fluxbox

Name

`fluxbox` — A lightweight window manager for the X Windowing System

Synopsis

fluxbox [-vhi] [-rc rcfile] [-l logfile] [-d display] [-screens scr,scr]

DESCRIPTION

Fluxbox provides configurable window decorations, a root menu to launch applications and a toolbar that shows the current workspace name, a set of application names and the current time. There is also a workspace menu to add or remove workspaces. The ‘slit’ can be used to dock small applications, e.g. most of the bbtools can use slit.

Fluxbox can iconify windows to the toolbar, in addition to adding the window to the *Icons* submenu of the workspace menu. One click and they reappear. A double-click on the titlebar of the window will *shade* it, i.e. the window will disappear, only the titlebar remains visible.

Fluxbox uses its own graphics class to render its images on the fly. By using style files, you can determine in great detail how your desktop looks. Fluxbox styles are compatible with those of Blackbox 0.65 or earlier versions, so users migrating can still use their current favourite themes.

Fluxbox supports the majority of the Extended Window Manager Hints (EWMH) specification, as well as numerous other Window Hinting standards. This allows all compliant window managers to provide a common interface to standard features used by applications and desktop utilities.

OPTIONS

Fluxbox supports the following command-line options:

-d display | -display display

> Start Fluxbox on the specified display. Programs started by Fluxbox will share the DISPLAY environment variable also.

-h | -help

> Display command line options.

-i | -info

Display useful information concerning the defaults and compiled-in options.

-l logfile

Starting Fluxbox with this option will designate a file in which you want to log events to.

-rc rcfile

Use a different config file other than the default ~/.fluxbox/init.

-v | -version

The version of Fluxbox installed.

STARTING FLUXBOX

Fluxbox comes with a program called startfluxbox(8) usually located wherever you installed Fluxbox. This script provides you with many options and variables that can be set when starting Fluxbox. To actually call Fluxbox and begin using it, you should place exec startfluxbox in your ~/.xinitrc or ~/.xsession (depending on your distributions and/or display manager) as the last executed command. This is assuming that the location of fluxbox(8) and startfluxbox(8) are in your shell's $PATH. Also note that you may need to create the $_{\text{/.xinitrc file or your setup may use}}$ /.xsession instead, depending on your X setup. For more information on your shell, please visit your shell's manual page.

By using fluxbox -i you'll see the defaults used by Fluxbox. These are what Fluxbox looks for upon startup. In the list of 'Defaults:' you'll see a menu file location, this is where you can provide a system-wide menu file for your users.

On exit or restart, Fluxbox will save user defaults in the file ~/.fluxbox/init. Resources in this file can be edited by hand. Fluxbox also has many tools to edit these, look through the main menu once Fluxbox has started to find different ways of managing your session.

USING FLUXBOX

When Fluxbox is initially started, it can be very intimidating for the new user. There really isn't much on the screen and it becomes tough to know where to begin. We'll give a quick summary of the common things you'll find in this section. However, we recommend that you consult the referenced sections of this manual to further develop your understanding of what you can do with Fluxbox.

Root Window (Main)

Looking at the Fluxbox desktop immediately after startup you'll generally see only one thing. The toolbar. If you were to right click (mouse button 3) somewhere else blank, you would be able to access

the root menu, a middle click (mouse button 2) shows you the Workspace menu. It is there, you just won't be able to see it while empty, but the slit is also hiding out somewhere as well.

Root Menu and Workspace Menu

From the Root Menu you can launch applications and configure fluxbox. The Workspace Menu shows all windows and on which workspaces they are. See section MENUS on how to customize thess menus.

Toolbar

The toolbar contains up to eight fields/tools:

- **Workspace Name**: Name of the current visible workspace
- **Iconbar**: Show Windows
- **System Tray**: Area for applets
- **Clock**: Date and Time
- **Workspace Arrows**: Previous/Next Workspace
- **Window Arrows**: Previous/Next Application Window

The contents of the toolbar can be configured in the 'init' file, we discuss the 'init' file at great length in the RESOURCES section.

Slit

Initially you won't be able to see the slit. It is there, it just isn't being utilized yet. The slit confuses some people initially. Think of it as a dock where you can place smaller programs. If you've looked at any screenshots on the offical Fluxbox web site you'll have noticed some small programs on the edge of some of the screens. These were more than likely those docked programs in the slit. To learn more about the slit, we have an entire section below that goes into detail about the options you have.

Layers

Fluxbox manages the following layers (from highest to lowest layer):

- Above Dock
- Dock
- Top
- Normal
- Bottom

- Desktop

Windows on a higher layer will always appear above those on a lower one. These layers can be used on application windows, the slit or the toolbar. You can assign applications to a certain layer by specifying it in the ‘apps’ file. We discuss the ‘apps’ file in the APPLICATIONS section. We discuss layers in more detail in the LAYERS section.

Focus Model

The window that has the focus is the one that receives key and mouse events. The focus model is selectable via the Configuration menu located in the root menu. We’ll discuss the different types of focus below in the FOCUS MODEL section.

Windows

A left click (mouse button 1) on any part of the window’s border will raise it. Dragging then moves the window to another part of the desktop. Dragging the resize grips at the left and right bottom corners resizes the window. Middle clicking on a border or titlebar will immediately lower the window. Right clicking on a border or titlebar pops the Window menu up. The commands in this menu alone are discussed in detail in the Window Menu section of MENUS.

Tabs

Fluxbox allows windows to be ‘grouped’ by middle clicking and holding on a window’s title bar and dragging it onto another window. This will ‘tab’ the titlebars, allowing you the user to select each window individually. This ‘tabbing’ allows you to put multiple applications in one location on the desktop, and do several operations (for example, moving or resizing) to all windows in the group.

Miscellaneous

When you want to drag a window, but cannot see either the bottom handle or its titlebar you can press (and hold!)

```
ALT + Left Mousebutton (mouse button 1)
```

and move it anywhere in the current workspace. This key combination can also be used to raise a partially visible window.

