pListenD

COLLABORATORS			
	TITLE : pListenD		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		March 29, 2025	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

1 pListenD **1**

1.1 pListenD.doc . . . 1
1.2 List.lib/ListAllgemeines() . . . 1
1.3 List.lib/pOS_ListInit() . . . 2
1.4 List.lib/pOS_ListInsert() . . . 2
1.5 List.lib/pOS_ListAddHead() . . . 3
1.6 List.lib/pOS_ListAddTail() . . . 3
1.7 List.lib/pOS_ListRemove() . . . 4
1.8 List.lib/pOS_ListRemHead() . . . 4
1.9 List.lib/pOS_ListRemTail() . . . 5
1.10 List.lib/pOS_ListMoveListTail() . . . 5
1.11 List.lib/pOS_ListMoveListHead() . . . 6
1.12 List.lib/pOS_ListMoveListInsert() . . . 7
1.13 List.lib/pOS_ListCntSucc() . . . 7
1.14 List.lib/pOS_ListGetSuccNodeCnt() . . . 8
1.15 List.lib/pOS_ListSucc() . . . 9
1.16 List.lib/pOS_ListPred() . . . 9
1.17 List.lib/pOS_ListIsEmpty() . . . 10
1.18 List.lib/pOS_ListCheckMember() . . . 10
1.19 List.lib/pOS_ListEnqueue() . . . 11
1.20 List.lib/pOS_ListEnqueueName() . . . 11
1.21 List.lib/pOS_ListEnqueueIName() . . . 12
1.22 List.lib/pOS_ListFindName() . . . 13
1.23 List.lib/pOS_ListFindIName() . . . 13
1.24 List.lib/pOS_ListFindINameV() . . . 14
1.25 List.lib/pOS_ListSwapNode() . . . 15

Chapter 1

pListenD

1.1 pListenD.doc

```
                                List.lib

ListAllgemeines()          pOS_ListAddHead()
pOS_ListAddTail()          pOS_ListCheckMember()
pOS_ListCntSucc()          pOS_ListEnqueue()
pOS_ListEnqueueIName()     pOS_ListEnqueueName()
pOS_ListFindIName()        pOS_ListFindINameV()
pOS_ListFindName()         pOS_ListGetSuccNodeCnt()
pOS_ListInit()             pOS_ListInsert()
pOS_ListIsEmpty()          pOS_ListMoveListHead()
pOS_ListMoveListInsert()   pOS_ListMoveListTail()
pOS_ListPred()             pOS_ListRemHead()
pOS_ListRemove()           pOS_ListRemTail()
pOS_ListSucc()             pOS_ListSwapNode()
```

1.2 List.lib/ListAllgemeines()

```
 STRUKTUREN

        struct pOS_ExList
        struct pOS_ExNode
        struct pOS_List
        struct pOS_Node

 INCLUDES

        pExec/List.h
        pExec/Node.h
        proto/pList.h

 ENUMERATION

        enum pOS_ExNodeType
```

```
BESCHREIBUNG
```

1.3 List.lib/pOS_ListInit()

```
PROTOTYP
      VOID pOS_ListInit
        (
          pOS_List *list
        );

FUNKTION
      Installieren einer leeren Liste.

PARAMETER
      list (_R_A0)
        Zeiger auf die Liste

HINWEIS
      Eine Liste muß zuerst installiert werden, bevor sie benutzt
      (gefüllt/ausgelesen) werden darf!

      /* Head zeigt auf den ersten Eintrag oder Tail in einer leeren Liste */
      list->lh_Head     =(pOS_Node*)&list->lh_Tail;
      /* Tail ist immer NULL */
      list->lh_Tail     =NULL;
      /* TailPred zeigt auf den letzten Eintrag oder Head in einer leeren Liste  ↩
         */
      list->lh_TailPred =(pOS_Node*)&list->lh_Head;

AMIGA FUNKTION
      VOID NewList(struct List *list);
```

