Fluxbox Documentation

Tobias Klausmann

Edited by

Engelbert Gruber

Fluxbox Documentation

by Tobias Klausmann

Edited by Engelbert Gruber

Revision History

Revision 0.4 2003-09-16 Revised by: eg
New editor in charge. Add revision history. Revisite Key Actions (crosscheck with FbCommandFactory.cc).

Table of Contents

List of Tables

Chapter 1. Introduction

1.1. About this documentation

This is the new documentation for Fluxbox, an X11 window manager. It was created from the older HTML only documentation with the goal of making it easily adaptable to various output formats. Its source is written in DocBook 4.1, a format that was specifically designed for documentation.

There have been a few Fluxbox documentation maintainers so far. The docs your reading now are mostly based on work by Rando Christensen <rando@babblica.net (mailto:rando@babblica.net)> or were originally written by him, followed by <klausman@users.sourceforge.net (mailto:klausman@users.sourceforge.net)> and several others have contributed. If credit is missing, please notify the <editor (mailto:grubert@users.sourceforge.net)> to fix it.

Questions regarding Fluxbox itself are better directed to the Mailing lists. You can find subscription information for them on the official Fluxbox page (http://www.fluxbox.org).

> **Submissions and Translation:** I'll accept almost all submissions of well-written documentation for anything not covered here which people think should be added; Simply send me an email at the address mentioned above if you have any questions about how you can help or if you have some documentation you'd like to add.

Both the source of this documentation and various converted formats such as HTML and PostScript are available from the Fluxbox site. If you would like to use a different DTD or another output format, the source is your friend, although you'll probably need some DocBook expertise to modify DTDs. If you would like to convert the source file to some other format than the ones provided on the Fluxbox site, the package docbook2x might prove helpful.

1.2. About Fluxbox

1.2.1. What is Fluxbox

Fluxbox is yet another window manager for X. It's based on the Blackbox 0.61.1 code. Fluxbox looks like Blackbox and handles styles, colors, window placement and similar things exactly like Blackbox (100% theme/style compatibility).

So what's the difference between Fluxbox and Blackbox then? The answer is: LOTS! Here is a list of features that Fluxbox already has, or are in the works...

1.2.2. Features

Implemented:

- Configurable window tabs.
- UTF-8 support
- Iconbar (for minimized/iconified windows)
- Windows/Tabs auto grouping
- MouseWheel scroll changes workspace
- Configurable titlebar (placement of buttons, new buttons etc.)
- KDE and Gnome (including Gnome 2) support
- Extended Window Manager Hints (ewmh) support
- Native integrated keygrabber (supports emacs like keychains)
- Maximize over slit option
- Slit dockapp ordering
- Configurable toolbar

Planned:

- Session Management

Aside from all this many changes and improvements have been made to the code.

1.2.3. Getting Fluxbox

Fluxbox source can be downloaded from the main Fluxbox site, http://fluxbox.sourceforge.net/. Most Linux distributions and other free Unixes also include source and binary packages of Fluxbox. When in doubt, use your latest distribution packages.

1.2.4. Asking questions and getting support

While Fluxbox is pretty easy to use and configure, chances are that at some point questions or problems turn up. A great deal of those have been answered or solved before. Therefore, before you start asking questions on the mailing lists or on the Fluxbox IRC channel, please have a look at the FAQ in Appendix B.

Chapter 2. Getting started

2.1. Installing Fluxbox

This section was contributed by Jason Gillman Jr. aka "Ircaddict".

Many distributions provide binary packages of Fluxbox (or, in the case of Gentoo and FreeBSD, ports/ebuilds) which make installing Fluxbox pretty painless. There are reasons to compile from source, however. For example, the most recent version of Fluxbox might not be available as a distribution package. Additionally, it might be desirable to compile Fluxbox with a certain set of compiler flags. If you would like to use your distributions packages, consult the distribution docs. For some distributions, both source and binary packages are provided on the Fluxbox site.

The purpose of this document is to help people who are fairly new to the X11 windowing system (or Linux in general) compile and install the Fluxbox window manager.

2.1.1. Acquiring the sources

The first thing that you will want to do is go to the download page (http://fluxbox.sourceforge.net/download.php) and download the source tarball (the extension will be .tar.gz).

From time to time, special developer releases are made. Those allow a preview of things to come, new features and the like. Due to their developer-release quality, they sometimes lack functionality or stability. Try them if you want bleeding edge Fluxbox - with all the benefits and dangers. Directions on how to fetch the can be found in the News section of the official Fluxbox site.

2.1.2. Extract and compile

Okay, now that you have the source tarball, you have to extract the goods. This can be done by running the following command, replacing the filename with the one of the file you downloaded:

```
$ tar xzvf fluxbox-0.1.12.tar.gz
```

It will then throw out a list of the files that are being un-archived. After you do this, change into the directory that was created (it will be something like `fluxbox-0.1.12/`, but it depends on the version). The next step is to configure and make Fluxbox. During the configure run, you can enable or disable some features in Fluxbox. For most people, the defaults should be okay. If you want Fluxbox' slit to work with KDE panel icons, you would add the option **`--enable-kde`**. To find out what other options

are provided by the configure script, use the option **`--help`**. If you don't want to enable KDE, this should suffice:

```
$ ./configure
$ make
```

After Fluxbox is compiled, become root and run:

```
# make install
```

Congratulations, you now have compiled and installed Fluxbox.

2.1.3. Running Fluxbox

It's all nice and good if you have it installed, but what use is it if you can't run it?

There are two generally different ways to start X11 (and thus Fluxbox). The traditional way is using the command `startx`. The other way is using a graphical login manager (also called "display manager"). The most common display manager is `xdm` which is part of the XFree86 distribution. The display manager provided by Gnome is called `gdm`, the one from KDE is `kdm`.

If X11 is started the former way (via `startx`), the file that is important is called `.xinitrc` and resides in your home directory. In the case of starting via a display manager, the file is `.xsession` which resides at the same location.

The next step is to find the executable for Fluxbox. For most people, this is `/usr/local/bin/fluxbox`. Now you need to edit (or create) the file I just mentioned. Just put the following line at the bottom of the file:

```
exec /usr/local/bin/fluxbox
```

Change the `/usr/local/bin/fluxbox` to where ever your Fluxbox executable is, the above is the default location when compiling from source. Once that is done, save it and close whatever editor you used to edit it. Now you need to run the following command if you use `startx`:

```
$ chmod 700 .xinitrc
```

In the case of `.xsession` that is not needed. In both cases, you should create the directory in which Fluxbox stores its configuration:

```
$ mkdir .fluxbox
```

If you don't create it, when you exit Fluxbox and restart, you will lose all your setting.

Maybe copy `init` and `menu` files from the `/usr/local/share/fluxbox` directory to the `.fluxbox/` directory in your home directory.

2.1.4. Adding to KDE Displaymanager

To get Flux on the KDM menu store

```
[Desktop Entry]
Encoding=UTF-8
Type=XSession
Exec=/usr/local/bin/fluxbox
TryExec=/usr/local/bin/fluxbox
Name=Fluxbox
```

into the file `/usr/share/apps/kdm/sessions/09fluxbox.desktop`.

The file location is for Mandirva and might be different on other systems mabye look for `/etc/kdm` on others.

The numberprefix `09` might have to do with the position in the session selection.

2.2. Commandline options

The available options are output with a short description when fluxbox is invoked with the option `-help`.

```
Fluxbox 1.0rc2: (c) 2001-2006 Henrik Kinnunen
Website: http://www.fluxbox.org/

-display <string>  use display connection.
-screen <all|int,int,int> run on specified screens only.
-rc <string>   use alternate resource file.
-version   display version and exit.
-info    display some useful information.
-log <filename>   log output to file.
-help    display this help text and exit.
```

Chapter 3. Tools

3.1. Introduction

Fluxbox comes with an assortment of tools that make life with a bit easier or provide nifty features. By default, they are installed in the same location as the `fluxbox` binary, which is `/usr/local/bin` unless you used a different prefix during configuration. Most distributions also pick a different location, so if you installed binary packages, you may find them in `/usr/bin`.

