NodeMaster

COLLABORATORS			
	TITLE : NodeMaster		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		February 14, 2023	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

NodeMaster

1.1 Amiga-E : NodeMaster

```
                                ** NodeMaster - Original By Fabio Rotondo **

          ** This Object is part of the Amiga Foundation Classes **

                        ** DOCUMENTATION GUIDE **


               Introduction

               Author Info

               History
                    Amiga Foundation Classes

Requires:  resourceTracker

Base:     $0001

COMMANDS                              BRIEF DESCRIPTION
-------------------------------------------------------------------------

               nodemaster(resTracker=NIL)
                Initializes the NodeMaster object

               add(object, mode=NM_ADD_TAIL)
                Add obejct to the list

               addr()
                Get the pointer to the Exec list

               change(object)
                Change current node object

               changenum(newnum)
                Change current node ordinal number

               changepos(node)
                Change current node position
```

```
clear()
 Clear ALL list

clearstack()
 Clears all push()ed items

clone()
Clones the current object

del()
 Delete the current node

empty()
 Check if there are nodes in list

error()
Returns last error code

first()
 Jump to the first node

get()
 Get the pointer to the current node

insert(object)
 Insert an object AFTER the current one

isfirst()
Checks if the current item is the first

islast()
Checks if the current item is the last

item(numitem)
 Jump to the specific item

last()
 Jump to the last node

obj()
 Get the PTR of the current object

numitems()
 Return number of nodes in memory

numpos()
 Returns current ordinal position

pop(pos=TRUE)
 Get a node from the stack

prev()
 Go to the previous node

push()
 Push a node on the stack
```

```
sort(sortroutine, info=NIL)
 Sort a list of items

succ()
 Go to the next node

version()
 Returns class version AND revision


Error Table
```

1.2 Author(s) Info


```
Original By:  Fabio Rotondo  (fsoft@intercom.it)

E Version By: Fabio Rotondo

C++ Version By: Massimo Tantignone (tanti@intercom.it)

Address:

        Fabio Rotondo
        C.so Vercelli 9
        28100 Novara
        ITALY

        e-mail: fsoft@intercom.it
                Fabio.Rotondo@deagostini.it

        Phone:  (ITA) - (0)321 -   459676  (home)
                (ITA) - (0)321 -   424272  (office)
                (ITA) - (0)338 -  7336477  (GSM Phone)


        Web:    http://www.intercom.it/~fsoft               (my home page)

                http://www.intercom.it/~fsoft/ablast.html   (Amiga Blast Home  ↩
                   Page)
        ------------------------------------------------------------------------ ↩


        Massimo Tantignone

        ITALY

        e-mail: tanti@intercom.it

        Web:    http://www.intercom.it/~amigaws
```

