

Fluxbox 文書

Tobias Klausmann

Fluxbox 文書

Tobias Klausmann

Rando Christensen

Section "Installing Fluxbox": Jason Gillman, Jr.

Section "Theme Basic": Justin Rebelo

日本語訳: SATOH, Satoru

目次

1. はじめに	1
1.1. この文書について	1
1.2. Fluxbox について	1
1.2.1. Fluxbox とは何か	1
1.2.2. 特徴	1
1.2.3. Fluxbox の入手	2
1.2.4. 質問やサポート	2
2. はじめに	3
2.1. Fluxbox のインストール	3
2.1.1. ソースの取得	3
2.1.2. 展開、コンパイル	3
2.1.3. Fluxbox の実行	4
2.1.4. その他の雑多な事柄	4
3. ツール	5
3.1. はじめに	5
3.2. fbrun	5
3.3. fluxbox-generate_menu	6
3.4. fluxspace	6
3.5. wmctrl	6
3.6. ページャー	6
4. タブ	8
4.1. はじめに	8
4.2. 発展	8
4.2.1. いい加減なウィンドウのグループ化	8
4.2.2. 単一ウィンドウクラスのタブ化	8
4.2.3. 完全にタブをオフに	8
4.2.4. タブの配置	9
4.2.5. タブの自動グループ化	9
4.2.6. テーマでのタブ	10
5. キーバインド	11
5.1. キーグラバー	11
5.2. キーの名前	12
5.3. アクション	12
6. デスクトップの背景	16
7. slit	17
8. ツールバー	19
9. メニューの編集	20
9.1. メニューファイルの位置の指定	20
9.2. 利用可能なコマンド	20
10. テーマ	23
10.1. テーマの基本	23
10.2. より高度なテーマ編集	24
A. .xinitrc / .xsession の設定	27
B. よくきかれる質問と回答 (FAQ)	29
C. テーマリファンレンス	33
C.1. テーマディレクティブ	33
D. Fluxbox で Artwiz フォントを使うには	38
E. デバッグ	41

表目次

5.1. ナビゲーション	12
5.2. ウィンドウ操作	14
5.3. ウィンドウのサイズ変更	14
5.4. ウィンドウの移動	14
5.5. その他	15
10.1. テクスチャディレクティブ	25
C.1. ツールバースタイル	33
C.2. Menu styles	34
C.3. ウィンドウスタイル	35
C.4. ハンドルバースタイル	35
C.5. リサイズグリップスタイル	35
C.6. ウィンドウボタンスタイル	36
C.7. ウィンドウフレームスタイル	36
C.8. タブスタイル	36
C.9. ウィンドウラベル上のフォントスタイル	36
C.10. その他のスタイル	37

例目次

3.1. ユーザー foobar の ~/.ssh/config	5
4.1. groups ファイル	9
5.1. 設定ファイル	11
5.2. XMMS キーバインド	11
5.3. マルチメディアキーで XMMS を操作	11
5.4. 右矢印キーに対する xev の出力	12
7.1. slit リスト指定	18
8.1. ツールバー、ツールの例	19
9.1. メニューファイルの設定	20
9.2. ~/.fluxbox/menu	21
10.1. ツールバーの時計のスタイル	24
10.2. メニュースタイル	24
10.3. フォントスタイル	24
10.4. テクスチャスタイル	25
10.5. 典型的な簡略スタイル	26
A.1. .xinitrc	28
B.1. スタイルファイルでのタブのカスタマイズ	31
D.1. fonts.dir	39

第1章 はじめに

1.1. この文書について

これは Fluxbox - X11 ウィンドウマネージャ - の新しい文書です。旧 HTML 文書から様々な出力形式に簡単に変換できるように作成されました。文書のために特別に設計された DocBook 4.1 形式で書かれています。[訳注: この訳そのものは DocBook XML 4.2 形式で書かれています]

Fluxbox の文書はこれまでのところ二、三人で保守してきました。今あなたが読んでいるものは Rando Christensen<randoc@babblca.net> が書いたもの、あるいはそれに基づくもので、後に <klausman@users.sourceforge.net> や他の何人かが貢献しています。もしクレジットがなかったなら修正するので <编者> まで知らせて下さい。