The key combination

```
ALT + Right Mousebutton (mouse button 3)
```

will allow you to resize the window. These can be disabled in the resource file with:

```
session.session0.useMod1: <boolean>
```

MENUS

Fluxbox installs a default menu file in /usr/local/share/fluxbox/menu. You can also use fluxbox -i to confirm this action. Of course this system-wide menu can be customized for all users at once, but it is also possible to create an individual menu file for each user. By convention, users create a menu file in ~/.fluxbox . Once you've created your own menu file, you'll want to make sure that you properly declare this location in your 'init' file so that Fluxbox knows where to look. The value you'll want to add or change is:

```
session.session0.menuFile: <menufile>
```

For this change to take effect, Fluxbox must be restarted. Be sure that your menu is usable, then choose 'Restart' from the default Fluxbox root menu. This restart is only necessary if you make changes to the 'init' file, otherwise a 'Reload Config' is acceptable. A menu reload can also be forced by sending SIGUSR2 signal (see the SIGNALS section).

Root Menu

The root menu is where you can change different aspects of Fluxbox by simply clicking on a menu item. Most of the changes in this menu can also be done in the 'init' file. However it makes it very easy to change certain options without having to open up an editor and find the resource. In the root menu, you usually have a 'fluxbox menu' or 'Settings' submenu, where you will find lots of different options. We'll take a look at most, if not all, of those here.

- **Configure**: The next level under this menu is where you can set certain resources and really begin to cus- tomize the look and feel of your desktop.
- **System Styles**: This is where the standard styles are listed. You can select one of these by clicking on it. You may have to 'reload' the config or 'restart' to get every graphical element to change to the new style. System styles are located in /usr/local/share/fluxbox/styles/ upon a default install. Remember that you can confirm this with fluxbox -i.
- **User Styles**: ~/.fluxbox/styles This is the location where you will store new styles that you grab from the Internet. If you create your own styles this is also where you will put yours (provided that you follow the *standards* described in fluxstyle(1)).
- **Workspace List**: This is a list of the workspaces configured in your 'init' file. If there are programs running on any of the workspaces, they will be listed one level down.
- **Tools**: Listed here are different tools that you can use. You can rename your workspace, run programs from a command line or regenerate your menu.

- **Window**: Allows you to switch your window manager. (Only listed if you have other window managers/desktop environments installed.)
- **Lock Screen**: Locks the screen. . .
- **Fluxbox Command**: A little Commandline will popup where you can enter a Fluxbox command.
- **Reload Config**: Use this to reload any menu files or style files. Just a basic re-read of the files by a running Fluxbox.
- **Restart**: Restart the whole darn thing, this rereads files and redraws all graphical elements.
- *Exit: Exits Fluxbox and shuts down the X Window server.

Configuration Menu

This menu offers the opportunity to set up fluxbox. It ca also achieved by editing the init file, but this is a easier and faster way to most users.

- **Focus Model**: Please read the FOCUS MODEL section at the end of this manual.
- **Slit**: This Menu can be opend by right clicking the slit (if visible).
- **Placement**: This lets you set the position of the slit.
- **Layer**: Look above for the layer priorities.
- **Auto hide**: If enabled, the slit will disappear after a given amount of time and hide from the view of the user. You can make it appear if you move the mouse to the edge of the desktop where the slit is psitioned.
- **Maximize over**: If this is enabled, all windows, if you maximize them, will stretch over/under the slit. Otherwise the will be limited to the slit's edge.
- **Alpha**: By changing the value the slit (only the decoration not the apps in the slit) will become transparent. 0 (transparent) - 255 (opaque)
- **Slit direction**: Changing the value will set the slit's direction for ordering apps sitting in the slit. There is no effect with only on application.
- **Clients**: This submenu lets you reorder the the applications running in the slit. You are able to hide apps from the slit by unselecting them in the list showing. This will not kill the app. You can make them appear by selecting them in the list. The "Save SlitList" option saves the new order to you slitlist located in ~/.fluxbox (useful if you reordered the apps with the cycle option).
- **Toolbar**: Please take a look at the "Configuration via the Toolbar Menu" part of the TOOLBAR section.
- **Image Dithering**: Enable or disable dithering of images.
- **Opaque Window Moving**: If enabled, you will see the window content while dragging it. Otherwise the window will be shown as a "border".
- **Full Maximization**: Enabling this will override the seperate settings for the slit/toolbar. Windows will always maximize over/under both of them.
- **Focus New Window**: If enabled, a newly opend window will gain focus.

- **Focus Last Window on Workspace**: This focuses the last window if switching back to a worspace if the option is enabled.
- **Windows Warping**: If enabled, you can drag windows from one to another workspace.
- **Desktop MouseWheel Switching**: You will be able to change the workspace with your mousewheel if used on the desktop or over the toolbar if the option is enabled.
- **Decorate Transient Windows**: With this option enabled all temporary windows will have a border and grips.
- **Click Raises**: If enabled a click anywhere on a window area (including the decorations) will raise it. Otherwise you can only raise it by clicking the titlebar.
- **Transparency**: This sets the transparency for an focused, unfocused window and the menu.

Window Menu

The Window menu is displayed when you right click on the titlebar or border of a window. The options available are:

- **Send To...**: Send window to another workspace. When you select the workspace with a middle click, Fluxbox will send you along with the application to the selected workspace.
- **Shade**: Shade the window (display the titlebar only).
- **Iconify**: Iconify window. The 'icon' can be found in the Icons submenu of the workspace menu as well as in the toolbar (if a Toolbar mode showing Icons is selected).
- **Maximize**: (Un)Maximize window. Depending on your toolbar and slit configuration, maximize may cover them. You can use the different mouse buttons for different aspects of maximize function.
  - Button 1 (Un)Maximize as normal.
  - Button 2 (Un)Maximize window vertically.
  - Button 3 (Un)Maximize window horizontally.

- **Raise**: Raise the window.
- **Lower**: Lower the window.
- **Stick**: (Un)Stick window. A 'stuck' window will always be displayed on all workspaces.
- **Next Client**: Activate the next client in this window's group.
- **Prev Client**: Activate the previous client in this window's group.
- **Layer...**: Change the layer of this window.
- **Remember...**: Specify which window settings should be stored in the apps file, covered later on in the APPLICATIONS section.
- **Close**: Close the application softly.