1.3 History

```
                Legend: NEW - New feature
          ADV - Advanced feature
          FIX - Bug fix

V3.50 - NEW: Added  resourceTracker  support.
      - NEW: Added
                clone()
                method.

V3.40 - NEW: Added error() method.
      - NEW: Added isfirst() method.
      - NEW: Added islast() method.

V3.30 - FIX: Fixed a minor bug in the clear() method.
      - ADV: Clear routine optimized.
      - ADV: Better docs.
      - NEW: Added islast() method.
      - NEW: Added isfirst() method.
      - NEW: Added version() method.
      - ADV: List constructors are INSIDE NodeMaster source for better porting.

V3.20 - ADV: Optimized code.
      - ADV: Now nodemaster() calls clearstack() instead of freeing stack's memory ↩
        by itself.
      - ADV: Now clear() calls clearstack() instead of freeing stack's memory by  ↩
        itself.
      - ADV: clearstack() is faster.
      - ADV: add() now is faster (and shorter!)
      - ADV: removed all internal self.empty() calls and replaced with the (faster ↩
        )
            IsListEmpty() macro.
      - ADV: Now item() patches positions <0

V3.10 - ADV: clear() modified: now the LIST IS correctly initialized.
      - FIX: now frees the LIST pointer correctly.


V3.00 - ADV: General Optimizations
      - ADV: Now first(), prev(), succ(), last(), change(), item(), insert(), add ↩
        ()
            return the item address (instead of TRUE) and FALSE when there are
            no more items.
      - ADV: Now stacked push() and pop() have 8 stack levels!!
      - ADV: Now pop() has an optional parameter (pos=TRUE): set it to FALSE if
            you want just to free a stacked push()
      - NEW: method clearstack() to free all push()ed positions.
      - FIX: empty() used to crash when used after a del() with no items.
      - NEW: sort() method. Incredibly fast and general-purpose sort algo
            written by Andrea Galimberti.
      - ADV: optimized prev() and succ() methods. Now about 20% faster.
      - ADV: changepos() modified: now checks againgsno items, and return the
            data inside the item or NIL.
      - ADV: del() modified: now returns the pointer to the data inside the
            next active node.
```

```
      - FIX: pos() has been renamed numpos()
      - FIX: now SN_ADD_TAIL, SN_ADD_HERE, SN_ADD_HEAD are called:
                NM_ADD_TAIL, NM_ADD_HERE, NM_ADD_HEAD (short: change the SN_
                with NM_)
      - ADV: change() method is now smarter and returns PTR TO obj data or
             FALSE.
      - ADV: better docs and examples.
```

1.4 Introduction

```
INTRODUCTION

NodeMaster  is  a generic object handling module which will allow you to easily
create  lists  of  whatever  you  want.  Its  generic  structure  is ideal for
polimorphism  and  inheritance in other objects.  This object just take care of
creating  Exec  Lists  and  Nodes  of something you pass to it (we will call it
"object")  and  then  it has great power in Lists manipulation.  You can easily
add/del nodes from the list, go to a specific item by its ordinal number and so
on...

I have worked out this module structuring it on Amiga Exec's Nodes.  This means
that  everything  you  will add/remove to a NodeMaster will be done on a System
List Node.

This code is very Exec List-based, so you can do whatever you want.

Main features are:

    * Push/Pop commands to save/restore a special node position.

    * Insert command to add a object AFTER another (usually it is added as Last
      node)

    * Item   command to go to a specific item by its ordinal number.

NEW V2.0 Version features:

    * New method pos() which returns the actual ordinal node position!

    * Enhanced item() method: now it does not scans from first() item to
      selected, but just +/- delta objects from the starting point.

    * Internal sniff-snuff ;)

    * NO MORE ENFORCER'S HITS! ;)

    * Returns Raise() errors instead of return codes!!

    * More examples!

NEW V2.10 Version features:

    * New changenum() method to force NodeMaster to change the
      internal ordinal number of current node.
```

```
    * More code sniff-snuff ;)

NEW V3.00 Version features:

    * General improvements in speed.

    * NEW! Low-Level tree-algo sort (FAST!!)

    * Now push() and pop() has 8 levels of stack!!!

    * Now NodeMaster is part of the  Amiga Foundation Classes
```

1.5 NodeMaster / Error Table

```
ERROR VALUE     | DESCRIPTION
----------------+-------------------------------------
    $0000       | No Memory
```

1.6 Amiga Foundation Classes: NodeMaster/NodeMaster()

```
           NAME: NodeMaster(resTracker=NIL : PTR TO resourceTracker)

       SYNOPSIS: nodemaster(resourceTracker * resTracker)

    DESCRIPTION: Use this command to inizialize a NodeMaster object.

          INPUT: resTracker   - (Optional) Pointer to the Resource Tracker.

        RESULTS: NONE.

       SEE ALSO:
```

