

InstallingPV.hyper

COLLABORATORS

	<i>TITLE :</i> InstallingPV.hyper		
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>	<i>SIGNATURE</i>
WRITTEN BY		August 26, 2022	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

1	InstallingPV.hyper	1
1.1	Installing PowerVisor (Tue Jul 14 16:48:08 1992)	1
1.2	Installing PowerVisor : Commands used in this tutorial	2
1.3	Installing PowerVisor : Functions used in this tutorial	2
1.4	Installing PowerVisor : Introduction	2
1.5	Installing PowerVisor : The 'mode' command	2
1.6	Installing PowerVisor : The 'prefs' command	4
1.7	Installing PowerVisor : The maximum number of lines in the history	4
1.8	Installing PowerVisor : Some standard keys	5
1.9	Installing PowerVisor : Setting the startup screen size	6
1.10	Installing PowerVisor : Setting the stack fail level	6
1.11	Installing PowerVisor : Setting default logical window parameters	6
1.12	Installing PowerVisor : The default commandline length	9
1.13	Installing PowerVisor : Some debug preferences	9
1.14	Installing PowerVisor : Debug mode	10
1.15	Installing PowerVisor : Installing pens	10
1.16	Installing PowerVisor : Setting the default font	13
1.17	Installing PowerVisor : The config file	13
1.18	Installing PowerVisor : Other installation factors	13
1.19	Installing PowerVisor : Setting the colours	14
1.20	Installing PowerVisor : Setting alias commands	14
1.21	Installing PowerVisor : Attaching commands to keys	15
1.22	Installing PowerVisor : Files	15
1.23	Installing PowerVisor : Logical windows	17
1.24	Installing PowerVisor : For the wizards	17

Chapter 1

InstallingPV.hyper

1.1 Installing PowerVisor (Tue Jul 14 16:48:08 1992)

Contents:

Introduction

The 'mode' command

The 'prefs' command

The maximum number of lines in the history

Some standard keys

Setting the startup screen size

Setting the stack fail level

Setting default logical window parameters

The default commandline length

Some debug preferences

Debug mode

Installing pens

Setting the default font

The config file

Other installation factors

Setting the colours

Setting alias commands

Attaching commands to keys

Files

Logical windows

For the wizards

Various:

Commands used in this tutorial

Functions used in this tutorial

[Back to main contents](#)

1.2 Installing PowerVisor : Commands used in this tutorial

mode	Set PowerVisor preferences
prefs	Set preferences
screen	Set PowerVisor on another screen

1.3 Installing PowerVisor : Functions used in this tutorial

None

1.4 Installing PowerVisor : Introduction

At this moment (after you have read `GettingStarted` , `Expressions` and `Screens` and `Windows`) you are probably ready to customize PowerVisor. This is a complex business and you will probably want to change some parameters later when you really start to use PowerVisor.

1.5 Installing PowerVisor : The 'mode' command

With the `mode` command you can set some parameters. The 'mode' command expects any number of keywords. These keywords need not be in any particular order.

The following keywords are supported :

