The PCLinuxOS magazine

Openbox Special Edition

March, 2012

Welcome to Another PCLinuxOS Magazine Special Edition!

Just as we did with all the other window manager series we have done, we have compiled all the Openbox articles into a handy, easy to access pdf.

If you don't find an answer to your question here, it may be in our previously released Special Edition about XFCE and LXDE, since LXDE uses Openbox as its window manager.

We hope you find this Special Edition useful.

Meemaw
Assistant Editor

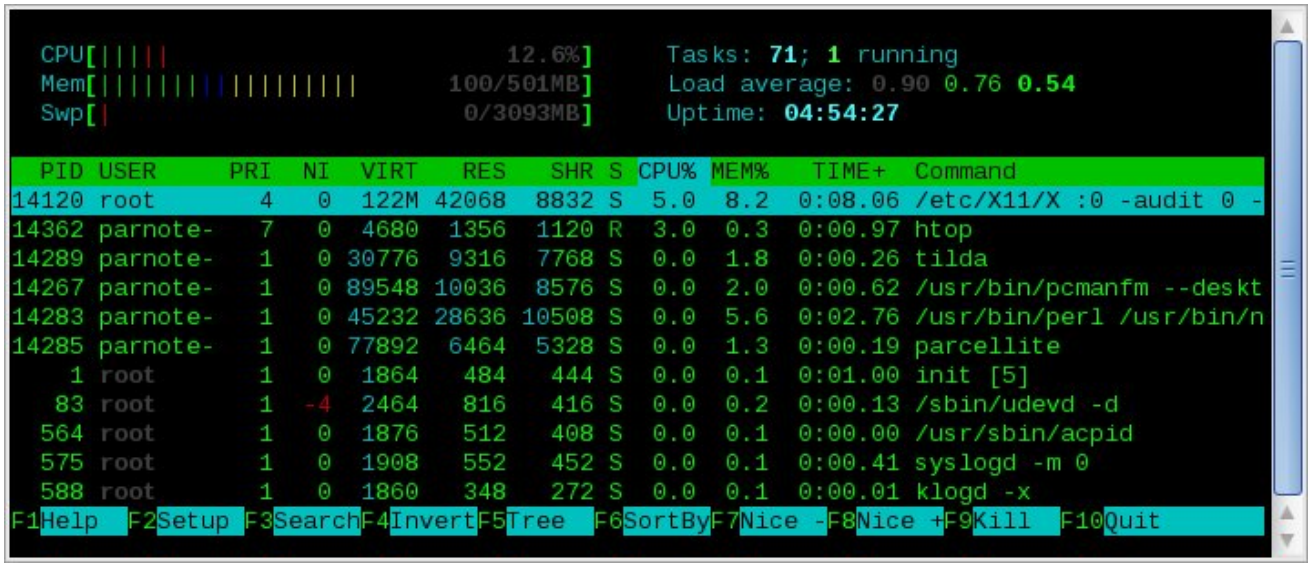
Table of Contents

Openbox: An Overview

by Paul Arnote (parnote)

If you are like most PCLinuxOS users, you have an old computer stuck back in a closet. Like most PCLinuxOS users, you cannot bear the thought of an older computer that has ***any*** life left in it sitting idle, especially if you can find a good use for that computer.

Openbox can resurrect and re-purpose that old computer that's just gathering dust in the back of that closet. With minimal hardware requirements, Openbox can breathe new life into that old computer, and provide a very usable second computer.



History & Background

To get a real feel for the history of Openbox, you have to go back in time to a point before Openbox came to fruition. Openbox was originally derived from the X window manager Blackbox. Blackbox was created in 1997 as a lightweight X window manager, and was written in C++ with entirely original code. Along about Blackbox 0.65, Openbox was spun off, and is written entirely in C.

Openbox, currently at version 3.4, has been devoid of any remaining Blackbox code since version 3.0. The Openbox project is written primarily by Dana Jansens of Carleton University in Ottawa, Ontario, Canada. Openbox also serves as the window manager for the LXDE desktop environment.

Putting It To Use

I have installed Openbox on an IBM Thinkpad T23, running a Pentium III 1.13 GHz processor, with 512 MB RAM, and a S3 SuperSavage IX/C graphics card with 8 MB video RAM. To say that the T23 is fast and responsive running Openbox is an understatement. With the low CPU demands and low memory overhead of Openbox, the T23 acts like a new computer. I can only imagine how fast and responsive Openbox is on a newer computer with a faster, more modern processor and more memory.

Of course, I have fully updated my system, since the latest Live CD is from November, 2010. I am running the 2.38.x kernel, along with X.org 1.9.5. Above is a screen shot that shows my CPU and memory usage. These values are ***after*** all the tweaks I've applied to my Openbox installation. For example, I'm running **tilda** in the background, and using **gnome-power-manager** as my battery notification and power manager. **Tint2**, a lightweight panel replacement, serves as my panel. All information comes from running **htop** at the command line prompt in tilda.

Below (next page) is my tweaked Openbox desktop, displaying a custom wallpaper I created a few years ago. I've applied a custom Openbox window theme,

called Appleish. After my tint2 configuration took a nosedive, I went with the default tint2 configuration – of course, with a few custom, handmade tweaks to the tint2 configuration file.

Summary & Things To Come

The PCLinuxOS Openbox ISO, created by PCLinuxOS community member melodie, is rock solid. It goes a long way to breathing new life into an older computer you may have lying around. Will it make your Pentium III with 512 MB RAM behave like a new computer with a quad-core processor and 4 GB RAM? Certainly not. But then again, you will have another computer running PCLinuxOS, and one that can do service as a media server, or as a computer that does the "basic tasks" (like checking email, creating documents, browsing the web, etc.) very, very well.

Openbox is ***not*** for beginning Linux users. Tweaking and tuning Openbox involves, in many cases, hand editing various configuration files that are tucked away in your Linux file system. In keeping with the lightweight nature of Openbox, there aren't a lot of GUI tools available to assist with the management of those files, as there are with the bigger and heavier full-blown desktop environments like KDE and Gnome. But if you ever wanted to learn more about Linux and how it all comes together, learning the ins and outs of Openbox will be help you along that path. If you are a power Linux user, then you will feel right at home with Openbox, and with making all of the manual edits to the configuration files to tweak and tune your Openbox installation.

One difference that I did notice was the inclusion of **sudo** in the PCLinuxOS Openbox ISOs. Traditionally, sudo is NOT included with official PCLinuxOS releases, since an improper use of sudo could compromise the security of a computer it is being used on. In somewhat of a defense, there is not a user predefined in the sudoers file. Still, it has become pretty much standard operating procedure for PCLinuxOS to not embrace the use of sudo. Having sudo pre-installed gives the opposite impression.

As we here at the magazine explore Openbox, we'll be bringing you articles on ways to customize your PCLinuxOS Openbox installation. Hopefully, these articles will show you the elegance, simplicity and various uses of Openbox, as well as ways you can customize and tailor your Openbox installation.

Through the articles we have planned, we hope that we take the "sting" out of configuring Openbox, and show you the options you have for tweaking Openbox. As you will see, it is actually quite easy to make your Openbox installation quite unique.

Want To Help?

Would you like to help with the PCLinuxOS Magazine? Opportunities abound. So get involved!

You can write articles, help edit articles, serve as a "technical advisor" to insure articles are correct, create artwork, or help with the magazine's layout.

Join us on our **Google Group mailing list.**

Screenshot Showcase

Posted by Taco.22 on December 13, 2011

Openbox: Edit rc.xml To Gain Control

by Paul Arnote (parnote)

Last November, we covered the **lxde-rc.xml** file, back when we were covering the LXDE desktop. Since LXDE uses Openbox as its window manager, a lot of the information we covered then is also applicable to the Openbox release of PCLinuxOS.

Openbox uses a file similar to that used by LXDE, called **rc.xml**. The file, stored in your **/home/username/.config/openbox** folder, is responsible for helping define various aspects of how your windows are displayed on the screen, mousebindings, keybindings, and many other settings. You will have to enable the display of hidden files in your file manager in order to see the hidden folder.

So that you can tweak, tune and customize your Openbox installation fully and completely, let's take a look at the different sections of the rc.xml file, as well as some things you can do with it.

Basic File Structure

As you might have already guessed, the rc.xml file uses the XML format, which stands for E**x**tensible **M**arkup **L**anguage. XML is very much like HTML. Just like with HTML, the command sets have opening and closing statements, or tags. For example, to make bold text in HTML, you would use *some text here*, placing the text you want to make bold between the and tags.

In XML, every opening tag has a closing tag. So, if you have a <keybind *something something*> tag, you will also have a </keybind> tag to close it out. XML commands can also be nested (and most often are), as this excerpt from the rc.xml file shows:

```
<keybind key="C-A-x">
    <action name="Execute">
        <command>xchat</command>
     </action>
</keybind>
```

As you can see, the nested commands in XML work from the outside to the middle, then back out again. In the excerpt above, we start with the keybinding tag (<keybind …>), define the keys on the keyboard to use (C-A-x), specify the action to take when those keys are pressed (Execute), specify the command to execute (<command>xchat), then we back our way out, closing out the command tag (</command>), then closing out the action tag (</action>), then closing out the keybind tag (</keybind>). Failure to close out the tags, or closing out the tags in the wrong order, will result in a corrupt XML file. Typos and misspellings will also result in a corrupt XML file. ***So, double check everything before you save the file!*** In fact, it would be a prudent decision to make a backup copy of your original, unaltered rc.xml file. This way should something go horribly wrong, you can always restore your computer to a previous working state, simply by replacing the bad rc.xml file with one that you know works.

Fortunately, the text editor that comes installed on the PCLinuxOS Openbox releases is **Geany**, which is very good at syntax highlighting. While it may not initially seem like a really big deal, syntax highlighting in a text editor can save you hours of debugging time, in the event that you end up with a corrupt XML file. Since the highlighting is done "on the fly," it's also easy and quick to see if you have made any mistakes ***as you are typing*** the commands.

You may notice comments interspersed throughout the rc.xml file. Comment lines are those that look like this:

```
<!-- comment placed here -->
```

If you make additions to your rc.xml file, it would be a great idea to also insert your own comments. Six months after you make changes or additions, you may not remember exactly what you did, unless you left yourself a "calling card" of sorts, in the form of a comment that describes what you did. Also, the comments that are built into the default rc.xml file can go a long way in helping you understand what to do in any given section of the file.

When you open up the rc.xml file, you will notice that the first tag in the file starts off <openbox-config *something something*>. This means that the last tag in the rc.xml file will be </openbox-config>, to close out the rc.xml file. In between will be all of the other XML tags that define the Openbox options.

First Section: Resistance

Think of the resistance setting as how hard you have to push a window on your screen against the screen edge before it moves that window to the next virtual desktop. The higher the number, the harder you have to "push" before that window will move to the next desktop. Here are the default settings in the installed Openbox:

```
<resistance>

     <strength>10</strength>

     <screen_edge_strength>20

     </screen_edge_strength>

</resistance>
```

The "strength" setting determines how much resistance there is between adjacent windows, before one is allowed to overlap the other. The "screen_edge_strength" setting determines how much resistance there is at the screen edge, before allowing the selected window to move to the next desktop.

On my copy of Openbox, I've changed the strength to 50, and the screen_edge_strength setting to 100. This requires me to push fairly hard against the screen edge, before the window moves over to the next desktop. I've done this because sometimes I simply want a window positioned at the screen edge, and not moved to another desktop.

Second Section: Focus

The second section of rc.xml deals with how Openbox focuses the windows on your desktop. There are a number of options you can choose from. Here is the "focus" section from my installation of Openbox:

```
<focus>

     <focusNew>yes</focusNew>

     <!-- always try to focus new windows
      when they appear. other rules do
      apply -->

     <followMouse>no</followMouse>

     <!-- move focus to a window when you
      move the mouse into it -->

     <focusLast>no</focusLast>

     <!-- focus the last used window when
      changing desktops, instead of the one
      under the mouse pointer. when
      followMouse is enabled -->

     <underMouse>no</underMouse>

     <!-- move focus under the mouse, even
      when the mouse is not moving -->

     <focusDelay>200</focusDelay>

     <!-- when followMouse is enabled, the
      mouse must be inside the window for
      this many milliseconds (1000 = 1 sec)
      before moving focus to it -->

     <raiseOnFocus>no</raiseOnFocus>

     <!-- when followMouse is enabled, and
      a window is given focus by moving the
      mouse into it, also raise the window
      -->

</focus>
```

To start with, the "FocusNew" setting tells Openbox to automatically focus on any new windows that are displayed on your desktop. The "followMouse" setting tells Openbox to change the window focus to the window under your mouse. The default setting here is "yes," but I have changed that to "no."

The "focusLast" setting only has an effect when "followMouse" is enabled, or set to "yes." The default value for "focusLast" is "no." Changing it to "yes" will automatically re-focus the last active window on a different desktop, instead of allowing the focus to change to whichever window is currently under your mouse pointer.

With the "underMouse" setting, the window focus is moved to the window under the mouse, even if the mouse isn't moving. The default value is "no." The "focusDelay" setting selects the number of milliseconds delay before the window is focused when moving a mouse into a window. The default value is 200 ms. The "raiseOnFocus" setting causes the window receiving the focus to be raised to the top, automatically. The default setting is set to "no." These three settings are dependent on the

"followMouse" setting, and if the "followMouse" setting is turned off (set to "no"), then these settings effectively do nothing.

Third Section: Placement

The "Placement" section of the rc.xml file tells Openbox how to place the windows on your desktop. The default values are shown below:

```
<placement>

     <policy>Smart</policy>

     <!-- 'Smart' or 'UnderMouse' -->

     <center>yes</center>

     <!-- whether to place windows in
      the center of the free area found
      or the top left corner -->

     <monitor>Active</monitor>

</placement>
```

You can choose whether the windows are placed under the mouse, or if Openbox places the windows (the "smart" policy). You can also determine if the windows are centered on your desktop or not, by changing the "center" setting.

Fourth Section: Theme

The "Theme" section of rc.xml tells Openbox what theme to use when displaying your windows, as well as what options (fonts, window decorations, window decoration order, etc.) to use. Below is a snippet of the theme section of the rc.xml file:

```
<theme>

     <name>Appleish</name>

     <titleLayout>NSLIMC</titleLayout>

     <!--
       avaible characters are NDSLIMC,
       each can occur at most once.
       N: window icon
       L: window label (AKA title).
       I: iconify
       M: maximize
       C: close
       S: shade (roll up/down)
       D: omnipresent (on all desktops).
      -->

     <keepBorder>yes</keepBorder>

     <animateIconify>yes</animateIconify>

     <font place="ActiveWindow">

     <name>Liberation Sans</name>

     <size>10</size>

     <!-- font size in points -->

     <weight>Bold</weight>

     <!-- 'bold' or 'normal' -->

     <slant>Normal</slant>

     <!-- 'italic' or 'normal' -->

     </font>

     ...
```

The first setting, "name," tells Openbox which theme I've chosen to use. In my case, that is the "Appleish" theme, which I installed via Synaptic.

The next setting, "titleLayout," tells Openbox which window decorations you want to have displayed on the window title bar, along with the placement of those decorations. Fortunately, there is a "key" provided in the comments of this section of the rc.xml file. Each character can be used only once in any given theme.

The "keepBorder" setting tells Openbox if it should keep the window borders if window decorations are turned off. The default value here is "yes." The "animateIconify" setting enables a slight animation feature when the windows are minimized to the panel. The default setting is "yes."

Next, we specify the attributes of the various text parts of the theme. Starting with the title bar of the "ActiveWindow," we specify the "name" of the font, "size" of the font, "weight" of the font, and the "slant" of the font. We repeat this for each of the following:

InactiveWindow, MenuHeader, MenuItem and OnScreenDisplay.

We'll discuss themes in more depth, a little later, in a separate article.

Fifth Section: Desktops

As you may guess, this section of the rc.xml file specifies the details about your virtual desktops.

```
<desktops>

    <!-- this stuff is only used at
    startup, pagers allow you to change
    them during a session

    these are default values to use when
    other ones are not already set by
    other applications, or saved in your
    session

    use obconf if you want to change these
    without having to log out and back in
    -->

    <number>4</number>

    <firstdesk>1</firstdesk>

    <names>

    <name>Water</name>

    <name>Fire</name>

    <name>Earth</name>

    <name>Air</name>

    </names>

    <popupTime>500</popupTime>

    <!-- The number of milliseconds to
    show the popup for when switching
    desktops. Set this to 0 to disable the
    popup. -->

</desktops>
```

The first setting, "number," sets the number of virtual desktops available. The "firstdesk" setting specifies which desktop should be displayed when Openbox is first started. Under the "names" setting, you can give a name to each of your virtual desktops. You can give them any name you want. You can call them the names of your children, give them philosophical names, or keep it simple and give them numbers. By default, they are named Water, Fire, Earth and Air. The last setting, "popupTime," setts the number of milliseconds to show the popup window on your screen, when changing desktops. The default setting is 500 ms, or one-half second.