1.4 List.lib/pOS_ListInsert()

```
PROTOTYP
      VOID pOS_ListInsert
        (
          pOS_List *list,
          pOS_Node *node,
          const pOS_Node *pred
        );

FUNKTION
      Einfügen eines Nodes in eine Liste

PARAMETER
      list (_R_A0)
        Zeiger auf die Liste
      node (_R_A1)
        Zeiger auf den einzuhängenden Node,
```

```
        der sich in keiner Liste befinden darf
      pred (_R_A2)
        Zeiger auf den Node, hinter dem der aktuelle Node eingehängt
        werden soll; NULL für Listenanfang

SIEHE AUCH
      pOS_ListAddHead(), pOS_ListAddTail(), pOS_ListRemove()

AMIGA FUNKTION
      VOID Insert(struct List *list, struct Node *node, struct Node *pred);
```

1.5 List.lib/pOS_ListAddHead()

```
PROTOTYP
      VOID pOS_ListAddHead
        (
          pOS_List *list,
          pOS_Node *node
        );

FUNKTION
      Einfügen eines Nodes am Anfang einer Liste

PARAMETER
      list (_R_A0)
        Zeiger auf die Liste
      node (_R_A1)
        Zeiger auf den einzuhängenden Node,
        der sich in keiner Liste befinden darf

SIEHE AUCH
      pOS_ListInsert(), pOS_ListAddTail(), pOS_ListRemHead(),
      pOS_ListRemove()

AMIGA FUNKTION
      VOID AddHead(struct List *list, struct Node *node);
```

1.6 List.lib/pOS_ListAddTail()

```
PROTOTYP
      VOID pOS_ListAddTail
        (
          pOS_List *list,
          pOS_Node *node
        );

FUNKTION
      Einfügen eines Nodes am Ende einer Liste

PARAMETER
```

```
      list (_R_A0)
        Zeiger auf die Liste
      node (_R_A1)
        Zeiger auf den einzuhängenden Node,
        der sich in keiner Liste befinden darf

SIEHE AUCH
      pOS_ListInsert(), pOS_ListAddHead(), pOS_ListRemTail(),
      pOS_ListRemove()

AMIGA FUNKTION
      VOID AddTail(struct List *list, struct Node *node);
```

1.7 List.lib/pOS_ListRemove()

```
PROTOTYP
      VOID pOS_ListRemove
        (
          pOS_Node *node
        );

FUNKTION
      Aushängen eines Nodes aus der Liste

PARAMETER
      node (_R_A0)
        Zeiger auf den auszuhängenden Node

HINWEIS
      Im Debugmodus wird beim ausgehängten Node der Succ- und Pred-
      Zeiger auf NULL gesetzt.

SIEHE AUCH
      pOS_ListInsert(), pOS_ListAddHead(), pOS_ListAddTail()

AMIGA FUNKTION
      VOID Remove(struct Node *node);
```

1.8 List.lib/pOS_ListRemHead()

```
PROTOTYP
      pOS_Node *res = pOS_ListRemHead
        (
          pOS_List *list
        );

FUNKTION
      Aushängen des ersten Nodes einer Liste

PARAMETER
```

```
      list (_R_A0)
        Zeiger auf die Liste

ERGEBNIS
      res (_R_D0)
        Adresse des ausgehängten Nodes
        oder NULL wenn die Liste leer ist

HINWEIS
      Im Debugmodus wird beim ausgehängten Node der Succ- und Pred-
      Zeiger auf NULL gesetzt.

SIEHE AUCH
      pOS_ListRemTail(), pOS_ListRemove()

AMIGA FUNKTION
      struct Node *RemHead(struct Node *node);
```