pal	set monitor to pal (only AmigaDOS 2.0) the non-interlaced resolution is 640x256 and the interlaced resolution is 640x512. This resolution will be bigger if you use overscan.
ntsc	set monitor to ntsc (only AmigaDOS 2.0) non-interlaced : 640x200 interlaced : 640x400
vga	set monitor to vga (only AmigaDOS 2.0 and new denise) non-interlaced : 640x480 interlaced : 640x960
viking	set monitor to a2024 (only AmigaDOS 2.0)

```
resolution : 1024x1008

lace          use interlace
nolace       no interlace (default)

fancy        use two bitplanes for the PowerVisor screen (default)
nofancy      use only one bitplane

sbottom      include the size gadget in the bottom border. This means
             that you loose a line but you gain some columns (default)
nosbottom    include the size gadget in the right border of the window.
             You loose some columns but you gain a line or so

space        add a space after a snapped word (default)
nospace      don't add a space. Simply snap the word as it is

lonespc      snap a space if you click on an empty place in a logical
             window
nolonespc    don't snap a space (default)

shex         show hex words when disassembling instructions. The
             disadvantage is that these words could overwrite the right
             side of the symbolname if present (default)
noshex       don't show hex words. Symbols will be completely visible

dec          display all printed integers as decimal only
hex          display all printed integers as hexadecimal only
hexdec       first display hex, than display decimal (default)

more         enable -MORE- check for the 'Main' logical window (default)
nomore       disable -MORE- check

auto         perform an automatic list whenever the current list changes
noauto       don't do this (default)

byte         list memory as bytes (for the memory command)
word         list memory as words
long         list memory as longs (default)
ascii        list memory as ascii

fb           feedback each command as it is typed in by the user on
             the current logical window (default)
nofb         don't do this

patch        patch the Exec AddTask function. When this function is
             patched by PowerVisor crashtrapping will work better
             for all new tasks created after the patch is applied.
             This is recommended if you use resident breakpoints
             (see Debugging ). Note that it is not safe to
             run other debuggers (like CodeProbe or MonAm) when
             the patch is applied. They will probably crash when
             you try to trace with them. There will (probably) be
             no problems if you start the other debugger and load
             the debug program with this debugger BEFORE you
             apply the patch (before you start PowerVisor or before
             you type 'mode patch'). (default)
nopatch      Don't patch the Exec AddTask function. When the AddTask
```

function is not patched, PowerVisor will trap a crash a bit later (too late if you plan to use resident breakpoints). With the patch applied PowerVisor traps crashes just on the spot while if the patch is not applied PowerVisor will only trap the crash just before the guru would normally arrive (you will even have to press 'cancel' on the 'task-held' requester before PowerVisor notices the crash). But 'mode nopatch' is the only safe way to run other debuggers concurrently with PowerVisor.

```
intui      if this option is set, PowerVisor will also open a
           physical window (or Intuition window) everytime you
           open one of the standard logical windows ('Extra',
           'Debug', ...) with the standard commands ( xwin ,
           dwin , ...) (not with openlw ). This is useful if
           you prefer to work with Intuition windows instead of
           PowerVisor logical windows. The logical window
           is of course opened in this physical window (with the
           same name).
nointui    Simply open each standard logical window in the 'Main'
           physical window (default)
```

Example, to set everything other than default you can use :

```
< mode lace fancy nospace lonespc noshex dec nomore auto byte nofb nopatch
<      intui <enter>
```

All the parameters set with the 'mode' command will be saved with the saveconfig command (see later).

1.6 Installing PowerVisor : The 'prefs' command

Using the prefs command you can install some additional parameters. All parameters installable with the 'prefs' command will be saved when you use saveconfig . The following sections deal with all parameters you can set with 'prefs'.

1.7 Installing PowerVisor : The maximum number of lines in the history

Using 'prefs history' (see prefs) you can get or set the maximum number of lines in the history buffer. The default is 20.

```
< prefs history <enter>
> 00000014 , 20
```

You can set another value with :

```
< prefs history 500 <enter>
```

Note that the history buffer is dynamic. If there are 10 lines in the

history buffer you only loose memory for 10 lines even if the maximum number of lines is 500.

The command also clears the history buffer.

1.8 Installing PowerVisor : Some standard keys

Using 'prefs key' (see prefs) you can get or set keycodes for standard keys. The following keys are supported :

nr	name	default key	code	qualifier
0	interrupt key	<esc>	045	0
1	hot key	<r-shift>+<r-alt>+'/'	03A	022
2	pause key	<r-alt>+<help>	05F	020
3	cycle active logwin	<tab>	042	0
4	history up	<arrow up>	04C	0
5	history down	<arrow down>	04D	0

You can look at the current values with :

```
< prefs key 2 <enter>
> 005F 0020
```