3.2. `fbrun`

`fbrun` is basically equivalent to the "Run..." dialog in other desktop environment. This means that it is an easy way to start a program that isn't contained in the menu (or needs a special set of parameters for this particular invocation).

Another way `fbrun` can be useful is to be called from the menu with a preloaded command line that you can edit and then execute. An example might be sshing to a very long host name with lots of options of which one changes all the time. In this case, you could add an entry for fbrun to your menu that contains all the options and the host name. When you use said entry, you could edit the line as necessary and execute it.

`fbrun` has various options:

```
-font [font name]           Text font
-title [title name]         Set title
-text [text]                Text input
-w [width]                  Window width in pixels
-h [height]                 Window height in pixels
-display [display string]   Display name
-pos [x] [y]                Window position in pixels
-fg [color name]            Foreground text color
-bg [color name]            Background color
-na                         Disable antialias
-hf [history file]          History file to load (default ~/.fluxbox/history)
-help                       Show this help
```

Most of those options should be self-explanatory. The options **`-text`** and **`-hf`** maybe need some explanation. The former is used to specify the pre-loaded (editable) text inside the `fbrun` window. If you'd like to specify multiple arguments (like **`ssh -X -f`**), be sure to wrap them in quotes:

```
fbrun -text "ssh -X -f"
```

The **`-hf`** option specifies the history file, which is a place where `fbrun` keeps its "memory" of the command lines you used (just like bash does). Normally you don't need this option, then the default is used. It might prove useful if you have multiple `fbrun` entries in your menu and want to keep separate histories for each of them.

3.3. fluxbox-generate_menu

FIXME: This section has to be written.

3.4. fluxspace

(Quoted from :http://fluxspace.sourceforge.net):

> Fluxspace blends Fluxbox's window management with new desktop management capabilities. It leverages existing components and the power of Python to help build a flexible desktop environment around Fluxbox and other lightweight window managers.
>
> - Add per-workspace desktop icons and panels to Fluxbox by integrating tools like Rox Filer and Idesk.
> - Decorate each workspace with different wallpaper.
> - Manage your startup applications.
> - Automatically start and stop applets as you enter and leave workspaces, allowing each workspace to have its own unique tools and dockapps.

3.5. wmctrl

http://sweb.cz/tripie/utils/wmctrl/

The wmctrl program is a command line tool to interact with an EWMH/NetWM compatible X Window Manager.

3.6. Pagers

A desktop/workspace pager aids in working with more than one workspace. By displaying workspaces in a small view and allowing to drag windows in this view. A pager is not required when working with multiple workspaces, managing workspaces is done by the windowmanager, the workspace name and contents can be seen in the toolbar and windows can be sent to other workspaces from the window menu or configured in the apps file.

But pagers give a better overview, some show only window borders, others pictures of window contents or icons of the running application.

- fbpager (http://www.fluxbox.org/fbpager/): a slit pager with transparency, mouse gestures and icons.
- fluxter: is a modified bbpager, configuration is closer to fluxbox. fluxter (http://benedict.isomedia.com/homes/stevencooper/projects/fluxter.html)
- Rox pager: the pager of ROX desktop, a desktop environment. ROX (http://rox.sourceforge.net/)
- 3d-Desktop: OpenGL pager showing workspaces in 3D. desk3d (http://desk3d.sourceforge.net/)
- and a lot ...

Chapter 4. Tabs

4.1. An introduction to tabs

Fluxbox' tabs aren't in any way a new idea. The implementation is much like that of the PWM Window Manager. The way it works: Multiple windows are grouped together, and they share the same geometry- they're the same size, exactly in the same position, and moving one window results in moving them all. Think of them as a stack of papers. The tabs are like those little little plastic tabs you can stick on papers to make flipping to a certain page very easy and fast.

In versions up to 0.1.14 the tabs actually were attached to the window. From 0.9.x tabs are embedded into the window titlebar. Version 1.0rc1 and later has both options, configurable in the config menu.

That's exactly how tabs work in Fluxbox. Simply by selecting the proper tab for the window you want, and that window pops up to the top of that window stack. Now, let's try it.

Basic tabbing

The first thing to remember is that *all* tab manipulation actions use the middle mouse button. So, to start, pick two windows that you'd like to group together. Middle-click on the tab for the first one, and drag it onto the tab for the second. Congratulations, they're stuck together! You can use the tabs now to switch between the two windows.

To remove a tab, it's the same thing. Middle-Click on a grouped tab and drag it away.

4.2. Advanced Tabbing

4.2.1. Sloppy Window Grouping

"But It feels awkward for me to drop the tab onto the small other tab."

Good news, then. From the Fluxbox 'configuration' menu, pick the 'Sloppy Window Grouping' Option. This allows you to drop your tab anywhere on the target window to perform the grouping.

4.2.2. Tabbing single window classes

"That's great, but I only want to tab program X!"

There's two different ways you can accomplish this, depending on how many programs you want to have tabs. You can either switch tabs on/off by window. (Right click on titlebar, click 'tab' option), *or* you can turn them off globally with 'Configuration'->'Use Tabs'. After turning them off, You can turn tabs on for individual windows as described above.

4.2.3. Turning tabs off entirely

"I don't think I like tabs. Can I turn them off?"

Certainly. Select 'Use Tabs' from the Fluxbox 'Configuration' menu. As this is a toggle, selecting it again will turn tabs back on. There is also a setting in the `init` configuration file for this:

```
session.tabs:   true
```

Disabling tabs can be accomplished by replacing **`true`** with **`false`**.

4.2.4. Tab Placement

In versions up to 0.1.14 the tabs actually were attached to the window, from 0.9.x tabs are embedded into the window titlebar.

(0.1.14) There's a configuration menu option that's called 'Tab Placement'. These are places on the window where the tabs will be located. This is pretty straightforward, so the only thing we really need to mention is the 'Relative' options. These options make it so that the total length of all tabs on the window is equal to the length of the window. As in, if there's one tab on a window, the tab is the length of the window. If there are two tabs, each tab's length is 50% of the window length. This option often makes the tabs look unobtrusive and is very popular.

4.2.5. Tab Autogrouping

Sometimes you'd like some apps to be automatically grouped together when they start. This is logically called "Autogrouping". This part explains how it works. First of all, you need Fluxbox v0.1.11 or higher. Autogrouping doesn't work with older versions. Then you need to create a `~/.fluxbox/groups` file if it is not there already. Then, edit your `~/.fluxbox/init` file and add this line (or change it if it's there and looking different):

```
session.groupFile:  ~/.fluxbox/groups
```

Okay, everything is in place. Now you simply have to fill in the groups file.

Groupfile format

There is one group for each line in the file and you just type the instance name of the program to be grouped. Example:

Example 4-1. `groups` file

```
Navigator nedit
xterm
```

This will make two groups, one with `netscape` and `nedit` and one with `xterm`. The new window will only group itself to other windows on the same workspace and to the last window that was focused. You get the name to put in the group file with:

```
xprop |awk '/WM_CLASS/{print $4}'
```

and click on the window. If this does not display anything, try changing the **`$4`** to **`$3`**.

Autogrouping from Tabs

This will allow you to popup the root menu, if you right click on the tab and select an application it'll start grouped to the tab.

> **Note:** Grouping this way (from tabs) might interfere with normal autogrouping in a bad way.