[訳注: この日本語訳は訳者が個人的な興味でつくったものです。日本語訳に関連する指摘や修正などは <訳者(佐藤 暁)> まで連絡を下さい。

Fluxbox 自体についての質問はメーリングリストに直接投稿した方が良いでしょう。Fluxbox 公式ページでメーリングリストの講読情報を見つけることができます。

意見、翻訳について

ここではふれられておらず追加するべきであると考えられるような事項については、よく書かれた文書が提供されるならほとんどすべてを受け入れるつもりです。どうやって手助けすればいいか知りたい、あるいは追加したい文書があるなら、単純に私の上述のアドレス宛にメールを下さい。

この文書のソースと HTML や PostScript などの様々な形式に変換したものの双方とも Fluxbox サイトで入手可能です。おそらく DTD を変更するのにいくらかの DocBook の知識が必要となりますが、異なる DTD を使いたい、あるいは他の出力形式を得たいならソースが役立つことでしょう。Fluxbox サイトで提供されていない他の形式にソースを変換したいなら docbook2x パッケージが役立つでしょう。

[訳注: この訳では DocBook XML 4.2 を使い、XSL-T (libxml2 に含まれる xsltproc) および XSL-FO (apache project の FO プロセッサ FOP) を使って HTML と PDF にフォーマット変換しているので、原書とはその変換プロセスおよびソースが異なります。訳者の個人的な主観を言えば、もう DocBook / DSSSL の組み合わせは obsolete だと思いますが、ここでそれにふれると本筋と離れてしまいますので、詳細について知りたければ、この文書のソースおよび他の DocBook XML 関連のリソースをあたって下さい。]

1.2. Fluxbox について

1.2.1. Fluxbox とは何か

Fluxbox は X のウィンドウマネージャの一つです。Blackbox 0.61.1 のコードを基にしています。Fluxbox は Blackbox と似た見た目を持っていて、スタイル、色、ウィンドウ配置、他同様の機能をちょうど Blackbox と同じように扱います (テーマ/スタイルについては 100 % 互換です)。

それでは Fluxbox と Blackbox の違いは何でしょう? 答え: 違いは一杯あります! 次のリストは Fluxbox が既に持っているかあるいは作業中である機能のリストです。

1.2.2. 特徴

実装済み:

- ・ 調整可能なウィンドウタブ
- ・ アンチエイリアスサポート (Xft)
- ・ UTF-8 サポート
- ・ (最小/アイコン化ウィンドウのための) アイコンバー
- ・ ウィンドウ/タブの自動グループ化
- ・ マウスホイールスクロールによるワークスペースの移動
- ・ (ボタンの配置、新規ボタンなどを)設定可能なタイトルバー
- ・ KDE、GNOME (GNOME 2 を含む)サポート
- ・ 拡張ウィンドウマネージャヒント (ewmh) サポート
- ・ 組み込みで統合されたキーグラバー (Emacs に似たキーチェーンをサポート)
- ・ スリットを覆って最大化するオプション
- ・ スリットのドックアプリケーションの順番付け

[訳注: 開発版の Blackbox では Xft/UTF-8 や EWMH サポートについては既に実装済み、または実装を計画/テスト中です。]

計画中:

- ・ セッション管理
- ・ ウィンドウスナップ
- ・ 調整可能なツールバー
- ・ その他の細かな機能

これらすべてだけでなく、多くの変更と改良がコードに加えられています。

1.2.3. Fluxbox の入手

Fluxbox のソースは公式サイト <http://fluxbox.sourceforge.net/> からダウンロードすることができます。また、ほとんどの Linux ディストリビューションと他のフリーの Unix にも Fluxbox のソースとバイナリパッケージが含まれています。[パッケージがあるかどうか]疑わしいのなら、御利用のディストリビューションの最新パッケージを使ってください。

1.2.4. 質問やサポート

Fluxbox を使い、設定するのはとても簡単にできますが、いくつか質問があったり問題が生じることもあるでしょう。それらの内の非常に多くのものについては既に解答があったり、解決済みです。従って、メーリングリストや Fluxbox IRC チャンネルで質問する前にまず 付録 B. よくきかれる質問と回答 (FAQ) を見るようにして下さい。