Or you can set it to other values with :

```
< prefs key 0 16 0 <enter>
```

This command will set the interrupt key to the 'q' key (or the 'a' key if you have an AZERTY keyboard). This is only an example. I recommend that you choose better keys :-)

Here are some codes that are often used :

<esc>	045
<tab>	042
<F1>	050
.	.
.	.
.	.
<F10>	059
<backspace>	041
<help>	05F
	046
<up>	04C
<down>	04D
<left>	04F
<right>	04E
<return>	044
<enter> (numeric keypad)	043

Here are the qualifers. You must add two qualifiers if you want to use a combination :

Left shift	0001
------------	------

Right shift	0002	
Capslock	0004	(this one is ignored by PowerVisor)
Control	0008	
Left alt	0010	
Right alt	0020	
Left Amiga	0040	
Right Amiga	0080	

1.9 Installing PowerVisor : Setting the startup screen size

With 'prefs screen' (see `prefs`) you can define the width and height of the PowerVisor screen. If you have AmigaDOS 2.0 you can use any size for the screen. If this size is big you can scroll by moving the mouse out of the visible part of the screen.

The default is (65535,65535) (or (-1,-1) as words).Which means that PowerVisor will take the default width and height for the screen.

```
< prefs screen <enter>
> -1 -1
```

For example, to make the screenheight equal to 1024 use the following command :

```
< prefs screen -1 1024 <enter>
```

You need to quit PowerVisor and start it again to see the result of this command.

1.10 Installing PowerVisor : Setting the stack fail level

With 'prefs stack' (see `prefs`) you can set or get the amount of stackspace left before PowerVisor will halt the task and give a warning (only if account is on with the `account` command). The default value is 40.

```
< prefs stack <enter>
> 00000028 , 40
```

Change it with :

```
< prefs stack 80 <enter>
```

Be careful not to make it too big, or else all tasks will be stopped even if nothing is wrong. This will almost certainly crash your Amiga.

1.11 Installing PowerVisor : Setting default logical window parameters

Using 'prefs logwin' (see prefs) you can set or get default parameters for the standard logical windows ('Main', 'Extra', 'Refresh', 'Debug', 'Rexx', 'PPrint' and 'Source').

Four values are remembered for each standard logical window :

- the number of columns
 - if 0 we scale the number of columns to the visible size at the moment the logical window is created. After that the number of columns is fixed
 - if -1 we use horizontal autoscale. The number of columns is adapted to the visible size everytime the horizontal visible size changes. The disadvantage of this is that the logical window is cleared
 - every other value sets a fixed number of columns
- the number of rows
 - (like the number of columns)
- the flag mask
- the flag values in this mask

The following flags are supported (also see Screens and Windows) :
(bit 0/bit 1)

-MORE- disable/enable	4
top-visible/real-top	32
statusline on/off	64
interrupt on/off	128
auto output snap	256

All other bits are used or reserved and should not be used in the mask.

For example. To set the -MORE- check off, the statusline off and leave all other flags as default you can use :

```
mask = 4+64
value = 64
```

Or with -MORE- check on :

```
mask = 4+64
value = 4+64
```

The following default values are used when a logical window is opened :

```
-MORE- disabled
Interrupt/Pause enabled
Home position is top-visible
Status line on
Auto Output Snap off
```

The standard logical windows have the following defaults (Note that the -MORE- setting of the 'Main' logical window is dependent on the mode command) :

- Main (0,0,160,0)
number of columns and rows is set to a fixed value. This value is the maximum number of columns and rows at the time the logical window is created.
- Interrupt/Pause enabled
- Home position is top-visible

- Auto Output Snap is on
- Extra (0,0,160,0)
number of columns and rows is set to a fixed value. This value is the maximum number of columns and rows at the time the logical window is created.

Interrupt/Pause enabled
Home position is top-visible
Auto Output Snap is on
 - Refresh (0,50,160,160)
the number of columns is set to a fixed value. This value is the maximum number of columns at the time the logical window is created. The number of rows is fixed and always set to 50.