4.2.6. Tabs in Themes

We have an entire section (Chapter 10) devoted to the way tabs look in themes. You can check it out if you're interested in changing the way the tabs look in your theme. (Tabs in themes will normally default to what the titlebars look like; sometimes people like to change that.)

Chapter 5. Key Bindings

5.1. The Keygrabber

The keygrabber works very similarly to `bbkeys`, which is an excellent tool but had some limitations (and was license incompatible), but with a totally new syntax for the config file, and a couple of new features which makes Fluxbox even more powerful.

For one, the new keygrabber supports chains of key sequences (like emacs)... so you could for example have: **`Mod1 + M + Mod1 + F`** to switch to the next workspace (not that anyone would actually use that particular key sequence.)

And if you've typed in a part of the sequence but decide not to continue (abort) then you can hit another key sequence (that you configured in your keys file) to simply abort (with: AbortChain).

In addition to this you can also bind a key sequence to switch between grouped/tabbed windows in a group (with: NextTab and PrevTab).

It is possible to execute more than one command with one keybinding. The commands will be executed parallel.

Example 5-1. macro command

```
Mod4 Shift x :MacroCmd { ExecCommand xterm } { ExecCommand gv 1.ps }
```

Finally, for your convenience we (or rather vlaad and tarzeau) have supplied you with two scripts (they both do the same thing) to convert a `bbkeys` conf-file to a Fluxbox keys-file. (usage of the scripts is described in them). Download: convertkeys (http://fluxbox.sourceforge.net/download/convertkeys) or convertkeys2 (http://fluxbox.sourceforge.net/download/convertkeys2).

Example 5-2. A Fluxbox keysfile

```
Mod1 Tab :NextWindow
Mod1 F1 :Workspace 1
Mod1 F2 :Workspace 2
Mod1 F3 :Workspace 3
Mod1 F4 :Workspace 4
Control n Mod1 n :NextTab
```

So, as you can see, First there is a modifier, then a key (then a modifier and key again if you wish to have a longer sequence) and finally a colon with an action.

For a list of valid keys actions read a little further down in this document.

And why this is good for you? Well, now you can master your xmms, for example:

Example 5-3. XMMS keybindings

```
Mod1 P :ExecCommand xmms -p
Mod1 F :ExecCommand xmms -f
```

If you have some additional keys (e.g multimedia keys), you can control xmms this way if you have configured those keys properly in XFree86:

Example 5-4. Multimedia keys for XMMS

```
None XF86AudioPlay :ExecCommand xmms -u
None XF86AudioStop :ExecCommand xmms -s
```

I would say check xmms --help for more info, but you are doing this right now probably...

5.2. Key Names

You probably ask yourself how to find out key names. Run `xev`, move your mouse to the newly created window and press a key and see what it says about it. Here is the example after pressing the Right Arrow key:

Example 5-5. `xev` output for the right arrow key

```
KeyPress event, serial 18, synthetic NO, window 0x2c00001,
root 0x60, subw 0x0, time 3745737930, (373,380), root:(504,526),
state 0x10, keycode 102 (keysym 0xff53, Right), same_screen YES,
XLookupString gives 0 characters:  ""
```

The interesting bit is the key name, which is what's inside the parentheses together with the `keysym`. In this example, that reads `(keysym 0xff53, Right)`. Thus, the name for the key is `Right`.

Special Keys

Here are some special keys for your convenience. Note that they will be displayed by `xev` immediately after you press them (not as modifiers to other keys).

Key	X11 Name
Control, Strg	Control
Alt	Mod1
Super, Meta, Win* Keys	Mod4

5.3. Actions

These are the actions currently provided by Fluxbox. They cover most of the stuff one might want to place on keypresses. Note that in the keybindings file, the last character before the action should be a `:`.

Actions are case insensitive.

Table 5-1. Navigation

Action	Result
`Workspace`	Go to a particular workspace. Use :Workspace 1, :Workspace 2, etc.
`NextTab`	Switch to the Next Tab in the current group.
`PrevTab`	Switch to the Previous Tab in the current group.
`NextWindow N`	Go to Next Window. See Note 1.
`PrevWindow N`	Go to previous window. Again, see Note 1.
`NextWorkspace`	Go to the Next Workspace.
`PrevWorkspace`	Go to the Previous Workspace.
`NextGroup`, **`PrevGroup`**	Go to the next/prev window group.
`LeftWorkspace`	Same as PrevWorkspace.
`RightWorkspace`	Same as NextWorkspace.

1. `NextWindow` / `PrevWindow`: The **`NextWindow`**/**`PrevWindow`** has a numerical argument, which is a bit too complicated to be explained inside that table above. Here's how it works.

The integer parameter to configure the behaviour:

Parameter value	Behaviour
0 or unspecified	Default/current behavior - no skipping
1	Skip lower tabs
2	Skip stuck windows
3	Skip lower tabs/stuck windows
4	Skip shaded windows
5	Skip lower tabs/shaded windows
6	Skip stuck windows/shaded windows
7	Skip lower tabs/stuck windows/shaded windows

Table 5-2. Window Operations

Action	Result
`Close`	Close the Window.

Action	Result
KillWindow	The equivalent of calling `xkill` and clicking on the window.
Minimize	Also known as "iconify". Make the window iconified.
ShadeWindow	Put window in the 'shaded' state, or restore from the 'shaded' state.
StickWindow	Toggle a Window's 'sticky' state.
ToggleDecor	Toggle whether or not current window has a border, buttons, and titlebar.
Raise	Bring the window to the 'Top', it will appear 'Above' windows that it overlaps.
Lower	Opposite of Raise.
NextTab, **PrevTab**	Activate next, prev tab.
MoveTabLeft, **MoveTabRight**	Move activate tab by the **n** tabs left/right.
DetachClient	Take client out of tab-group.

Table 5-3. Window Sizing

Action	Result
MaximizeHorizontal	Maximize the window horizontally.
MaximizeVertical	Maximize the window vertically.
MaximizeWindow	Maximize the Window.
Resize	Resize the active window by the specified delta, e.g. **resize −8 −8**.
ResizeTo	Resize the active window to the specified geometry. **resize −8 −8**.
ResizeHorizontal	Resize horizontal by the specified delta.
ResizeVertical	Resize vertical by the specified delta.
ArrangeWindows	Tile Windows magically.
ShowDesktop	Iconify all windows.

> **Note:** When resizing "one unit" might be different for different applications. `xterm`/`aterm`/`Eterm` instead of resizing by one pixel, will add another character-width worth of space.
>
> Other programs should just resize by one pixel.

Table 5-4. Window Movement

Action	Result
SendToWorkspace n	Send current window to a specified workspace. e.g. :SendToWorkspace 1

Action	Result
TakeToWorkspace n	Send current window to a specified workspace and change to the workspace.
Move	by delta-x delta-y.
MoveLeft	Guess.
MoveRight	Guess.
MoveUp	Guess.
MoveDown	Guess.

Table 5-5. Miscellaneous

Action	Result
AbortKeychain	In Multi-binding keychains, cancel the keybinding.
ExecCommand	Execute a command. Example :ExecCommand xmms -t.
RootMenu	Summon the Root Menu.
WorkSpaceMenu	Summon the Workspace Menu.
Restart	Restart fluxbox.
Reconfigure	Reconfigure fluxbox, rereads configuration. e.g. if `keys` was changed, but `init` and `slitlist` might be written before read.
SetStyle	Load the specified file.
SetWorkspaceName	.
SaveRC	Save resource files.
Quit	Quit fluxbox.

Chapter 6. Desktop Backgrounds

Fluxbox, like Blackbox, just has two wrapper utilities for this - `fbsetroot` and `fbsetbg`. Let's see how they work.

`fbsetroot` is about the equivalent of `xsetroot`, which can set backgrounds as long as they're simple, like a solid color. `fbsetroot` can also set gradients.