第2章 はじめに

2.1. Fluxbox のインストール

この節は Jason Gillman Jr. ("Ircaddict") の貢献によるものです。

多くのディストリビューションでは、Fluxbox を極めて簡単にインストールできるようにするバイナリパッケージ (または Gentoo や FreeBSD のように ports/ebuilds) を提供しています。しかし、いくつかの理由からソースからコンパイルする場合があります。例えば、Fluxbox の最新版はディストリビューションでは提供されていないかもしれませんし、さらに、特定のコンパイルオプションでコンパイルしたい場合もあるでしょう。ディストリビューションのパッケージを使いたいなら、そのディストリビューションの文書を参考にして下さい。Fluxbox サイトでは、いくつかのディストリビューションについて、ソースとバイナリ両方のパッケージを提供しています。

この文章の目的は、X11 window system (や一般には Linux) にほとんど馴染みがない人が、Fluxbox をコンパイルし、インストールするのを手助けすることです。

2.1.1. ソースの取得

まず最初に [Fluxbox] ダウンロードページからソースアーカイブ (.tar.gz または .tar.bz2 形式) をダウンロードします。

時々特別な開発版リリースがつけられます。これらは将来取り入れられる新規の機能などのプレビュー版です。開発版リリースの品質によっては、ときどき機能や安定性を欠いていることがあります。利益と危険性すべてを受け入れた上で、最前線の Fluxbox を求めるなら試してみてください。入手方法については公式サイト News セクションに指示があります。

2.1.2. 展開、コンパイル

さて、ソースアーカイブを入手したら展開しなければなりません。次のようにして展開します (ファイル名はそのダウンロードしたファイルで置き換えます):

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

[訳注: solaris など一部のプラットフォームでは tar が gzip 拡張をサポートしていないので上の例のようにはできない場合があります。その場合は次のようにします:

```
$ gzip -dc fluxbox-0.1.12.tar.gz | tar xvf -
```

]

こうすると展開されるファイルのリストが出力されるはずですが。展開したら (fluxbox-0.1.12/ といったような名前) の作成されたディレクトリに移動します。次のステップは Fluxbox の configure、make です。configure 実行時にはいくつかの機能を有効または無効にすることができます。ほとんどの人にとってはデフォルトのままでもいいはずですが、例えば KDE パネルアイコンと slit が協調して動作するようにしたいなら オプション `--enable-kde` を追加します。configure スクリプトの提供する他のオプションを知りたいならオプション `--help` を使います。KDE 機能を有効にしたいと思わないのなら、次のようにすれば十分なはずですが:

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

コンパイルが終わったら root になってこうします:

```
# make install
```

おめでとう、Fluxbox のコンパイルとインストールが終了しました。

2.1.3. Fluxbox の実行

インストールはすべてうまくいきました。しかし実行するにはどうすればいいでしょうか。

X11 (すなわち Fluxbox) を起動させるには、一般に二つの異なる方法があります。伝統的なのはコマンド `startx` を使う方法です。もう一つはグラフィカルなログインマネージャ (ディスプレイマネージャとも呼ばれます) を使う方法です。もっとも一般的なディスプレイマネージャは、XFree86 の配布に含まれる `xdm` です。GNOME と KDE で提供されるディスプレイマネージャは、それぞれ `gdm` と `kdm` です。

前者の方法 (`startx`) で X11 を起動する場合は `~/.xinitrc` が、ディスプレイマネージャの場合は `~/.xsession` が重要となります。

次のステップは Fluxbox の実行ファイルがどこにあるかみつけることです。ほとんどの場合 `/usr/local/bin/fluxbox` がそれです。さて、前述のファイルを編集しなければなりません、それには、単純にそのファイルの終りに次の行を挿入します:

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

`/usr/local/bin/fluxbox` (ソースからコンパイルしたときのデフォルト値) の部分は環境に合わせて変えて下さい。編集が終わったら保存してエディタを終了します。そして `startx` を使っているなら、次を実行する必要があります:

```
$ chmod 700 .xinitrc
```

これは `.xsession` の場合は必要ありませんが、どちらの場合でも Fluxbox が設定を保存するディレクトリをつくっておくべきです:

```
$ mkdir .fluxbox
```

このディレクトリをつくっておかないと、Fluxbox を終了して再起動するとすべての設定が失なわれてしまいます (残念ながらディレクトリは自動生成されません)。

2.1.4. その他の雑多な事柄

もし手助けが必要なら Fluxbox ヘルプフォーラムに行き質問を投稿すれば、きっと誰かが助けてくれるはずです。Fluxbox のメーリングリストで書くこともできます。メーリングリストへのリンクは Fluxbox サイトにあります。最後に、`/usr/local/share/fluxbox` ディレクトリから `init` と `menu` そして `titlebar` ファイルを `.fluxbox/` にコピーしておくことをおすすめします。

