNode(list:PTR TO lh,nodename,adr)
Para         : address of list,the name of a node,adr to copy node if adr<>0.
Return       : the number of the new selected node in the list.
Description  : Add a node and return the new current node (for LISTVIEW_KIND).

   To add a node:

           currentnode:=p_AjouteNode(mylist,'New',0)

   To Add a OBJECT obj (see
               p_CleanList()
               :

       myobj:=New(SIZEOF obj)
       curnode:=p_AjouteNode(mylist,'New Object',myobj)

   you can also Allocate data to the object before add it to the list:

       myobj:=New(SIZEOF obj)
       myobj.mystring:=String(EstrLen(other_string))
       StrCopy(myobj.mystring,other_string,ALL)
       myobj.mydata:=New(400)
       curnode:=p_Ajoutenode(mylist,'New Object',myobj)
```

1.9 p_emptylist()

```
Fonction     : p_EmptyList(list:PTR TO lh)
Para         : Address of a list.
Return       : TRUE if list is empty,else the adress list.
Description  : Look if a list is empty.
```

1.10 p_countnodes()

```
Fonction     : p_CountNodes(list:PTR TO lh)
Para         : address of a list
Return       : number of nodes in the list.
Description  : count nodes in the list.
```

1.11 p_doupnode()

```
Fonction     : p_DoUpNode(list:PTR TO lh,numnode)
Para         : address of a list,num of node.
Return       : the number (and not the address) of the new node selected.
Description  : move up a node.
```

1.12 p_dodownnode()

```
Fonction     : p_DoDownNode(list:PTR TO lh,numnode)
Para         : address of a list,num of node.
Return       : the num of the new selected node.
Description  : make down node.
```

1.13 p_sortlist()

```
Fonction     : p_SortList(list:PTR TO lh)
Para         : address of list.
Return       : NONE.
Description  : Sort a list (found in toolmanager sources).

       This procedure need the Utility.library.
```