`fbsetbg` is a wrapper that tries to find a suitable background-setting app and then tries to set the wallpaper using that app. You don't have to configure `fbsetbg`. It just uses the first app it can find.

Overriding theme backgrounds

Fluxbox allows to ignore the background specified in a style file by specifying it in your `~/.fluxbox/init` file:

```
session.screen0.rootCommand: fbsetbg -f ~/backgrounds/zimdib_dark.png
```

To avoid having to change the `~/.fluxbox/init` file, `fbsetbg` will store the used wallpaper in `~/.fluxbox/lastwallpaper` and a reload it when called with option **`-l`**. The rootCommand should then be.

```
session.screen0.rootCommand: fbsetbg -l
```

Chapter 7. The Slit

One of the most frequently asked questions is "What is the slit?" In fact, when this document was written, I also copied it to the top of the FAQ to point to this section in order to stop it from being asked in #fluxbox ten times a day, and to end the myth that the Slit is another name for the Toolbar.

The slit is one of the many parts of Fluxbox that has been inherited from Blackbox. It is designed to hold WindowMaker Dockapps, (and anything that runs in that mode which is called 'withdrawn' or (less often) 'swallowed'). Such applications often have a **`-w`** option, but some are automatically in withdrawn mode.

Well, the first thing is to make sure that it's compiled into your copy of Fluxbox. It's generally safe to assume it is. As far as I know, all packages for different distributions ship with it. If you think you're better off without it, you can disable it at compile time. Note however, that an unused slit occupies *no* screen space and only very little memory, so it's usually only necessary to disable it if it interferes with some other software you use.

Other than that, You can run any 'dockable' application. (This is also known as running in 'withdrawn' mode). As an example, `xmms` ships with the `wmxmms` application. Simply run **`wmxmms &`**, and it will appear in the slit. As mentioned above, some applications (for example `gkrellm`) need the **`-w`** command line switch to appear in the Slit.

Where can I get dockapps?

The best place to start is the Dockapp warehouse (http://www.bensinclair.com/dockapp). There is a very large repository of different dockapps available there. Apart from that, you could search Freshmeat (http://freshmeat.net/), or check with your distribution.

The bbtools (http://bbtools.sourceforge.net) page holds a bunch of Blackbox/Fluxbox utils, most of which can run in the slit.

Also, if you have KDE support enabled, KDE dock applets will appear in the slit.

Dockapps.Org (http://www.dockapps.org/) is a new website exclusively dedicated to dockapps.

Can I change Slit behavior?

Of course. Simply right-click on the visible portion of the slit, and select options from there. Most of these options are the exact same as on the Taskbar. The only thing that isn't is `Direction`. Your slit can run either `Horizontal` or `Vertical`.

There's also a **`Maximize Over Slit`** option in the configure menu; this allows maximized windows to cover the slit.

I want my slit apps to remember order!

In Fluxbox 0.1.10 and later, this is possible, using a `slitlist` file. Here are some instructions on using it.

The current order of dockapps is stored in a file, by default `~/.fluxbox/slitlist`. When loading dockapps into the slit, it attempts to maintain the previous ordering, matching previously-run dockapps by name.

A simple procedure for getting the slit sequences the way you like is:

Ordering dockapps

1. Run Fluxbox with no pre-loaded dockapps.
2. Run dockapps individually in the order you want them.
3. Re-add dockapps to your auto-run script, for example `.xinitrc` or `.xsession`. Order doesn't matter here.

This sequence is saved to `~/.fluxbox/slitlist` by default and will be maintained in future Fluxbox sessions.

You are free to manually edit the `slitlist` file. It is a simple list of window names, one line per dockapp. This file should be edited while not running Fluxbox. Otherwise changes may get overwritten.

You also have the option of choosing a different path for the slit list file. The following example `init` file entry changes the path:

Example 7-1. Slitlist specification

```
session.slitlistFile: /home/me/etc/slitsort
```

Note that there is no option to disable the sorting entirely. The patch author could not think of any benefit to arbitrary ordering.

Chapter 8. The Toolbar

The toolbar is a small area to display information by fluxbox, like a clock, or buttons for running programs.

It is configured by the init-file and the style-file. Init settings usually are changed by the toolbar menu.

The toolbar can be switched off (made invisible by setting `session.screen0.toolbar.visible: false` in fluxbox's init file.

The tools to be displayed are configured in the resource file, usually .fluxbox/init, with the entry toolbar.tools.

Example 8-1. toolbar tools example

```
session.screen0.toolbar.tools: clock, iconbar, workspacename
```

Possible tools: workspacename, prevworkspace, nextworkspace, iconbar, systemtray, prevwindow, nextwindow, clock

The toolbar width and transparency and layer can be configured via the toolbar menu. The toolbar menu is displayed if one right-clicks the clock or the workspacename in the toolbar.

The iconbar can be configured to display None, Icons from all workspaces, Workspace Icons, all windows from Workspace, All Windows from all workspaces.

How this is displayed is configured in the style file.

Chapter 9. Editing Menus

So you have Fluxbox installed and now you see that nifty little application launcher called the menu when you right click your desktop. This is of little use if you can't edit it to launch the applications that you use. This doc will attempt to answer all questions regarding this.

First off, there is a nice utility shipped with Fluxbox, called `fluxbox-generate_menu`. It tries to glean the paths of various commonly installed programs such as browsers and terminal emulators from your environment and creates a menu file. `fluxbox-generate_menu` is covered in detail in Section 3.3.

9.1. Setting the Menu File Location

The Fluxbox menu is by default `~/.fluxbox/menu`. This setting can be changed, however, via your `init` file. Here is an example of the line:

Example 9-1. Setting the menu file

```
session.menuFile:       ~/.fluxbox/menu
```

Just change the `~/.fluxbox/menu` if you wish to use a different file for your menu. The current structure, however, should be fine for most people.

9.2. Available Commands

The Fluxbox menu is a text file that allows you to make subfolders, launch applications, control workspaces, configure Fluxbox, and exit X. The menu can take the following commands:

```
# menu file 2004-06-22
[begin] (MenuTitle)
[submenu]  (SubMenuName) {SubMenuTitle}
[exec] (ApplicationName) {/path/to/program}
[include] (/path/to/menufile) <icon file>
[end]
[nop] (--------)
[workspaces] (SubMenuName)
[wallpapers] (/path/to/background/images) {background-setting-command}
[wallpapermenu] (directory) {command}
[stylesdir] (/path/to/stylesdir)
[config] (FluxboxConfiguration)
[reconfigure] (Reconfigure)
[restart] (Restart)
[exit] (Exit)
```

- Comments start with **#**, everything upto the line's end is ignored.
- Text in **[]** is the command to fluxbox.
- Text in **()** is the text shown in the menu.
- Text in **{}** is the command that is started if this entry is clicked.
- Text in <> specifies an icon file. The icon files path should be absolute and in xpm-format.
- If the parameter to include directive is a directory, every file in this path will be included.

  Included files must start with **[begin]** and end with **[end]**.

Not all entries are necessary in all cases, e.g. **[end]** has no use for an icon.

Note that there is no hard limit on the nesting depth of submenus. There probably is a practical one, though.

[nop] - This allows you to put text or an empty line if you wish that will not execute anything but just act as a separator in your menu.

[reconfigure] - If you use the menu to change your Fluxbox configuration, the changes will be lost after you exit Fluxbox. They need to be written to the `init` file to be permanent changes, which can be done by hitting reconfigure after you've made the changes you want.

[restart] - I just want to make sure that everyone knows restart only restarts Fluxbox, and not your whole system.