Workspace Menu

The workspace menu can be found by middle clicking on the background. A menu will popup giving you the option to add or remove a workspace. You will also see your workspaces listed there, in a lower menu under these the programs that are running on those respective workspaces will be displayed. Last but not least you will notice the Icons menu. This is for applications which have been 'iconified'.

Menu Behavior

The behavior of the submenus in a menu can be configured in the 'init' file, with the following entries (default for both is 0):

```
session.screen0.menuDelay: <msec>
session.screen0.menuDelayClose: <msec>
```

Menu Syntax

There are up to four fields in a menu line. They are of the form:

```
[tag] (label|filename) {command|filename} <icon file>
```

The supported tags are:

[begin] (label)

> This tells Fluxbox to start parsing the menu file. This tag is required for Fluxbox to read your menu file. If it cannot find it, the system default menu is used in it's place.

[end]

> This tells Fluxbox that it is at the end of a menu. This can either be a submenu or the main root menu. There must be at least one of these tags in your menu to correspond to the required [begin] tag.

[exec] (label) {command}

> Inserts a command item into the menu. When you select the menu item from the menu, Fluxbox runs *command*.

[exit] (label)

> Inserts an item that shuts down and exits Fluxbox. Any open windows are reparented to the root window before Fluxbox exits.

[include] (file-or-directory-name)

Parses the file specified by filename inline with the current menu. The filename can be the full path to a file or it can begin with ~/, which will be expanded into your home directory. If the path is a directory, then all files in the directory are included.

[nop] (label)

Insert a non-operational item into the current menu. This can be used to help format the menu into blocks or sections if so desired. This tag does support a label, but one is not required in which case a blank item will be used instead.

[separator]

This will create a nice separation line. Useful for splitting up sections in a *pretty* way.

[style] (label) {filename}

This tells Fluxbox to insert an item that, when selected, reads style file named filename and apply the new textures, colors and fonts to the current running session.

[stylesmenu] (directory)

Reads all filenames from the specified directory, assuming that they are all valid style files, and creates menu items in the current menu for every filename, that, when selected by the user will apply the selected style file to the current session. The labels that are created in the menu are the filenames of the style files.

[stylesdir] (label) {directory}

Creates a submenu entry with label (that is also the title of the new submenu), and inserts in that submenu all filenames in the specified directory, assuming that they are all valid style files (directories are ignored) in the same way as the [stylesdir] command does. Both [stylesdir] and [stylesmenu] commands make it possible to install style files without editing your init file.

[submenu] (label) {menutitle}

This tells Fluxbox to create and parse a new menu. This menu is inserted as a submenu into the parent menu. These menus are parsed recursively, so there is no limit to the number of levels or nested submenus you can have. The title for the new menu is optional, if none is supplied, the new menu's title is the same as the item label. An [end] tag is required to end the submenu.

[reconfig] (label)

When selected this item re-reads the current style and menu files and applies any changes. This is useful for creating a new style or theme, as you don't have to constantly restart Fluxbox every time you save your style. However, Fluxbox automagically rereads the menu whenever it changes.

[restart] (label) {command}

This tells Fluxbox to restart. If command is supplied, it shuts down and runs the command (which is commonly the name of another window manager). If the command is omitted, Fluxbox restarts itself.

[config] (label)

Inserts a Fluxbox native submenu item, containing numerous configuration options concerning window placement, focus style, window moving style, etc.

[wallpaper] (label)

This allows you to list your backgrounds. This tag is built in to use fbsetbg(1) and allows you to simply click on an image to set your wallpaper. See? Fluxbox makes it easy...

[workspaces] (label)

This tells Fluxbox to insert a link to the workspaces menu directly into your menu. This is handy for those users who can't access the workspace menu directly (e.g. if you don't have a 3 button mouse, it is rather hard to middle click to show the workspace menu).

Any line that starts with a *#* or *!* is considered a comment and ignored by Fluxbox. Also, in the label/command/filename fields you can escape any character. Using \ inserts a literal back-slash into the label/command/filename field.

```
Menu Example
  # Fluxbox menu file
  [begin] (Fluxbox)
    [exec] (rxvt) {rxvt -ls} </usr/X11R6/share/icons/terminal.xpm>
    [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] (Window Manager)
      [exec] (Edit Menus) {nedit ~/.fluxbox/menu}
      [submenu] (Style) {Which Style?}
        [stylesdir] (~/.fluxbox/styles)
        [stylesmenu] (Fluxbox Styles) {@pkgdatadir@/styles}
      [end]
      [config] (Config Options)
      [reconfig] (Reconfigure)
      [restart] (Restart)
    [end]
    [exit] (Log Out)
  [end]
```

TOOLBAR

The toolbar is a small area to display information by Fluxbox like a clock, the identifier for the workspaces, a systemtray or a taskbar (iconbar) that can contain the running programs. The color, look, font etc. is defined in the the style and can't be defined as a global setting.

The parts of the Toolbar can be enabled/disabled in the Init-File with the arguments given to the line:

```
session.screen0.toolbar.tools
```

The order and the count of the Tools is freely selectable and has to be seperated by a ",". E.g.:

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

The possible parts (Tools) of the Toolbar are:

- **Clock**: This will show an area to display a clock and the date according to the format specification listed in "man strtftime"
- **Iconbar**: This is the area that contains all windows (all running applications, all minimized windows or maybe no window, all depending on the Toolbar Settings).
- **Systemtray**: The Systemtray can hold Applications that are made to sit in it.
- **WorkspaceName**: This displays the name of the actual name of the Workspace.
- **PrevWorkspace**: This displays an arrow that allows to switch to the next Workspace left of the actual. Same as MouseWheelDown with "Desktop MouseWheel Switching" enabled.
- **NextWorkspace**: This displays an arrow that allows to switch to the next Workspace right of the actual. Same as MouseWheelUp with "Desktop MouseWheel Switching" enabled.
- **PrevWindow**: This displays an arrow that allows to gain focus of the previous visible window on the actual workspace.
- **NextWindow**: This displays an arrow that allows to gain focus of the next visible window on the actual workspace.