Sixth Section: Resize

The "Resize" section governs how Openbox displays windows when you are resizing or moving windows. Below is the "resize" section of the rc.xml file, with all the default values displayed:

```
<resize>

    <drawContents>yes</drawContents>

    <popupShow>NonPixel</popupShow>

    <!-- 'Always', 'Never', or 'Nonpixel'
    (xterms and such) -->

    <popupPosition>Center</popupPosition>

    <!-- 'Center' or 'Top' -->

    <popupFixedPosition>

    <x>0</x>

    <y>0</y>

    </popupFixedPosition>

</resize>
```

The "drawContents" setting tells Openbox to redraw the program inside the window whenever you are resizing or moving the window. The "popupShow" setting, set to "NonPixel," only displays a popup window on the screen when the resizing window specifies that it is being resized more than one pixel. The popup window shows the screen coordinates for the resized window, along with the pixel size of the window. This usually applies to terminals. You can also set the "popupShow" setting to always be displayed, or to never be displayed.

The "popupPosition" setting allows you to specify the location where the popup window will appear. With

the default setting of “center,” the popup window is displayed at the center of the window being resized or moved. The other choices are “top,” where the popup window is displayed above window’s title bar, or “fixed,” where the popup window is displayed at a location defined by “popupFixedPosition.” The latter accounts for the two settings, x and y, under the “popupFixedPosition” setting.

Seventh Section: Margins

The “margins” section of the rc.xml file literally creates margins on your screen.

```
<margins>

    <top>1</top>

    <bottom>2</bottom>

    <left>2</left>

    <right>2</right>

</margins>
```

Specify the number of pixels you want to have as a screen margin in the top, bottom, left and right settings, and no items (except for wallpaper) will be drawn in those areas of your screen.

Eighth Section: Dock

Openbox makes good use of items from other desktops and window managers, and allows you to use any number of dockapps that are available for WindowMaker, Xfce, KDE, Gnome, and many others. You can find a whole slew of dockapps available for use with Openbox at dockapps.org. The “dock” section of the rc.xml file helps govern their placement, and is in effect only if you are running a dockapp.

```
<dock>

    <position>TopLeft</position>

    <!-- (Top|Bottom)(Left|Right|)
    |Top|Bottom|Left|Right|Floating -->

    <floatingX>0</floatingX>

    <floatingY>0</floatingY>

    <noStrut>yes</noStrut>

    <stacking>Above</stacking>

    <!-- 'Above', 'Normal', or 'Below' -->

    <direction>Horizontal</direction>

    <!-- 'Vertical' or 'Horizontal' -->

    <autoHide>no</autoHide>

    <hideDelay>300</hideDelay>

    <!-- in milliseconds (1000 = 1 second)
    -->

    <showDelay>300</showDelay>

    <!-- in milliseconds (1000 = 1 second)
    -->

    <moveButton>Middle</moveButton>

    <!-- 'Left', 'Middle', 'Right' -->

</dock>
```

The “position” setting allows you to set where the dockapp appears. The default value is TopLeft. Other possibilities are TopRight, BottomLeft, BottomRight, Top, Bottom, Left, Right or Floating. If you select floating, then the next two settings, floatingX and floatingY set the horizontal and vertical positioning, respectively.

The “noStrut” setting allows windows to be placed over the dockapp. The “stacking” setting determines which layer of the desktop to place the dockapp. You can decide if your dockapps are positioned in a vertical or horizontal row with the “direction” setting.

By toggling the “autoHide” setting (the default is “no”), you can cause your dockapps to automatically hide until you mouse over their position. If the “autoHide” setting is activated, then the “hideDelay” and “showDelay” settings are activated. The setting for both are in milliseconds, and the default value is 300 ms.

Finally, the “moveButton” setting determines which mouse button to use to move the dockapp to a new location on your desktop. The default value is “Middle.”

Ninth Section: Keyboard

The “keyboard” section is typically the largest section of the rc.xml file, and one where you can make some very interesting changes in the functionality of Openbox. Below is an excerpt from the first few lines of the keyboard section of my rc.xml file:

```
<keyboard>

    <!-- Keybindings for desktop switching
-->

    <keybind key="C-A-Left">

      <action name="DesktopLeft">

        <dialog>no</dialog>

        <wrap>yes</wrap>

      </action>

    </keybind>

    <keybind key="C-A-Right">

      <action name="DesktopRight">

        <dialog>no</dialog>

        <wrap>yes</wrap>

      </action>

    </keybind>

    <keybind key="C-F1">

      <action name="Desktop">

        <desktop>1</desktop>

      </action>

    </keybind>
```

I previously covered keybindings fairly thoroughly in the November, 2010 article LXDE: Meet The Heart & Soul – lxde-rc.xml. Since LXDE uses Openbox as its window manager, the information in that article applies equally to the Openbox rc.xml file. Instead of repeating all of that information here, I’ll simply refer you to that article for a more complete discussion of keybindings.

One thing that I did find in my installation of Openbox were keybindings that were either duplicated, or that are assigned to keystroke combinations that typically are reserved for other functions. For example, for the “DesktopLeft” and “DesktopRight” settings in my excerpt above, the default values are defined as C-Left (Control key + Left cursor key) and C-Right (Control key + Right cursor key), respectively. However, those keystroke combinations are typically reserved for use in word processing programs to move your cursor through your document one word at a time. Thus, I changed my keybindings for “DesktopLeft” and “DesktopRight” to C-A-Left (Control key + Alt key + Left cursor key) and C-A-Right (Control key + Alt key + Right arrow key). This preserves the proper functioning of the original keystroke combination for use in my word processing programs.

Another dubious keybinding (for me, at least) is the definition of the C-w (Control key + “w” key) to execute the “ShowMenu” function, that brings up a menu displaying all of the desktops and the applications running on each one. I typically use the C-w key to close out child windows in an application, like individual tabs in my web browsers. I simply changed my keybindings so that C-A-w executes the “ShowMenu” function, preserving the use of C-w for what I normally use it for.

Additionally, many keybindings are defined to use the Super key (a.k.a. the Windows key). However, my IBM T23 does not have a Super key, so those keystroke combinations are impossible for me to use. Instead, I changed those keybindings that use the Super key to keybindings that use Control + Alt.

One real beauty of the “keyboard” section of the rc.xml file is that I can change the keystroke combinations used so that they are more closely tailored to how I tend to work with my computer, and to better fit the configuration of my hardware. For the predefined keybindings that use the the Super key to launch programs I never or rarely use, or that perform tasks that I customarily don’t perform from the keyboard, I simply left them unchanged.

Another real beauty of the “keyboard” section is that if there is a bash script you like to routinely run, you can assign it a keybinding so that it is only a keystroke or two away from execution.

I recommend that you study how the keybinding section is set up. It won’t take long before you catch on to the format and start coming up with your own custom keybindings. If you want more information on keybindings, you can visit the Openbox Wiki entry on keybindings, where there is a complete breakdown of all of the keybinding settings. We’ll also be mentioning keybindings later on, in other Openbox articles in The PCLinuxOS Magazine.

Tenth Section: Mouse

Under the “mouse” section of the rc.xml file, you can control most of the settings for how your mouse functions under Openbox. Below is an excerpt from the rc.xml file on my Openbox installation (this section is too long to print in its entirety):

```
<mouse>

    <dragThreshold>8</dragThreshold>

    <!-- number of pixels the mouse must
move before a drag begins -->

    <doubleClickTime>200</doubleClickTime>

    <!-- in milliseconds (1000 = 1 second)
-->

    <screenEdgeWarpTime>400
    </screenEdgeWarpTime>

    <!-- Time before changing desktops when
    the pointer touches the edge of the
    screen while moving a window, in
    milliseconds (1000 = 1 second). Set
    this to 0 to disable warping -->

    <context name="Frame">

      <mousebind button="A-Left"
      action="Press">

        <action name="Focus"/>

        <action name="Raise"/>

      </mousebind>

      <mousebind button="C-A-Left"
      action="Click">

        <action name="Unshade"/>

      </mousebind>

      <mousebind button="A-Left"
      action="Drag">

        <action name="Move"/>

      </mousebind>

      <mousebind button="A-Right"
      action="Press">

        <action name="Focus"/>

        <action name="Raise"/>

        <action name="Unshade"/>

      </mousebind>
```

Some basic mouse settings, like how fast a double click of the mouse has to occur before it is recognized as a double click, the number of pixels to move the mouse before it is recognized that you are dragging a window or other screen element, or how much time you have to push a window against a screen edge before it warps to the adjacent desktop, are taken care up right up front.

The bulk of the “mouse” section deals with mousebindings. Mousebindings are very, very similar to keybindings, and define what action to take when different mouse buttons are pushed, in combination with certain key presses, depending on the context in which Openbox detected them. The latter is determined by the “context name” setting.

For a full discussion of the settings you can make, check out the Openbox Wiki page on mousebindings. Most users probably won’t find much of a reason to mess around with the mousebindings – unless you are a tweak-aholic, or unless you have the latest, greatest gaming mouse with 56 button combinations.

Eleventh Section: Menu

As you might imagine, the "menu" section controls the behavior of menus on Openbox. Below is the entire menu section from the rc.xml file on my Openbox installation:

```
<menu>

    <!-- You can specify more than one menu
    file in here and they are all loaded,
    just don't make menu ids clash or,
    well, it'll be kind of pointless -->

    <!-- default menu file (or custom one
    in $HOME/.config/openbox/) -->

    <file>menu.xml</file>

    <hideDelay>200</hideDelay>

    <middle>no</middle>

    <submenuShowDelay>100

    </submenuShowDelay>

    <applicationIcons>yes

    </applicationIcons>

</menu>
```

The first setting, "file," specifies which file contains the data used to create the application menu in Openbox, which is accessible via a right click on the desktop. You can have multiple menu files, but as the comments above indicate, you have to insure that there are no clashes between menu ids across all of the menu files. Typically, the menu file being specified here resides in your **/home/username/.config/openbox** folder. There is another menu.xml file in **etc/xdg/openbox**, but to be totally honest, I cannot see that it is used or where it fits into the picture.

The "hideDelay" setting causes the menu to immediately disappear if you click longer than the specified length of time (default is 200 ms). If you are a "slow mouse clicker," you may want to specify a longer time. Just remember that 1000 ms equals 1 second, so 500 ms would be one-half of a second. Clicking and releasing the menu for a time less than that specified will cause the menu to continue being displayed when you release the mouse button.

The "middle" setting causes the menus to be center aligned vertically, as opposed to being top aligned. The "submenuShowDelay" and "submenuHideDelay" (not shown) determine the time delay before showing, and subsequently hiding, a submenu. Both values must be less than the "hideDelay" setting, or they are ignored.

With the "applicationsIcons" setting, you can tell Openbox whether or not to show the application icons on the desktop and in the application menu. I have found that this setting has no effect with the right click Openbox application menu, as I've never seen any application icons displayed in that menu, regardless of the setting. However, it may play a role when you use a bona fide application launch menu, like that which you get when using the LXDE panel (lxpanel).

One setting not included in the menu section of the rc.xml file on my Openbox installation is the "manageDesktops" setting. This allows you to add or subtract desktops as you need, on the fly, right from the right click Openbox menu. If you are working along, and all of a sudden discover that having another it would help your productivity to have an additional desktop desktop, you can add it right there, on the spot, from the right click Openbox menu. Apparently, the default value for this setting is "yes," since I have that ability in my right click Openbox menu, despite its absence from the menu section of my rc.xml file.

Twelfth Section: Applications

Under the "applications" section, you can define the behavior of individual applications in Openbox. Below is a copy of the "applications" section of my rc.xml file:

```
<applications>

    <application name="draklive-install">

      <maximized>true</maximized>

    </application>

    <application name="tint2">

      <layer>below</layer>

    </application>
```

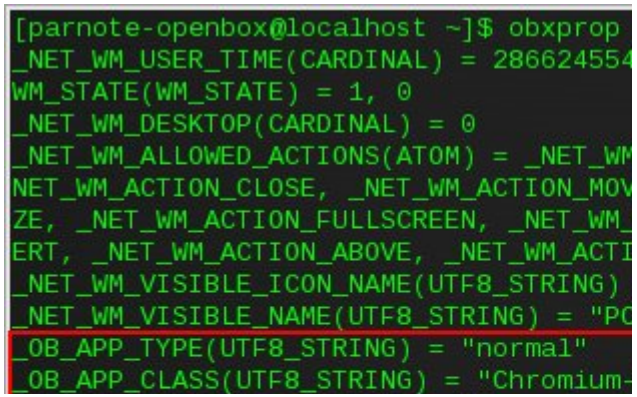
```
    … (example deleted)

</applications>
```

This section of your rc.xml file can easily become quite a bit larger than what is shown here. With the settings in this section, I can specify positioning, desktops, display states, layers and a whole host of other options, for each of the applications you typically run.

Call me a creature of habit if you will, but I've gotten into the habit of having my web browser window on desktop 1, and my open IRC chat client on desktop 3. I do this on all of my computers, all running different versions of PCLinuxOS. It just one thing that helps keep me organized when I have several applications open at the same time.

So, if I click on a link in IRC, I really don't want Firefox to open up on desktop 3. It puts a chink in my routine. With the settings in the "applications" section of my rc.xml file, it's very easy to restrict Firefox to opening only on desktop 1.

In order to provide Openbox the information it needs, it's a good idea to run a command line program, called **obxprop**. When you run obxprop, the cursor will change to a cross-hair. Move your cursor over the window you want information on, and click your cursor. Immediately, you will be presented with a lot of information, as in the image below:



You are looking for the information highlighted in the red box. You need to specify either the _OB_APP_CLASS or _OB_APP_NAME, or both in order for Openbox to find the proper application. You can shorten the output by entering **obxprop | grep "^_OB_APP"** to list only the information needed by Openbox.

Now that you have the information that Openbox needs, you can start to set up your application specific controls. First, we need to tell Openbox which application we want to make rules for. We do this by specifying the name and/or class of the application, with the information provided by obxprop.

```
<application name="chromium-browser"
class=”Chromium-browser>
```

Let's say we always want Chromium to start up on desktop 1. We next need to enter the following:

```
<desktop>1</desktop>
```

If we wanted to make sure that it always opened up maximized, we'd enter another line:

```
<maximized>true</maximized>
```

Finally, we close out the "application" tag:

```
</application>
```

You can also use wildcard characters in the name and class fields. A "*" matches any number of characters, while a "?" matches any single character.

There are several options available when making application specific settings or rules. Refer to the Openbox Wiki section on per application settings to see examples on how to apply those rules and settings. You can also take a look at your rc.xml file to see some examples of how to set up application specific rules and settings.

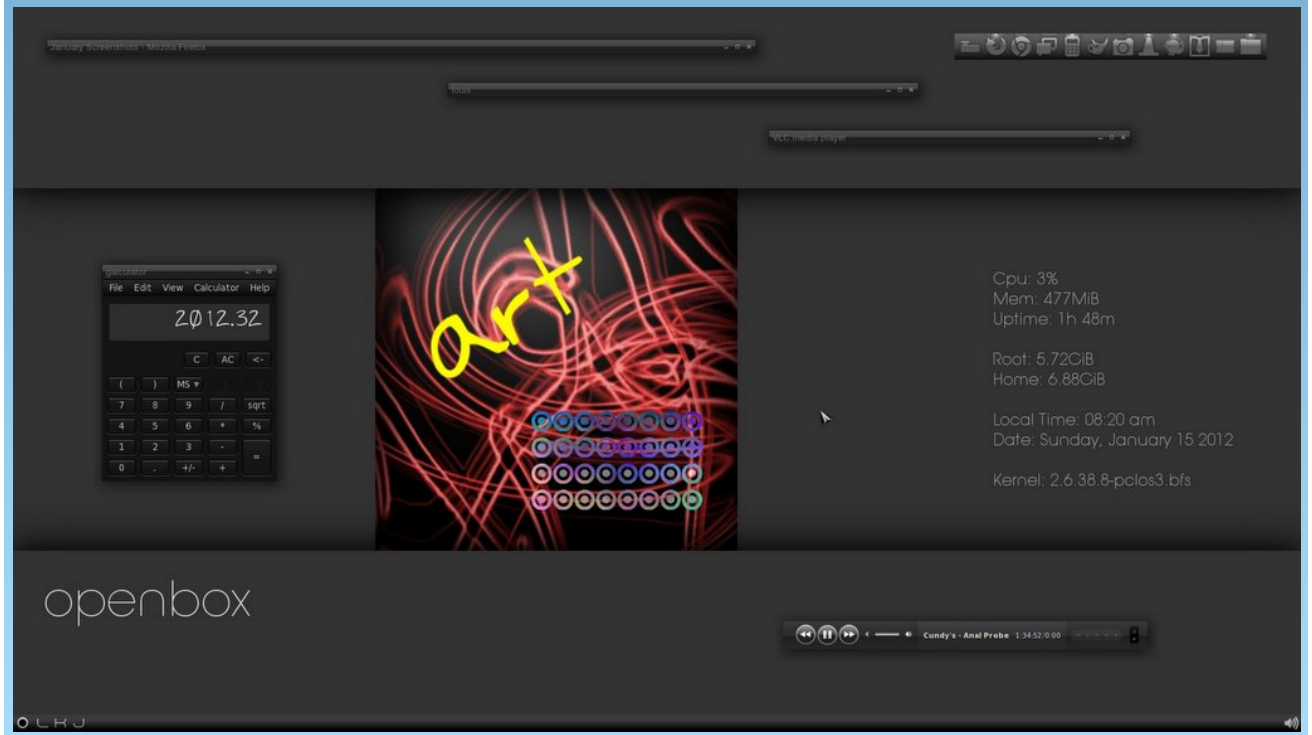
Thirteenth Section: Coordinates

This section of the rc.xml file is not included in my Openbox installation. As such, this section can be considered to be optional, since everything works just fine without it. If you are interested in learning more about the coordinates section of the rc.xml file, I'll refer you to the coordinates section of the Openbox Wiki.