第3章 ツール

3.1. はじめに

Fluxbox には、より簡単に使えるようにする、または粋な機能を提供する様々なツールがついてきます。デフォルトでは、それらのツールはfluxboxバイナリと同じ場所 (configure 時に異なる prefix を指定していなければ /usr/local/bin、ほとんどのディストリビューションでは別の場所 /usr/bin) にインストールされています。

3.2. fbrun

fbrun は、基本的には他のデスクトップ環境の "Run ..." ダイアログの同等品です。つまり fbrun を使えば、メニューに含まれていない (あるいは実行時に特別なパラメータセットを必要とする) プログラムを簡単に実行することができます。

他に便利なのは、メニューから fbrun を編集が可能であらかじめ読み込まれるコマンド行とともに呼出し、実行できるということです。例えば非常に長いホスト名のホストにいつも一定でない沢山のオプションをつけて ssh する場合などです。この場合、すべてのオプションとホスト名を含むメニューを fbrun 用に追加できます。そのエントリを使う際に必要なら編集してから実行できます。

[訳注: 本題とは離れますが、ssh では \$HOME/.ssh/config にホスト毎の設定 (ホスト名の別名、X11 転送を有効にするかどうかなどの固有のオプション) を次のようにあらかじめ指定しておくことができます:

例 3.1. ユーザー foobar の ~/.ssh/config

```
Host shn
  Hostname very-long-host-name
  User foo
  ForwardX11 yes
  Port 22
  Protocol 2
```

]

fbrun は様々なオプションを持っています:

-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
-a	Antialias
-hf [history file]	History file to load (default ~/.fluxbox/history)
-help	Show this help

ほとんどのオプションは見ただけでわかるはずですが、オプション `-text` と `-hf` については少し説明が必要かもしれません。前者は `fbrun` のあらかじめ読み込まれる(編集可能な)テキストを指定するのに使います。 `ssh -X -f` のように複数の引数を指定したいのなら、次のように引用符で囲むのを忘れないで下さい:

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

`-hf` オプションは `fbrun` が (ちょうど `bash` がそうするように) 使ったコマンド行の“履歴”を記録しておくファイルの場所を指定します。通常このオプションは必要ではなく、未指定ならデフォルトの設定が使われます。メニューに複数の `fbrun` エントリがあり、各々について履歴を別々に保存したい場合は便利でしょう。

3.3. fluxbox-generate_menu

要修正: このセクションの内容を書くべき

3.4. fluxspace

(<http://fluxspace.sourceforge.net> から引用):

fluxspace は Fluxbox のウィンドウ管理を新しいデスクトップ管理機能と融合します。既存の構成部品と Python の力を活用し、Fluxbox や他の軽量ウィンドウマネーのまわりに柔軟なデスクトップ環境を構築するのを手助けします。

- ・ Rox Filer や Idesk のようなツールと協調し、ワークスペース毎のデスクトップアイコンやパネルを追加
- ・ ワークスペース毎に別の壁紙で飾る
- ・ (ウィンドウマネージャ)起動時に起動するアプリケーションを管理
- ・ ワークスペースの出入り時に自動的にアプレットを起動/停止し、各ワークスペースで各々固有のツールとドックアプリケーションを使えるようにする

3.5. wmctrl

`wmctrl` プログラムは EWMH/NetWM 互換なウィンドウマネージャと対話するためのコマンド行ツールです。

3.6. ページャー

デスクトップ/ワークスペースページャーは複数のワークスペースで作業するのを手助けします。ワークスペースを縮小表示したり、この表示内でウィンドウをドラッグできたり。複数のワークスペースで作業していても、ウィンドウマネージャがワークスペースを管理しているならページャーが必要ではありません。ワークスペース名とその内容はツールバー内に見ることができ、ウィンドウはウィンドウメニューかそのアプリケーションファイルに設定することで他のワークスペースに移せます。

しかしページャーはもっと良くて、ウィンドウの枠だけ、ウィンドウ内容の他の画像、あるいは動作しているアプリケーションのアイコンを表示することができます。

- ・ bbpager: blackbox から. blackbox のものに似た設定ファイルのスタイル. ウィンドウ枠だけを表示. bbtools
- ・ fluxter: bbpager の派生で fluxbox により近い設定. fluxter
- ・ Rox pager: ROX デスクトップ環境のデスクトップのページャー. ROX
- ・ 3d-Desktop:ワークスペースを 3D で表示する OpenGL ページャー. desk3d
- ・ などなど ...