Example 9-2. `menu`

```
# Fluxbox menu file
[begin] (Fluxbox)
  [exec] (rxvt) {rxvt -ls}
  [exec] (netscape) {netscape -install}
  [exec] (The GIMP) {gimp}
  [exec] (XV) {xv}
  [exec] (Vim) {rxvt -geometry 132x60 -name VIM -e screen vim}
  [exec] (Mutt) {rxvt -name mutt -e mutt}
  [submenu] (mozilla)
    [exec] (browser) {mozilla -browser}
    [exec] (news) {mozilla -news}
    [exec] (mail) {mozilla -mail}
    [exec] (edit) {mozilla -edit}
    [exec] (compose) {mozilla -compose}
  [end]
  [submenu] (Startup)
    [exec] (gkrellm) {gkrellm -w}
    [exec] (xmms) {xmms -p}
    [exec] (galeon) {galeon -s}
```

```
        [exec] (kdeinit) {kdeinit}
      [end]
[submenu] (user wallpapers)
  [wallpapers] (~/.backgrounds) {fbsetbg -f}
[end]
      [submenu] (Window Manager)
        [exec] (Edit Menus) {nedit ~/.fluxbox/menu}
        [submenu] (Style) {Which Style?}
          [stylesdir] (~/.fluxbox/styles)
          [stylesmenu] (Fluxbox Styles) {/usr/local/share/fluxbox/styles}
        [end]
        [config] (Config Options)
        [reconfig] (Reconfigure)
        [restart] (Restart)
      [end]
      [exit] (Log Out)
    [end]
    # end of menu file
```

or consult the rather complete sample `menu` file that comes with Fluxbox.

Chapter 10. Themes

10.1. Theme basics

This section was contributed by Justin Rebelo (mailto:rebelo@shaw.ca) aka "demerol".

What is a style and how does it work?

A style is basically a theme for Fluxbox. It is a simple ASCII text file that tells Fluxbox how to generate the appearance of the different components of the window manager. They are usually located in `~/.fluxbox/styles` and in the global Fluxbox share directory, which will vary depending on the method of installation used.

How do I make my own?

Start off by opening a style in your favorite text editor (I recommend `vim`). Look at the style, how it is structured and organized. Just looking at it will explain most of the questions you may have.

Structure of a style

The style is made up of a few major components which then have their own sub-directives. The **`toolbar`**, **`menu`** and **`window`** are the major components. The **`window.*`** directives control the appearance of your window frames, **`window.tab.*`** controls the appearance of the window tabs. **`menu.*`** controls the appearance of the popup menu that you see when you right click on the desktop. **`toolbar.*`** is the bar you will see at the top or bottom of your screen. The Slit (also called a dock, wharf, etc in other WMs) is controlled by the **`toolbar`** settings, too if you don't set its style specifically.

How do I change the appearance of the slit?

The slit usually uses the same options as the toolbar. Most of the time this should work reasonably well. If you want to specifically style the slit, there are three style directives you can use:

```
slit: [texture option]
slit.color: [color value]
slit.colorTo: [color value]
```

These commands work just like those for menu, window etc. when texturing the Slit.

Can I set a background image/color?

Somewhere in the style file you might see a line with **`rootCommand`** at the beginning and it will be

followed by a command to set the background color or image of the style. Fluxbox includes the programm `bsetroot` to set color and gradient background and `fbsetbg` to display images.

Can I add note/comments in my styles?

Sure, just start a line with a hash (**#**), a bang(**!**), or use C++ style comments (**//**).

I still have more questions...

Take a look at the styles provided with Fluxbox, you should be able to find the answer there or by trying different settings. If you still can't get it, stop by #fluxbox on OPN. My nick is demerol.

10.2. Advanced Theme Editing

The majority of this section is taken from or heavily influenced by the man page as of Fluxbox 0.1.13. Usually, the man page is the definite authority on this matter, but this document might be more enlightening to the beginning style author.

To understand how the style mechanism works, it is nice to know a little about how X11 resources work.

X11 resources consist of a key and a value. The key is constructed of several smaller keys (sometimes referred to as children), delimited by a period (**.**). Keys may also contain a asterisk (*****) to serve as a wildcard, which means that one line of typed text will match several keys. This is useful for styles that are based on one or two colors.

Fluxbox allows you to configure its three main components: the toolbar, the menus and the window decorations. The slit automatically inherits its style from the toolbar but can be styled differently if need be. The little window that shows the x-y position while dragging windows, borrows its style from the window's titlebar.

Here are some quick examples to illustrate the basic syntax:

Example 10-1. Toolbar Clock style

```
toolbar.clock.color:  green
```

This sets the color resource of the toolbar clock to **`green`**. Another example:

Example 10-2. Menu style

```
menu*color: rgb:3/4/5
```

This sets the color resource of the menu and all of its children to **`rgb:3/4/5`**. For a description of color names, see the X11 man pages. So this one also applies to **`menu.title.color`** and **`menu.frame.color`**. And with this:

Example 10-3. Font style

```
*font: -b&h-lucida-medium-r-normal-*-*-140-*
```

you set the font resource for all keys to this font name all at once. For information about the fonts installed on your system, you can use a program like `xfontsel`, `gfontsel`, or `xlsfonts`.

Now, what makes Fluxbox just so spectacular, is its ability to render textures on the fly. Texture descriptions are specified directly to the key that they should apply to, e.g.:

Example 10-4. Texture style

```
toolbar.clock:   Raised Gradient Diagonal Bevel1
toolbar.clock.color: rgb:8/6/4
toolbar.clock.colorTo: rgb:4/3/2
```

Don't worry, we will explain how these directives work. A texture description consists of up to five fields, which are as follows:

Table 10-1. Texture directives

Directive	Description
Flat / Raised / Sunken / Tiled	give the component either a flat, raised or sunken appearance. Tiled does only affect pixmaps and it will not scale them.
Gradient / Solid	draw either a solid color or a gradiented texture.
Horizontal / Vertical / Diagonal / Crossdiagonal / Pipecross / Elliptic / Rectangle / Pyramid	Select one of these texture types. They only work when Gradient is specified, too.
Interlaced	interlace the texture (darken every other line). This option is most commonly used with gradiented textures, but from Blackbox version 0.60.3 on (and thus in all versions of Fluxbox), it also works in solid textures.
Bevel1 / Bevel2	type of bevel to use. Bevel1 is the default bevel. The shading is placed on the edge of the image. Bevel2 is an alternative. The shading is placed one pixel in from the edge of the image.

Apart from the texture description, the option **`ParentRelative`** is also available, which makes the component appear as a part of its parent.

All gradiented textures are composed of two color values: the **`color`** and **`colorTo`** resources. When Interlaced is used in **`Solid`** mode, the **`colorTo`** resource is used to find the interlacing color.

The complete list of components and which kind of value they can contain can be found in Appendix C.

Now, that seems a long list, but remember, when you create your own style, you can easily set lots of keys with a single command, e.g.

Example 10-5. Typical short style

```
*color:             slategrey
*colorTo:           darkslategrey
*unfocus.color:     darkslategrey
*unfocus.colorTo:   black
*textColor:         white
*unfocus.textColor: lightgrey
*font:              lucidasans-10
```

Appendix A. Setting up `.xinitrc`/`.xsession`

Contributed by Verin.

The place of xinit in things

A window manager is just one more application for X11, like `netscape` or `gimp` or `xterm`. Many people new to X11 come to believe that X11 runs the window manager and the window manager runs programs. But that's not true. If configured right, you can run all your applications under X11, kill the window manager, and start another window manager up.

The *real* program that X11 runs, that runs other programs, is your `.xinitrc` or `.xsession` script. When X11 is started, your `.xinitrc` or `.xsession` script is run, and when the script is done, X11 comes down. Let me repeat that, its important: *when `.xinitrc` is finished, that is when X ends*. It isn't when your window manager exits.