Summary

As you can see, the rc.xml file exerts a ***lot*** of control over the behavior of Openbox. Once you get the hang of the XML format of the file, making changes is very easy. Because it is so easy to make changes, and because it offers so many options, you have a lot of leeway in customizing the behavior of your Openbox installation. You can truly tailor and trim your Openbox installation to be uniquely yours.

Screenshot Showcase



Posted by LKJ on January 5, 2012

Openbox: ZRAM - What It Is & How To Use It

by Darrel Johnston (djohnston)

ZRAM (formerly ramzswap) is a block device which is created in your computer's memory, or RAM. It looks and acts to the system like a disk drive. However, this disk drive has only one function, to act as a swap disk. One difference between ZRAM and a normal swap partition is that the pages swapped to ZRAM are compressed before being stored. This technique has two advantages. (1) More page data can be stored because the data is compressed. (2) Because the swapped memory is stored in RAM, rather than a normal disk drive, read and write access times are much quicker.

The author of the program, nitingupta910, mentions another possible use on his Google code webpage. "With compcache at hypervisor level, we can compress any part of guest memory transparently - this is true for any type of Guest OS (Linux, Windows etc.). This should allow running more number of VMs for given amount of total host memory."

Melodie has the older version (ramzswap) enabled on the live CDs for Openbox Bonsai and Openbox full, "out of the box" (pun intended). She has the new version (ZRAM) enabled on the Openbox Edu live CD. Both versions initialize a block device in RAM from the /etc/rc.d/rc.local file. The first (ramzswap) version creates the device /dev/ramzswap0. The size of the ramzswap0 device is defined by using the rzscontrol executable. The default size is 15% of RAM. The device can be used as a front end to a swap partition on a disk drive by defining the swap partition in the rc.local file as shown below.

```
# Ramzswap will act like a swap front end
if RAMZ_BACKING_SWAP is defined.

# Writes are forwarded to this device when
memory limit is reached or data
# is not compressible.

# i.e. RAMZ_BACKING_SWAP="/dev/sda3"

RAMZ_BACKING_SWAP=""
```

For kernel version 2.6.37 or higher, the device created is /dev/zram0. The rzscontrol executable can not be used to modify the zram0 device, so is no longer needed. Since it uses a different kernel module, the older /etc/rc.d/rc.local file must be replaced with the newer version. You can obtain it here. If you use the default parameters, the zram0 device will be 25% the size of available RAM. For example, if you have 1GB of available RAM, the created zram0 device will be 256MB in size, leaving you with 768MB of RAM. The default size is defined in the portion of the rc.local file shown below.

```
# Swap size = 25% of free memory;
    ZRAM_SWAP_SIZE=$(($FREE_MEM/4))

echo $(($ZRAM_SWAP_SIZE * 1024 * 1024)) >
/sys/block/zram0/disksize
```

In order to change the size to a fixed amount, comment the ZRAM_SWAP_SIZE= line and replace the string $($ZRAM_SWAP_SIZE * 1024 * 1024)) with an integer representing the total number of bytes. As an example, to get a fixed size of 256MB of ZRAM, you would edit the rc.local file as shown below.

```
# Swap size = 25% of free memory;
#    ZRAM_SWAP_SIZE=$(($FREE_MEM/4))

# Swap size = 256MB
echo 268435456 > /sys/block/zram0/disksize
```

I understand the advantages of using a ZRAM device, and agree that storing virtual machines within ZRAM devices is a more efficient method for utilizing available RAM. What I struggle with is understanding how it is more useful on a desktop or laptop than a conventional swap partition, other than the speed advantage.

Openbox: Tint2 vs Lxpanel

by Paul Arnote (parnote)

As you may have noticed, the PCLinuxOS version of Openbox comes in two versions: a “full” version with a full compliment of pre-installed applications, and the “Bonsai” version, which is a lightweight installation that comes with a minimum of pre-installed applications. But the differences don’t stop there.

One notable difference is the selection of the panel used by the two versions. The “full” version uses tint2 as the panel, while Bonsai uses lxpanel to provide the end user with a useful panel.

Lxpanel is “borrowed” from the LXDE desktop. Tint2 is a product of a “Google Summer of Code” project, with the aim to create a simple, easy to use and lightweight panel. Currently, tint2 is up to version 0.11.

Both work exceptionally well, and both are excellent choices for a lightweight panel. Which you use depends on what you are expecting from a panel.

At first glance, the most noticeable difference is that lxpanel has an application menu, a quick launch area, and a desktop pager, while those items are lacking in the tint2 panel. However, both lxpanel and tint2 have a task bar area, a system tray and a clock. My initial reaction to the tint2 panel wasn’t all that positive, since it was missing some of the items I was accustomed to using on my panel. However, the “missing items” on the tint2 panel really aren’t all that missed, since all of those items are available

Lxpanel

Tint2

from the Openbox right click menu, regardless of which panel you choose to use.

Since we have already covered lxpanel in a previous issue (October, 2010) when we were covering the LXDE desktop, much of the rest of this article will deal with tint2 and how to configure it.

Tint2

Let’s start off by taking a closer look at the tint2 panel (bottom).

The tint2 panel, by default, does ***not*** show you all desktops in a single view, nor does it separate the icons by desktop. Thankfully, the tint2 panel is configurable and easy to configure, thanks to the tint2rc file. It is located in your **`/home/username/.config/tint2 folder`**. With just a few simple edits of the tint2rc file, you can easily configure tint2 to be the panel you want.

Here’s the tint2rc file from my Openbox installation:

```
# Tint2 config file
# Generated by tintwizard
(http://code.google.com/p/tintwizard/)
# For information on manually configuring
tint2 see
http://code.google.com/p/tint2/wiki/
Configure

# Background definitions
# ID 1
rounded = 7
border_width = 2
background_color = #000000 60
border_color = #FFFFFF 16

# ID 2
rounded = 5
border_width = 0
background_color = #FFFFFF 40
border_color = #FFFFFF 48

# ID 3
rounded = 5
```

```
border_width = 0
background_color = #FFFFFF 16
border_color = #FFFFFF 68

# Panel
panel_monitor = all
panel_position = bottom center horizontal
panel_size = 94% 30
panel_margin = 0 0
panel_padding = 7 0 7
panel_dock = 0
wm_menu = 0
panel_layer = top
panel_background_id = 1

# Panel Autohide
autohide = 0
autohide_show_timeout = 0.3
autohide_hide_timeout = 2
autohide_height = 2
strut_policy = follow_size

# Taskbar
taskbar_mode = multi_desktop
taskbar_padding = 2 3 2
taskbar_background_id = 0
taskbar_active_background_id = 0

# Tasks
urgent_nb_of_blink = 8
task_icon = 1
task_text = 1
task_centered = 1
task_maximum_size = 140 35
task_padding = 6 2
task_background_id = 3
task_active_background_id = 2
task_urgent_background_id = 2
task_iconified_background_id = 3

# Task Icons
task_icon_asb = 70 0 0
task_active_icon_asb = 100 0 0
task_urgent_icon_asb = 100 0 0
task_iconified_icon_asb = 70 0 0
# Fonts
task_font = sans 7
task_font_color = #FFFFFF 68
task_active_font_color = #FFFFFF 83
task_urgent_font_color = #FFFFFF 83
task_iconified_font_color = #FFFFFF 68
font_shadow = 0

# System Tray
systray = 1
systray_padding = 0 4 5
systray_sort = ascending
systray_background_id = 0
systray_icon_size = 16
systray_icon_asb = 70 0 0

# Clock
time1_format = %H:%M
time1_font = sans 8
time2_format = %a %B %d
time2_font = sans 6
clock_font_color = #FFFFFF 74
clock_padding = 1 0
clock_background_id = 0
clock_rclick_command = orage

# Tooltips
tooltip = 0
tooltip_padding = 2 2
tooltip_show_timeout = 0.7
tooltip_hide_timeout = 0.3
tooltip_background_id = 1
tooltip_font = sans 10
tooltip_font_color = #000000 80

# Mouse
mouse_middle = none
mouse_right = close
mouse_scroll_up = toggle
mouse_scroll_down = iconify

# Battery
battery = 0
battery_low_status = 10
battery_low_cmd = notify-send "battery low"
battery_hide = 98
bat1_font = sans 8
bat2_font = sans 6
battery_font_color = #FFFFFF 74
battery_padding = 1 0
battery_background_id = 0

# End of config
```

I do NOT recommend using the “Tint Wizard” program to control and change your settings. Rather, I recommend editing the tint2rc file by hand, by loading it into Geany – or any other plain text editor. From my personal experience, I can attest to the fact that the Tint Wizard program is quite capable of producing corrupt tint2rc files. While a good idea, it doesn’t seem to be well implemented. You will have more consistent results by hand editing the file – at least for now, until the Tint Wizard bugs are worked out.

Background IDs

The very first thing that is listed are the background definitions. We can define as many background definitions as we feel are necessary. In the default setup of tint2 in Openbox, there are three

backgrounds defined. These blocks of settings determine whether or not the particular background has rounded corners, border widths, and the colors of items.

Even though all of our background definitions set the border width to zero, we must also set a color definition for the border. Likewise, we also set a color definition for the background. All color definitions are preceded by a "#" symbol, followed by the hexidecimal color code for the color we want to use. The number after the hexidecimal color definition sets the transparency for that color, where 100 is opaque and zero is completely transparent.

As we go through each of the other sections, you will note that each section specifies which background ID to use when it is drawn on your screen.

Panel

The panel section of the tint2rc file determines much of the overall appearance of the tint2 panel. The first item, panel_monitor = all, tells tint2 to draw the panel on all the monitors it finds connected to your computer. You can also specify it to draw the tint2 panel on specific monitors connected to your computer, should you have multiple monitors connected.

The second item, panel_position, tells tint2 where to draw the panel on your screen. The default value, at least in the PCLinuxOS Openbox installation, is to draw the panel at the bottom of your screen, in a horizontal aspect, and centered on your screen.

The panel_size parameter tells tint2 the width and height of your panel. My panel is set to occupy 94% of the width of my screen, with a height of 30 pixels. Setting your width to "0" will cause the tint2 panel to occupy the full width of your screen.

Panel_margin tells tint2 whether or not to employ a margin in relationship to your monitor edge. The defaults in PCLinuxOS Openbox are "0 0" and place the tint2 panel up against the screen edge, without a margin. However, if you want to insure that there is a little "breathing room" around you panel, specify how many pixels you want your horizontal margin (the first number), and how many pixels you want your vertical margin to be (the second number), in relationship to the nearest screen border as defined in the panel_position parameter.

The fifth item, panel_padding, tells tint2 the horizontal, left-to-right, padding (the first number), the vertical padding (the second number), and the horizontal spacing (the third number).

Panel_dock tells tint2 whether or not to place the tint2 panel in the window manager's dock. The default value is "0," which causes tint2 to bypass the Openbox dock.

The seventh item, wm_menu, determines whether or not the default window manager menu is displayed when you right click on the tint2 panel. The default value is "0." Changing this to "1" may be useful, depending on how you work with your computer.

The panel_layer setting allows you to specify if the panel is drawn on the top layer, the bottom layer, or is treated like a normal window. The default value in PCLinuxOS Openbox is "top."

Finally, the panel_background_id parameter tells tint2 which of the previously defined background IDs to use when drawing the panel. In our case, that would be to use the first background ID that we defined, with a black background color and an opacity of 60%, and a white border color, with an opacity of 16%.

Panel Autohide

As you might expect, the "panel autohide" section of the tint2rc file controls the autohide capabilities of the tint2 panel. By default, the tint2 panel's ability to autohide is turned off (autohide = 0). When the tint2 panel is in this state, the rest of the settings have no effect. Changing it to "autohide = 1" will cause the tint2 panel to autohide, and only appear when you mouse over its intended location.

When autohide is activated, the other settings become active. The autohide_show_timeout (default 0.3 seconds) specifies how many seconds (or tenths of a second) delay before the panel is shown when you move your mouse over the intended location of the tint2 panel. The autohide_hide_timeout parameter (default of 2 seconds) specifies how many seconds the panel is shown after you move your mouse outside the boundaries of the revealed panel, before it is hidden again. The autohide_height parameter (default of 2 pixels) specifies how many pixels the hidden panel occupies on your screen.

The last setting in this section of the tint2rc file, strut_policy, actually belongs to the panel section. STRUTs are used by the Openbox to decide the size of maximized windows. It determines if 'maximized windows' should follow tint2 size (follow_size, and default) or use the minimum size (minimum), or use the screen size (none).

Taskbar

Here, we define the appearance of the taskbar section of the tint2 panel. The taskbar_mode settings, as they exist in PCLinuxOS Openbox, default to showing the icons only from the current desktop (taskbar_mode = single_desktop). However, changing “single_desktop” to “multi_desktop” will show all of the icons of running applications on all desktops.

Given that tint2 has no pager, per se, changing the setting to “multi_desktop” mimics pager-like activity, since all of the running applications are grouped on the taskbar by desktops. Additionally, you can click and drag applications from one desktop to another, plus you can switch desktops simply by clicking your mouse on the corresponding section of the taskbar.

The taskbar_padding setting determines the horizontal_left_right padding (pixels from the horizontal edge of the taskbar), vertical padding (pixels from the top and bottom edge of the taskbar), and the horizontal spacing between items on the taskbar. The default value in PCLinuxOS Openbox is “2 3 2,” providing two pixels of horizontal padding from the horizontal edge of the taskbar, three pixels of padding between the upper and lower edge of the taskbar, and two pixels of spacing between items.

The taskbar_background_id setting determines which background to use when drawing the taskbar on your computer screen, while taskbar_active_background_id determines which background ID to use for the current desktop. If you changed the taskbar_mode to “multi_desktop,” you won’t notice any effect from the taskbar_active_background_id setting.

Tasks

The tasks section of the tint2rc file controls how each of the tasks are drawn on the taskbar. To start off, the urgent_nb_of_blink setting tells the tint2 panel how many times to blink a taskbar element when urgent attention is requested. The task_icon and task_text settings determine, respectively, if an icon or text is displayed on the taskbar item. By default, both items are displayed in PCLinuxOS Openbox. The task_centered setting determines if the task name is centered (1, the default) or not (0).

To economize space on the taskbar, it would be quite easy to mimic the KDE Smooth Tasks plasmoid by choosing to display an icon only on the taskbar by turning off the display of the text label – and altering one other setting, which we’ll mention here shortly.

The task_maximum_size determines the maximum size of the task item. It consists of two numbers, the width and height. By default in PCLinuxOS Openbox, these are set to 140 pixels wide and 35 pixels tall. If you like displaying the text on your task buttons, as is the usual case, these “measurements” typically work out fine. But if you want to mimic the KDE Smooth Tasks plasmoid’s way of displaying running tasks on your taskbar, change the task_maximum_size width to 40 pixels, turn off the text (task_text = 0), and your tint2 panel will look much like the screenshot above.

With the task_padding setting, we can control the horizontal and vertical padding of the individual task buttons displayed on the taskbar. The rest of the settings in the tasks section of the tint2rc file deal with which background ID to use to display tasks in their various states (active, urgent and iconified).