The Toolbar can be configured in two ways. Either through the Configure-Menu for the Toolbar, which is accessable in the Configuration Part of the Root-Menu or with a right-click on the Workspace Name/Arrows/Clock in the Toolbar, or by editing the Init-File by hand (Check the RESOURCES section for more information about that).

Configuration via the Toolbar Menu

All Changes work on the fly and you can notice them immediately, except for a change of the "Toolbar Alpha", that needs a restart to make the change visible:

- **Visible**: Sets the toolbar either to visible or invisible (Well, this should be obvious).

```
session.screen0.toolbar.visible: <boolean>
```

- **Auto hide**: If this is enabled the toolbar will disappear after a defined time when the mouse-pointer leaves the area of the toolbar. It will slide in when the cursor hits the remaining edge of the toolbar. The delay-time can be set in init.

  ```
  session.screen0.toolbar.autoHide: <boolean>
  session.autoRaiseDelay: <int>
  ```

- **Toolbar width percentage**: Sets the width of the toolbar in percent. Use the left mouse button to decrease and the right mouse-button to increase the value. The value can be from 0-100.

  ```
  session.screen0.toolbar.widthPercent: <int>
  ```

- **Maximize Over**: Enabling this option will prevent windows from maximizing over the toolbar. With this switched on they will only dock to the edge of the bar. To use this option, "Full Maximazition" from the fluxbox Configuration menu has to be DISABLED. Otherwise this option will not work.

  ```
  session.screen0.toolbar.maxOver: <boolean>
  session.screen0.fullMaximization: <boolean>
  ```

- **Layer...**: This sets the layer on which the toolbar is set. With this you can set the toolbar to "Always on top".

  ```
  session.screen0.toolbar.layer: <layer>
  ```

- **Placement**: Sets the toolbar to either the top or the bottom edge of the screen with a left, right or center alignment

  ```
  session.screen0.toolbar.placement: <direction>
  ```

- **Alpha**: This sets the alpa value for the toolbar. Use the left mouse-button to decrease and the right mouse-button to increase the value. 0 is invisible, 255 is not transparent at all.

  ```
  session.screen0.toolbar.alpha: <int>
  ```

- **Iconbar Mode**:

  ```
  Specifies the mode of the iconbar:
  ```

- **None**: will show not a single window
- **Icons**: will only show the windows of all workspaces that are minimzed (iconified)
- **NoIcons**: will only show the windows of all workspaces that are not minimzed (iconified)
- **WorkspaceIcons**: will only show the windows of the current workspace that are minimzed (iconified)
- **WorkspaceNoIcons**: will only show the windows of the current workspace that are not minimzed (iconified)
- **Workspace**: will show all windows of the current workspace
- **All Windows**: will show all windows of all workspaces

  ```
  session.screen0.iconbar.mode: <mode>
  ```

- **Alignment**:
- **Left**: all Icons/Windows will be left aligned according to the width set in init
- **Relative**: all Icons/Windows will be averaged so that the iconbar will always be completely filled
- **Right**: all Icons/Windows will be left aligned according to the width set in init

```
session.screen0.iconbar.alignment: <alignment>
session.screen0.iconbar.iconWidth: <int>
```

- **Show Pictures**: If enabled the iconbar will show the Application's Icon (if it is available)

  ```
  session.screen0.iconbar.usePixmap: <boolean>
  ```

- **Clock**: Lets you switch between the 00:00am - 12:00pm and 00:00-24:00 notation
- **Edit Clock Format**: clicking this entry will pop up a little window in which the clock format according to *man strftime* can be set.

  ```
  session.screen0.strftimeFormat: <format>
  ```

RESOURCES

Usually the ~/.fluxbox/init resource file is created and maintained by Fluxbox itself. You can use the [config] menu to set most of these options. However, we'll cover all of the resource options that are available to the user. If you edit this file while Fluxbox is running, you must 'restart' as to reload the resource options.

When running Fluxbox in a multiple desktop environment the screen0 key can also be screen1, screenN etc. You can customize the behaviour of Fluxbox on each desktop accordingly. Here is an example, and a favourite of the Fluxbox documentation manager:

```
session.screen0.toolbar.onTop: False
session.screen0.toolbar.autoHide: True
session.screen0.toolbar.placement: BottomCenter
session.screen0.toolbar.widthPercent: 42
session.screen0.slit.onTop: False
session.screen0.slit.autoHide: True
session.screen0.slit.placement: TopLeft
session.screen0.slit.direction: Vertical
session.screen0.strftimeFormat: %I:%M %p
session.screen1.toolbar.onTop: True
session.screen1.slit.autoHide: False
session.screen1.slit.placement: CenterRight
session.screen1.slit.direction: Vertical
session.screen1.strftimeFormat: %a %d %R [%s]
```

Here are the resources that are currently available:

```
session.screen0.menu.alpha: <integer>
session.screen0.slit.alpha: <integer>
session.screen0.toolbar.alpha: <integer>
session.screen0.window.focus.alpha: <integer>
session.screen0.window.unfocus.alpha: <integer>
      These resources are available to the user to set different lev-
      els of transparency for different components of Fluxbox.  Each
      one accepts a value between 0-255, 255 being opaque and 0 being
```

```
      completely transparent. Default: 255

session.screen0.slit.autoHide: <boolean>
session.screen0.toolbar.autoHide: <boolean>
      The autoHide resources allow the user to set the behaviour of
      the toolbar and slit.  This behaviour can be that they disappear
      when they are not being used actively by the user, or they
      remain visible at all times.  Default: <boolean>

session.screen0.desktopwheeling: <boolean>
session.screen0.toolbar.wheeling: <boolean>
      These set the ability to utilize the users mouse scroll wheel.
      Setting these values to '<boolean>' allows the user to essentially
      scroll through workspaces or applications on the toolbar.
      Default: <boolean>

session.screen0.slit.layer: <layer>
session.screen0.toolbar.layer: <layer>
      With these two resources, you can set the layer you want the
      toolbar and the slit to appear on.  Please read the LAYER sec-
      tion for more information.  Default: Desktop

session.screen0.slit.onTop: <boolean>
session.screen0.toolbar.onTop: <boolean>
      A user can set whether or not the toolbar or slit are always on
      top of the screen.  Setting these resources will put the slit
      and toolbar above everything visible in the window.  Default:
      False

session.screen0.slit.placement: <placement>
session.screen0.toolbar.placement: <placement>
      These allow a user to place the slit and toolbar where ever they
      like.  Possible options are:
      -   BottomCenter
      -   BottomLeft
      -   BottomRight
      -   LeftCenter
      -   RightCenter
      -   TopCenter
      -   TopLeft
      -   TopRight

session.screen0.slit.maxOver: <boolean>
session.screen0.toolbar.maxOver: <boolean>
      Setting these to '<boolean>' will allow application windows to maximize
      over the complete screen.  Setting to '<boolean>' allows the slit and
      toolbar to hold their territory and will always be visible when an
      application is maximized. Default: <boolean>

session.screen0.toolbar.height: <integer>
      Set the height of the toolbar.  Default: 0
      If the value is set to 0, the style file will gain control over the
      toolbar height. It is possible to set a fixed height by changing it in
      the init to something >0.
```