1.9 List.lib/pOS_ListRemTail()

```
PROTOTYP
      pOS_Node *res = pOS_ListRemTail
        (
          pOS_List *list
        );

FUNKTION
      Aushängen des letzten Nodes einer Liste

PARAMETER
      list (_R_A0)
        Zeiger auf die Liste

ERGEBNIS
      res (_R_D0)
        Adresse des ausgehängten Nodes
        oder NULL wenn die Liste leer ist

HINWEIS
      Im Debugmodus wird beim ausgehängten Node der Succ- und Pred-
      Zeiger auf NULL gesetzt.

SIEHE AUCH
      pOS_ListRemHead(), pOS_ListRemove()

AMIGA FUNKTION
      struct Node *RemTail(struct Node *node);
```

1.10 List.lib/pOS_ListMoveListTail()

```
PROTOTYP
      VOID pOS_ListMoveListTail
        (
          pOS_List *listsource,
          pOS_List *listdest
        );

FUNKTION
      Einfügen einer Liste am Ende einer anderen Liste.
      Dabei werden alle Nodes aus listsource nach listdest übertragen.

PARAMETER
      listsource (_R_A0)
        Zeiger auf die Source-Liste, dessen Nodes am Ende von listdest
        eingehängt werden
      listdest (_R_A1)
        Zeiger auf die Destination-Liste, die erweitert wird

HINWEIS
      Nach der Funktion ist die Source-Liste leer und kann
      ganz normal bearbeitet werden.

SIEHE AUCH
      pOS_ListMoveListHead(), pOS_ListMoveListInsert(),
      pOS_ListAddTail()

AMIGA FUNKTION
```

1.11 List.lib/pOS_ListMoveListHead()

```
PROTOTYP
      VOID pOS_ListMoveListHead
        (
          pOS_List *listsource,
          pOS_List *listdest
        );

FUNKTION
      Einfügen einer Liste am Anfang einer anderen Liste.
      Dabei werden alle Nodes aus listsource nach listdest übertragen.

PARAMETER
      listsource (_R_A0)
        Zeiger auf die Source-Liste, desses Nodes am Anfang von listdest
        eingehängt werden
      listdest (_R_A1)
        Zeiger auf die Destination-Liste, die erweitert wird

HINWEIS
      Nach der Funktion ist die Source-Liste leer und kann
      ganz normal bearbeitet werden.
```

```
SIEHE AUCH
      pOS_ListMoveListTail(), pOS_ListMoveListInsert(),
      pOS_ListAddHead()

AMIGA FUNKTION
```

1.12 List.lib/pOS_ListMoveListInsert()

```
PROTOTYP
      VOID pOS_ListMoveListInsert
       (
         pOS_List *listsource,
         pOS_List *listdest,
         pOS_Node *pred
       );

FUNKTION
      Einfügen einer Liste in eine andere Liste.
      Dabei werden alle Nodes aus listsource hinter pred in
      listdest übertragen.

PARAMETER
      listsource (_R_A0)
        Zeiger auf die Source-Liste, desses Nodes in listdest
        eingehängt werden
      listdest (_R_A1)
        Zeiger auf die Destination-Liste, die erweitert wird
      pred (_R_A2)
        Zeiger auf den Node in listdest, hinter dem die listsource-
        Nodes eingefügt werden sollen; NULL für Listenanfang

HINWEIS
      Nach der Funktion ist die Source-Liste leer und kann
      ganz normal bearbeitet werden.

SIEHE AUCH
      pOS_ListMoveListHead(), pOS_ListMoveListTail(),
      pOS_ListInsert()

AMIGA FUNKTION
```

1.13 List.lib/pOS_ListCntSucc()

```
PROTOTYP
      pOS_Node *res = pOS_ListCntSucc
        (
          const pOS_List *list,
          const pOS_Node *node,
          ULONG cnt
        );
```

```
FUNKTION
      Ermitteln eines Nodes anhand seiner Position in der Liste

PARAMETER
      list (_R_A0)
        Zeiger auf die Liste
      node (_R_A1)
        Zeiger auf den Node als Suchbasis oder NULL für Listenanfang
      cnt (_R_D0)
        gewünschte Position in der Liste
        (relativ ab node oder absolut ab list)

ERGEBNIS
      res (_R_D0)
        gefundener Node, der an der cnt. Position in der Liste hängt
        oder NULL

SIEHE AUCH
      pOS_ListGetSuccNodeCnt()

AMIGA FUNKTION
```