Task Icons

The task icons section tells tint2 how to display icons on the panel. The task_icon_asb setting controls all icons that don’t fall into the other designated special categories. The task_active_icon_asb setting controls how the icon of the active window is displayed. The task_urgent_icon_asb setting controls how an icon is displayed when a window is requesting your urgent attention. The task_iconified_icon_asb setting controls how the

Tint2 panel with tasks set up to mimic KDE Smooth Tasks plasmoid

icons of an iconified (minimized) window are displayed.

Each setting has three sets of numbers, separated by a space. The numbers indicate the alpha (transparency) of the icon (0 to 100), followed by the saturation (-100 to 100), then the brightness (-100 to 100).

Fonts

The fonts section specifies how tint2 displays fonts. To save space on my tint2 panel, I have the text turned off. If you do the same, none of the following settings will really have any effect. If, however, you prefer to see a more traditional panel and choose to have the text of the window's title bar displayed on your panel items, then you can customize how tint2 displays that text.

The task_font setting tells tint2 which font to use, any optional special style instructions (bold, italic or bolditalic), followed by the size of the font. The task_font_color setting sets the color of the font (specified with a "#" sign, followed by a 6 character hexidecimal color code), a space, then the opacity of the text color (0 to 100, with 100 being opaque and 0 being transparent).

The task_active_font_color sets the color and opacity of the text for the active window icon, while task_urgent_font_color and task_iconified_font_color sets the color and opacity of the text for icons of windows requiring your attention and icons of iconified windows, respectively.

The font_shadow setting (0 or 1) tells task2 whether or not to draw a shadow under the text displayed on the panel.

System Tray

As the name indicates, this section controls the system tray area of the panel. The systray setting allows you to set whether the system tray is embedded in the tint2 panel (set with a value of 1), or whether the system tray is disabled (set with a value of 0), and is not displayed at all. The systray_padding uses three sets of numbers, separated by a space, that govern the horizontal left-to-right padding, vertical padding, and horizontal spacing.

The systray_sort setting allows you to tell tint2 the ordering to use when displaying the icons of the system tray. In Openbox, the default value is "ascending," which means that the icons will be placed in alphabetical order, based on the name of the application, from A to Z, left to right. The other options are: descending (reverse alphabetical order), left2right (display icons from left to right, based on the order in which the applications are loaded), and right2left (display icons from right to left, based on the order in which the applications are loaded).

With the systray_background_id setting, you control which of the previously defined panel background definitions to use when drawing the system tray. The systray_icon_size setting sets the size, in pixels, to draw the icons on the system tray. The systray_icon_asb, as with the previously discussed task_icon_asb settings, sets the alpha, saturation and brightness levels to use when displaying the icons in the system tray.

Clock

Just as its name suggests, the Clock section of the tint2rc file controls how tint2 displays clock information on the tint2 panel. The format for the clock information displayed uses the format of strftime. Refer to the manual page at the previous link.

The time1_format setting controls the formatting for the top item displayed in the clock, while the time2_format setting controls the formatting for the bottom item displayed in the clock. By default, in PCLinuxOS Opbenbox, those correspond to the time on the top, followed by the day and date on the bottom. To eliminate either line of the clock display, simply comment out the line you don't want displayed by placing a # at the beginning of the line. Commenting out both lines will prevent the clock from being displayed at all on your tint2 panel.

The time1_font and time2_font settings allow you to specify what font, style (bold, italic or bolditalic) and fontsize to use to display your clock information. The clock_font_color allows you to set the hexidecimal font color and the opacity of the text. The clock_padding sets how many pixels, horizontal and

vertical, is padded around the clock information when it is displayed. The clock_background_id setting tells tint2 which of the predefined backgrounds to use when displaying the clock information.

The clock_rclick_command setting tells tint2 what command to execute when you right click on your clock. The default in PCLinuxOS Openbox is to open the Xfce calendar program, called Orage. However, Orage is not installed in my default installation of Openbox. It should be easy enough to install it via Synaptic, though.

One setting that is not in the default PCLinuxOS Openbox configuration of tint2 is the clock_tooltip setting. Here is the entry on my Openbox installation:

```
clock_tooltip = %r, %A, %B %d, %Y
```

With the addition of this line to the Clock section of your tint2rc file, the time, day of week, month, day and year will appear as a tooltip whenever you hover your mouse over the clock.

Tooltips

The tooltips section of the tint2rc file controls the display of the tooltips on your tint2 panel, if activated. Setting tooltips = 1 turns tooltips on, while tooltips = 0 turns them off and prevents their display.

The tooltip_padding, tooltip_background_id, tooltip_font, and tooltip_font_color settings work exactly like their counterparts in the other sections. However, there are two other settings that are unique to the display of tooltips. The tooltip_show_timeout setting sets the number of seconds of delay to show the tooltip when hovering your mouse cursor over an item on the tint2 panel. The default in PCLinuxOS Openbox is 0.7 seconds. The tooltip_hide_timeout setting sets the number of seconds before the tooltip disappears when your mouse leaves the item on the tint2 panel. The default in PCLinuxOS Openbox is 0.3 seconds.

Mouse

The mouse section of the tint2rc file contains instructions for how your tint2 panel should respond to various mouse clicks. The mouse action choices should be rather obvious by reading their description. Each of the four mouse actions can have any of the following as their setting:

```
close:              close the task

toggle:             toggle the task

iconify:            iconify the task

toggle_iconify:     toggle or iconify the
task

maximize_restore:   maximized or minimized
                    the task

desktop_left:       send the task to the
                    desktop on the left

desktop_right:      send the task to the
                    desktop on the right

next_task:          send the focus to next
                    task

prev_task:          send the focus to
                    previous task
```

The default right click action in PCLinuxOS Openbox is to close the task. Since I'm used to right clicking on a panel task to access its window menu in all the other panel applications, I found myself inadvertently closing applications that I wanted to remain open. So, I changed the right click mouse action to toggle_iconify. This way, the application remained opened, and it only changed its state from visible to iconified – or vice versa.

Battery

Unfortunately, I could not test the battery section of the tint2rc file. Whenever I tried to activate the battery notification, I received an error in my open terminal session that read:

```
ERROR: battery applet can't open
energy_now
```

After this message, tint2 exited with a segmentation fault. Instead, I'm using the Gnome Power Manager to monitor the battery states on my laptop.

Summary

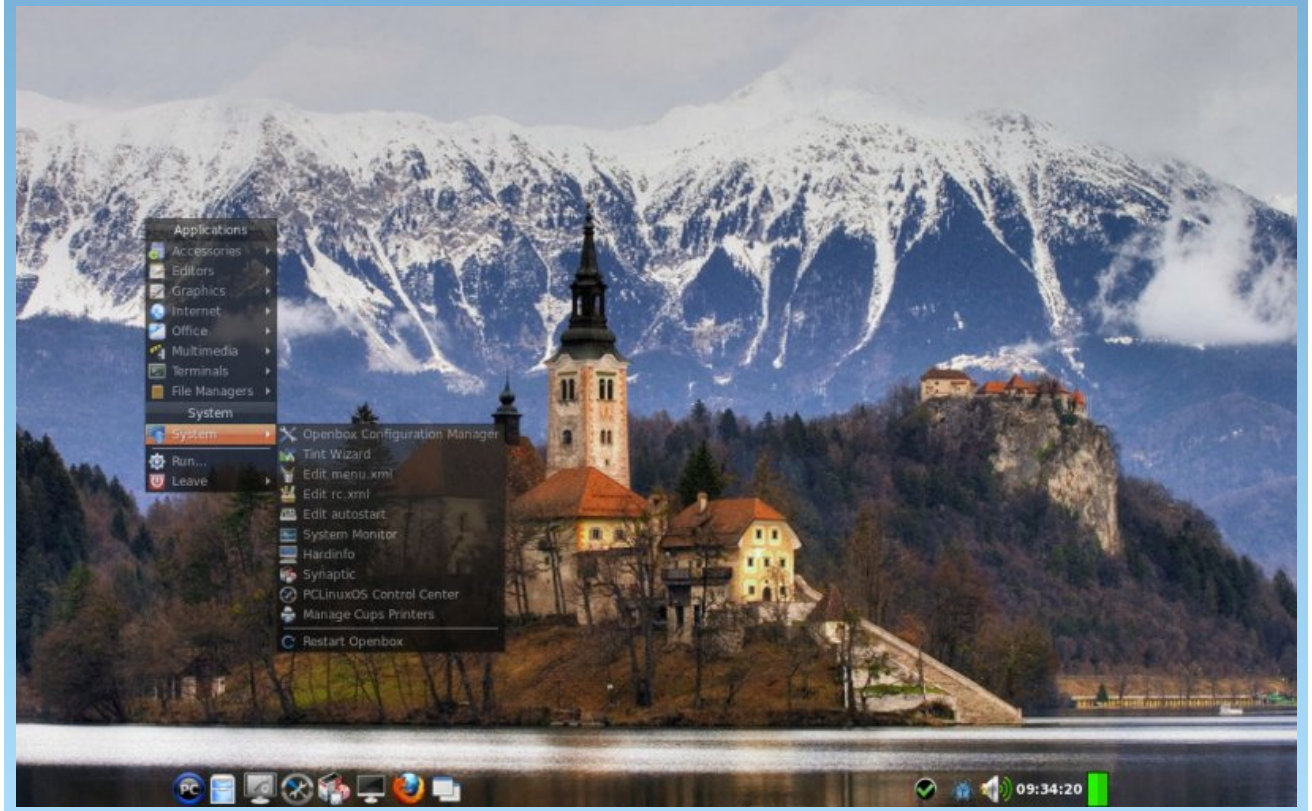
Tint2 makes a very suitable panel replacement. It's lean, mean, stable and attractive. You will find it easy to control, as well. There are reports of users running multiple panels on a desktop, and tint2 can be made to load up a custom resource file. Simply issue the command tint2 mysecondpanel.tint2rc, where mysecondpanel.tint2rc is the resource file that contains the information that controls the behavior of the second panel.

Feel free to explore all the options of tint2, by checking out its configuration page. You will find some entries that don't work with our current version of tint2 in the PCLinuxOS repository. The version in our repository is the latest stable version, and there are some instructions on the configuration page that apply only to the beta version.

The PCLinuxOS Magazine

Created with Scribus 1.3.9

Screenshot Showcase



Posted by ferry_th on January 6, 2012

Openbox: Using feh To Manage Your Wallpaper

by Darrel Johnston (djohnston)

Feh is an imlib2 based image viewer. It can also be used to manage your desktop wallpaper. It's not used for that purpose very often because most desktop environments have their own wallpaper managers. Two exceptions are Fluxbox and Openbox.

In the PCLinuxOS community remaster of Openbox, Melodie has chosen to use the PCManFM file manager to manage the wallpaper. However, if you look in the ~/.config/openbox directory, you will see the autostart.sh file which is run every time you log into an Openbox session. In the file, she has a couple of commented lines for using feh to display a wallpaper. Let's take a look at that section of the file.

a random desktop background. There can be one only, it
works with the same command line. I insist : uncomment ONE LINE ONLY !
tip : take the second one if you have several wallpapers and want a different
one at each session.
feh --bg-scale "$(find ~/.local/wallpapers -type f)" &
feh --bg-scale "$(find ~/.local/wallpapers -type f |sort -R |tail -1)" &

Let's examine what each of the two options does. The first example is the simpler one. She uses the find command to find a file in (~ = your home directory) the ~/.local/wallpapers directory. The resulting string value is read by feh. The --bg-scale parameter tells feh to scale the image to the desktop size. Possible parameters are:

--bg-center
Center the file on the background. If it is too small, it will be surrounded by a black border

--bg-fill
Like --bg-scale, but preserves aspect ratio by zooming the image until it fits. Either a horizontal or a vertical part of the image will be cut off

--bg-max
Like --bg-fill, but scale the image to the maximum size that fits the screen with black borders on one side.

--bg-scale
Fit the file into the background without repeating it, cutting off stuff or using borders. But the aspect ratio is not preserved either

--bg-tile
Tile (repeat) the image in case it is too small for the screen

Note that if you have more than one file in the ~/.local/wallpapers directory, you will get an error: "feh ERROR: Couldn't load image in order to set bg".

The second option also uses the find command to find a file in the ~/.local/wallpapers directory. But this option is for more than one wallpaper image. The list of files is passed to the sort command, and the -R parameter sorts the list in a random order. The randomly sorted list is then passed to the tail command, which finds the last file in the sorted list. The string value of the resulting file is read by feh, which scales the image to the desktop size. Using this option will give you a randomly selected wallpaper every time you login to an Openbox session.

Suppose you want to cycle the wallpaper every few minutes or hours. To rotate the wallpaper randomly, create a script with the code below (for example, wallpaper.sh). Make the script executable (chmod +x wallpaper.sh) and call it from ~/.xsession. You can also put the source directly in ~/.xsession instead of in a separate file. Change the "15m" delay as you please (see man sleep for options).

```
#!/bin/sh
while true; do
      feh --bg-scale "$(find
~/.local/wallpapers -type f |sort -R
|tail -1)" &
      sleep 15m
done
```

Another way of doing it is shown below.

```
#!/bin/sh
while true; do
      find ~/.wallpaper -type f -name
'*.jpg' -o -name '*.png' -print0 |
                  shuf -n1 -z | xargs
-0 feh --bg-scale
      sleep 15m
done
```

After feh has been run for the first time, it creates the hidden file .fehbg in your home directory. If you wish

to use the same wallpaper each time you login, you can add the line sh ~/.fehbg & to your ~/.config/openbox/autostart.sh file.

Note that if you want to use feh to manage your wallpaper in the Openbox edition, you must comment the line /usr/bin/pcmanfm --desktop & contained in the ~/.xsession file. Most of the instructions I've seen say to edit ~/.xinitrc. Commenting out the pcmanfm line in the ~/.xinitrc file did nothing for me. Also note that I'm using the openbox-bonsai-2010.11 version.

Screenshot Showcase

Posted by Ferdes Fides on December 23, 2011

Openbox: Customize Your Right Click Menu

by Paul Arnote (parnote)

Unless you are running the LXDE Panel (lxpanel) on your copy of Openbox (or even if you are running lxpanel), the “traditional” way of accessing the applications menu in Openbox is by right clicking on an empty spot of your Openbox desktop. Fortunately, it’s quite easy to tweak and tune your Openbox right click menu.

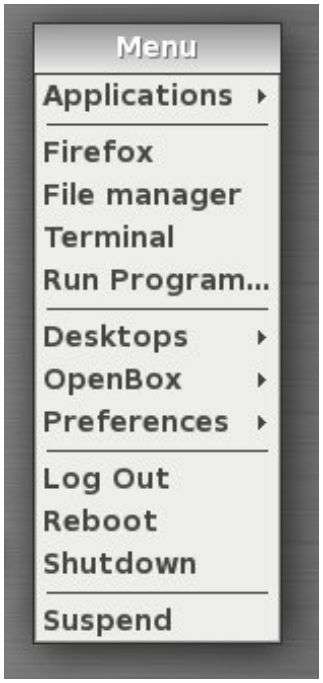
The information for the Openbox right click menu is stored in the `menu.xml` file, located in your `/home/username/.config` folder. For a brief description of the XML file format, take a look at the PCLinuxOS Magazine Openbox article on editing your rc.xml file. Use the PCLinuxOS Openbox default text editor **Geany** to edit any of your Openbox configuration files.

