PopupMenu

COLLABORATORS

	TITLE : PopupMenu	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		December 31, 2022	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

1 PopupMenu **1**

1.1 PopupMenu Library V7.1 . . . 1
1.2 What is this?? . . . 2
1.3 Requirements . . . 3
1.4 Author . . . 3
1.5 DISCLAIMER . . . 3
1.6 Important things to think about when you're using popupmenu.library! . . . 4
1.7 History & Future . . . 5
1.8 " . . . 8
1.9 pm_alterstate . . . 9
1.10 pm_exlsta . . . 9
1.11 pm_filterimsga . . . 9
1.12 pm_finditem . . . 10
1.13 pm_freepopupmenu . . . 11
1.14 pm_getitemattrsa . . . 11
1.15 pm_itemchecked . . . 12
1.16 pm_makeidlista . . . 13
1.17 pm_makeitema . . . 14
1.18 pm_makemenua . . . 16
1.19 pm_openpopupmenua . . . 17
1.20 pm_setitemattrsa . . . 18
1.21 Macros Example . . . 19

Chapter 1

PopupMenu

1.1 PopupMenu Library V7.1

What is this??

Requirements

The author

Disclaimer, Copyright and Distrubition

History and Future of popupmenu.library

Things you need to know. IMPORTANT!

Acknowledgements

Functions in the library

PM_ExLstA()

PM_FilterIMsgA()

PM_FindItem()

PM_FreePopupMenu()

PM_GetItemAttrsA()

PM_ItemChecked()

PM_MakeIDListA()

PM_MakeItemA()

PM_MakeMenuA()

PM_OpenPopupMenuA()

PM_SetItemAttrsA()

Some examples

Run the demo.

Some examples on macros.

1.2 What is this??

Introduction

This is The One and Only Popup Menu Library you use when you want a nice popup menu in your programs!
The included examples are compiled using SAS/C 6.55, they should work with other compilers. If they don’t, let me know.

Features

* User configurable.

* Support for pulldown menus.

* Unlimited number of submenus.

* Unlimited number of mutually excluded items.

* Supports MultiSelect.

* Supports images and icons.

* MagicMenu images by Mario Cattaneo.

* Bold, italic, underlined and shadowed text.

* Coloured text.

1.3 Requirements

PopupMenu.library requires the following:

* AmigaOS 3.0 or higher.

Recommended:

* AmigaOS 3.1

* Picasso96 or CyberGfx

1.4 Author

This is where you send things that can't be digitized:

Henrik Isaksson
Garvarvägen 33
950 40 Töre
SWEDEN

And this is where you send the rest:

hki@hem1.passagen.se

You might reach me here too:
(But I change ISP's often, so don't rely on this one..)

henisak@algonet.se

This is the official popupmenu homepage:

http://www.algonet.se/~henisak/pm/pm.html

1.5 DISCLAIMER

compile any part of it.

* You may use this library for free if your product is free to the user
  (FreeWare, Public Domain or similar).

* If your application is ShareWare, you should send me the full release
  or a keyfile.

DISTRIBUTION
============

This package is freely distributable. That means you are allowed to re-
distribute this package as long as you follow these points:

* You may NOT ADD any files to the archive!
* You may NOT CHANGE any files in this archive!
* You may NOT REMOVE any of the files in this archive!
* You may include the library file, and the preferences editor in your own
  distributions, if you follow the copyright rules stated above.

This package may be freely distributed via BBSs, InterNet/UseNet,
software libraries such as Fred Fish's and Aminet® CD-ROM, and other
similar electronic channels.

Disk magazines and services that charge extra for file transfers may
NOT distribute it without written permission by the developer(s)!

DISCLAIMER
==========

By using this product, you accept the FULL responsibility for any damage
or loss that might occur through its use or the inability to use it. The
developer(s) of the software and the author and the translators of this
"Copyright Note" can NOT be held responsible.

Some names used in this text are trademarks or registered trademarks.
The use of these names does not imply that they are free.

1.6 Important things to think about when you're using popupmenu.library!

Most things are handled automatically, but there are a few things that need
to be kept in mind.