Script layout

Well, first realize something you already know. When you type a command in a shell, you can't do anything else until that command is done, when it exits. Your `.xinitrc` or `.xsession` script is just the same. When it starts going through it, if it hits any program that takes a long time to run (like most X11 programs), it stops right there until that program is finished.

Ideally, you should only have one place where the script 'hangs'. And usually you want this to be at the end. So, if you have any programs you want to run under X11 before you get to this 'hang' spot, you should background them. You put an **`&`** at the end of the line. So, say you want `xclock` to run in addition to other things, put this line before your 'hang' spot:

```
xclock &
```

Now, the next thing is the **`exec`** thing you see, where lots of sources recommend how to add your window manager to your script. But honestly, its not really necessary, if you put your window manager on the last line of your script, it will hang there just fine without the **`exec`**.

So why the **`exec`**? Well, lets say you want to put lots of window manager start lines in your script, and you want only one to work. Well, with **`exec`** you can put your chosen start-line at the top. Because this is what **`exec`** means:

"Replace myself with this program, i.e. start it and terminate myself immediately when it finishes. "

So if you put an **`exec wmaker`** line atop of a **`exec enlightenment`** line, when `wmaker` is done, so is the script, it never gets to the next line.

See what I mean by being unnecessary? You could just put a bunch of commented-out window manager lines, and it would work just the same.

Another way to do things

As an alternative, you could start up your window manager *first*, and store the process ID in a environment variable:

```
wmaker & wmpid=$!
```

that puts it in the background (**`&`**) and puts the process id (**`$!`**) in a variable (**`wmpid`**). Then, to make your hang point, you can **`wait`**:

```
wait $wmpid
```

or you could hang on a program you always want to use, like maybe `gkrellm`, by just not backgrounding it. But remember that as soon as you terminate it, so will your X11 session.

Now, I use the `wait` method, because I like picking my window manager before I launch my dockapps and stuff. Also, before doing anything else, I like to change the settings on my X11 server, like the dpms, the screen saver, and even add some directories to my font path (fonts I don't want to install universally). And then after everything is done, I like to clean up my fontpath, mainly because if I ran a display manager, its not good at resetting the font path all the time.

Example A-1. `.xinitrc`

```
# turn off screen blanking and turn on energy star features
xset s off
xset dpms 600 60 60

# add my optional fonts to the font path
xset +fp "$X_FONTPATH"
xset fp rehash

# export the current environment, in case it needs to be debugged
env > ~/.xenv

# window manager
fluxbox & wmpid=$!

bbrun &
wmCalClock &
wmxmms &

# HANG POINT - wait for window manager to exit
wait $wmpid

# restore the x fontpath
xset fp default
```

Appendix B. Frequently asked questions (FAQ)

1. What is the Slit?

The first thing to know about the Slit is that it is *not* the Fluxbox taskbar.

The Slit is a place where dockable applications can 'dock'. We have an entire chapter devoted to the Slit: Chapter 7. Read it before asking any questions about what the slit is and how it works.

2. Is there a way to have Slit dockapps be in a certain order?

In Fluxbox versions after 0.1.10, yes. There's an explanation in Chapter 7.

3. How do I change the toolbar Time format?

Change this line in your `init`:

```
session.screen0.strftimeFormat: %a %d %H:%M
```

For information on the format, run **`man 3 strftime`** on your machine.

4. I make changes to my `~/.fluxbox/init`, but they are getting overwritten.

This is a bug in versions of Fluxbox prior to 0.1.8-bugfix2. Please upgrade to the latest version / bugfix before reporting this.

5. How do tabs work?

See Chapter 4.

6. It seems that I can't use the old fonts like Snap with AA... What's the deal?

As soon as you activate AA with Fluxbox, it relies on Freetype2 to render the characters. If you don't use AA, X11s native font rendering can be used. As FT2 does *not* support all the formats that X11 supports, the choice of fonts for Fluxbox becomes limited. As of this writing, FT2 supports the following formats:

- TrueType files (.ttf) and collections (.ttc)
- Type 1 font files both in ASCII (.pfa) or binary (.pfb) format
- Type 1 Multiple Master fonts
- Type 1 CID-keyed fonts
- OpenType/CFF (.otf) fonts
- CFF/Type 2 fonts
- Adobe CEF fonts (.cef)
- Windows FNT/FON bitmap fonts

Freetype 1, on the other hand, does only support TrueType fonts, although support for GX and OTF fonts can be enabled by using another library. Details can be found on the Freetype homepage.

7. I keep hearing about these Artwiz fonts. What gives?

There is an explanation in Appendix D

If you like the Artwiz fonts, but don't like the way they look in terminals, consider checking out the LFP fontpack, from the Linux Font Project (http://dreamer.nitro.dk/linux/lfp/). There are two sets of fonts there, The LFP Fixed-Width Fonts (good for terminals), and LFP Variable-Width Fonts (good for other things). The fixed-width fonts are also available for the Linux console.

8. How do I set my background?

This is explained in Chapter 6.

9. My background changes to an ugly one when I change themes.

There is a solution in Chapter 6.

10. Can I use my existing `.blackboxrc` for Fluxbox?

You may certainly, be sure to add some lines for titlebar and keygrabbing, though. It may also be wise to symlink your `blackboxrc` and `~/.fluxbox/init` together somehow.

11. How do i launch apps automatically on Fluxbox startup?

See Appendix A.

12. Can I use Blackbox styles (themes) with Fluxbox?

Yes. The tarballs for both packages should be 100% interchangeable. This should also stay true for the Waimea and Openbox projects, although I can't guarantee it. I've not seen such a promise from either of the two projects, but one of the goals of Fluxbox is to stay compatible with Blackbox styles.

13. How do I set up my `.xinitrc`/`.xsession`?

See Appendix A.

14. Is there KDE support?

Yeah, use the `configure` option **`--enable-kde`**. This ensures that KDE tray icons will appear in the slit.

15. Is there Gnome support?

Yeah, use the `configure` option **`--enable-gnome`**. This enables the Gnome hints. In Fluxbox versions 0.1.12 and later this is the default.

16. BBtools don't use my current style settings, even after restart.

Simply link your `~/.blackboxrc` to your `~/.fluxbox/init`, for example with this command:

```
$ ln -s ~/.fluxbox/init ~/.blackboxrc
```

17. The tabs look ugly with some styles, how to fix it?

0.1.14 or preceeding versions.

To make the tabs look (even more) pretty you'll need to add some extra entries to your desired style (theme). However, note that you don't have to do this, Fluxbox is very capable of setting the tabs to a proper color/style by itself, but if you want to have more control of how they look, you might want to add a few lines like this:

Example B-1. Customized tabs in style file

```
! -- tab style (for Fluxbox)
window.tab.justify:                     Right
window.tab.label.unfocus:               Flat Solid
window.tab.label.unfocus.color:         rgb:AC/AC/AC
window.tab.label.unfocus.textColor:     black
window.tab.label.focus:                 Raised Solid
window.tab.label.focus.color:           rgb:CC/CC/CC
window.tab.label.focus.textColor:       black
window.tab.borderWidth:                 1
window.tab.borderColor:                 rgb:10/10/10
window.tab.font:                        fixed
! --- end, tab style
```

Okay, so what does all this do then? Well the same stuff as any other thing in a theme, I'm sure you get it if you have ever made a Blackbox theme before (Chapter 10 might be interesting if you haven't).

Also note that a style containing these extra entries will still work perfectly in Blackbox, so you loose nothing by adding this.

18. How do I put icons on Fluxbox' desktop?

Fluxbox has its own companion program for this kind of functionality, it is called fbdesk (http://fluxbox.sourceforge.net/fbdesk). `fbdesk` is currently not packaged with fluxbox. There are also alternatives: the RoxFiler project, idesk (http://idesk.timmfin.net), xdesk (http://garuda.newmail.ru/xtdesk_e.dhtml).