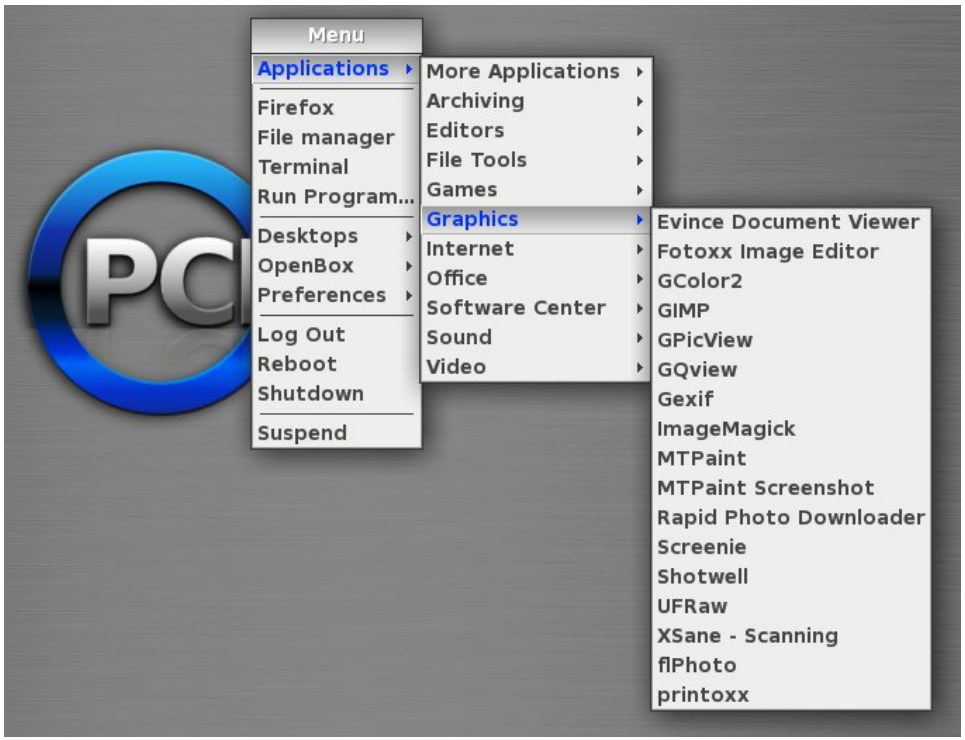
Below is a copy of the menu.xml file from my installation of Openbox:

```
<?xml version="1.0" encoding="UTF-8"?>

<openbox_menu
xmlns="http://openbox.org/3.4/menu">

    <menu id="desktop-app" label="Applications"
execute="openbox-menu -x -t 'sakura -e'" />

    <menu id="openbox-menu" label="OpenBox">
        <item label="ObConf">
            <action
name="Execute"><command>obconf</command></ac
tion>
        </item>
        <item label="Reload Openbox">
          <action name="Reconfigure" />
        </item>
    </menu>

    <menu id="preferences" label="Preferences">
        <item label="Desktop Prefs">
            <action
name="Execute"><command>pcmanfm --desktop-
pref</command></action>
        </item>
                    <item label="No effects">
                <action name="Execute">
                    <execute>

~/.config/openbox/scripts/xcompmgr.sh unset
                    </execute>
                </action>
            </item>
            <item label="Transparency">
                <action name="Execute">
                    <execute>

~/.config/openbox/scripts/xcompmgr.sh set
                    </execute>
                </action>
            </item>
            <item label="Transparency, fadings">
                <action name="Execute">
                    <execute>

~/.config/openbox/scripts/xcompmgr.sh setshaded
                    </execute>
                </action>
            </item>
            <item label="Transparency, fadings,
shadows">
                <action name="Execute">
                    <execute>
~/.config/openbox/scripts/xcompmgr.sh
setshadowshade
                    </execute>
                </action>
            </item>
    </menu>

    <menu id="root-menu" label="Openbox 3">
        <separator label="Menu" />
        <menu id="desktop-app" />
        <separator />
        <item label="Firefox">
            <action
name="Execute"><command>firefox</command></act
ion>
        </item>
        <item label="File manager">
            <action
name="Execute"><command>pcmanfm</command></
action>
        </item>
        <item label="Terminal"><action
name="Execute">
            <command>gnome-
terminal</command></action>
        </item>
        <item label="Run Program..."><action
name="Execute">
            <command>gnome-run-
dialog</command></action>
        </item>
        <separator />
        <menu id="client-list-menu" />
        <menu id="openbox-menu" />
        <menu id="preferences" />
        <separator />
        <!--<item label="Exit">
            <action name="Exit" />
        </item>-->
        <item label="Log Out">
            <action name="SessionLogout">
                <prompt>yes</prompt>
            </action>
```

```
        </item>
          <item label="Reboot">
                <action name="Execute">
                      <execute>dbus-send --system --
dest=org.freedesktop.Hal --print-reply
/org/freedesktop/Hal/devices/computer
org.freedesktop.Hal.Device.SystemPowerManagement
.Reboot</execute>
                </action>
              </item>
          <item label="Shutdown">
                <action name="Execute">
                      <execute>dbus-send --system --
dest=org.freedesktop.Hal --print-reply
/org/freedesktop/Hal/devices/computer
org.freedesktop.Hal.Device.SystemPowerManagement
.Shutdown</execute>
                </action>
              </item>
              <separator />
              <item label="Suspend">
                <action name="Execute">
                      <execute>dbus-send --system --
dest=org.freedesktop.Hal --print-reply
/org/freedesktop/Hal/devices/computer
org.freedesktop.Hal.Device.SystemPowerManagement
.Suspend int32:0</execute>
                </action>
              </item>
      </menu>
</openbox_menu>
```

Below is a the basic menu that appears when I right click on my Openbox desktop:





When I hover my mouse over the “Applications” menu, the menu expands, as shown in the screenshot (above, right):

You most likely won’t want to mess much with the “Applications” entry of your installation. This portion of the menu is controlled more by the xdg menu entries, and you can cause chaos to reign by messing around here. The real beauty of the menu.xml file is the ability to customize the rest of your Openbox menu. My menu, depicted above, has been tweaked and customized by me.

After the first separator in the menu, you have a “quick launch” area. You can define applications that you may want quick and ready access to in this area. The changes I made here was to call Gnome Terminal, instead of Sakura, which is the default terminal application in the PCLinuxOS Openbox release. Since Gnome Terminal was pre-installed on my Openbox installation, it was a simple matter of changing the line in menu.xml that executes “sakura” to “gnome-terminal.”

I further expanded the “quick launch” area of my Openbox menu by adding in the “Run Program...” entry. Openbox, as it’s installed from the Live CD, doesn’t have any way of accessing a “Run Program...” dialog box. So, I installed “gnome-run-dialog” from Synaptic, a Python/Gtk+ application that provides you a “Run Program...” dialog box.

Adding it to the “quick launch” section of my Openbox menu was as easy as adding these three lines to my menu.xml file:

```
<item label="Run Program..."><action
name="Execute">
        <command>gnome-run-
dialog</command></action>
</item>
```

To increase the usefulness of the “Run Program...” entry, I also subsequently added a keybinding to launch “gnome-run-dialog” from the keyboard, with the press of the Alt + F2 key combination, to my rc.xml file.

Getting “fancy”

You can just as easily create top level menus populated with sub-menu items. Because of various issues, I tend to switch between the use of Firefox and Chromium browsers for different tasks that I routinely perform on my computers. For example, I prefer to use Chromium for working in Google Docs (the vehicle preferred for magazine article submission). Despite its larger size, Chromium experiences fewer “freezes” than Firefox, which is a problem I’ve experienced with Firefox ever since version 3.2. However, I prefer to use Firefox for the bulk of my web browsing.

As such, I prefer to have both Chromium and Firefox listed in the “quick launch” area of the Openbox right click menu. To keep things neat and tidy (and to prevent the “quick launch” area from becoming too long and unweildly), I prefer to have a top level menu that, when selected, allows me to choose between Firefox and Chromium. It’s easier than you might think. Here’s the new “snippet” from my altered menu.xml file, inserted right after the line that reads **`<menu id="desktop-app" />`**:

```
<menu id="browsers" label="Browsers">
      <item label="Firefox">
   <action
name="Execute"><command>firefox</command></act
ion>
      </item>
        <item label="Chromium">
   <action name="Execute"><command>chromium-
browser</command></action>
      </item>
</menu>
```

To start with, you define a new “top level” menu with the <menu …> tag. Be sure to give it a unique menu ID. Here, I called mine “browsers,” which I knew was not already in use. What appears after “label=” is the text that appears in the menu.

Next, add the menu items that you want to populate that top level menu, as I have done with the “item” entries for Firefox and Chromium. Finally, close out the top level menu with the </menu> tag.

If you want to get even fancier – and more complex – you can further define sub menus for top level menus, with each sub menu containing additional sub menus and menu items. How fancy and just how complex you want to make your menu additions is entirely up to you. However, I subscribe to the K.I.S.S. principle (Keep It Super Simple), and this is as complex as I want to get.

Summary

You will find that editing your Openbox menu.xml file will add extra functionality to your Openbox installation. Editing, changing and expanding your Openbox menu is very easy to do, and gives you the opportunity to further customize Openbox to work in a manner that more closely matches the way that you work with your computer.

Openbox Live CDs: A Comparison

by Meemaw

"Which Openbox Live CD should I use? What are the differences?"

In this article we'll explore the differences between the full Openbox Live CD and the Openbox-Bonsai mini Live CD. Just like the other 'mini' versions of PCLinuxOS, Openbox-Bonsai is a smaller Live CD, with only a few needed programs to get you started. You can pick and choose what programs you want and not have to use what someone else has chosen. For this article, we will use the most recent official releases, Openbox-Bonsai-2011.03 and Openbox-2010.11.

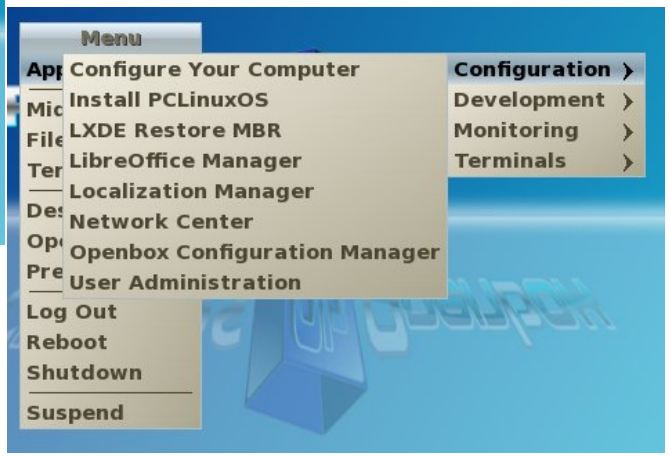
Bonsai

Bonsai is the minimum version of Openbox. The ISO is 266 MB, and contains just a bare minimum of programs to get you started in the Openbox experience. The rest you can pick and choose for yourself, via Synaptic. Programs on the Live CD/default installation include:

Midori and Firefox web browsers
flPhoto Photo Viewer
LibreOffice Manager
PCManFM File Manager
Geany (Text Editor)
File Roller (Archiver)
HTop system monitor
NetApplet
Sakura and XTerm Terminals

The current official version of Bonsai includes lxpanel as the default panel. Rumor has it that in the upcoming release of Bonsai, it will use tint2 as the default panel.

You can see here the menu structure and the choices that Bonsai offers in the 'More Applications' section.



Openbox Full

The full-size Live CD of Openbox is 685 MB and, naturally, includes loads more programs.

In the Internet section, programs include:

Firefox and Midori web browsers
Sylpheed
Gajim and XChat
Deluge
Pino
Filezilla
Transmission

In the Graphics section you see:

Fotoxx
Gimp
Rapid Photo Downloader
flPhoto, GQView, GPicView
Screenie
XSane

The Office section contains:

OpenOffice Manager
Galculator
Gnumeric
Abiword
Sunbird and Osmo Organizers
Evince Document Viewer

For file management, editing and monitoring you will see:

PCManFM and Thunar File Managers

Geany Text Editor
HTop
EeeControl
NetApplet
System Info
Gnome, Root, Sakura, Urxvt and XTerm Terminals
XArchiver
Gnomebaker CD Burning program

There is only one program in the Audio section:

Audacious

The Video section includes two:

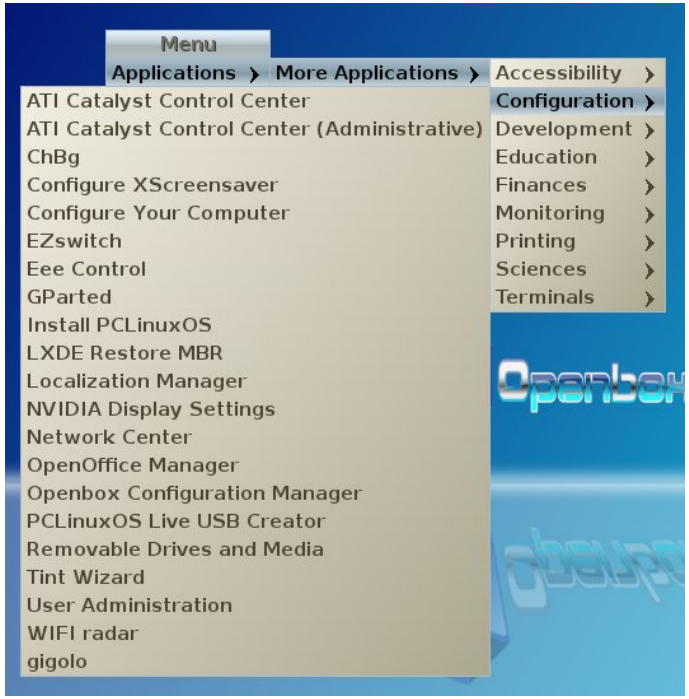
Coriander
SMPlayer

You also have extras:

Stellarium
Money Manager Ex
5 or 6 Games

The current full version of Openbox uses tint2 as the panel. Lxpanel is scheduled to replace tint2 as the default panel in the forthcoming new version.

You can see here the menu structure and the choices that the full version of Openbox offers in the 'More Applications' section. Notice that it includes the Printing section which is not in Bonsai. Stellarium is accessible in the Education as well as the Sciences section, and that the Finances section is where you will find Money Manager Ex. Also, the configuration contains many more programs than are on the Bonsai iso (top center).



New versions of Openbox and Openbox-Bonsai are in the works now, and should be released soon. You know the PCLinuxOS mantra: It'll be released when it's ready.

Disclaimer

1. All the contents of the NEW PCLinuxOS Magazine are only for general information and/or use. Such contents do not constitute advice and should not be relied upon in making (or refraining from making) any decision. Any specific advice or replies to queries in any part of the magazine is/are the person opinion of such experts/consultants/persons and are not subscribed to by the NEW PCLinuxOS Magazine.
2. The information in the NEW PCLinuxOS Magazine is provided on an "AS IS" basis, and all warranties, expressed or implied of any kind, regarding any matter pertaining to any information, advice or replies are disclaimed and excluded.
3. The NEW PCLinuxOS Magazine and its associates shall not be liable, at any time, for damages (including, but not limited to, without limitation, damages of any kind) arising in contract, rot or otherwise, from the use of or inability to use the magazine, or any of its contents, or from any action taken (or refrained from being taken) as a result of using the magazine or any such contents or for any failure of performance, error, omission, interruption, deletion, defect, delay in operation or transmission, computer virus, communications line failure, theft or destruction or unauthorized access to, alteration of, or use of information contained on the magazine.
4. No representations, warranties or guarantees whatsoever are made as to the accuracy, adequacy, reliability, completeness, suitability, or applicability of the information to a particular situation.
5. Certain links on the magazine lead to resources located on servers maintained by third parties over whom the NEW PCLinuxOS Magazine has no control or connection, business or otherwise. These sites are external to the NEW PCLinuxOS Magazine and by visiting these, you are doing so of your own accord and assume all responsibility and liability for such action.

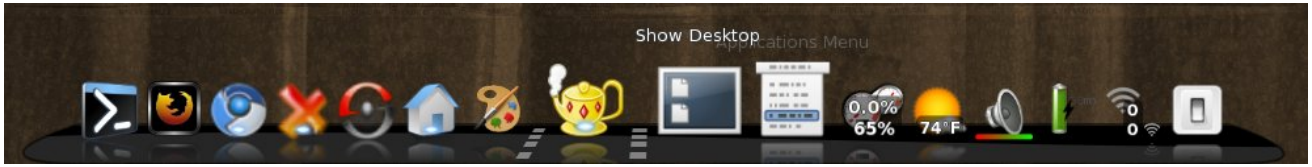
Material Submitted by Users
A majority of sections in the magazine contain materials submitted by users. The NEW PCLinuxOS Magazine accepts no responsibility for the content, accuracy, conformity to applicable laws of such material.

Entire Agreement
These terms constitute the entire agreement between the parties with respect to the subject matter hereof and supersedes and replaces all prior or contemporaneous understandings or agreements, written or oral, regarding such subject matter.

Openbox: Add A Quick Launch Bar

by Paul Arnote (parnote)

My customized Cairo-Dock without a mouse over.



My customized Cairo-Dock with the mouse hovering over one of the icons.

Since Openbox doesn't have a panel of its own (borrowing lxpanel from LXDE or using Tint2 instead), it almost begs for us to use one of the quick launch bars that are out there. Plus, if you are using Tint2 as your panel, it does not currently allow launchers to be used.

Fortunately, there are choices under PCLinuxOS for a launch bar for your Openbox desktop. All are relatively lightweight, and give your desktop some flash and panache. All mimic (to varying degrees) the Mac OS-X actions of "zooming" when you mouse over the individual icons.

What are they called? Well, they are **adeskbar, wbar** and **Cairo-Dock**. Of the three, adeskbar is the lightest weight launch bar, weighing in at only 472 KB, while wbar fills in the middle, weighing in at 842 KB. Cairo-Dock, on the other hand, weighs in at 7.2 MB, with another 8.5 MB for the required plug-ins, and taking up another 14.1 MB for the optional themes. As you might imagine, Cairo-Dock offers the flashiest effects on your desktop. Your choice will be somewhat dictated by how fancy you want your launch bar to be, how fast your computer is, how much RAM you have, and how much hard drive space you have. Obviously, if neither of those are of any concern to you, then the choice falls strictly in the arena of aesthetics and personal preference.