1.14 List.lib/pOS_ListGetSuccNodeCnt()

```
PROTOTYP
      ULONG res = pOS_ListGetSuccNodeCnt
        (
          const pOS_Node *node
        );

FUNKTION
      Liefert die Position des Nodes in der Liste.
      Gezählt wird ab 0=erster in der Liste.

PARAMETER
      node (_R_A1)
        Zeiger auf den Node

ERGEBNIS
      res (_R_D0)
        Position ab 0 gezählt, an welcher der Node in der Liste hängt

HINWEIS
      Wird für node NULL übergeben, so wird 0 zurückgeliefert.
      Hängt der gesuchte Node als erstes in der Liste, wird ebenfalls
      0 zurückgeliefert.

SIEHE AUCH
      pOS_ListCntSucc()

AMIGA FUNKTION
```

1.15 List.lib/pOS_ListSucc()

```
PROTOTYP
      pOS_Node *res = pOS_ListSucc
        (
          const pOS_List *list,
          const pOS_Node *node
        );

FUNKTION
      Nachfolger eines Nodes ermitteln

PARAMETER
      list (_R_A0)
        Zeiger auf die Liste
      node (_R_A1)
        Zeiger auf den Node, der sich in der Liste befindet und
        dessen Nachfolger ermittelt werden soll
        oder NULL wenn der erste Eintrag der Liste ermittelt werden soll

ERGEBNIS
      res (_R_D0)
        nächster vorhandener Node in der Liste
        oder NULL wenn kein weiterer vorhanden ist

BEISPIEL
      /* Vorwärtsauslesen aller Nodes in einer Liste */
      pOS_List *liste;  // kann mittels pOS_ListInit() installiert
                        // und mittels pOS_ListInsert() gefüllt werden
      pOS_Node *hilf;
      for(hilf=NULL; hilf=pOS_ListSucc(liste,hilf); )
        printf("Node 0x%08lX\n",hilf);

SIEHE AUCH
      pOS_ListPred(), pOS_ListIsEmpty()

AMIGA FUNKTION
```

1.16 List.lib/pOS_ListPred()

```
PROTOTYP
      pOS_Node *res = pOS_ListPred
        (
          const pOS_List *list,
          const pOS_Node *node
        );

FUNKTION
      Vorgänger eines Nodes ermitteln

PARAMETER
      list (_R_A0)
```

```
        Zeiger auf die Liste
      node (_R_A1)
        Zeiger auf den Node, der sich in der Liste befindet und
        dessen Vorgänger ermittelt werden soll
        oder NULL wenn der letzte Eintrag der Liste ermittelt werden soll

ERGEBNIS
      res (_R_D0)

SIEHE AUCH
      pOS_ListSucc(), pOS_ListIsEmpty()

AMIGA FUNKTION
```

1.17 List.lib/pOS_ListIsEmpty()

```
PROTOTYP
      BOOL pOS_ListIsEmpty
        (
          const pOS_List *list
        );

FUNKTION
      Prüfen ob die angegebene Liste leer ist

PARAMETER
      list (_R_A0)
        Zeiger auf die Liste

ERGEBNIS
      res (_R_D0)
        TRUE wenn die angegebene Liste leer ist,
        sonst FALSE.

SIEHE AUCH
      pOS_ListSucc(), pOS_ListPred()

AMIGA FUNKTION
      BOOL IsListEmpty(struct List *list);
```