Interrupt/Pause disabled
Home position is real-top
Auto Output Snap is off
 - Debug (90,42,160,160)
the number of columns is fixed and set to 82. The number of rows is fixed and set to 42.

Interrupt/Pause disabled
Home position is real-top
Auto Output Snap is off
 - REXX (0,50,160,128)
the number of columns is set to a fixed value. This value is the maximum number of columns at the time the logical window is created. The number of rows is fixed and always set to 50.

Interrupt/Pause disabled
Home position is top-visible
Auto Output Snap is off
 - PPrint (0,50,160,128)
the number of columns is set to a fixed value. This value is the maximum number of columns at the time the logical window is created. The number of rows is fixed and always set to 50.

Interrupt/Pause disabled
Home position is top-visible
Auto Output Snap is off
 - Source (-1,-1,160,160)
the number of columns and rows are autoscalable.

Interrupt/Pause disabled
Home position is real-top
Auto Output Snap is off
-

To get the defaults :

```
< prefs logwin debug <enter>
> 90 42 00A0 00A0
```

To set other defaults :

```
< prefs logwin debug 100 50 160 160 <enter>
```

You need to quit PowerVisor and start it again to see the result of this command when applied to the 'Main' logical window. For all other logical windows you can simple close/open them to see the changes. You can also use the `setflags` command to set the flags for a logical window immediatelly (also for 'Main'), and the `colrow` command to set the number of columns and rows immediatelly.

1.12 Installing PowerVisor : The default commandline length

With 'prefs linelen' (see `prefs`) you can set or get the maximum length of the stringgadget commandline. This is also the maximum number of characters that may be used in scripts (not ARexx scripts). The default is 400. Note that PowerVisor uses 800 bytes with a linelength of 400 (always double).

To get the current length :

```
< prefs linelen <enter>
> 00000190 , 400
```

To set another value :

```
< prefs linelen 1000 <enter>
```

You need to quit PowerVisor and start it again to see the result of this command.

1.13 Installing PowerVisor : Some debug preferences

With 'prefs debug' (see `prefs`) you can set or get some preferences for the debugger. The first argument is the number of instructions to disassemble after each trace and the second argument is 0 or 1 if you want the disassemble the previous instruction (no or yes resp.) after each trace. The values set with this command are also used by the fullscreen debugger. Default is 5,1. This means 5 instructions and yes, disassemble the previous executed instruction too.

```
< prefs debug <enter>
> 00050001 , 327681
```

You can set other values using :

```
< prefs debug 15 0 <enter>
```

(15 instructions and no previous instruction)

1.14 Installing PowerVisor : Debug mode

Using 'prefs dmode' (see prefs) you can set the information that should be displayed after a trace. This parameter is not used by the fullscreen debugger since the fullscreen debugger always shows all information.

You can set the following parameters :

```

n          show no info at all. This is useful when you are using the
           fullscreen debugger and you do not want to be disturbed
           by output on the current logical window
r          show only registers
c          show only code (using the format describe in the previous
           section)
f          show registers and code (default)

```

For example :

```
< prefs dmode n <enter>
```

1.15 Installing PowerVisor : Installing pens

Every color you see on the PowerVisor screen or window can be changed. You can do this with the color command. This command changes the RGB values for a color. The disadvantage is of course that this change affects all uses of that color. But don't worry, PowerVisor gives you the option to change virtually every color. This is because PowerVisor uses a pen array (the idea is stolen from AmigaDOS 2.0) to access each color. Using 'prefs pens' (see prefs) you can view or change this pen array.

There are 48 pens. The first 24 are for fancy screens (2 or more bitplane screens) and the last 24 are for 1 bitplane screens. Only the first 21 pens are used for both the fancy and no-fancy screens. The other pens are reserved for future use.

With no arguments this command lists all pens on two lines: the first line containing all pens for fancy screens and the second line for no-fancy screens.