* If you want to let the user select when the menu should appear,
  use the PM_Code tag. The argument to that tag should be one of the
  following:

  SELECTUP
  SELECTDOWN
  MENUUP
  MENUDOWN

  This can be done by just copying the IntuiMessage->Code field, IF you
  are responing to a IDCMP_MOUSEBUTTONS, and only then.

```
   If you are using IDCMP_MENUVERIFY to open the menu, you should send
   a MENUDOWN.

 * If you want to use keyboard shortcuts in your application, you have to
   use the PM_FilterIMsg() function.

 * If you want the menus to be able to handle multiple selections, you have
   to use the PM_MenuHandler tag.

 * If you want to follow Mario 'padrino' Cattaneo's MagicMenu style
   conventions, here are his rules:

   All contextual menus should have a title describing it's context, such
   as "Trashcan" for a trashcan icon in a WB replacement software.

   The title should be printed using shadowed, centered text, and it should
   be white. (use the tags PM_Centered, PM_Shadowed and PM_ShinePen)
   Under the title
   The simplest way to do this is to use the PMMenuTitle macro.
```

1.7 History & Future

```
                The history of popupmenu.library:

7.1 Bugfix. Sometimes the menu could show up on the wrong place.

7.0 The PM_FilterIMsg() function has been moved to a new library offset.
  This was done because the argument passing used the wrong registers.
  To provide backwards compatibillity, the old function is still left.

  The Image system is entirely rewritten. All used images are now
  stored in the configuration file. This saves some memory and will,
  in future releases, let the user load his/her own images.
  Image drawing should also be much faster, especially on chunky
  screens and on machines with chunky-to-planar converter.
  There may be some problems with prefsfiles from release 6.0.

  Fixed a bug in the separator bars. (The MM2 style was used even
  without MagicMenu2 borders selected.)

  Added the
                PM_ColourBox
                 tag.

6.0 Made some changes to the API, for keyboard shortcuts and
  multiselect support. The library should still be 99% backwards
  compatible. Hacks like the previous version of the BigMenu demo
  will crash.

  Added keyboard shortcut support through
                PM_FilterIMsg()
                .
```

```
                PM_ExLstA()
                 added for easier listing of mutually exclusive item ID's.
```

Added support for multiselect.

Added menu border settings.

Fixed the "standard 2d images".

Fixed a problem with pulldown menus when "compact" was not selected.

Moved some of the images out of the library, and in to the prefs editor. The needed images will then be stored in the prefsfile. (wich means that only the needed images will be in memory)

5.4 Moved the submenus 5 pixels up.

Improved the deadlock detection.

Improved the look of separator bars.

5.35 Optimized the pen allocation a bit.

5.3 Moved the submenus a bit to the left. Now it looks more like the other menus, and it's easier to open the submenus.

Fixed a bug in pen allocation/freeing.

Made a few adjustments to the separator bars.

Fixed bug in "Submenu Delay" code.

Fixed an enforcer hit in window mode.

Got rid of a few unnecessary bytes again. (about 1k smaller)

5.2 Fixed delay bug in non-window mode.

Improved the remapping of magic images.
Right look and a few hundred percent faster. :)

Pulldown menus should now work in non-window mode too.

Fixed the shadows. They were 1 pixel to wide before.

Added a new tag, PM_Shadowed, to make the text shadowed.

Changed the PMMenu macro so the titles will use shadowed text.

Fixed refresh problem in window-mode.

Made the separator bars look more like MM2.

And this time the library is actually 1272 bytes smaller!

5.1 Fixed the deadlock bug in non-window mode.

Added MagicMenu2 Images, and it's special remapping.
(relative color values)

Added support for pulldown menus.
(works only in window-mode at the moment)

5.0 Opening the menu does no longer deactivate the active window.

The menus can now use the blitter instead of windows, wich is much faster. (this has to be activated in the prefs-editor.)

The menu font is now fetched from the DrawInfo, and not the menu's parent window.

There is now a space between the bottom of the text and the bottom of the select bar.

Clicking the mousebutton very fast should no longer result in a menu that doesn't dissapear. (as it should, when you release the button)

Some of the tags send to OpenPopupMenu() has become obsolete.
(The preferences has taken over)

The input method has changed, so the library now requires commodities.library (wich is a disk library).

The menu shadows now look like MagicMenu2 shadows.
(the size increases for each submenu opened)

Starting with this version, kickstart 3.0 is required. At least for the non-window mode. In window mode it _may_ still work with 2.0, but it's not tested. So if you have ks 2.0, I would appreciate if you could try it out.

Fixed a bug. Submenus could appear at the bottom of the screen when the mouse was moved quickly over an item.