1.18 List.lib/pOS_ListCheckMember()

```
PROTOTYP
      BOOL pOS_ListCheckMember
        (
          const pOS_List *list,
          const pOS_Node *node
        );

FUNKTION
```

```
      Prüfen ob ein Node in der angegebenen Liste eingehängt ist

PARAMETER
      list (_R_A0)
        Zeiger auf die Liste
      node (_R_A1)
        Zeiger auf den zu prüfenden Node

ERGEBNIS
      res (_R_D0)
        TRUE wenn der node in der angegebenen list eingehängt ist,
        sonst FALSE.

AMIGA FUNKTION
```

1.19 List.lib/pOS_ListEnqueue()

```
PROTOTYP
      VOID pOS_ListEnqueue
        (
          pOS_ExList *list,
          pOS_ExNode *node
        );

FUNKTION
      Einfügen eines Nodes in eine Liste anhand seiner Priorität

PARAMETER
      list (_R_A0)
        Zeiger auf die Liste
      node (_R_A1)
        Zeiger auf den Node, der noch in keiner Liste eingehängt
        sein darf und dessen node->ln_Pri-Wert als Einfügekriterium
        dient

HINWEIS
      Bei mehreren gleichen Prioritätswerten, wird der neue Node
      hinter alle anderen (prioritätsgleichen) Nodes eingehängt.

SIEHE AUCH
      pOS_ListEnqueueName(), pOS_ListEnqueueIName()

AMIGA FUNKTION
      VOID Enqueue(struct List *list, struct Node *node);
```

1.20 List.lib/pOS_ListEnqueueName()

```
PROTOTYP
      VOID pOS_ListEnqueueName
        (
```

```
          pOS_ExList *list,
          pOS_ExNode *node,
          SLONG mode
        );

FUNKTION
      Einfügen eines Nodes in eine Liste anhand seines Namens

PARAMETER
      list (_R_A0)
        Zeiger auf die Liste
      node (_R_A1)
        Zeiger auf den Node, der noch in keiner Liste eingehängt
        sein darf und dessen node->ln_Name-Eintrag als Einfügekriterium
        dient
      mode (_R_D0)
        1: Alphabetisch aufsteigend einsortieren (A,B,C,...,a,b,c,...)
        0: Alphabetisch absteigend einsortieren (...,c,b,a,...,C,B,A)

HINWEIS
      Wechseln Sie den Modus beim Einfügenen mehrerer Nodes in
      eine Liste, ist die Liste danach nicht korrekt sortiert!

SIEHE AUCH
      pOS_ListEnqueue(), pOS_ListEnqueueIName()

AMIGA FUNKTION
```

1.21 List.lib/pOS_ListEnqueueIName()

```
PROTOTYP
      VOID pOS_ListEnqueueIName
        (
          pOS_ExList *list,
          pOS_ExNode *node,
          SLONG mode
        );

FUNKTION
      Einfügen eines Nodes in eine Liste anhand seines Namens
      ohne Beachtung von Groß- und Kleinschreibung

PARAMETER
      list (_R_A0)
        Zeiger auf die Liste
      node (_R_A1)
        Zeiger auf den Node, der noch in keiner Liste eingehängt
        sein darf und dessen node->ln_Name-Eintrag als Einfügekriterium
        dient
      mode (_R_D0)
        1: Alphabetisch aufsteigend einsortieren (A=a,B=b,C=c,...)
        0: Alphabetisch absteigend einsortieren (...,C=c,B=b,A=a)

HINWEIS
```

```
      Wechseln Sie den Modus beim Einfügenen mehrerer Nodes in
      eine Liste, ist die Liste danach nicht korrekt sortiert!

SIEHE AUCH
      pOS_ListEnqueue(), pOS_ListEnqueueName()

AMIGA FUNKTION
```