Here follow the currently defined pen numbers and there default color values. Add 24 to the pen number to get the no-fancy screen pen number :

nr	name	default fancy	default no-fancy
0	BoxBackground	0	0
1	LogWinBackground	0	0
2	NormalText	1	1

3	PromptText	1	1
4	StatusTextInactive	1	1
5	StatusTextActive	2	0
6	InActiveBar	0	0
7	ActiveBar	3	1
8	TopLeft3D	2	1
9	BottomRight3D	1	1
10	BoxLine	1	1
11	PositionBox	0	0
12	TopLeftBox	1	1
13	BottomRightBox	2	1
14	PositionIndicator	3	1
15	SGInactiveText	3	1
16	SGInactiveBackground	0	0
17	SGActiveText	1	1
18	SGActiveBackground	0	0
19	HilightPen	2	0
20	HilightBackPen	0	1

(To see the effect of pen changes you should open/close your windows first. For most pens the effect is immediately visible and for some pens (like the SG... pens) the change is only visible after exiting PowerVisor)

The 'BoxBackground' pen is used to draw the background of a box (see the 'Screen' tutor file for more information about boxes). This pen color is usually the same as the 'LogWinBackground' pen explained below. If you insist on using another value you will see a small border in each of your logical windows. This small border will grow if you make the number of columns and lines in the logical window smaller than the visible size

The 'LogWinBackground' pen is the background pen used in a logical window. This pen is only used on places where characters CAN appear. This explains the small border that you see when you change the 'BoxBackground' pen. The small border is the region of the logical windows where no characters can appear

The 'NormalText' pen is used to print all text in logical windows. This pen should be different from 'LogWinBackground', otherwise you will see nothing

The 'PromptText' pen is used to print the prompt (the current list indicator). This pen should be different from 'BoxBackground', otherwise the prompt will be invisible

The 'StatusTextInactive' pen is used to print the name of the logical window in the logical window bar when the logical window is inactive (the active logical window is the window where you can scroll)

The 'StatusTextActive' pen is used to print the name of the logical window in the logical window bar when the logical window is active

The 'InActiveBar' pen is used as background for the logical window bar when the logical window is inactive. This pen should be different from 'StatusTextInactive'

The 'ActiveBar' pen is used as background for the logical window bar

when the logical window is active. This pen should be different from 'StatusTextActive'

The 'TopLeft3D' and 'BottomRight3D' pens are used for all 3D borders (at this moment only the border round the logical window bar uses these pens). The 'TopLeft3D' pen should be a light color and 'BottomRight3D' should be dark. This is to ensure the raised 3D effect of the border

The 'BoxLine' pen is used to draw the line between horizontally arranged logical windows. It should be different from 'BoxBackground'

The 'PositionBox' pen is used to draw the background of the position indicator at the right of the logical window bar

The 'TopLeftBox' and 'BottomRightBox' pens are used for the recessed 3D border round the position indicator. Note that this border is dithered when PowerVisor is in no-fancy mode

The 'PositionIndicator' pen is used for the little knob in the position indicator. This pen should be different from the 'PositionBox' pen

'SGInactiveText' is used for the color of the text when the stringgadget is inactive (only AmigaDOS 2.0)

'SGInactiveBackground' is used for the color of the stringgadget background when it is inactive (only AmigaDOS 2.0)

'SGActiveText' is used for the color of the text when the stringgadget is active (only AmigaDOS 2.0)

'SGActiveBackground' is used for the color of the stringgadget background when it is active (only AmigaDOS 2.0)

'HilightPen' is used for the color of the hilighted text (used by the fullscreen debugger for example)

'HilightBackPen' is the background color for hilighted text. With this you can achieve the effect of inverse video if you want

Example :

```
< prefs pens <enter>
> 0 0 1 1 1 2 0 3 2 1 1 0 1 2 3 3 0 1 0 2 0 0 0 0
> 0 0 1 1 1 0 0 1 1 1 1 0 1 1 1 1 0 1 0 0 1 0 0 0
```