1.7 Amiga Foundation Classes: NodeMaster/add()

```
                          NAME: add(object:PTR TO LONG, mode=SN_ADD_TAIL)

       SYNOPSIS: APTR add(APTR s, ULONG mode = NM_ADD_TAIL)

    DESCRIPTION: Use this command to add an object to the list.

          INPUT: object         - PTR TO LONG. This is the object to add.
                 mode           - (OPTIONAL) This flag is very useful to
                                  choose _where_ a new node will be added.
                                  Default is as last one, but you can
                                  add it as the first line or in the middle
                                  of the list (same as insert() command).

                                  Possible values are:

                                  NM_ADD_HEAD - Use this one to add the node
```

```
                                        as the first in list.

                                NM_ADD_HERE - Use this one to add the node
                                              AFTER the current one.
                                              (Same as insert() method)

                                NM_ADD_TAIL - (Default) Use this one to add
                                              the node as the last in list.


     RESULTS: PTR TO obj     - PTR TO add()ed obj: everything went fine.

              Raise()        - An exception in case of problems.

        NOTE: Starting from V3.00, NodeMaster will return the PTR TO obj
              you have just add()ed, if successfull.

              Note also that constants SN_ADD_* are now called NM_ADD_*.

              Please, be careful and *NOT* use NM_ADD_HERE if you do not
              know for sure that your list is not empty: empty list can
              cause problems. If you want to do things in a safe way
              use the insert() method, instead of add().

    SEE ALSO:
              insert()

              del()

              item()
```

1.8 Amiga Foundation Classes: NodeMaster/get()

```
                        NAME: get()

      SYNOPSIS: Node *get(void)

   DESCRIPTION: Use this command to get a pointer to the current Exec List
              node.

         INPUT: NONE.

       RESULTS: A PTR TO ln (An Exec List Node)

      SEE ALSO:
              addr()
```

1.9 Amiga Foundation Classes: NodeMaster/addr()

```
                        NAME: addr()

      SYNOPSIS: List *addr(void)
```

```
DESCRIPTION: Use this command to get the addr of the Exec List.

      INPUT: NONE.

    RESULTS: A PTR TO lh  (an Exec List Header).

       NOTE: This command is useful expecially with ListView gadgets
             which requires a PTR TO an Exec List Header. All you have
             to do is:
                Gt_SetGadgetAttrsA(listgad, win, req,
                    [GTLV_LABELS, NodeMasterobj.addr(), 0,0])

             And the ListView will show your new list.
             Please refer to RMKM and autodocs for more infos regarding
             Gadtools and ListViews.

   SEE ALSO:
             get()
```

1.10 Amiga Foundation Classes: NodeMaster/push()

```
                     NAME: push()

   SYNOPSIS: BOOL push(void);

DESCRIPTION: Use this command to memorize the current node position.

      INPUT: NONE.

    RESULTS: TRUE  - push()ing successful.

             FALSE - push()ing failed (out of stack space, or no items)

       NOTE: * From V2.00 it does not restore the internal ordinal number.
               See
             item()
             for a better
               explanation.

             * Starting from V3.00 now push() and pop() have 8 stack levels.

   SEE ALSO:
             pop()

             first()

             last()

             changepos()

             changenum()
```

1.11 Amiga Foundation Classes: NodeMaster/pop()

```
                        NAME: pop(pos=TRUE)

    SYNOPSIS: APTR pop(BOOL autopos = TRUE)

 DESCRIPTION: Use this command to restore current node to the one
              previously Push()ed.

       INPUT: pos        - (default TRUE) this is a boolean flag.

                           TRUE, restores the previously push()ed
                                 node in list.

                           FALSE, just removes the push()ed node from
                                  the stack.

     RESULTS: PTR TO obj - if pop() succeded and pos=TRUE.

              FALSE      - not pop()ed or pos=FALSE.

        NOTE: * If no node was Push()ed the current node won't change.

              * From V2.00 it does not restore the internal ordinal number.
                See
              item()
              for a better
                explanation.

              * Starting from V3.00, push() and pop() have 8 stack levels.

    SEE ALSO:
              push()

              changepos()

              changenum()
```

