Linked

COLLABORATORS			
	TITLE : Linked		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		August 26, 2024	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

1 Linked **1**

1.1 Linked List . . . 1
1.2 addelement . . . 1
1.3 changecurrentelement . . . 2
1.4 clearlist . . . 2
1.5 countlist . . . 2
1.6 firstelement . . . 3
1.7 insertelement . . . 3
1.8 killelement . . . 3
1.9 lastelement . . . 3
1.10 listbase . . . 3
1.11 listindex . . . 4
1.12 nextelement . . . 4
1.13 previouselement . . . 4

Chapter 1

Linked

1.1 Linked List

```
Pure Basic - Linked List library V1.10

  The Linked Lists are object which are dynamically allocated
  depending of your need. This is a list of elements and each
  elements is fully independant of the other. You can add any
  elements you want, inserting elements at the position you need
  deleting some other and more... These kind of datamanagement is
  very used in the AmigaOS as it's the best way to handle data
  when you don't know how many there are.

Commands summary in alphabetical order:

   AddElement
   ChangeCurrentElement
   ClearList
   CountList
   FirstElement
   InsertElement
   KillElement
   LastElement
   ListBase
   ListIndex
   NextElement
   PreviousElement


Example:

   Linked List demo
```

1.2 addelement

```
  SYNTAX
AddElement(linkedlist())
```

```
  COMMAND
Add a new empty element after the actual position. This new element become
the current element of the list.
```

1.3 changecurrentelement

```
  SYNTAX
ChangeCurrentElement(linkedlist(), *NewElement)

  COMMAND
Change the current element of the specified list to the given *NewElement.
The *NewElement must be a pointer to another element which exists in this
list. This function is very useful to remember an element, and calling it
back after doing other processing.


Example:

  *Old_Element = @mylist()      ; Get the address of the actual element

  ResetList(mylist())           ; Doing a search loop to all elements named
  While(NextItem(mylist())      ; "John" and change them to "J"
      If mylist()\name = "John" ;
          mylist()\name = "J"   ;
      EndIf                     ;
  Wend                          ;

  ChangeCurrentElement(mylist(), *Old_Element) ; Restore our element before the  ↩
     search
```

1.4 clearlist

```
  SYNTAX
ClearList(linkedlist())

  COMMAND
Kill all the elements in this list and release their memory. After this
call the list is still usable, but no more elements are in the list.
```

1.5 countlist

```
  SYNTAX
CountList(linkedlist())

  COMMAND
Count how many elements there is in the linked list. It doesn't change the
actual current element.
```

1.6 firstelement

```
  SYNTAX
FirstElement(linkedlist())

  FUNCTION
Change the current list element to the first list element.
```

1.7 insertelement

```
  SYNTAX
InsertElement(linkedlist())

  COMMAND
Add a new empty element before the actual position. This new element become
the current element of the list.
```

1.8 killelement

```
  SYNTAX
KillElement(linkedlist())

  FUNCTION
Remove the current element from the list. After this call, the new current
element is the element which follow or null if you've kill the last element.
```

1.9 lastelement

```
  SYNTAX
LastElement(linkedlist())

  FUNCTION
Change the current list element to the last list element.
```

1.10 listbase

```
  SYNTAX
*ListBase = ListBase(linkedlist())

  FUNCTION
It returns the address of the list base structure ('List' on the AmigaOS)
```

1.11 listindex

```
  SYNTAX
Index = ListIndex(linkedlist())

  STATEMENT
Return the current list element position, considering that the first element
is at the position 1.
```

1.12 nextelement

```
  SYNTAX
Result = NextElement(linkedlist())

  STATEMENT
Change the current list element with the next element and return its
address or return NULL if there is no more elements.
```

1.13 previouselement

```
  SYNTAX
Result = PreviousElement(linkedlist())

  STATEMENT
Change the current list element with the previous element and return its
address or return NULL if the current element was the first element.
```