Change the color of normal text to blue (or the color installed on color 3) :

```
< prefs pens 2 3 <enter>
```

1.16 Installing PowerVisor : Setting the default font

With the `setfont` command it is possible to set the font for one logical window. But this command only sets the font for the text in the logical window. The font is not set for the size bar. With the `'prefs font'` command (see `prefs`)

you can set the default font used by everything in PowerVisor. This default font is used by all logical windows (except when you `'setfont'` them with another font), the stringgadget, the screen titlebar and the size bars between all logical windows.

Example :

```
< prefs font <enter>
> topaz.font : 8 0 0
```

This is the default (`topaz.font`, size 8, style 0 and flags 0)

If you want to use `topaz.font 9` for example you can use :

```
< prefs font topaz.font 9 0 0 <enter>
< saveconfig <enter>
< quit <enter>
```

```
CLI< run pv <enter>
```

And the default font for everything will be `topaz 9`.

1.17 Installing PowerVisor : The config file

All parameters set with the `mode` or `prefs` commands can be saved using the `saveconfig` command. This command will create a file called `s:PowerVisor-config`. If this file is not present when PowerVisor starts, the default values are used.

Example :

```
< saveconfig <enter>
```

1.18 Installing PowerVisor : Other installation factors

There are still other things that you can install. But these things are not saved in the `s:PowerVisor-config` file. If you want to set them automatically you'll have to include the right commands in the `s:PowerVisor-startup` file. See the `Scripts` chapter for more information about scripts in general.

1.19 Installing PowerVisor : Setting the colours

With the `color` command you can set the default colours for the PowerVisor screen.

Example, to get the 3D colours used by Workbench 2.0 when you have opened PowerVisor on a 2 bitplane screen you can use :

```
< color 0 10 10 10 <enter>
< color 1 0 0 0 <enter>
< color 2 15 15 15 <enter>
< color 3 6 7 9 <enter>
```

If you use these values the 3D design (the statusline) will be very nice.

You can't use the 'color' command when PowerVisor is on another screen.

1.20 Installing PowerVisor : Setting alias commands

Using the `alias` command you can define other commands.

Note that there are already a lot of predefined aliases in the `s:PowerVisor-startup` file. See the [Alias Reference](#) chapter for more information about these aliases.

Some extra examples :

```
< alias lfd 'loadfd [] fd:[]_lib.fd' <enter>
```

You have defined a command 'lfd'. When you type something like :

```
< lfd exec <enter>
```

The command will be expanded and the following command will be executed :

```
< loadfd exec fd:exec_lib.fd <enter>
> ...
```

This command loads an fd-file in memory.

```
< alias opwin '{openpw [] 0 0 300 200;openlw [] [] 80 40}' <enter>
```

Defines an alias 'opwin' that opens a physical window and a logical window in this physical window.

If you use the group operator in the alias string, alias expansion will be done again. So you can use aliases in an alias allowing for recursive alias definitions. The following two commands define a new command 'fact' to compute the facultaty of its argument :

```
< alias _fact 'void if(([])==1,1,{_fact ([]) -1}*([])'' <enter>
```

```
< alias fact 'disp {_fact []}' <enter>
```

```
< fact 5 <enter>
> 00000078 , 120
```

'_fact' is the recursive alias. 'fact' is only provided to give a more command like syntax. Note that this recursion is limited by both the available stack (PowerVisor will give an error when the end of the stack is getting close) and the maximum length of the commandline (you can increase this maximum length with the `prefs` command).

If you want to use `[]` in the string you can use quotes :

```
< alias test 'print \[\]' <enter>
```

The first string expansion leads to the definition of 'test' as the command 'print \[\]'. The expansion done by 'print' leads to the correct execution.