Added a new function,
PM_AlterState()
.

4.3 Fixed a bug. (read from adress 0)

4.2 Now the menus can have shadows.

4.1 Images are now remapped correctly.

Small changes for colour and image prefs.

4.0 More bugs fixed.

A new demo, MenuVerify, shows how to use popupmenu.library with IDCMP_MENUVERIFY.

3.6 Bug fix. (never released, i got more bug reports...)

```
3.5 Now checks for the file ENV:PopupMenu.cfg, and if it exists,
  loads it, and replaces the default settings.

  Bug in PM_SubMenuTimer fixed.

  PM_Code added.
               Read
                this!

3.0 Added PM_GetItemAttrs(), PM_SetItemAttrs(), PM_IsChecked
  and PM_FindItem.

  Added the tags PM_Left and PM_Top.

  Added a few more demos.

  Changed the naming of functions and macros to avoid interfering
  with other libraries.

2.0 Entirely rewritten and is now a shared library!

1.3 Added submenu support

1.2 Lots of new flags, and checkable menuitems

1.1 OpenPopupMenuPos()

1.0 First release

The future:

1. Datatype loading of images.

2. Font settings. (maybe)

3. Mail me if there's anything else!
```

1.8 "


```
 Special thanks goes to:

 Thanks to Stefan Sommerfeld for his very good bugreports, help with the
 SwapBitsClipRectRastPort() bit, and the remapping code (too bad I coudn't
 use it ;( ).

 Thanks to Trond Werner Hansen for his shadow-drawing code!

 Thanks to Mario 'padrino' Cattaneo for all the MagicMenu2 images and the
 other MM2 specs. BTW, if you don't want to upset him, remember to  spell
 padrino with a small p... ;)

 Thanks to Olaf 'Olsen' Barthel for his useful hint about deadlock detection.
```

```
Thanks to all the rest of you!
```

1.9 pm_alterstate

```
 NAME
PM_AlterState -- Simulate a (de-)selection of a checked item. (V5)

 To be continued... (sometime, but right now I don't think you need this
 function)
```

1.10 pm_exlsta

```
                 NAME
PM_ExLstA -- Build a list of ID numbers for mutual exclusion. (V6)

 SYNOPSIS
list = PM_ExLstA(id);
D0               A1

list = PM_ExLst(id1, ...);

struct PM_IDLst *PM_ExLstA(ULONG *id);

struct PM_IDLst *PM_ExLst(ULONG id1, ...);

 FUNCTION
This function is used to build lists of ID numbers, that are needed
for the PM_Exclude tag.
The array of ID's is ended by NULL.

 INPUTS
id - array of ID numbers

 RESULT
Returns a list of ID's, or NULL if it ran out of memory.
You don't really need to care about what it returns, just pass it
on to
              PM_MakeItemA()
              .

 SEE ALSO

              PM_MakeIDListA()

              PM_MakeItemA()
```

1.11 pm_filterimsga

```
                  NAME
PM_FilterIMsgA -- Handle keyboard shortcuts. (V6)

 SYNOPSIS
userdata = PM_FilterIMsgA(window, menu, imsg, tags);
D0                        A1      A2    A3    A5

userdata = PM_FilterIMsg(window, menu, imsg, tags, ...);

APTR PM_FilterIMsgA(struct Window *, struct PopupMenu *,
  struct IntuiMessage *, struct TagItem *);

APTR PM_FilterIMsg(struct Window *, struct PopupMenu *,
  struct IntuiMessage *, ULONG, ...);

 FUNCTION
This function handles keyboard shortcuts.
It compares 'imsg' against IDCMP_VANILLAKEY, if one is found, that
item's UserData is returned, or the MenuHandler hook is called.
Remember to set IDCMP_VANILLAKEY for the window(s) you use for user
input.

 INPUTS
window = pointer to the parent window.
menu   = pointer to a popup menu.
imsg   = IntuiMessage to be filtered.

 TAGS
Accepts the same tags as PM_OpenPopupMenuA, in addition to those
listed below:

PM_AutoPullDown - (BOOL) Set this to TRUE if you want a pulldown
      menu opened automatically, when the user presses
      RMB.
      You will have to use WFLG_RMBTRAP and
      IDCMP_MOUSEBUTTONS to get it working.

 RESULT
The UserData of the item that was selected, or NULL.
If MultiSelect is enabled, this result should not be used, since it
would not be reliable when the user selects several items.
(The user can ofcourse only select more than one item if the tag
PM_AutoPullDown is used)

 SEE ALSO

              PM_OpenPopupMenuA()
```