1.12 Amiga Foundation Classes: NodeMaster/obj()

```
                        NAME: obj()

    SYNOPSIS: APTR obj(void);

 DESCRIPTION: Use this command to get the current node's PTR TO object.

       INPUT: NONE.

     RESULTS: PTR TO obj data to the current node object.

    SEE ALSO:
              add()
```

1.13 Amiga Foundation Classes: NodeMaster/del()

```
                        NAME: del()

    SYNOPSIS: APTR del(void)

 DESCRIPTION: Use this command to delete the current node.
              After deletion the CURRENT NODE will be the next one.
              If the node you deleted was the last one, then the next
              will be the previous one.

       INPUT: NONE.

     RESULTS: PTR TO next obj data. Starting from V3.00 del() method
              returns the pointer to the next actual obj data, or NIL
              if it was the last object avaible in list (ie. list empty)

    SEE ALSO:
              clear()

              add()
```

1.14 Amiga Foundation Classes: NodeMaster/clearstack()

```
                        NAME: clearstack()

    SYNOPSIS: void clearstack(void)

 DESCRIPTION: Use this method to clear all push()ed nodes in stack.

       INPUT: NONE.

     RESULTS: push() stack will be cleared.

    SEE ALSO:
              push()

              pop()
```

1.15 Amiga Foundation Classes: NodeMaster/numitems

```
                        NAME: numitems()

    SYNOPSIS: ULONG numitems(void)

 DESCRIPTION: Use this command to know how many items are added to the
              list.

       INPUT: NONE.

     RESULTS: items  - LONG. Number of items.
```

```
   SEE ALSO:
             add()

             del()
```

1.16 Amiga Foundation Classes: NodeMaster/empty()

```
       NAME: empty()

   SYNOPSIS: BOOL empty(void)

DESCRIPTION: Use this command to check whether the list is empty or not.

      INPUT: NONE.

    RESULTS: TRUE       - List is empty
             FALSE      - At least one item is present.

   SEE ALSO:
```

1.17 Amiga Foundation Classes: NodeMaster/first()

```
                       NAME: first()

   SYNOPSIS: APTR first(void)

DESCRIPTION: Use this command to jump to the first object in the list.

      INPUT: NONE.

    RESULTS: PTR TO first obj  - Position correct.
             FALSE             - Cannot go to the first (maybe list empty).

       NOTE: Before V3.00 first() used to return TRUE when positioning
             correctly to the first item, now it returns directly the
             obj data, so you don't have to call the "obj()" method.

   SEE ALSO:
             last()

             del()

             obj()
```

1.18 Amiga Foundation Classes: NodeMaster/last()

```
                       NAME: last()

   SYNOPSIS: APTR last(void)
```

```
DESCRIPTION: Use this command to position current node to the last one.

      INPUT: NONE.

    RESULTS: PTR TO obj data  - Operation successfull.

             FALSE            - No items.

       NOTE: V3.00 - Now last() returns directly the obj data, so you
             do not have to do a "obj()" anymore.

   SEE ALSO:
             first()

             item()
```

1.19 Amiga Foundation Classes: NodeMaster/succ()

```
                      NAME: succ()

   SYNOPSIS: APTR succ(void)

DESCRIPTION: Use this command to position current node to the next one in
             list.

      INPUT: NONE.

    RESULTS: PTR TO obj    - PTR TO obj data: positioning successful.
             FALSE         - No next items.

       NOTE: It is a structural problem linked TO this method. If you
             add to NodeMaster some numbers (eg. 1, 2, 3, 4,...,n)
             the class could give you a wrong result, when it encounters
             a number "0", because it will return a "0" and you could assume
             that it is a FALSE (no next items) value.
             You should check using the
             islast()
             method.

   SEE ALSO:
             prev()

             islast()

             last()
```