1.21 Installing PowerVisor : Attaching commands to keys

Using the `attach` command you can attach a command to a key (such a key with a command attached to it is called a macro)

Example :

To add the `list` command to the `<F1>` key you can use :

```
< attach 'list' 050 0 <enter>
```

If you press `<F1>` 'list' is put on the stringgadget and an enter is simulated (this means that 'list' is also put in the history buffer).

If you do not want the stringgadget to be disturbed you can use :

```
< attach 'list' 050 0 e <enter>
```

With `remattach` you can remove macros.

1.22 Installing PowerVisor : Files

If you like you can change the online help file. Read the 'QuickHelp' document for more information about the quickhelp file format.

Note that if you start a line in the online help file with a '`\ensuremath{\lnot}`' \leftrightarrow it will not be printed on screen (with the `help` command).

When you have changed the online help file you must update PowerVisor-ctrl with the following command :

```
CLI< makehelp pv:c/PowerVisor-help pv:c/PowerVisor-ctrl word 2 <enter>
```

(Note that the online help files are probably in the S: subdirectory if you have AmigaDOS 1.3 or older because the PROGDIR: feature is only supported starting with AmigaDOS 2.0. With this feature a program can see where the program executable is located. PowerVisor uses PROGDIR: to locate the online help files, the error file and the menu file)

You can also change the error file PowerVisor-errors. Each line in this file must always be 70 bytes long (including the newline). The first line in the file corresponds with error -6.

If you use the AmigaDOS 2.0 version of PowerVisor you can also use menus. All menus PowerVisor uses are in the PowerVisor-menus file. Each line in this file has the following format (spaces are not important) :

```
<command> [<title string> [<command string> [<shortcut>] ] ]
```

<command> is one of

```
'title' for a menu titlebar
'item' for a menu item
'sub' for a sub item
'ibar' for a horizontal bar in an item menu
'sbar' for a horizontal bar in a sub menu
'end' to end the menus (is required)
```

The other strings are strings in the PowerVisor syntax. This means that you can use quotes (or MUST use quotes if there are spaces in the string or other special things not supported in names (see Expressions)) or not (simple names).

The 'title' command only uses the <title string>

The 'item' command uses the <title string> and if the item has no sub menus the <command string> is also used
<shortcut> is optional

The 'sub' item used the <title string> and the <command string>
<shortcut> is optional

The other commands have no <title string>, <command string> or <shortcut>

Example :

```
> title Project
>   item 'About' 'print \'Hello world\0a\''
>   ibar
>   item Quit quit Q
>
> title misc
>   item 'Testing it' 'disp 1' '1'
>   item 'Testing it 2' '{disp 1;disp 2}'
>   item 'Sub menus'
>       sub 'Sub 1' 'disp 1'
```

```
>      sub 'Sub 2' 'disp 2' '2'
> end
```

1.23 Installing PowerVisor : Logical windows

Although there are already a lot of logical window parameters you can set with the `prefs` command, there can be times that you need more. See the `Screens and Windows` chapter to see how you can open logical and physical windows.

Here is an example :

This sequence of commands opens two extra logical and physical windows and tile them in a certain fashion :

Make the 'Main' physical window a non-backdrop window with `screen` :

```
< screen 0 <enter>
```

Go to two bitplane mode and interlace with `mode` :

```
< mode fancy lace <enter>
```

Open two physical windows and two logical windows with the `openlw` and `openpw` commands :

```
< openpw debug 0 0 500 200 <enter>
```

```
< openpw rexx 500 0 140 200 <enter>
```

```
< openlw debug debug 82 42 <enter>
```

```
< openlw rexx rexx 80 40 <enter>
```

Size the main physical window with `size` and `move` :

```
< size main 640 200 <enter>
```

```
< move main 0 200 <enter>
```

1.24 Installing PowerVisor : For the wizards

And last for the maximum flexibility... the `PVCALL` command!

Note that this command is not for the casual user. It is also not for the experienced PowerVisor user. It is for the PowerVisor wizard :-) !!!

See the `The Wizard Corner` if you think you are a wizard.