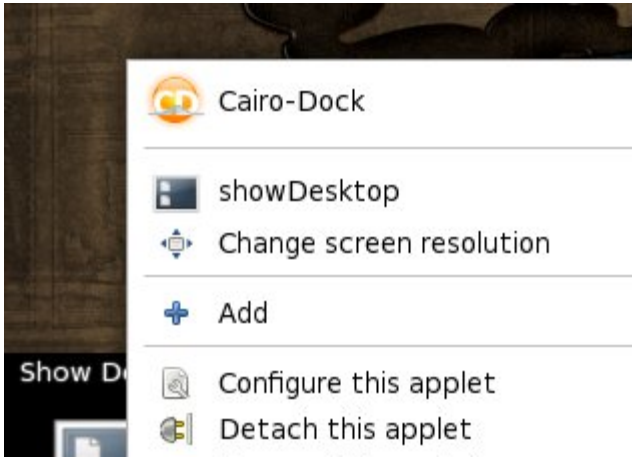
Cairo-Dock

The "flashiest" of the three, by far, is Cairo-Dock. Just looking at it, you get the impression that there is more to it. With the default installation of Cairo-Dock from Synaptic, you get reflections of the icons on the bar beneath them. When you put it "into motion," by moving your mouse over the individual icons, the icons "zoom," but can just as easily be made to rotate and do other "tricks." Being the largest download of the three launch bars, Cairo-Dock is also the most capable.

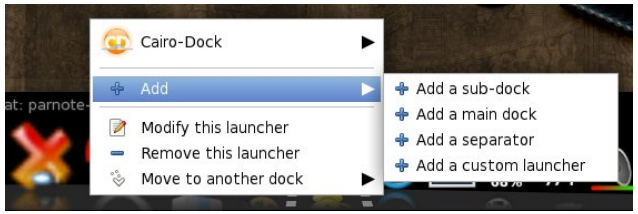
On my IBM Thinkpad T23 with Openbox installed (Pentium III, 512 MB RAM, 8 MB video RAM with no OpenGL capabilities), Cairo-Dock consumes very few resources, despite being the largest in file size. Even while activated by moving my mouse over the launch bar, Cairo-Dock consumed no more than 9% of the CPU and no more than 8% of my RAM.

Cairo-Dock is divided up into sections. The left-most section contains your launchers. The middle section (between the dotted lines) contains the icons of your minimized applications. The right-most section contains the icons of your Cairo-Dock plug-ins.

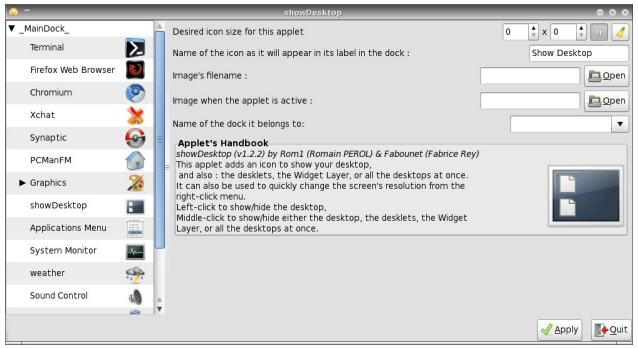
Arranging your icons on Cairo-Dock is as simple as dragging and dropping the icons to where you want them, with one caveat: plug-ins cannot co-mingle with launchers, and launchers cannot co-mingle with plug-ins.



When I first installed Cairo-Dock from Synaptic, it contained a host of Gnome-specific applications that I do not run on Openbox. Fortunately, it's as easy as right clicking on the icon you do not want, and selecting "Remove this launcher" or "Remove this applet."

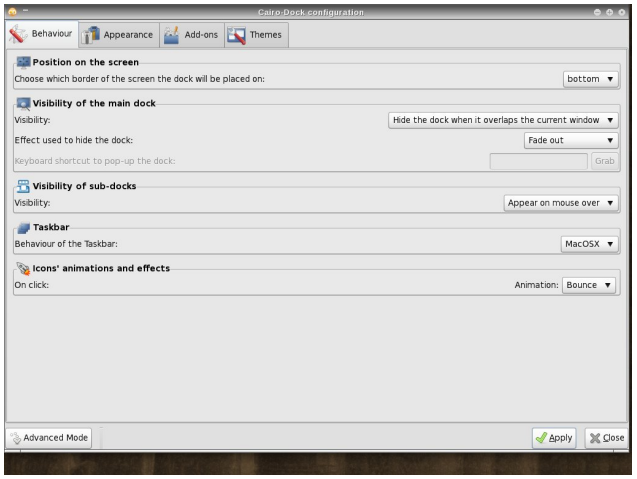


Similarly, it's just as easy to add a launcher or applet. To add a launcher, right click on the left side of the Cairo-Dock, go to the "Add" menu, then select "Add a custom launcher" from the menu. A new icon will be placed in the launcher area.



Right click on the new icon, then "Modify this launcher." You will then be able to give the new launcher a name, specify the command you want to run, along with the icon you want to use to represent your new launcher.

When it comes to adding applets to Cairo-Dock, it is almost as easy. Right click anywhere on your Cairo-Dock, select the Cairo-Dock menu item, then "Configure." You will see the window above. Select the "Add-ons" tab, and select the applets you want to display on your Cairo-Dock.



Once you've added the applet, you can right click on it and select "Configure this applet." When you do, the window shown two screenshots ago will be displayed. There, you can set display options for the selected applet.

As you can see in the previous screen shot, there are lots of options for Cairo-Dock, divided into four tabs. Feel free to explore the options. After all, you can always change them back if you don't like them. To make it easier to remember what you changed, it might be prudent to only change one thing at a time, just in case you don't like that particular selection.

wbar

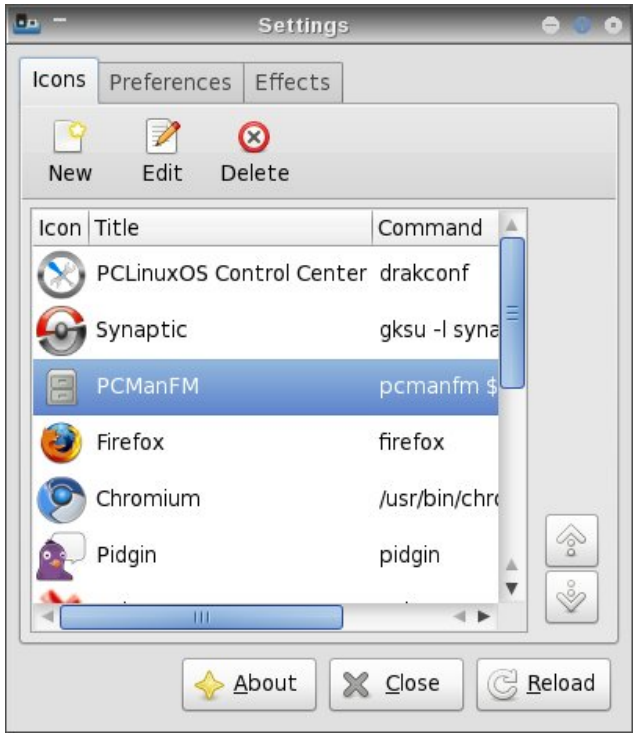
With wbar, you get a Mac OS-X like launch bar that occupies a minimum of space, while remaining relatively light on resources and disk space. While wbar does not have all the options that Cairo-Dock has, lacking additional plug-in applets that add some additional functionality, it's still a quite capable launch bar. It does one thing, and does it well.

On my Thinkpad T23 running Openbox, wbar doesn't even show up in the list of applications when I run the top command in a terminal window. When I mouse over wbar, it then shows up in the list of applications in the top command, consuming approximately 12% of the CPU and approximately 10% of the available memory, a bit more than Cairo-Dock, despite its smaller file size.

Like with Cairo-Dock, wbar came set up out of the box for some Gnome applications that I never use. Configuring wbar is relatively easy, and it comes with the configuration tool included on wbar. In fact, it's the sixth icon from the right in the screen shot below.

Clicking on the configuration icon will bring up the screen shot (next page). Notice the three tabs at the top of the dialog box.

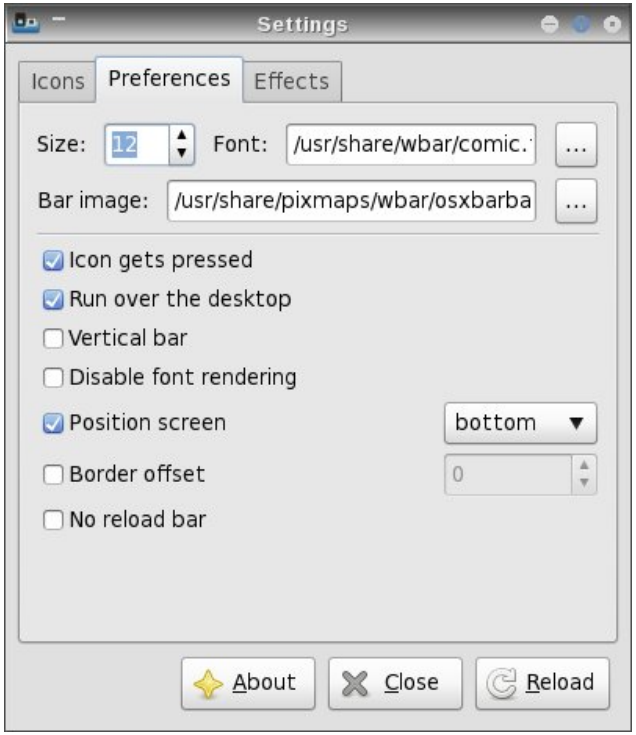
My customized wbar with the icons zooming from a mouse over.



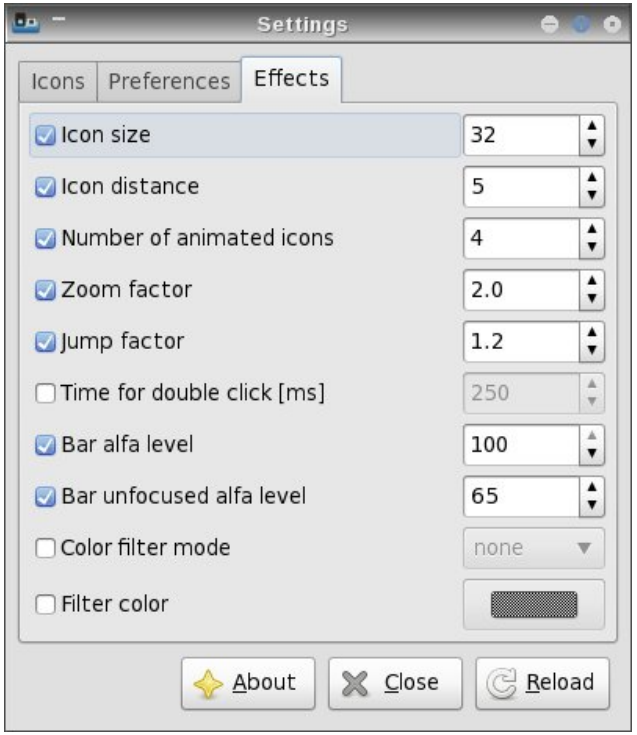
To add a new icon toyo ur wbar, click on the "New" button. Provide a title for your new launcher, the command to execute, and the icon file to use to represent your new launcher. If you're not sure where icon files are stored on your system, they are typically found in /usr/share/icons and /usr/share/pixmaps. There are even some that come with wbar, found in the /usr/share/pixmaps/wbar directory.

When you select the "Edit" button, the same dialog box is displayed, with the current information already filled into the fields. Simply make your changes and select the "Accept" button. Then, select the "Reload" button in the configuration dialog box to make your new (or edited) launcher visible.

You need to be aware, however, of one limitation of wbar when selecting your icons. Currently, wbar cannot display *.svg icon files. Rather, it can only display *.png icon files.



Under the "Preferences" tab, you can set the font and font size you want wbar to use when it displays the text of the icon when you mouse over, as well as the background image you want wbar to use. Among other things, you can also set wbar's screen position.



Under the "Effects" tab, you can set the icon size, the spacing between the icons, the zoom factor (2.0 is double size), the jump factor (the higher the number, the more "elevated" the icon is above the

others when you mouse over the icon), and a number of other items.

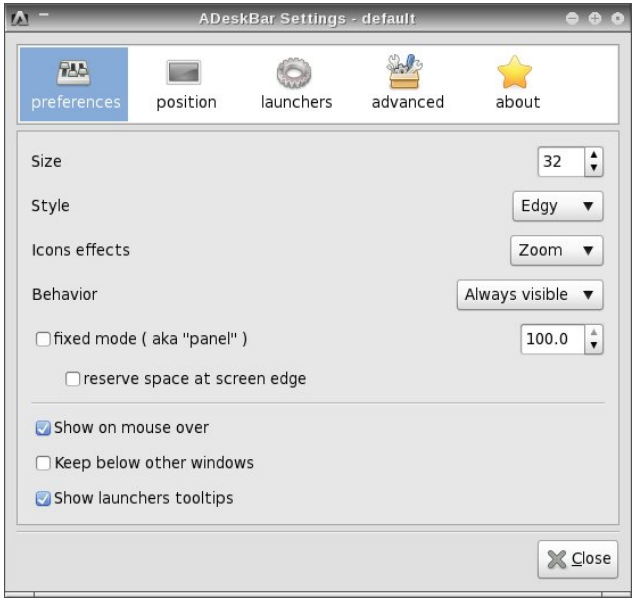
Whenever you make any changes, be sure to hit the "Reload" button to make your changes take effect in wbar.

Adeskbar

Adeskbar not only has the smallest file size, but it's also the lightest when it comes to using computer resources. Nothing I did could get adeskbar to show up in the list of applications when I ran the top command in a terminal. It didn't show up when it was idling, and it did not show up when I moused over the icons, either.

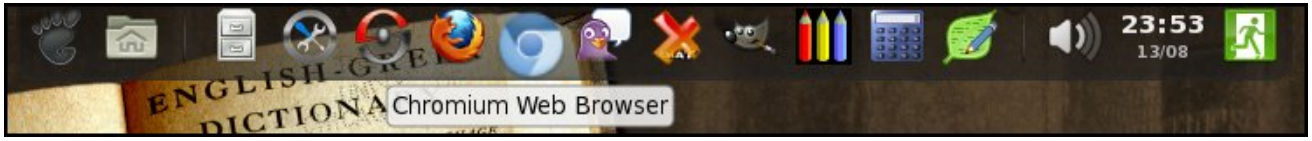
To be perfectly honest, I originally wasn't going to include adeskbar in this article. The first time I launched it from a terminal session, it was hidden behind Cairo-Dock, which I had forgotten to stop before launching adeskbar. So, I had mistakenly thought that it was not running. Mea culpa. It wasn't until I moved my tint2 panel to the top of my screen and I was running wbar that I decided to give it another try. Imagine my surprise when adeskbar appeared!

When you first launch adeskbar, it looks a bit sparse. Only the menu, audio volume control, clock, a pair of separators and the session control objects are present. It is between the separators where you will want to place your launcher icons.



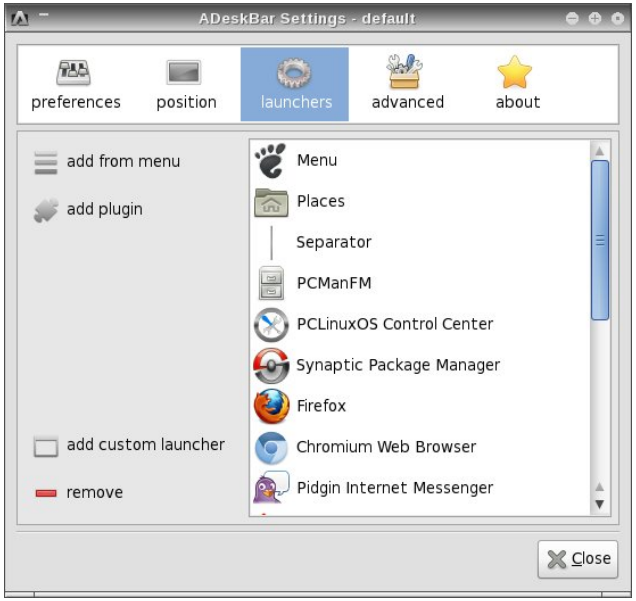
Configuring adeskbar is quite easy. Simply right click on the launch bar and select "Preferences." You will see the screenshot above.