session.screen0.toolbar.visible: <boolean>
      The user can set whether they want to have a toolbar on screen at
      all. Setting to '<boolean>' removes the toolbar from the screen. This
      ultimately depends on whether or not the toolbar was compiled into
      the Fluxbox build. The default is that the toolbar will be visible.
      Default: <boolean>

session.screen0.toolbar.widthPercent: <integer>
      This resource sets the width of the toolbar on the screen to
      integer. Default: 100

session.screen0.toolbar.tools: <tools>
      This resource specifies the tools plugged into the toolbar. Read
      the TOOLBAR section in this manual for a description of each of
      these. Possible tools::
      -   clock
      -   iconbar
      -   nextwindow
      -   prevwindow
      -   nextworkspace
      -   prevworkspace
      -   systemtray
      -   workspacename

session.screen0.slit.onhead: <integer>
session.screen0.toolbar.onhead: <integer>
      For those that have dual head systems, users can set this value
      to the number of the screen where they would like to see the
      slit and toolbar. Default: 0

session.screen0.iconbar.iconWidth: 70
session.screen0.iconbar.mode: <mode>
      This value is set in the Iconbar Mode menu. The available
      options are::
      -   All Windows
      -   Icons
      -   None
      -   Workspace
      -   WorkspaceIcons

session.screen0.iconbar.usePixmap: <boolean>
      This is also set in the Iconbar Mode menu. When set to '<boolean>'
      this will show the native icon of applications. Default: <boolean>

session.screen0.iconbar.iconTextPadding: 10l
session.screen0.iconbar.deiconifyMode: Current
session.screen0.iconbar.wheelMode: Screen
session.screen0.iconbar.alignment: <position>
      This value should be changed in the Iconbar Mode menu. Default:
      Relative

Available options:
-   Left: Fixed width, aligned left
-   Relative

```
-   Right: Fixed width, aligned right

session.screen0.iconbar.clientWidth: <integer>
      Used to specify the iconbar button width for Left/Right align-
      ment.  Default: 0

session.screen0.overlay.lineWidth: 1
session.screen0.overlay.lineStyle: LineSolid
session.screen0.overlay.joinStyle: JoinMiter
session.screen0.overlay.capStyle: CapNotLast
session.screen0.slit.direction: Vertical
session.screen0.strftimeFormat: <date>
      This adjusts the way the current time is displayed in the tool-
      bar.  The strftime(3) format is used.  Default: %I:%M %p

session.screen0.tab.alignment: Left
session.screen0.tab.height: 16
session.screen0.tab.placement: Top
session.screen0.tab.rotatevertical: True
session.screen0.tab.width: 64
session.screen0.followModel: Ignore
session.screen0.rowPlacementDirection: LeftToRight
session.screen0.colPlacementDirection: TopToBottom
session.screen0.resizeMode: Bottom
session.screen0.focusModel: ClickToFocus
session.screen0.autoRaise: <boolean>
session.screen0.clickRaises: <boolean>
session.screen0.workspacewarping: <boolean>
session.screen0.showwindowposition: <boolean>
      Setting this resource to '<boolean>' shows the user, in a little window,
      the exact position of the application window while the user is
      dragging it. Allows a precise placement of windows on a screen.
      Default: <boolean>

session.screen0.decorateTransient: <boolean>
session.screen0.showposinsidewindow: <boolean>
session.screen0.menuMode: Delay
session.screen0.focusNewWindows: <boolean>
session.screen0.workspaceNames: workspace1, workspaceN
      Here is where the user can name their workspaces. However it is
      recommended to use the tool available in the Configuration Menu to
      set these. Default: one, two, three, four

session.screen0.menuDelayClose: 0
      This value sets the delay (in milli-sec) that you would like the
      menu to remain visible after you've clicked out of it. Default: 0

session.screen0.edgeSnapThreshold: <integer>
      When moving a window across your screen, Fluxbox is able to have it
      'snap' to the edges of the screen for easy placement. This variable
      tells Fluxbox the distance (in pixels) at which the window will jump
      to the edge. Default: 0

session.screen0.windowPlacement: RowSmartPlacement
```

```
session.screen0.fullMaximization: <boolean>
session.screen0.sloppywindowgrouping: <boolean>
session.screen0.rootCommand: <command>
      This overrides the styles rootCommand. When this value is set, it
      will keep your background the same, regardless of what any style
      would like your background to be. NOTE: Setting this command can be
      dangerous. Please make sure you know what you are doing when setting
      this resource to a value other than a desktop wallpaper command.

session.screen0.imageDither: <boolean>
session.screen0.opaqueMove: <boolean>
      Sets the visibility level of application windows while being
      dragged. Default: <boolean>

session.screen0.menuDelay: <msec>
session.screen0.workspaces: <integer>
      Set this to the number of workspaces the users wants. Default: 4

session.screen0.focusLastWindow: <boolean>
session.screen0.windowMenu:

session.appsFile: <location>
session.groupFile: <location>
session.keyFile: <location>
session.menuFile: <location>
session.slitlistFile: <location>
session.styleFile: <location>
      All of these resources require a pathname to their specific
      requests.This is where you can specify different files. Most of
      the defaults will be located in the users ~/.fluxbox directory.

session.autoRaiseDelay: <integer>
      Adjusts the delay (in milli-sec) before focused windows will raise
      when using the Autoraise option. Default: 250

session.cacheLife: <integer>
      This tells Fluxbox how long (in minutes) unused pixmaps may stay in
      the X server's memory. Default: 5

session.cacheMax: <integer>
      This tells Fluxbox how much memory (in Kb) it may use to store
      cached pixmaps on the X server. If your machine runs short of
      memory, you may lower this value. Default: 200

session.colorsPerChannel: <integer>
      This tells Fluxbox how many colors to take from the X server on
      pseudo-color displays.  A channel would be red, green, or blue.
      Fluxbox will allocate this variable ^ 3 and make them always
      available. Value must be between 2-6. When you run Fluxbox on an
      8bpp display, you must set this resource to 4. Default: 4

session.doubleClickInterval: <integer>
      Adjust the delay (in milli-sec) between mouse clicks for Fluxbox
      to consider a double click.  Default: 250
```

```
session.forcePseudoTransparency: <boolean>
session.focusTabMinWidth: 0
session.iconbar: <boolean>
      Set this value to '<boolean>' or '<boolean>' to enable or disable Fluxbox
      using the toolbar to display iconified windows. Default: <boolean>

session.ignoreBorder: <boolean>
session.imageDither: <boolean>
      Set '<boolean>' or '<boolean>', respectively, to enable or disable dithering
      of images. Only necessary on systems with small colour depths (8bpp
      or less). Default: <boolean>

session.numLayers: 13
session.opaqueMove: <boolean>
      When moving a window, setting this to '<boolean>' will draw the window
      contents as it moves (this is nasty on slow systems). If '<boolean>' it
      will only draw an outline of the window border. Default: <boolean>

session.tabs: <boolean>
session.tabPadding: 0
session.tabsAttachArea: Window
session.titlebar.left: Stick
session.titlebar.right: Minimize Maximize Close
session.updateDelayTime: 0
session.useMod1: <boolean>
```