1.22 List.lib/pOS_ListFindName()

```
PROTOTYP
      pOS_ExNode *res = pOS_ListFindName
        (
          const pOS_ExList *list,
          const CHAR *name
        );

FUNKTION
      Suchen eines Nodes anhand seines Namens in einer Liste
      unter Beachtung von Groß- und Kleinschreibung.

PARAMETER
      list (_R_A0)
        Zeiger auf die Liste
      name (_R_A1)
        Name nach dem gesucht werden soll. Es wird dabei zwischen
        Groß- und Kleinschreibung unterschieden.

ERGEBNIS
      res (_R_D0)
        Zeiger auf den gefundenen Node,
        oder NULL wenn kein passender Node in der Liste vorhanden ist

SIEHE AUCH
      pOS_ListFindIName(), pOS_ListFindINameV()

AMIGA FUNKTION
      struct Node *FindName(struct List *list, STRPTR name);
```

1.23 List.lib/pOS_ListFindIName()

```
PROTOTYP
      pOS_ExNode *res = pOS_ListFindIName
        (
          const pOS_ExList *list,
          const CHAR *name
        );

FUNKTION
      Suchen eines Nodes anhand seines Namens in einer Liste
```

```
      ohne Beachtung von Groß- und Kleinschreibung.

PARAMETER
      list (_R_A0)
        Zeiger auf die Liste
      name (_R_A1)
        Name nach dem gesucht werden soll. Es wird dabei nicht
        zwischen Groß- und Kleinschreibung unterschieden.

ERGEBNIS
      res (_R_D0)
        Zeiger auf den gefundenen Node,
        oder NULL wenn kein passender Node in der Liste vorhanden ist

SIEHE AUCH
      pOS_ListFindName(), pOS_ListFindINameV()

AMIGA FUNKTION
```

1.24 List.lib/pOS_ListFindINameV()

```
PROTOTYP
      pOS_ExNode *res = pOS_ListFindINameV
        (
          const pOS_ExList *list,
          const CHAR *name,
          pOS_ExNode *node
        );

FUNKTION
      Suchen eines Nodes anhand seines Namens in einer Liste
      ohne Beachtung von Groß- und Kleinschreibung.

PARAMETER
      list (_R_A0)
        Zeiger auf die Liste
      name (_R_A1)
        Name nach dem gesucht werden soll. Es wird dabei nicht
        zwischen Groß- und Kleinschreibung unterschieden.
      node (_R_A2)
        Zeiger auf den, in der Liste vorhandenen Node, ab dem die
        Suche begonnen werden soll oder NULL für komplette Liste.

ERGEBNIS
      res (_R_D0)
        Zeiger auf den gefundenen Node,
        oder NULL wenn kein passender Node in der Liste vorhanden ist

BEISPIEL
      const pOS_ExList *liste;  // zu durchsuchende Liste
      const CHAR       *name;   // zu suchender Name
      const pOS_ExNode *hilf;
      for(hilf=NULL; hilf=pOS_ListFindINameV(liste,name,hilf); ) {
        printf("Node 0x%08lX, Name '%s'\n",hilf,hilf->ln_Name);
```

```
        if((hilf=pOS_ListSucc(liste,hilf)) == NULL) break;
      }

SIEHE AUCH
      pOS_ListFindName(), pOS_ListFindIName()

AMIGA FUNKTION
```

1.25 List.lib/pOS_ListSwapNode()

```
PROTOTYP
      VOID pOS_ListSwapNode
        (
          _R_A0 const pOS_List *list,
          _R_A1 pOS_Node *node1,
          _R_A2 pOS_Node *node2
        );

FUNKTION
      Vertauschen zweier Nodes in einer Liste.

PARAMETER
      list (_R_A0)
        Zeiger auf die Liste
      node1 (_R_A1)
        Zeiger auf den ersten zu tauschenden Node
      node2 (_R_A2)
        Zeiger auf den zweiten zu tauschenden Node

SIEHE AUCH
      pOS_ListInsert(), pOS_ListRemove()

AMIGA FUNKTION
```