Under the first tab, "Preferences," you can set the size of the launch bar, the "style" used to display adeskbar, what icon effects you want to use, whether it's always visible or if it autohides, and a few other settings which should be fairly self-explanatory.



Adeskbar running at the top of my screen.

Under the "Position" tab, you can select where on your screen you want adeskbar to appear.
Under the "Launchers" tab, you can start to fill in the launchers you want to include on your launch bar.



The easiest way to add applications to your launch bar to to select them from the "Add from menu" button. However, due to a problem that adeskbar has displaying some of the submenu entries on your system, everything you want to add to your launch bar may not be available.

In that event, you will need to click on the "Add custom launcher" button, and fill in the fields in the dialog box shown on the next page. The "PCLinuxOS Control Center" entry in the screen shot was added this way.

To select the icon you want to display, simply click on the large button on the left side of the dialog box, and travel to the location where your icon is stored. Unlike with wbar though, adeskbar appears to be perfectly capable of displaying either *.svg or *.png icon files.

Adding plug ins to adeskbar is even easier. Simply click on the “Add plug ins” button, and select the plug in that you want to add. I will caution you, however, that not all of the plug ins will work. Your “clue” that the selected plug in will not work will be that the plug in’s icon will not immediately appear in the adeskbar launch bar.

Under the “Advanced” tab, you can set some of the finer aspects of how adeskbar is displayed on your screen. Feel free to play with the settings (preferably one at a time) to customize adeskbar on your computer. One thing I noticed is that you cannot set the “zoom” level for the icons on your adeskbar to more than 1.30, or 130%, of the icon’s original size.

Also, you can change the menu icon by double clicking on the default Gnome “footprint” icon in the “Launchers” tab, and changing it to something you might like better in the dialog box that appears. (Hint: click on the Gnome icon on the large button on the left and choose your new icon).

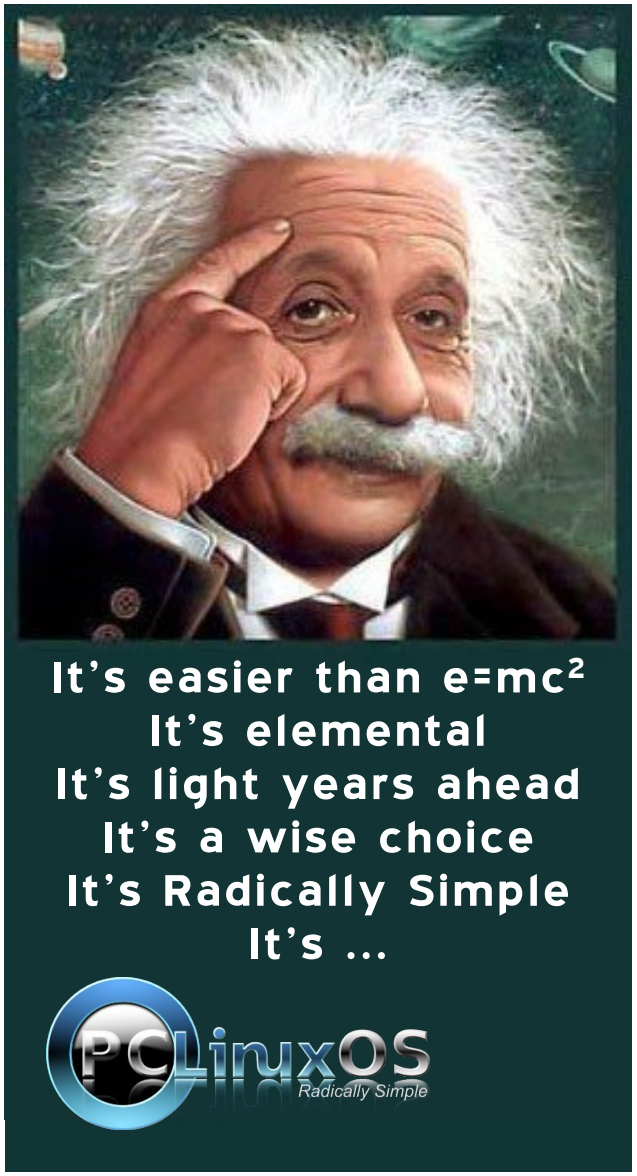
Summary

As you can see, there are three excellent choices in the PCLinuxOS repository for adding a launch bar to your Openbox installation. In fact, all three of these can be added to any desktop, regardless if it’s Gnome, KDE, Xfce, LXDE, or any of the other graphical desktops.

The adesktop launch bar is the obvious choice if you are concerned about computer resources and the amount of disk space consumed – and if you can tolerate its limitations. The wbar launch bar represents a good compromise between the low resource usage of adesktop and the relatively large hard drive space required by Cairo-Dock. The Cairo-Dock launch bar represents the pinnacle of “flash” for your launch bar, if you want the flashiest launch bar on your desktop.

All three can be set to start up automatically simply by adding their command to the Openbox autostart.sh file, as we covered in the July issue of The PCLinuxOS Magazine.

So what are you waiting for? Adding a launch bar to your desktop is fun, and it puts your most commonly used applications only a mouse click away. Plus, it’ll dazzle your friends when they see it in action.



Openbox: Customize Your Window Themes

by Paul Arnote (parnote)

One nice thing about many of the Linux desktop environments and window managers is the ability to customize the appearance of your desktop. This doesn't mean only wallpaper. This also includes window appearance. Openbox is no different in this aspect. Additionally, the information in this article applies equally to LXDE, since it uses Openbox as its window manager.

While it's possible to create your own Openbox theme from scratch by hand (obviously, since ***someone*** had to have created the first Openbox theme), you will find it easier to find a theme that you like and modify it to suit your individual tastes. The latter is the path we'll take with this article. Typically, you install and change your Openbox themes through the OBConf utility. However, you can also manually install them, provided you place them in the proper location. You can find many ready-to-go Openbox themes from other Openbox users.

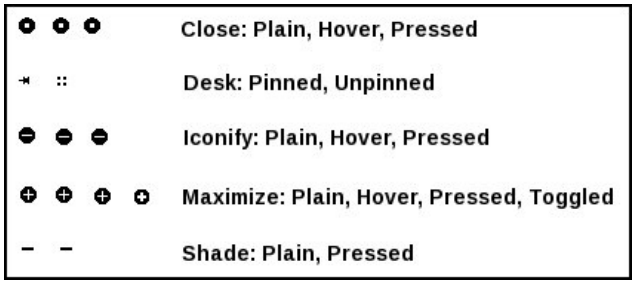
If you choose to modify one of your favorite themes, I strongly urge you to make a copy of the theme and make your modifications to the copy. This way, if you royally screw up the modification, the original remains unaltered and intact.

To get started, it's important to know where Openbox themes are stored. Your themes can by "system-wide" themes (accessible to all users on a computer) or user-specific themes (accessible to only that particular user). System-wide themes are stored in **`/usr/share/themes`**, and you will need root access to make modifications. User-specific themes are stored in either **`~/.local/themes`** or **`~/.themes`**. The advantage to these themes is that you can modify them without having root access. The disadvantage is that they are available only to one particular user, unless you copy the theme to **`/usr/share/themes`** for all users to access. To do the latter, you will need root access. I tend to use the user-specific location, and if I come up with something that I like, I can then move them to the folder for system-wide themes, sharing my new theme with all the users on my computer.

Much of what controls the appearance of a theme in Openbox is stored in an X resource database file, called **themerc**. It's no more than a specially formatted text file. Typically, this file is stored in the themes folder, which takes on the form of another folder with the name of the theme, containing another folder named openbox-3. So, the "Appleish" themerc file, the theme upon which I based my modifications off of, is stored in **`/usr/share/themes/Appleish/openbox-3`**. The other files in the folder are the graphics files that form the window decorations.

Most likely a byproduct of its light weight, the choice of graphics format used to create the window decorations in Openbox imposes some inherent limitations. The graphics have to be in the *.xbm format, which is a binary color format. In case that's not striking you just right, let me put it this way: you can use any colors you want, so long as they are black or white. Essentially, the *.xbm file acts like a mask on the window title bar, and the instructions in the themerc file tell Openbox how to paint that mask and with what color. Face it: there aren't a lot of things you can do with a binary color graphic file format.

Below is a collection of all the graphic files in the Appleish theme, with labels:



Pretty plain, huh? As you might already be able to tell, there aren't a lot of options, and this somewhat restricts what you can do in an Openbox theme. Conversely, xfwm (the window manager for Xfce) allows the use of *.xpm files. Xpm files allow the use of color, giving you many, many more options in the appearance of your window decorations.

Below is the unaltered Appleish Openbox theme, showing an active and inactive window:

In the above image, I held the cursor over the exit button on the window title bar (putting the window decoration into the "hover" state), so you can see how the image is painted by the themerc file.

Below is my altered version of Appleish:

Again, I held the cursor over the close button of the active window ("hovered") so you can see how the themerc file has been altered to paint the window decorations in the customized version. Notice how the close button is now painted yellow (I can't help myself, since yellow has always been my favorite color), instead of dark gray.

Other differences you might notice include a darkening of the window titlebar, a change of the active titlebar font color to white, and the inclusion of the "raised" drawing flag for the window titlebar. In the altered Appleish theme, I left the inactive window settings unchanged from the original.

Remember that all of the painting of the window titlebars, the titlebar fonts, and the colors to use for many other settings, are under the control of the themerc file for the particular theme you are using. Although colors can be expressed as names of colors (as recognized by Xorg) and RGB:xx:xx:xx format, the most common format for specifying colors is the six digit hexidecimal color notation that most associate with how you express colors in an HTML file. Black becomes #000000, bright red is #ff0000, bright green is #00ff00, bright blue is #0000ff, and white is #ffffff. Other colors can be created by creating combinations of the three color intensities. Just be sure to keep each digit in the range of 0 to F.

Fortunately, all of the available options for the the themerc file are documented fully and extremely well in the Openbox Wiki. Instead of trying to cover them here, I'm going to refer you to this excellent resource instead. They have already done an outstanding job of explaining all the relevant information there.

Summary

A quick, cursory look at a themerc file may be enough to scare some users away from customizing their Openbox themes. But given the fact that the themerc file is little more than a specially formatted text file, and the additional fact that the themerc file options are so well documented, you owe it to yourself to at least give it a try.

Granted, while my first attempts at modifying my Appleish theme were quite horrendous in appearance, I quickly got a handle on what I needed to do and ended up making a theme modification that is uniquely mine, and one that suits me even more than the original Appleish theme.

Openbox: Use Pipe Menus For More Functionality

by Paul Arnote

We've already covered how to customize your Openbox right click menu in the August issue of The PCLinuxOS Magazine. However, you can further increase the functionality of your Openbox menu by using what's known as "pipe menus." Pipe menus are menus that activate an external script, and the information is dynamically displayed in your Openbox menu.

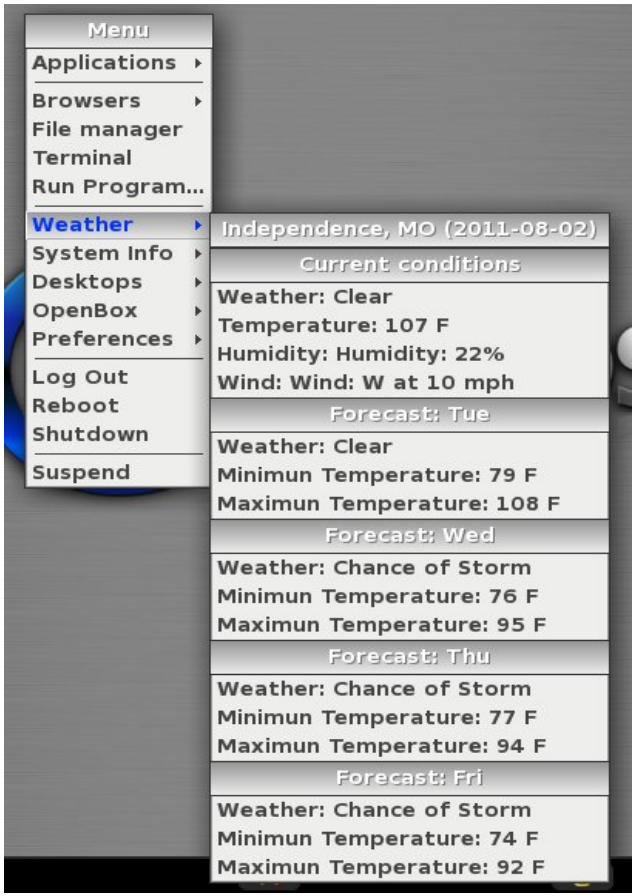
Pipe menus work fairly simply. First, you write a script (a bash script, a python script, etc.) that performs the "work." You then modify your ~/.config/openbox/menu.xml file to display the dynamic menu. As daunting as it may sound, it's actually easier than you may think.

To get started, you need to find a script to control your Openbox pipe menu – or write your own. If you choose to use one that someone else has already created, then your task will be quite a bit simpler. Fortunately, there are several "collections" of Openbox pipe menu scripts scattered around the web. One place you will definitely want to check out serves as, more or less, a central "clearing house" for all pipe menu scripts. That place is the pipes menu page on the Openbox Wiki.

If, however, you want to write your own custom script, the Openbox Wiki also has a page that details what you need to include in your script. Of course, you may want to take a look at the examples in the first link. I know that, for me anyway, it's so much easier to see an example of how to do it, in conjunction with the "technical directions" on some web page.

Putting Pipe Menus To Work

Take a look at this screen shot (below), that displays your local weather forecast:



To get the weather forecast to appear in your Openbox menu, here's what you need to do. First, go grab the python script. This particular version of the weather forecast script uses weather information from Google. If you prefer to use the weather information from Yahoo, you can grab a different python script. There is yet a third, different weather script, displaying information from weather.com. The setup steps for that script are very similar to the steps that follow for the Google and Yahoo weather information sources.

All of the scripts will give you similar information. The Google source will give you a four day forecast (today's, plus the next three days), while the Yahoo source will give you only a two day forecast (today's and tomorrow's).

I copied the script(s) into Geany, and saved them in ~/.config/weather. Next, go into the directory and mark the file as executable. In PCManFM, if you right click on the file and choose "Properties" from the context menu, select the second tab and place a check mark in the "Make the file executable" option. To avoid confusion, I saved the Google weather script as gweather.py, and the Yahoo weather script as yweather.py.

Next, you need to add a line to your ~/.config/openbox/menu.xml file, so that the menu displays in your Openbox menu. For the Google weather script, I added the following line (all on one line):

```
<menu id="pipe-weather" label="Google
Weather" execute="python
~/.config/weather/gweather.py 64052 en" />
```

For the Yahoo weather script, I added the following line (again, all on one line):

```
<menu id="yahoo-weather" label="Yahoo
Weather" execute="python
~/.config/weather/yweather.py 64052
Fahrenheit" />
```

As you can see, the options for both differ a little bit. For the Google weather script, you include the city code for your area (in the U.S., that's your ZIP code) and the language you want to use to display the information (in my case, "en" for English). For the Yahoo weather script, you include the city code for your area, along with the measurement units you want to use for displaying the temperature (Fahrenheit for the U.S., and Celsius for most everywhere else).

After editing your ~/.config/openbox/menu.xml file, right click your mouse on an empty spot on your desktop, and select the Openbox > Reload Openbox menu item to load your new menu into the Openbox menu. On subsequent reboots, this step will not be necessary.

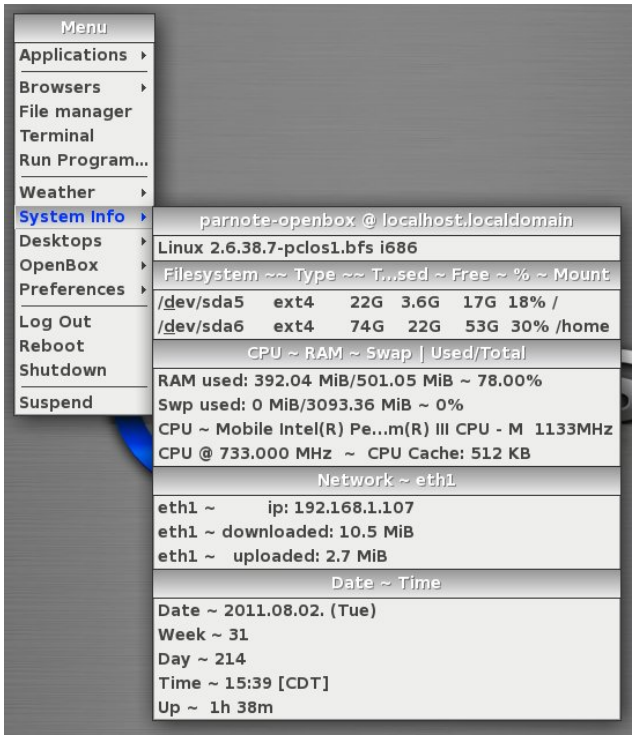
One caveat about the Yahoo weather script, however, is in order. The script is set up to cache the data from Yahoo, so that repeated access to the script doesn't keep retrieving data from Yahoo. The time length for the cache is set to six hours, meaning that despite how many times you access the Yahoo weather script during that time frame, you will be viewing the cached data. Near the top of the script (line 27), look for the entry named "CACHE_HOURS." Change the "6" to "1," and now the cached data will expire after one hour. This means that repeated attempts to access the Yahoo weather data will be refreshed if the data is more than one hour old. Accessing the weather data in less than the one hour time frame will result in the cached data being displayed.