KEYS

You can customize Fluxbox's key handling through the ~/.fluxbox/keys file. The file takes the format of:

```
<modifier> <key> :[...] <operation>
```

In the example below, Mod1 is the *ALT* key on the PC keyboard and Mod4 is one of the three extra keys on a pc104 branded with a familiar company logo. Lines beginning with a *#* or *!* are considered comments and unread by Fluxbox.

```
# Fluxbox keys file.
Mod1 Tab :NextWindow
Mod1 Shift Tab :PrevWindow
Mod1 F1 :Workspace 1
Mod1 F2 :Workspace 2
Mod1 F3 :Workspace 3
Mod1 F4 :Workspace 4
Mod1 F5 :Workspace 5
Mod1 F6 :Workspace 6
Mod1 F7 :Workspace 7
Mod1 F8 :Workspace 8
Mod1 F9 :Workspace 9
Mod4 b :PrevWorkspace
Mod4 c :Minimize
```

```
Mod4 r :ExecCommand rxvt
Mod4 v :NextWorkspace
Mod4 x :Close
Mod4 m :RootMenu
Control n Mod1 n :NextTab
```

As you can see from the last line, keybindings can be chained in a fashion similar to Emacs keybindings.

Some things to know: - Commands are case-insensitive. - Workspace numbering starts at "1". - Some commands have synonyms. - The space between the last key and the :Command is mandatory.

Here are Fluxbox key commands to use:

Window Manager Commands

- Restart <argument>
- Quit
- Reconfigure
- SetStyle <argument>
- ExecCommand <argument>

Currently Focused Window Commands

- Minimize
- MinimizeWindow
- Iconify
- Maximize
- MaximizeWindow
- MaximizeHorizontal
- MaximizeVertical
- ResizeTo <width> <height>
- Resize <delta-width> <delta-height>
- ResizeHorizontal <delta-width>
- ResizeVertical <delta-height>
- MoveTo <x> <y>
- Move <delta-x> <delta-y>
- MoveRight <delta-x>
- MoveLeft <delta-x>

- MoveUp <delta-y>
- MoveDown <delta-y>
- Raise
- Lower
- Close
- Shade
- ShadeWindow
- Stick
- StickWindow
- ToggleDecor
- SendToWorkspace <number>
- SentToWorkspace <number>
- KillWindow
- NextTab
- PrevTab
- MoveTabLeft
- MoveTabRight
- DetachClient

Workspace Commands

- NextWorkspace
- PrevWorkspace
- RightWorkspace <by-number>
- LeftWorkspace <by-number>
- Workspace <number>
- NextWindow <bitmask>
- PrevWindow <bitmask>
- NextGroup <by-number>
- PrevGroup <by-number>
- ArrangeWindows
- ShowDesktop (Iconifies all windows)
- RootMenu
- WorkspaceMenu
- WindowMenu

- SetWorkspaceName <name>

Special Commands

- MacroCmd
- ReloadStyle
- SetResourceValue <resourcename> <resource> value
- BindKey <key><value>: <action>

Couple of things

- SentToWorkspace: Will send you along with the window to the selected workspace. SendToWorkspace just sends the window.
- PrevWindow/NextWindow parameters take an integer: 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
- Bindkey will append key string and action to your keys file and bind the key.
- The *delta* value means the difference between the current setting and the requested setting. So if you have a window that is 100 pixels wide, you could set

```
Mod1 r :ResizeHorizontal 10
```

and when you use that key it would increase the size of your window to 110 pixels. If you had used

```
Mod1 R :ResizeHorizontal -10
```

then it would have decreased the size by 10, setting it to 90 pixels.

- Resize commands do not necessarily change the number of pixels. For instance, many terminals will use the size of a character as the resize unit. Most applications, however, use pixels.
- MacroCmd:

```
Mod1 r MacroCmd: {command1} {command2}
```

allows you to execute more than one command with one keybinding. The commands will be executed in serial.