1.20 Amiga Foundation Classes: NodeMaster/prev()

```
                      NAME: prev()

   SYNOPSIS: APTR prev(void)
```

```
DESCRIPTION: Use this command to go to the previous string in the list.

      INPUT: NONE.

    RESULTS: PTR TO obj   - PTR TO obj data: positioning successful.
             FALSE        - No previous items.

       NOTE: see note in
             succ()
                   SEE ALSO:
             succ()

             first()

             last()
```

1.21 Amiga Foundation Classes: NodeMaster/insert()

```
                      NAME: insert(object:PTR TO LONG)

   SYNOPSIS: APTR insert(APTR s)

DESCRIPTION: Use this command to add an object AFTER the current node.

      INPUT: object     - PTR TO CHAR. Object you want to add.

    RESULTS: PTR TO object data.

       NOTE: The current node WILL NOT change!!!

   SEE ALSO:
             add()

             del()

             item()
```

1.22 Amiga Foundation Classes: NodeMaster/item()

```
                      NAME: item(numitem)

   SYNOPSIS: APTR item(ULONG n)

DESCRIPTION: Use this command to position current node to the ordinal
             numitem node.

      INPUT: numitem     - LONG. Ordinal value of node position.

    RESULTS: PTR TO obj data - Position correct.

             FALSE           - List is empty.
```

```
NOTES: V3.00 - Now this method() returns directly obj data when
       called successfully, so you don't have to do an obj() anymore.

       From V2.00 this method changes a bit. Now it scans prev() or
       succ() item starting from the current position. So it is
       faster, but you have to be careful.
       If you use the method changepos(), the internal ordinal
       position is not changed. So you have to do a first() method
       before calling item().

       In any case, if you think that something in your source could
       behaves differently, just do a first() method call and
       everything will behaves as always.

       Also pop() and push() do not update internal ordinal position.
       Do a first() also before an item() after them.

SEE ALSO:
       numitems()

       changepos()

       changenum()
```

1.23 Amiga Foundation Classes: NodeMaster/change()

```
    NAME: change(data:PTR TO LONG)

SYNOPSIS:  PTR change(APTR s)

DESCRIPTION: Use this command to change the PTR to the data field of
       current object.

   INPUT: data  - PTR TO LONG. New data to change the old one with.

 RESULTS: PTR TO obj data - Changin successful.

       FALSE           - No change (maybe list empty)

SEE ALSO:
```

1.24 Amiga Foundation Classes: NodeMaster/clear()

```
                    NAME: clear()

SYNOPSIS  BOOL clear(void)

DESCRIPTION: Use this command to clear all items in the list.

   INPUT: NONE.
```

```
    RESULTS: The list will be completely empty.

   SEE ALSO:
             del()
```

1.25 Amiga Foundation Classes: NodeMaster/changepos()

```
                      NAME: changepos(node:PTR TO ln)

   SYNOPSIS:   APTR cngepos(Node *node)

DESCRIPTION: Use this command to change current node position to another.

      INPUT: node   - (PTR TO ln) new list node to change position to.

       NOTE: You *MUST* know exactly what you are doing. Passing a wrong
             node as parameter could get to Software Failures and so on.
             This command is designed only for "professional" user who
             intend build new object inheriting this one.

             From V2.00 note also that it does not update internal ordinal
             number. See
             item()
                    RESULTS: The current node position will be changed.

   SEE ALSO:
             first()

             last()

             item()
```

1.26 Amiga Foundation Classes: NodeMaster/pos()

```
                      NAME: numpos()

   SYNOPSIS: LONG numpos(void)

DESCRIPTION: Use this command to know the ordinal object position inside
             the list.

      INPUT: NONE.

    RESULTS: The current ordinal node position is returned.

   SEE ALSO:
             first()

             last()

             item()
```

1.27 Amiga Foundation Classes: NodeMaster/changenum()

```
                    NAME: changenum(newval)

   SYNOPSIS:   void chaenum(ULONG newnum)

DESCRIPTION: Use this command to change current node ordinal position
             number.

      INPUT: newval   - (LONG) new ordinal number to assign to the current
                        node.

       NOTE: You *MUST* know exactly what you are doing. Passing a wrong
             value as parameter could get to Software Failures and so on.
             This command is designed only for "professional" user who
             intend build new object inheriting this one.

    RESULTS: The current node ordinal number will be changed.

   SEE ALSO:
             first()

             last()

             item()

             changepos()
```