1.12 pm_finditem

```
 NAME
PM_FindItem -- Find an item in a popupmenu list. (V3)
```

```
 SYNOPSIS
item = PM_FindItem(menu, id);
D0                 A1    D1

struct PopupMenu *PM_FindItem(struct PopupMenu *, ULONG);

 FUNCTION
Find the pointer to an item using the ID number.

 INPUTS
menu = pointer to a popup menu list.
id   = ID number (PM_ID).

 RESULT
Returns a pointer to the found item, or NULL if unsuccessful.

 SEE ALSO
```

1.13 pm_freepopupmenu

```
                NAME
     PM_FreePopupMenu -- Free a menu list created by
             PM_MakeMenuA()
             .

 SYNOPSIS
     PM_FreePopupMenu(popupmenu);
                      a1

     void PM_FreePopupMenu(struct PopupMenu *);

 FUNCTION
     This function is used to free the list of menu items created by

             PM_MakeItemA()
             , and
             PM_MakeMenuA()
             .

 INPUTS
     popupmenu - pointer to a popup menu to free.

 SEE ALSO

             PM_MakeItemA()

             PM_MakeMenuA()
```

1.14 pm_getitemattrsa

```
                NAME
PM_GetItemAttrsA -- Get attribute values for an object. (V3)
```

```
 SYNOPSIS
result = PM_GetItemAttrsA(item, tags);
D0                        A2    A1

ULONG PM_GetItemAttrsA(struct PopupMenu *, struct TagItem *);

result = PM_GetItemAttrs(item, tag1, ...);

ULONG PM_GetItemAttrs(struct PopupMenu *, ULONG, ...);

 FUNCTION
Used to get attributes from an item.
item can be directly taken from
              PM_FindItem()
               as the input is
checked against NULL pointers.

 EXAMPLE

struct PopupMenu *menu;
struct Image *image;
BOOL checked;

....
/* Initialize the menu */
....

      PM_GetItemAttrsA(PM_FindItem(menu, itemid),
  PM_SelectImage, &image,
  PM_Checked, &checked,
  TAG_DONE);

 INPUTS
item = pointer to a popup menu item.
tags = array of TagItem structures with attribute/value pairs.

 RESULT
Returns the number of successfully copied attributes.

 SEE ALSO

              PM_FindItem()
```

1.15 pm_itemchecked

```
 NAME
PM_ItemChecked -- Find out if an item is checked. (V3)

 SYNOPSIS
item = PM_ItemChecked(menu, id);
D0                    A1    D1

BOOL PM_ItemChecked(struct PopupMenu *, ULONG);
```

```
 FUNCTION
Fast way to find out if an item is checked using the item ID.

 INPUTS
menu = pointer to a popup menu list.
id   = ID number (PM_ID).

 RESULT
TRUE (-1L) if the item is checked, FALSE (0L) if not checked,
PMERR (-5L) if the ID was not found in the list.

 SEE ALSO
```

1.16 pm_makeidlista

```
                NAME
      PM_MakeIDListA -- Create a list of ID's for exclusion/inclusion.

 SYNOPSIS
      list = PM_MakeIDListA(taglist);
      d0                 a1

      struct PM_IDLst *PM_MakeIDListA(struct TagItem *tags);

list = PM_MakeIDList(tag1, ...);

struct PM_IDLst *PM_MakeIDList(ULONG, ...);

 FUNCTION
      This function is used to create a list of ID's that is used to
      tell wich items an item should include, exclude, reflect or
      inverse reflect.

 INPUTS
      taglist - pointer to a taglist.

 TAGS
      PM_ExcludeID   (ULONG) ID of a item that should be unselected when
                     when this item is selected.

      PM_IncludeID   (ULONG) ID of a item that should be selected when
                     when this item is selected.

      PM_ReflectID   (ULONG) ID of a item that should copy the state
                     of this item, when it gets selected/unselected.

      PM_InverseID   (ULONG) ID of a item that should copy the inverse
                     state of this item, when it gets selected/
                     unselected.
                     Useful if you want to make sure only one of two
                     items is selected at a time.

 RETURNS
      Returns a pointer to a list of id's if successful.
```

```
SEE ALSO

             PM_MakeItemA()

             PM_ExLstA()
```