LAYERS

Layers affect the way that windows will overlap each other on the screen. Windows on a higher layer will always appear above those on a lower one, whether they are focused or not. By default, Fluxbox uses 13 layers, starting from 1 (highest). The number of layers can be changed by using the following resource:

```
session.numLayers: <integer>
```

There are two ways to assign a window to a different layer. When the window is open, you may select the layer in the 'Layer . . . ' submenu of the window menu. The menu gives six choices for the layer, which Fluxbox manages by name. The names are (from highest to lowest layer):

- 2 - Above Dock
- 4 - Dock
- 6 - Top
- 8 - Normal
- 10 - Bottom
- 12 - Desktop

The other way to set the layer for a window is through the 'apps' file. This method is described in the APPLICATIONS section.

FOCUS MODEL

The Focus Model defines how windows gain focus (i.e. become the active window, which receives keyboard and mouse events). The focus model can be changed in the configuration menu (usually located under *fluxbox menu* in the Root Menu.

There are two main aspects of the focus model: how windows gain focus and how tabs gain focus. Each of these has two options: focus follows mouse and click to focus. Focus follows mouse means that windows will gain focus when the mouse hovers over them. Click to focus means that windows will gain focus when the mouse clicks on them.

Thus, there are four main options when choosing a focus model. You should choose one of the first two and one of the last two. They are:

- **Click To Focus**: click to focus windows
- **Mouse Focus**: window focus follows mouse
- **ClickTabFocus**: click to focus tabs
- **MouseTabFocus**: tab focus follows mouse

There is one more option in the focus model menu. It is called AutoRaise. When AutoRaise is enabled, focused windows will appear on top of other windows in the same layer. When AutoRaise is disabled, you must explicitly raise a focused window, using the window menu or keybinding.

STYLES

Fluxbox enables you to use specialized files that contain X(1) resources to specify colors, textures, pixmaps and fonts, and thus the overall look of your window borders, menus and the toolbar.

The default installation of Fluxbox provides some of these style files. See fluxstyle(1) to accomodate the growing number of style components.

APPLICATIONS

It is possible to force an application to always have the same dimensions, position, and other settings when it is first launched. This is done using either the window-menu 'Remember... *submenu, or by directly using the ~/.fluxbox/apps file. Be careful to edit the apps file manually only when Fluxbox is not running. Otherwise your changes will be overwritten. Following is a listing of the valid entries for the 'apps* file. The 'Remember... *submenu has entries for most options that stores the current state in the 'apps* file for loading next time.

The format of a line in the 'apps' file is:

```
[app] (app-name) {count - optional}
        [Property1]  {value1}
        [Property2]  {value2}
        ...
[end]
```

Each app-name can be a string, or a regular expression. By default the name is matched against a windows WM_CLASS property (the first string in it, called the "instance"). You can match against the title, instance name (default), class name, or role (the WM_WINDOW_ROLE property) by explicitly specifying it. You can also specify multiple matches, which must ALL match for the properties to be applied. If a count is supplied in curly brackets at the end of the app line, then the entry will only match at most count at any time (default is to match all matching windows).

```
# match a standard xterm
[app] (xterm)
# match an xterm started like: xterm -name myshell
[app] (myshell)
# match any one Firefox window (the instance name is "Gecko")
[app] (class=Firefox-bin) {1}
# match the gaim buddy list window
[app] (role=buddy_list)
# match an rdesktop window to a particular host
```

```
[app] (title=rdesktop - hostname.*)
```

The following are the properties that can be defined in each [app] entry. Each name must be enclosed in square brackets, and the value is generally in curly brackets:

- [Workspace] {0-N}: Forces the application to open on the workspace specified. Workspaces are set by number, beginning with 0.
- [Dimensions] {Width Height}: Open the application with the specified width and height, in pixels.
- [Position] (**refspot**)) {X Y}: Position the application at a particular spot:
  - WINCENTER
  - CENTER
  - UPPERLEFT
  - UPPERRIGHT
  - LOWERLEFT
  - LOWERRIGHT

  You can optionally specify what X and Y are relative to. By default the upper left corner is placed at screen coordinates (X, Y). If you specify LOWERRIGHT, then the lower right corner of the window is positioned (X,Y) pixels from the lower right of the screen. Note that CENTER puts the top left corner of the window relative to the center of the screen (WINCENTER acts like the rest - positions the center of the window relative to the center of the screen).

- Specify the layer to open the window on (by number). Each layer has a number. The named ones are: 2-AboveDock, 4-Dock, 6-Top, 8-Normal, 10-Bottom, 12-Desktop.
- [Shaded] {yes|no}: The window is started shaded, or not.
- [Tab] {yes|no}: Whether this window can be tabbed with others.
- [IconHidden] {yes|no}: Hides the app from the icon bar
- [FocusHidden] {yes|no}: Hides the app from the window cycling list used Next/PrevWindow key bindings.
- [Hidden] {yes|no}: is both [IconHidden] plus [FocusHidden]
- [Deco] {NONE|NORMAL|TOOL|TINY|BORDER}: Specify the decoration state. There are several predefined dec- oration sets:

  o NORMAL - standard decorations
  o NONE - no decorations
  o BORDER - like NONE except keep the X window border
  o TINY - titlebar with an iconify button
  o TOOL - titlebar only

  A bitmask can also be used for fine-grained control. The bits are (from "1" to 1<<10): titlebar, handle/grips, border, iconify button, maximize button, close button, menu enabled, sticky button, shade button, tabbing enabled, focus enabled.

- [Sticky] {yes|no}: Whether the window is initially stuck or not.
- [Jump] {yes|no}: Jump to workspace. This one is only useful if 'Workspace' is set too. The workspace is changed to the workspace containing the application being launched.
- [Close] {yes|no}: Save settings on close. By default, application settings are not saved when a window is closed. Set this option if you want previous settings to be saved when the window is closed.

The apps file also allows you to specify applications that should be started could be used to specify the screen, not the workspace, on which the application should be started. Startup is not yet setable by menu.