19. Fluxbox 0.9.6 is slow...

Add the following line to your .xinitrc before you exec fluxbox:

```
export LC_ALL=C
```

should help on newer RedHats.

Appendix C. Theme Reference

C.1. Theme directives

This is a complete list of available theme directives specifying which values they can be assigned. See Chapter 10 for details on those directives.

Table C-1. Toolbar styles

toolbar	Texture
toolbar.height	Number
toolbar.color	Color
toolbar.colorTo	Color
Buttons	
toolbar.button	Texture or ParentRelative
toolbar.button.color	Color
toolbar.button.colorTo	Color
Color of unpressed button arrows	
toolbar.button.picColor	Color
Buttons in pressed state	
toolbar.button.pressed	Texture (e.g. **`Sunken`**) or **`ParentRelative`**
toolbar.button.pressed.color	Color
toolbar.button.pressed.colorTo	Color
Color of pressed button arrows	
toolbar.button.pressed.picColor	Color
Workspace label	
toolbar.label	Texture or **`ParentRelative`**
toolbar.label.color	Color
toolbar.label.colorTo	Color
toolbar.label.textColor	Color
Workspace label	
toolbar.workspace	Texture or **`ParentRelative`**
toolbar.workspace.pixmap	Pixmap
toolbar.workspace.color	Color
toolbar.workspace.colorTo	Color
toolbar.workspace.textColor	Color
toolbar.workspace.font	Font
Window label	
toolbar.windowLabel	Texture or **`ParentRelative`**
toolbar.windowLabel.color	Color

Window label	
toolbar.windowLabel.colorTo	Color
toolbar.windowLabel.textColor	Color
Clock	
toolbar.clock	Texture or **`ParentRelative`**
toolbar.clock.pixmap	Pixmap
toolbar.clock.color	Color
toolbar.clock.colorTo	Color
toolbar.clock.textColor	Color
toolbar.clock.font	Font
Iconbar if empty	
toolbar.iconbar.empty	Texture or **`ParentRelative`**
toolbar.iconbar.empty.pixmap	Pixmap
toolbar.iconbar.empty.color	Color
toolbar.iconbar.empty.colorTo	Color
toolbar.iconbar.empty	Texture or **`ParentRelative`**
Iconbar focused and unfocused	
toolbar.iconbar.focused	Texture or **`ParentRelative`**
toolbar.iconbar.focused.pixmap	Pixmap
toolbar.iconbar.focused.color	Color
toolbar.iconbar.focused.colorTo	Color
toolbar.iconbar.focused.textColor	Color
toolbar.iconbar.focused.font	Font
Text	
toolbar.justify	**`center`**, **`left`**, or **`right`**
toolbar.font	Font

Table C-2. Menu styles

Title	
menu.title	Texture
menu.title.color	Color
menu.title.colorTo	Color
menu.title.textColor	Color
menu.title.font	Font
menu.title.justify	**`center`**, **`left`**, or **`right`**
Frame	
menu.frame	Texture
menu.frame.color	Color
menu.frame.colorTo	Color
menu.frame.textColor	Color

Frame	
menu.frame.disableColor	Color
menu.frame.font	Font
menu.frame.justify	**`center`**, **`left`**, or **`right`**
Submenu bullets	
menu.bullet	**`empty`**, **`triangle`**, **`square`**, or **`diamond`**
menu.bullet.position	**`right`** or **`left`**
menu.submenu.pixmap	Pixmap
Highlighted item	
menu.hilite	Texture (e.g. **`Raised`**)
menu.hilite.color	Color
menu.hilite.colorTo	Color
menu.hilite.textColor	Color
menu.selected.pixmap	Pixmap
menu.unselected.pixmap	Pixmap

Table C-3. Window styles

Title	
window.title.focus	Texture
window.title.focus.color	Color
window.title.focus.colorTo	Color
window.title.unfocus	Texture
window.title.unfocus.color	Color
window.title.unfocus.colorTo	Color
window.title.height	Number
Label	
window.label.focus	Texture or **`ParentRelative`**
window.label.focus.color	Color
window.label.focus.colorTo	Color
window.label.focus.textColor	Color
window.label.unfocus	Texture or **`ParentRelative`**
window.label.unfocus.color	Color
window.label.unfocus.colorTo	Color
window.label.unfocus.textColor	Color

Table C-4. Handlebar styles

window.handle.focus.color	Color
window.handle.focus.colorTo	Color
window.handle.unfocus	Texture

window.handle.unfocus.color	Color
window.handle.unfocus.colorTo	Color

Table C-5. Resize grip styles

window.grip.focus	Texture
window.grip.focus.color	Color
window.grip.focus.colorTo	Color
window.grip.unfocus	Texture
window.grip.unfocus.color	Color
window.grip.unfocus.colorTo	Color

Table C-6. Window button styles

window.button.focus	Texture or **`ParentRelative`**
window.button.focus.color	Color
window.button.focus.colorTo	Color
window.button.focus.picColor	Color
window.button.unfocus	Texture or **`ParentRelative`**
window.button.unfocus.color	Color
window.button.unfocus.colorTo	Color
window.button.unfocus.picColor	Color
window.button.pressed	Texture
window.button.pressed.color	Color
window.button.pressed.colorTo	Color

Table C-7. Window frame styles

window.frame.focusColor	Color
window.frame.unfocusColor	Color

Table C-8. Tab styles

window.tab.justify	**`right`**, **`left`** or **`center`**
window.tab.label.unfocus	Texture
window.tab.label.unfocus.color	Color
window.tab.label.unfocus.textColor	Color
window.tab.label.focus	Texture
window.tab.label.focus.color	Color
window.tab.label.focus.textColor	Color

window.tab.borderWidth	Number of Pixels
window.tab.borderColor	Color
window.tab.font	Font

Table C-9. Font on window label styles

window.font	Font
window.justify	**`center`**, **`left`**, or **`right`**

Table C-10. Miscellaneous styles

Border drawn around all components	
borderWidth	Number of pixels
borderColor	Color
bevelWidth	Number of pixels
handleWidth	Number of pixels
frameWidth	Number of pixels
Command to be executed whenever the style is loaded	
rootCommand	Shell command, e.g. **`fbsetbg nicepiccy.jpg`**
Old BB 0.51 resources	
menuFont	Font
titleFont	Font

Appendix D. Artwiz fonts for Fluxbox

Introduction

The so-called Artwiz Fonts are fonts that were created (surprisingly enough) by a guy who calls himself Artwiz. You can download the fonts by aleczapka from sf (http://sourceforge.net/project/showfiles.php?group_id=95348) which are compatible with gtk2/kde3 apps or possibly older one from Han (http://www.xs4all.nl/~hanb/software/fluxbox/artwiz-fonts.tar.bz2).

You don't have to install the fonts by hand if you have the mandrake RPMs. They are already included. If you don't, well, here's how. There are two ways to install the artwiz fonts: system-wide and user-only.

System-wide install

If you want to install the fonts for all users on your system, download the tarball to `/tmp` and proceed like this:

```
# cd /usr/X11R6/lib/X11/fonts
# tar xjf /tmp/artwiz-fonts.tar.bz2
# cd fluxbox-artwiz-fonts
# mkfontdir
# chkfontpath -q -a /usr/X11R6/lib/X11/fonts/fluxbox-artwiz-fonts:unscaled
```

And restart the font-server. Note that your distribution might have a different location for system-wide fonts, like `/usr/share/fonts`. The directory above should be a sensible default, though.