1.17 pm_makeitema

```
                NAME
     PM_MakeItem -- Create a new menu item.

SYNOPSIS
     menu = PM_MakeItemA(taglist);
     d0               a1

     struct PopupMenu *PM_MakeItemA(struct TagItem *tags);

menu = PM_MakeItem(tag1, ...);

struct PopupMenu *PM_MakeItem(ULONG, ...);

FUNCTION
     This function is used to create a new menu item to be passed to

             PM_MakeMenuA()
             , for linking.

INPUTS
     taglist - pointer to a taglist listing your menu items.

TAGS
     PM_Title       (STRPTR) Pointer to the menu text you want.

     PM_UserData    (ULONG) Anything of your choice, can be used to
                    identify the item when it is selected. The value
                    stored here will be returned from
             PM_OpenPopupMenuA()
                                  when the user selects this item.

     PM_ID          (ULONG) An ID number, only needed if you want to
                    be able to read or change the attributes of this
                    item later. (for example, to find out if an item
                    is checked)

     PM_Sub         (struct PopupMenu *) A pointer to a menu list
                    returned from
             PM_MakeMenuA()
             . The item
                    will automatically get an arrow to the right
                    showing that it has a sub menu.

     PM_Flags       (ULONG) Used internally. Do not use this tag!
```

```
        PM_NoSelect     (BOOL) Make the item unselectable.

        PM_FillPen      (BOOL) Draw the item title in FILLPEN.

        PM_Checkit      (BOOL) Leave some space for a checkmark.

        PM_Checked      (BOOL) Put a checkmark to the left of the item.

        PM_Italic       (BOOL) Draw the text in italic.

        PM_Bold         (BOOL) Make the text bold.

        PM_Underlined   (BOOL) Underline the text.

        PM_WideTitleBar (BOOL)
        PM_TitleBar     (BOOL) Draw a horizontal separator instead of the
                        text.

        PM_ShadowPen    (BOOL) Draw the text in SHADOWPEN color.

        PM_ShinePen     (BOOL) Draw the text in SHINEPEN color.

        PM_Exclude      (struct PM_IDLst *) List of items to be selected
                        or unselected when this item gets selected.
                        The list should be created with
                PM_MakeIDListA().
                         PM_Disabled    (BOOL) Makes the item unselectable, and ↩
                            draws a
                        disable pattern over the item.

        PM_ImageSelected (struct Image *)
        PM_ImageUnselected (struct Image *)
                        Specifies an image to be rendered under the item
                        title.

        PM_IconSelected (struct Image *)
        PM_IconUnselected (struct Image *)
                        Specifies an image to be rendered to the left of the
                        item title.

        PM_AutoStore    (BOOL *) A pointer to a BOOL that will reflect the
                        state of the checkmark. The best way to find out
                        if an item is checked or not.

        PM_TextPen      (ULONG) A pen number for the text. You are
                        responsible for allocating/deallocating a pen
                        yourself.

PM_Shadowed     (BOOL) Give the the text a shadow using SHADOWPEN.

PM_CommKey      (STRPTR) Keyboard shortcut for this item.
          Only the first character will be used. This is a
          string pointer just to make it easier to use locale
          strings for the shortcuts.

PM_ColourBox    (ULONG) Draws a filled rectangle in using the
          specified pen.
```

```
          The box will be drawn at the end of the line in
          PM_CheckIt items, and at the beginning in other items.

   RETURNS
        Returns a pointer to an item if successful.

   SEE ALSO

                PM_MakeMenuA()

                PM_MakeIDListA()

                PM_ExLstA()

                PM_OpenPopupMenuA()
```

1.18 pm_makemenua

```
                    NAME
        PM_MakeMenu -- Create a new menu list.

   SYNOPSIS
        menu = PM_MakeMenuA(taglist);
        d0                  a1

        struct PopupMenu *PM_MakeMenuA(struct TagItem *tags);

  menu = PM_MakeMenu(tag1, ...);

  struct PopupMenu *PM_MakeMenu(ULONG, ...);

   FUNCTION
        This function is used to link menu items returned by

                PM_MakeItemA()
                .

   INPUTS
        taglist - pointer to a taglist listing your menu items.

   TAGS
        PM_Item - pointer to a menuitem returned from
                PM_MakeItemA()
                .

   RETURNS
        Returns a pointer to a list of items if successful.

   SEE ALSO

                PM_MakeItemA()
```