1.28 Amiga Foundation Classes: NodeMaster/sort()

```
                    NAME: sort(sortingroutine, info=NIL:PTR TO LONG)

   SYNOPSIS:   APTR sort(LONG (*comp)(APTR, APTR, APTR APTR info = NULL)

DESCRIPTION: Use this method to sort the list.

      INPUT: sortingroutine  - You MUST provide a comparison routine,
                               which will be used to sort your list.
                               The comp routine should accept three
                               params: item1, item2 and info.
                               Item1 and item2 are the two items you
                               should compare; while info is an
                               optional param containing to whatever
                               you want.
                               Your comp routine MUST return a value:

                                    >1 - Item1 > Item2
                                    =0 - Item1 = Item2
                                    <0 - Item1 < Item2

                               Then items will be sorted accordingly
                               by the sort() method.

             info            - (DEFAULT NIL) This is an optional param
```

```
                                that will be passed to your comp routine.
                                It can contain everything you like.

       RESULTS: PTR TO first obj data - list has been sorted.

                FALSE                  - sort() failed (maybe list empty)

          NOTE: After a sort():

                  + Stack will be cleared.
                  + Current item will be the first one.

      SEE ALSO:
                push()

                pop()

                clearstack()

                first()
```

1.29 Amiga Foundation Classes: NodeMaster/version()

```
           NAME: version()

       SYNOPSIS: ULONG version(BOOL rev = FALSE)

    DESCRIPTION: Returns class version and revision.

          INPUT: NONE

        RESULTS: This method returns TWO values: class version and revision.

  PORTING NOTES: Because in C++ a function cannot return two values, we have
                 arranged the version/revision return values in this way:
                 by calling version() method with rev=FALSE you will be
                 returned the Class VERSION. If you set rev=TRUE, then you'll
                 get the Class REVISION.

      SEE ALSO:
```

1.30 Amiga Foundation Classes: NodeMaster/islast()

```
                           NAME: islast()

       SYNOPSIS: BOOL islast(void)

    DESCRIPTION: This method checks if the current item is the last one.

          INPUT: NONE.

        RESULTS: TRUE          - The item is the last.
```

```
              FALSE         - The item is not the last.

      SEE ALSO:
                isfirst()
```

1.31 Amiga Foundation Classes: NodeMaster/isfirst()

```
                          NAME: isfirst()

      SYNOPSIS: BOOL isfirst(void)

   DESCRIPTION: This method checks if the current item is the first one.

         INPUT: NONE.

       RESULTS: TRUE          - The item is the first.
                FALSE         - The item is not the first.

      SEE ALSO:
                islast()
```

1.32 Amiga Foundation Classes: NodeMaster/error()

```
           NAME: error()

      SYNOPSIS: ULONG error(void)

   DESCRIPTION: This method checks if the current item is the first one.

         INPUT: NONE.

       RESULTS: last error code. 0 means "no errors".

      SEE ALSO:
```

1.33 Amiga Foundation Classes: NodeMaster/clone()

```
           NAME: clone()

      SYNOPSIS: Nodemaster * clone(VOID)

   DESCRIPTION: This method clones the current nodemaster and its contents
                and returns a new (cloned) NodeMaster object.

         INPUT: NONE.

       RESULTS: a PTR to a valid NodeMaster class. May be NIL.

          NOTE: - The cloned class resulting from this method should be
                  considered a NodeMaster class in all of it parts.
```

```
            So, you are supposed to END (delete) the class by
            yourself, when you have finished with it.

          - The Resource Tracker used by the "original" class is
            provided to the cloned one during creation.

SEE ALSO:  resourceTracker
```