User install

If you want to install the fonts for one user only, things are a bit easier. Download the tarball to your home directory, then:

```
$ tar xjf artwiz-fonts.tar.bz2
$ mv fluxbox-artwiz-fonts .fonts
$ mkfontdir $HOME/.fonts
```

Edit your `.xinitrc` or `.xsession` file (depends on on how you start X11) to contain the following line before any program calls:

```
xset +fp $HOME/.fonts
```

Then (re)start X11 and use `xlsfonts` or `xfontsel` to check if the fonts were recognized by the system.

Bugs

The artwiz-fonts sometimes conflict with your locale settings. To get them to work you may have to put the following at the beginning of your `.xinitrc` or `.xsession`:

```
export LC=C
export LC_ALL=C
```

These settings are for locales so if you are missing your fonts or have any other locale related problems remove them again. There is a different solution contributed by aleczapka below.

And here is the solution how to use your national settings and still be able to use the Artwiz Fonts in Fluxbox.

Locale settings fix

The solution is very simple. All you have to do is to fix your `fonts.alias` (and/or `fonts.dir`) file.

This will also fix problems with other applications (eg. Evolution and UTF-8). First you need to make proper `fonts.dir` file. It should be located in the directory where you have the Artwiz Fonts installed. If not, change to that directory and run `mkfontdir`.

The syntax of this file is simple. The first line contains only the number of fonts in the directory. All subsequent lines are of the form

```
font_filename fontname
```

Here's an example content of `fonts.dir` file:

Example D-1. `fonts.dir`

```
14
glisp.pcf.gz -artwiz-glisp-medium-r-normal--11-110-75-75-p-90-iso646.1991-irv
gelly.pcf.gz -artwiz-gelly-medium-r-normal--10-100-75-75-p-90-iso646.1991-irv
edges.pcf.gz -artwiz-edges-medium-r-normal--10-100-75-75-m-50-iso646.1991-irv
nu.pcf.gz nu
drift.pcf.gz drift
cure.pcf.gz cure
aqui.pcf.gz aqui
lime.pcf.gz -artwiz-lime-medium-r-normal--10-100-75-75-m-50-iso646.1991-irv
snap.pcf.gz -artwiz-snap-medium-r-normal--10-100-75-75-p-90-iso646.1991-irv
```

What interests us are entries that only consist of a short fontspec, in this case the fonts Nu, Drift, Cure and Aqui. The problem is that they lack full X11 font names.

Change the file to this:

```
14
glisp.pcf.gz -artwiz-glisp-medium-r-normal--11-110-75-75-p-90-iso646.1991-irv
gelly.pcf.gz -artwiz-gelly-medium-r-normal--10-100-75-75-p-90-iso646.1991-irv
edges.pcf.gz -artwiz-edges-medium-r-normal--10-100-75-75-m-50-iso646.1991-irv
nu.pcf.gz -artwiz-nu-medium-r-normal--11-110-75-75-p-90-iso646.1991-irv
drift.pcf.gz -artwiz-drift-medium-r-normal--11-110-75-75-p-90-iso646.1991-irv
cure.pcf.gz -artwiz-cure-medium-r-normal--11-110-75-75-p-90-iso646.1991-irv
aqui.pcf.gz -artwiz-aqui-medium-r-normal--11-110-75-75-p-90-iso646.1991-irv
lime.pcf.gz -artwiz-lime-medium-r-normal--10-100-75-75-m-50-iso646.1991-irv
snap.pcf.gz -artwiz-snap-medium-r-normal--10-100-75-75-p-90-iso646.1991-irv
```

The last thing to do is to fix your `fonts.alias` file (to use the fonts with different encodings than iso646).

Syntax is **`font_alias font_name`**. Eg. to make artwiz fonts working with ISO-8859-2 encoding make such an alias (all in *one* line).

```
-artwiz-anorexia-medium-r-normal--11-110-75-75-p-90-iso8859-2
-artwiz-anorexia-medium-r-normal--11-110-75-75-p-90-iso646.1991-irv
```

You can also have a peek at my fonts.dir (http://fluxbox.sourceforge.net/download/fonts.dir) and fonts.alias (http://fluxbox.sourceforge.net/download/fonts.alias) files, they support ISO-8859-1, ISO-8859-2, and iso10646-1 (UTF-8). If you set everything as written above, you don’t need to change your LC_* flags to C or POSIX. And all applications (not just Fluxbox), will work as they should; not complaining about "can’t convert character set" or similar.

Appendix E. Debugging

General information

Debugging is a skill which you can learn. There are quite a few good documents that describe how to make a good bug reports. Please read them before you get into action, your bug won't run away and it won't kill you. Rather we want to kill the bug and we want your help :-) So lets go to battle well prepared. Two nice documents are the Bugzilla Bug Reporting HOWTO (http://www.mozilla.org/quality/bug-writing-guidelines.html) and Simon Tathams How to Report Bugs Effectively (http://www.chiark.greenend.org.uk/~sgtatham/bugs.html).

Due to the nature of some unofficial patches, we can't support them all. This means (for you) that you should check if Fluxbox crashes *without* the patches you applied, too. If the patches of your Fluxbox were applied by some distribution maintainer, contact him about the issue.

If you require more help with debugging than this guide provides, you may also see if one of people on #fluxbox at irc.freenode.net is of help. As the main developers and quite a bunch of qualified people are usually there, chances are that we can sort things out.

Specific Fluxbox things

Output from Fluxbox

Like any other application Fluxbox sends all messages to the console it was started from. The problem is that under normal circumstances all output goes to the *text* console. Change your Fluxbox start line to **`exec xterm`** or whatever your favorite terminal emulator may be. Start X11 and start Fluxbox in that xterm, and lo and behold, you can easily see all messages from Fluxbox.

The info we want

We want to know a lot of things. Make sure you know all about them.

- Your operating system / distribution and its version.
- Fluxbox version, if CVS (or development tarball) then which day?
- When did it happen? Did you do something? Can you reproduce it?
- Your settings. They can be found in `~/.fluxbox/init`

What to do with core dumps.

So if you ever get a core dump with Fluxbox do the following: Go to the channel #fluxbox on OPN and tell fluxgen you got a core dump. Give him all the info he asks for. He will probably also ask you to do

the following. It's quite a lot of work but anybody with a bit of common sense and some Unix experience can do it. Oh, and you need the GNU debugger, called `gdb`.

Rebuild Fluxbox

Yes you read it right. To become a real debugger you have to rebuild Fluxbox with debugging symbols.

Build Fluxbox as usual and add the following option when you do the make:

```
$ CFLAGS=-Wall -g3 CXXFLAGS=-Wall -g3 make
```

If you use the source RPMs from the Fluxbox site or your distribution you can do something like this:

```
$ su
# rpm -ivh fluxbox-0.1.11.1mdk.src.rpm
# cd /usr/src/RPM/SPECS
# env DEBUG=true rpm -ba fluxbox.spec
# rpm -Uvh --force /usr/src/RPM/RPMS/i686/fluxbox*
# exit
$ mkdir -p ~/src/fluxbox
$ cp -R /usr/src/RPM/BUILD/fluxbox* ~/src/fluxbox
```

Go to the Fluxbox directory (So we get the core dump in the right place). The shell has a nifty feature that disables core-files so make sure you really get a core file with:

```
$ ulimit -c unlimited
```

Start X11 and let's debug. Do whatever it takes to make Fluxbox dump core. And now we start debugging:

```
$ gdb fluxbox core
```

And issue this command in `gdb` (the first part is `gdb`s prompt, don't type it :)):

```
(gdb) where
```

Now you get a lot of output. Fluxgen wants to know all the output that starts with a `#`.

Now paste that into an e-mail to fluxgen and also attach the four settings files in `~/.fluxbox/`.

Please do not remove the core and the source code directory yet. Fluxgen may want to ask you a few extra question and then you will need them. Please do not send the core file to fluxgen if he does not ask specifically - since it is specific to your system, it most probably is of no use to him and core files tend to be quite big.