1.19 pm_openpopupmenua

```
                NAME
      PM_OpenPopupMenuA -- Open a popup menu.

 SYNOPSIS
      userdata = PM_OpenPopupMenuA(prevwnd, taglist);
      d0                         a1       a2

      ULONG PM_OpenPopupMenuA(struct Window *prevwnd, struct TagItem *tags);

userdata = PM_OpenPopupMenu(prevwnd, tag1, ...);

ULONG PM_OpenPopupMenu(struct Window *, ULONG, ...);

 FUNCTION
      This function is used to open a popup menu based on an item list
      created with
              PM_MakeMenuA()
              .

 INPUTS
      prevwnd - pointer to parent window, used to find out screen, font
                and other drawing attributes.
      taglist - pointer to a taglist of menu options.

 TAGS
      PM_Menu             (struct PopupMenu *) Pointer to a menu list
                          created by
              PM_MakeMenuA()
              .

      PM_RecessSelected   OBSOLETE!
      PM_WideSelectBar    OBSOLETE!
      PM_Compact          OBSOLETE!
      PM_SubMenuTimer     OBSOLETE!
      PM_OldLook          OBSOLETE!
      PM_SameHeight       OBSOLETE!
      PM_CheckMark        OBSOLETE!
      PM_ExcludeMark      OBSOLETE!
      PM_SubMenuMark      OBSOLETE!
      PM_SmartRefresh     OBSOLETE!

      PM_Left             (ULONG) Horizontal position of the menu, relative
              to the menus left edge. (V3)

      PM_Top              (ULONG) Vertical position of the menu, relative
              to the menus top edge. (V3)

PM_Code       (UWORD) The contents of the Code field of the
        IntuiMessage structure. Used to find out if the
        mousebutton was pressed or released, so the user
        can specify in the preferences if she/he want the
        menu to open when the button is pressed or
        released. Must always be specified!
```

```
PM_PullDown     (BOOL) Turn the menu into a pulldown menu. (V5.1)

PM_MenuHandler      (struct Hook *) Menu handler function (hook). (V6.0)


 RETURNS
      Returns the value of UserData of the selected item, if no item
      was selected, NULL is returned.

 SEE ALSO

              PM_MakeMenuA()
```

1.20 pm_setitemattrsa

```
                  NAME
PM_SetItemAttrsA -- Specify attribute values for an object. (V3)

 SYNOPSIS
result = PM_SetItemAttrsA(item, tags);
D0                     A2    A1

ULONG PM_SetItemAttrsA(struct PopupMenu *, struct TagItem *);

result = PM_SetItemAttrs(item, tag1, ...);

ULONG PM_SetItemAttrs(struct PopupMenu *, ULONG, ...);

 FUNCTION
Specifies a set of attribute/value pairs with meaning as
defined in libraries/pm.h.
item can be directly taken from
              PM_FindItem()
               as the input is
checked against NULL pointers.

 EXAMPLE

struct PopupMenu *menu;

....
/* Initialize the menu... */
....

PM_SetItemAttrsA( PM_FindItem( menu, itemid ),
  PM_Checkit, TRUE,
  PM_Checked, TRUE,
  TAG_DONE);

 INPUTS
item = pointer to a popup menu item.
tags = array of TagItem structures with attribute/value pairs.

 RESULT
Returns the number of successfully changed attributes.
```

SEE ALSO

1.21 Macros Example

```
struct PopupMenu *menu;

/*
 Creating a menu with the macros...
*/

menu = PMMenu("The title of the menu"),
 End;

/*
 Creating a menu like the one above, but with items...
*/

menu = PMMenu("The menu title"),
   PMItem("The item text"),
     /* Tags for the item */
   End,
   PMItem("The item text (2)"),
     /* Tags for the item */
   End,
 End;

/*
 Creating a sub-menu for an item...
*/

menu = PMMenu("The menu title"),
   PMItem("Item with submenu"),
     PMSimpleSub,
       /* Here are the items... */
     End,
   End,
 End;

/*
 Creating an exclude ID list for PMExclude can be done in two ways.
 This is the first:
*/

 PM_Exclude,
   PMExl ExID(1),
     ExID(2),
 End,

/*
 This is the second way:
*/
```

```
  PM_Exclude, PM_ExLst(1, 2, 3, 0), End,
```