More than just weather reports

Of course, you can do more than just display weather information on demand via the Openbox menu. Another one that I found useful is called "sysinfo."

As you can see by the screenshot (top of next column), sysinfo provides lots of information about your computer system. This information includes the current kernel you are using, information about your drive partitions, data about RAM usage, swap file usage and CPU usage, information about your network connection, as well as time and date information.

To use this on your Openbox installation, first go grab the bash script that controls the display of this information. You may need to edit the bash script so that the information displayed reflects your computer and its hardware options. For example, I had to edit the bash script to display the proper hard drives for my system, as well as the network information.



Again, I copied the script into Geany, saved it at ~/.config/sysinfo as sysinfo.sh, and made the file executable. Next, I placed the following line in my ~/.config/openbox/menu.xml file (again, all on one line):

```
<menu id="sysinfo" label="System Info"
execute="~/.config/sysinfo/sysinfo.sh" />
```

Reload Openbox, via the Openbox > Reload Openbox menu item on your Openbox menu to access your new menu item.

Having some more fun

There are more items you may want to add to your Openbox menu. One pipe menu script adds RSS news feeds to your Openbox menu. Another checks your email, from the Openbox menu. Another displays a calendar and the current time. Yet others control playback of sound files, change wallpapers, and much more. Refer to the Openbox Wiki “clearing house” for a full list of pre-made Openbox pipe menus.

Summary

Basically, anything you can script can be formatted to work with Openbox’s pipe menus. This is where your custom scripting skills can help to truly make your Openbox experience unique.

If you want to read more about Openbox pipe menus, check out the TechRepublic articles that appeared at the end of July and in early August. This article had been planned since before we ever started doing Openbox articles, back when we were in the planning stages for the series of articles on Openbox. The TechRepublic articles help provide even more resources for those interested in learning more about Openbox’s pipe menus.

You can make the use of pipe menus as easy or as complex as you like. But use them you should, since they help provide a more complete, more customized user experience.

Screenshot Showcase

Posted by coffeetime on November 4, 2011

Openbox: Tips & Tricks

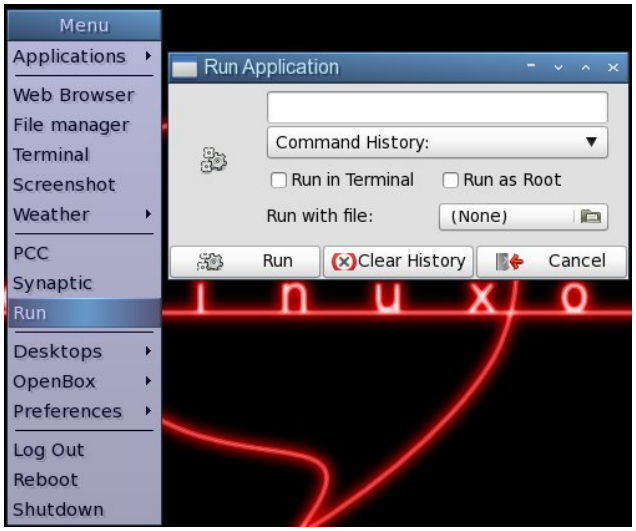
by Darrel Johnston (djohnston)
& Paul Arnote (parnote)

Add a run dialog to the Openbox menu

Open Synaptic and install the gnome-run-dialog package. Once that is accomplished, open the ~/.config/openbox/menu.xml file in a text editor. Add a section like the one shown below.

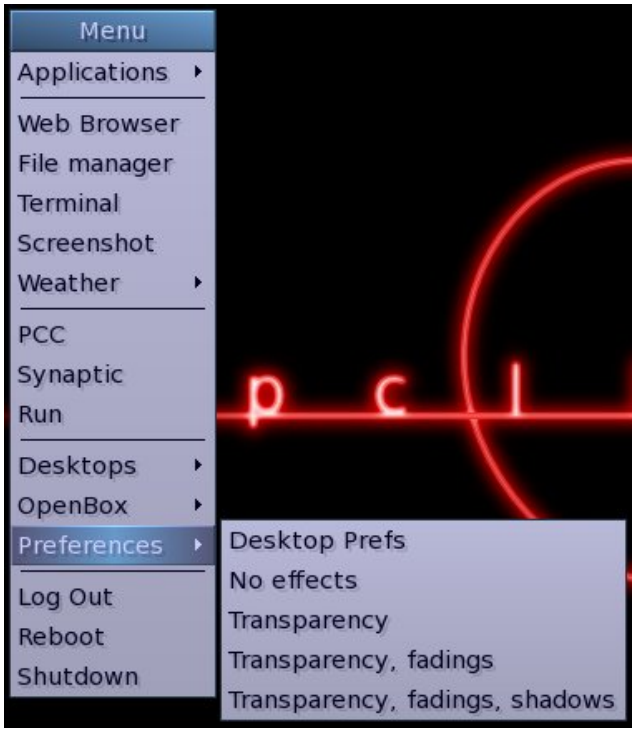
```
<item label="Run">
   <action name="Execute">
      <execute>gnome-run-dialog</execute>
   </action>
</item>
```

The item label is what we want shown in the Openbox menu. gnome-run-dialog is the program we want executed when we click on Run in the Openbox menu. Once you have edited and saved the menu.xml file, you can freshen the Openbox menu by logging out, rebooting, or clicking on Reconfigure or Restart in the submenu of the OpenBox label. Once you click on Run in the menu, the gnome-run-dialog window will appear.



Turn off fades and shadows to speed things up

In the Bonsai version of the PCLinuxOS Openbox edition, xcompmgr is disabled. We can enable it at login by opening the ~/.config/openbox/autostart.sh file and uncommenting the line #xcompmgr & by removing the # sign at the beginning of the line. However, this is unnecessary, as the composite manager effects can be turned on and off from the Openbox menu.

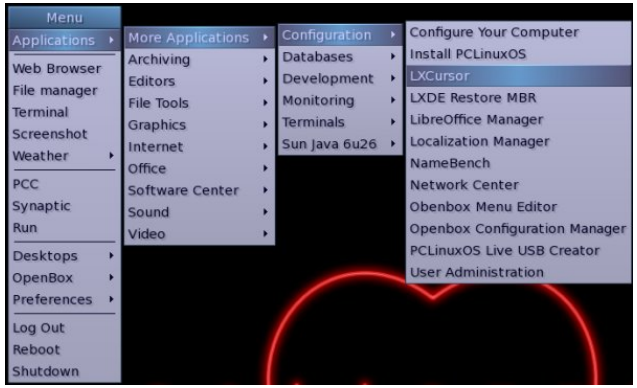


If the effects are off, selecting any of the Transparency menu items will turn the composite manager on. Doing so will automatically uncomment the xcompmgr & line in the autostart.sh file by executing one of the ~/.config/openbox/scripts/xcompmgr.sh options, as defined in the menu.xml file. Selecting Transparency will execute ~/.config/openbox/scripts/xcompmgr.sh set. Selecting Transparency, fadings will execute ~/.config/openbox/scripts/xcompmgr.sh setshaded. Selecting Transparency, fadings, shadows will execute ~/.config/openbox/scripts/xcompmgr.sh setshadowshade. Selecting No effects from the menu will comment the #xcompmgr & line in the autostart.sh file, and will execute ~/.config/openbox/scripts/xcompmgr.sh unset in the menu.xml file. Any changes made are kept at next login.

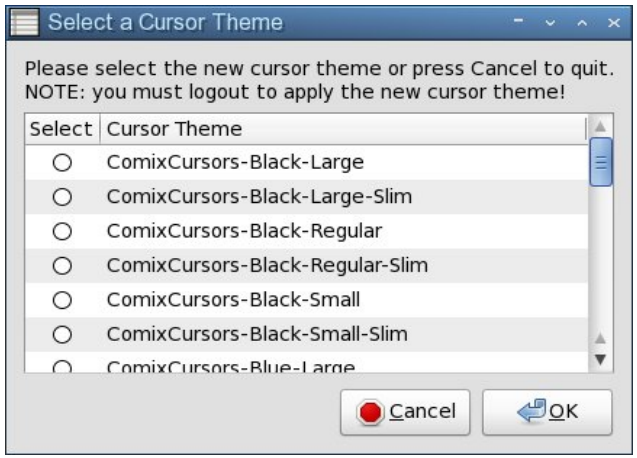
Use lxcursor to change cursor theme

We can use lxcursor to change our cursor theme. Open Synaptic and install the lxcursor package. Once that is accomplished, there is nothing to edit in the Openbox menu.xml file, unless you want to show the item in the main portion of the menu. We will, however, need to edit the desktop file, located at /usr/share/applications/lxcursor.desktop. As user root, open the desktop file in a text editor. Scroll down towards the bottom of the file and locate the line OnlyShowIn=LXDE;. Here, you can either

comment the line by adding the # symbol at the beginning of the line, or by deleting the entire line. Once the changes have been made, save the desktop file. To have the LXCursor item show in the Openbox submenu, freshen the Openbox menu by logging out, rebooting, or clicking on Reconfigure or Restart in the submenu of the OpenBox label.



If you change the current cursor theme, the change will not take effect until you have logged out and logged in again.



Screenshots Via The Keyboard – Revisited

In the November 2010 issue of The PCLinuxOS Magazine, back when we were wrapping up our series of articles on the LXDE desktop, we covered how to add keybindings to take screen shots (see the Advanced Keyboard Shortcuts section of the article).

One problem with the keybindings, as they were presented, is that they take the screen shot immediately. Normally, this isn't necessarily a problem. But it is if you want to capture menus in your screen shots, or some other on-screen animations (as I needed to do when taking the screen shots for the article on launch bars), taking the screen shots immediately won't work.

Fortunately, there is a solution, and it's quite simple. For example, take the command to capture the full screen shot (as excerpted from the original article):

```
bash -c "xwd -root | convert -
/tmp/screenshot-$(date +%s).png"
```

I simply added `sleep 5;` to the beginning of the command that is between the quotes. This provides a five second delay before carrying out the rest of the command that takes a screen shot of the full screen. The five second delay gives you ample time to activate a menu or animation that you might want to capture in the screen shot. This command uses the keybinding Ctrl + Print.

While I was "tinkering" with this command, I also took time to reformat the file name and where my screen shots were stored. As it was in the original article, it a) stored the image in your /tmp directory, and b) used a cryptic number after the date that specified how many seconds since 01-01-1970 UTC (the %s in the command above).

Instead, I changed the location where the screen shots are saved. I created a "Screenshots" directory under my "Pictures" directory, and used that instead. Second, I reformatted the information after the date to reflect something that is more easily read (and understood) by humans. I ended up with this:

```
bash -c "sleep 5;xwd -root | convert -
~/Pictures/Screenshots/screenshot-$(date
+%F-%H-%M-%S).png"
```

The %F uses the full date (YYYY-Month-Day), then prints a dash, then the hour (%H, based on a 24 hour clock) as two digits, another dash, then the minutes (%M) as two digits, another dash, then the seconds (%S) as two digits. Done this way, it makes it easier to locate the appropriate screen shot in a directory full of other screen shots.

Another Thing About Screen Shots

The above method does have one teensy-weensy problem: it won't capture transparency areas of a screen image that you may want to preserve. Instead of showing the transparency, it shows a transparent region as black. Most of the time, that is not a problem – unless I'm trying to show the transparency in the screen shot.

One of the applications I routinely use from the PCLinuxOS repository is MTPaint. When it comes to cropping an image for the magazine, there's little else that beats the simplicity of MTPaint. It's much faster to load than Gimp, and it makes sense to me to use a simple tool for a simple job.

Fortunately, MTPaint will also take screen shots. Anyone who has installed MTPaint from the PCLinuxOS repository will also notice that there are ***two*** entries in the Graphics section of the applications menu: one for the MTPaint program itself, and another one labeled MTPaint Screenshot. The latter will preserve any level of transparency that is displayed on your screen, as well, which is why I like to use it.

The only problem is that MTPaint Screenshot takes the screen shot image immediately. This doesn't allow me to capture any menu images or animations on the screen that I may also need to display. Fortunately, the solution was only a very short bash script away.

Borrowing from my five second delay I added to the keybinding method above, I created a bash script that executed a five second delay, and then used MTPaint to capture the screen shot. Here's the simple bash script:

```
#! /bin/bash
sleep 5
/usr/bin/mtpaint -s
```

The first line (of course, after the bash line) causes a five second pause before executing the second line, which runs MTPaint in the screen shot mode (hence, the -s command line option).

The only drawback here is that MTPaint will only grab a screen shot of the entire screen. But, what the hey. It also loads it into the MTPaint editor, where I can easily crop the image to only the part that I need.

Add A Power Manager & Monitor

Openbox, as it comes, doesn't have its own power manager or monitor. However, you can install the Gnome Power Manager, via Synaptic. With Gnome Power Manager, you can monitor your power and battery status. Additionally, it will warn you when your laptop battery gets low, and suspend, hibernate or shut down your computer when it the battery becomes critically low. Fortunately, Gnome Power Manager doesn't pull in many Gnome dependencies, helping to keep your Openbox installation light and nimble.

To insure that mine starts every time I start my computer, I added the following two lines to my autostart.sh file, in my ~/.config/openbox directory:

```
gnome-power-manager &
sleep 1
```

Now, Gnome Power Manager starts and runs in my system tray all the time, keeping a watchful eye on the status of my power and battery status.

Let There Be Sound

Well, okay. I may be exaggerating a bit, but out of the starting gates, Openbox doesn't have a sound volume manager running either. Check in Synaptic to see if VolumeIcon is installed already. If it isn't, go ahead and install it.

Next, much as we did with the Gnome Power Manager above, add the following two lines to your autostart.sh file:

```
volumeicon &
sleep 1
```

Now, whenever you start up your computer, VolumeIcon will be ever present to allow you quick and easy access to controlling the sound volume on your computer.

Openbox Resources: Learn More About It

by Paul Arnote (parnote)

After all the talk in these magazine pages about Openbox in the recent months, you may be wondering where you can find more information about Openbox. While the articles dealt with Openbox 3.4, Openbox 3.5 has just been released. Fortunately, all the information in the articles we've published over the last few months is equally applicable to Openbox 3.5. Melodie has been working on updated Openbox ISOs that feature the newer Openbox 3.5, and it should be released before too much longer.

Meanwhile, check out these resources below for more Openbox information.

Openbox Wiki

For all things Openbox, this is your one-stop-shop. You will find information about all sorts of Openbox options, as well as the "official" documentation. You can also find information regarding all sorts of Openbox add-ons. Just remember that it is not recommended to install applications from outside the official PCLinuxOS repository. Instead, make a post in the Package Suggest section of the PCLinuxOS forum for one of our packagers to package the add-on, so it can be added to the official PCLinuxOS repository.

Box-Look.org

There's nothing quite like redecorating from time to time, and when the urge strikes you to redecorate your Openbox desktop, make this site your first stop. Here, you will find new Openbox themes, wallpapers, fonts, logos and other cool stuff.

Customize.org

Find even more Openbox themes, wallpapers and icon sets at this site. The link above sorts out those user-submitted customizations that have the Openbox tag applied to them.

Urukrama's Openbox Guide

If a well written, well researched guide to Openbox, written in plain English is more to your liking, then look no further than Urukrama's Openbox Guide. The guide appears to be quite complete, and should be bookmarked by every Openbox user, so that the full potential of Openbox may be realized.

ArchLinux Wiki

Over at the ArchLinux Wiki, they maintain a very complete Openbox section, separate from the "official" Openbox Wiki. Of special interest is the special "Tips & Tricks" section. Topics in the "Tips & Tricks" section range from fairly simple to advanced and complex.

DeviantArt

To be honest, I would have never thought to look at DeviantArt for window manager themes. But, lo and behold, they are there. As with anything you might expect to find at DeviantArt, the quality is quite nice, so you should take the time to check out the offerings here.

I'm sure that with a little more digging, you can find other Openbox resources on the 'net. However, these six sites should go a long way towards getting you sailing a smooth course with Openbox.