Finally, you can set windows to group together by using the 'apps' file. This is achieved with either regular expressions using:

```
[app] (property=expr) ... {number}
```

Property can be one of the following tags:

o name - the name of the window (the first field of WM_CLASS) o class - class of the window (the second field of WM_CLASS) o title - title of the window (WM_NAME property) o role - role of the window (the WM_WINDOW_ROLE property)

If no 'property' is specified, the name property is assumed. You can find out the value for these fields for a particular window by running xprop(1).

You can also place [group] tag around several [app] tags, with an [end] tag to indicate the end of the group. You can also specify dimensions, positions, etc. for the group as for normal app entries. Here is a short example of an 'apps' file:

```
[startup] {xterm}
# match anything ending with term, up to 2 instances
[app] (.*[tT]erm) {2}
# match anything with 'gaim' in the title
[app] (title=.*gaim.*)
[app] (kate)
        [Dimensions]  (WINCENTER) {1022 747}
        [Position]    {0 0}
        [Close]       {yes}
[end]
[app] (konqueror)
        [Workspace]   {1}
        [Dimensions]  {1006 749}
        [Position]    {16 0}
        [Jump]        {yes}
[end]
# start all aterms without decorations
[app] (aterm)
        [Deco]        {NONE}
```

```
[end]
# a group with all windows called "special-term",
# appears on layer 4 (bottom)
[group]
[app] (special-term)
        [Layer] {4}
[end]
```

Parameters in the ‘apps’ file are case-sensitive. Application names are taken from the first X window WM_CLASS attribute by default (WM_NAME = title, WM_WINDOW_ROLE = role). You can see this attribute by using the xprop command. Transient windows are not affected by application settings. Take care when using regular expressions. If you are not familiar with regular expressions you can disable this feature by specifying --disable-regexp during configure. Plain strings will then be matched.

GROUPS

Since version 0.1.11, Fluxbox has a feature called autogrouping, that is apps are automatically grouped together if they are in the same group. NOTE: that this feature is deprecated since version 0.9.1 in favor of grouping using the ‘apps’ file, since it is much more powerful.

You can create groups simply by editing the ~/.fluxbox/groups file. This file takes the format of:

```
<app1> <app2> <app3> <...> <appN>
```

where elements can be found with this command:

```
$> xprop WM_CLASS
```

Just type this command into a terminal and use the mouse to click on the desired app and it will tell you what to write as an element (use the first of the two names returned). Each line forms a different group, e.g.:

```
Navigator nedit
xterm
```

This will create 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.

THE SLIT

The slit is a special Fluxbox window frame that can contain dockable applicatioins, e.g. *bbtools* or *wmapps*.

When applications are run in the slit they have no window borders of their own; instead they are framed in the slit, and they are always visible in the current workspace.

You can click button 3 on the edge of the slit window to get a menu to determine its position, whether its contained applications should be grouped horizontally or vertically and whether the slit should hide itself when the mouse moves away.

Most dockable applications use the -w option to run in the slit. For example, you could put in your ~/.xinitrc:

```
bbmail -w &
bbpager -w &
wmdrawer &
exec fluxbox
```

NOTE: You can also put all of these in the startfluxbox(8) script. This way you would only need to specify: exec startfluxbox in your ~/.xinitrc.

To use the slit you must have it compiled into Fluxbox, this is the default action.

Slitlist File

Fluxbox's slitlist file is available for those that use dockapps in the slit. This file helps Fluxbox keep track of the **order** of the dockapps that you want started. The file is generally located in ~/.fluxbox/slitlist

A simple procedure for getting the slit sequences the way you like it is: 1. Run Fluxbox with no pre-loaded dockapps 2. Run dockapps individually in the order you want them 3. Add dockapps to your auto-run script, or better yet your startfluxbox(8) script.

This sequence will be saved by default to ~/.fluxbox/slitlist and will be maintained in future versions of Fluxbox.

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

The user also has the option of choosing a different path for the slit list file. The following is the init file component that needs changed:

```
session.session0.slitlistFile: <filename>
```

ENVIRONMENT

HOME

Fluxbox uses HOME to find it's .fluxbox/init file, and to resolve style file and -directory names.

DISPLAY

When no other display was given on the command line, Fluxbox will start on the display specified by this variable.

Fluxbox can also take advantage of other environment variables if they are set before Fluxbox is started. For example, if $XTERM is set, then Fluxbox will allow $XTERM to be used in keys and menu files. So one can do:

```
Mod1 x ExecCommand :$XTERM
```

The way of using this in a clever way are endless.

SIGNALS

Fluxbox has the following signals and upon receipt of:

- SIGHUP Fluxbox loads the configuration.
- SIGUSR1 Forces reloading of configuration.
- SIGUSR2 Forces reloading of menu file.

AUTHOR and CREDITS

Fluxbox is written and maintained by Henrik Kinnunen <fluxgen@fluxbox.org>, Simon Bowden <rathnor@fluxbox.org> and Mathias Gumz <akira@fluxbox.org>. with contributions and patches merged from many individuals around the world.

Blackbox was written and maintained by Brad Hughes <blackbox@alug.org> and Jeff Raven <jraven@psu.edu>.

The Official Fluxbox website: http://www.fluxbox.org

Many compatible themes: - http://boxwhore.org - http://themes.freshmeat.net/

This manpage is the combined work of:

- Curt Micol <asenchi@asenchi.com> (>fluxbox-0.9.11)

- Tobias Klausmann <klausman@users.sourceforge.net> (<=fluxbox-0.9.11)
- Grupert <grubert@users.sourceforge.net> (fluxbox)
- Matthew Hawkins <matt@mh.dropbear.id.au> (blackbox)
- Wilbert Berendsen <wbsoft@xs4all.nl> (blackbox)
- Numerous other languages could be available if someone jumps in.

BUGS

If you find any bugs, please visit the #fluxbox irc channel on irc.freenode.net. Or you may subscribe to one of the mailinglists. More information can be found on the official website.

SEE ALSO

bsetroot(8) fbsetbg(8) fbrun(8) fluxstyle(8)