第4章 タブ

4.1. はじめに

Fluxbox のタブは全く新しい考えというわけではありません。実装は PWM ウィンドウマネージャのそれと非常によく似ています。複数のウィンドウは一緒にグループ化され、同じジオメトリ (正確に同じ位置と大きさ) を共有し、一つを動かせば結果としてすべてが動くというように機能します。積み重ねた紙を考えてみて下さい。タブは、とても簡単かつ速く紙をつまんでめくっていきけるようにする、小さな小さなプラスチックタブのようなものです。

バージョン 0.1.14 まではタブは実際にウィンドウにくっついていましたが、0.9.x からはタブはウィンドウのタイトルバーに組み込まれています。

以上がまさに Fluxbox のタブの機能です。単純に、目的のウィンドウの適切なタブを選択し、それが積み重なったウィンドウの一番前面にポップアップされます。それでは試してみましょう。

タブの基本

最初に覚えておく必要があるのは、すべてのタブ操作は第三ボタンで行うということです。それでははじめに一緒にグループ化したい二つのウィンドウを選びましょう。その一方のタブ上で第三ボタンをクリックし、もう一方のタブの上にドラッグします。おめでとう、積み重なりました。これでそのタブを使って二つのウィンドウを切り替えることができます。

タブを切り離すには同じようにします。グループ化されたタブをクリックし、ドラッグします。

4.2. 発展

4.2.1. いい加減なウィンドウのグループ化

“だけど、タブを小さな他のタブにドラッグするのはやりにくいように感じるけど?”

そこでいいニュースがあります。Fluxbox '設定(configuration)' メニューから'Sloppy Window Grouping' オプションを選択します。これでタブを目的のウィンドウのどこにドラッグしてもグループ化できます。

4.2.2. 単一ウィンドウクラスのタブ化

“素晴らしい。けれど私がタブ化したいのはプログラム X だけなんだ!”

タブをいくつかのプログラムに持たせたいかによって異なる二つの方法で、それを実現できます。ウィンドウ毎にタブをオン/オフできる (タイトルバーを右クリック、“タブ” オプションを選択) し、あるいは全体でオフにする ('設定 (configuration)' -> 'タブを表示 (Use Tabs)') こともできます。オフにした後で、前述の方法で個々のウィンドウ毎にタブをオンにします。

4.2.3. 完全にタブをオフに

“タブは好きじゃない。オフにできる?”

もちろん。'設定 (configuration)' メニューから 'タブを表示 (Use Tabs)' を選択します。切り替え項目なので、切り替えれば再度オンにできます。また init 設定ファイル内にも次のように設定があります:

```
session.tabs: true
```

タブのオン/オフは true/false で切り替えます。

4.2.4. タブの配置

バージョン 0.1.14 まではタブは実際にウィンドウにくっついていましたが、0.9.x からはタブはウィンドウのタイトルバーに埋め込まれています。

(0.1.14) 'タブの表示位置' という設定メニューオプションがあります。これらはウィンドウのどこにタブを配置するかを示しています。まさに字義どおりなので、本当に言及する必要があるのは '相対' オプションだけでしょう。これらのオプションは、すべてのタブの総長をウィンドウの長さと同じにします。もしタブが一つだけならウィンドウと同じ長さに、二つタブがあれば半分の長さになります。このオプションはタブを目障りにしないためにしばしばよく使われます。

4.2.5. タブの自動グループ化

ときにはいくつかのアプリケーションについて起動時に自動的にグループ化したいと思うかもしれません。これは論理的に "自動グループ化" と呼ばれています。この節ではどのように働くかを説明します。何よりもまず Fluxbox v0.1.11 以降が必要です。自動グループ化は古いバージョンでは動きません。それからもし既になければ ~/.fluxbox/groups ファイルをつくる必要があります。次に ~/.fluxbox/init ファイルを編集し次の行 (かもしあって内容が違っていれば) を追加します:

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

いいでしょう、完了です。あとは単にグループファイルを埋めていく必要があるだけです。

グループファイル構文

ファイルの各行に一つのグループがあり、ただグループ化するプログラムの名前を書きます。例:

例 4.1. groups ファイル

```
Navigator nedit
xterm
```

こうすると netscape と nedit、xterm の二つのグループができます。新規ウィンドウは、同じワークスペースのウィンドウ、そして最後にフォーカスされたウィンドウとだけグループ化されることになります。groups ファイルに書く名前は次のようにして得ます:

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

そしてそのウィンドウをクリックします。何も表示されないなら \$4 を \$3 に変えてみます。

[訳注: 本題とはまったく関係ありませんが同じことは awk を使わなくてもできます:

```
xprop | grep WM_CLASS | cut -d' ' -f 4
```

awk の場合と同様に何も表示されないなら末尾の 4 を 3 に変えます。

ちなみにそれぞれの実行ファイルの大きさは例えば次のとおりです:

```
$ ls -lh $(which grep) $(which cut) /usr/bin/gawk
-rwxr-xr-x  1 root    root          71K 2003-12-08 08:18 /bin/grep
-rwxr-xr-x  1 root    root          22K 2003-10-05 08:10 /usr/bin/cut
-rwxr-xr-x  1 root    root         249K 2003-09-09 09:19 /usr/bin/gawk
```

参考: 4.16 Think what shell commands you use (procmail tips から)]

タブで自動グループ化

タブを右クリックしてアプリケーションを選べば、ルートメニューをポップアップして起動したときからタブにグループ化できます。

注意

(タブで)グループ化すると、普通に自動グループ化するのに悪影響を及ぼすことがあるかもしれません。

4.2.6. テーマでのタブ

タブの見栄えをテーマでどのように扱う方法については完全に一節 (章 10. テーマ) をもうけています。テーマでタブの見栄えを変えることに興味を持っているならそれをチェックできます (テーマではタブは通常タイトルバーの外観と同じものが既定設定ですが、ときどき変えたいと思うこともあるでしょう)。

第5章 キーバインド

5.1. キーグラバー

キーグラバーは優れたツールであるものの、いくつか制限(とライセンス非互換だった問題)を持つ bbkeys と非常に良く似た働きをします。しかし、完全に新しい構文の設定ファイルと Fluxbox をより強力にするいくつかの新しい機能を持っています。

その一つとして、新しいキーグラバーは (Emacs のように) 一連のキーシーケンスをサポートしているので、例えば `Mod1 + M + Mod1 + F` で次のワークスペースに移動できます (ただし皆が実際にこの特定のキーシーケンスを使うわけではないでしょう)。

そして、もしシーケンスを途中まで入力し、しかし続けない (中止) することにしたなら、ただ他の (キーファイルで設定した `AbortKeychain` の) キーシーケンスを押すだけで中止することができます。

さらに、グループ内のグループ/タブ化されたウィンドウを切り替えるのに、キーシーケンス (`NextTab` と `PrevTab`) を割当てすることもできます。

最後に、便利のように bbkeys の設定ファイルを Fluxbox 用に変換する `vlaad` と `tarzeau` による二つのスクリプト(どちらも同じ働きをします)を紹介しておきます。 `convertkeys` と `convertkeys2` からダウンロードでき、使い方はそれらスクリプトの中に書かれています。

例 5.1. 設定ファイル

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

さて御覧頂けるように、最初にモディファイアがあり、次にキーが続く (もっと長いシーケンスを望むならさらにモディファイア、キーと続けることもできます)、最後はアクションを伴うコロンとなっています。

正しいキーアクションのリストについては、この文書のもう少し後でふれます。

それでこれがどう便利だというのでしょうか? 例えば `xmms` を支配することができます:

例 5.2. XMMS キーバインド

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

もしいくつかオプションキー (例えばマルチメディアキー) があり XFree86 で適切に設定しているなら次のようにして `xmms` をそれらのキーでコントロールすることができます:

例 5.3. マルチメディアキーで XMMS を操作

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

xmms --help により詳細な情報がありますが、これでおそらく適切に設定できるでしょう...

5.2. キーの名前

多分どうやってキーの名前を見つけるのかわからないでしょう。キーの名前は xev を実行してマウスを新規作成ウィンドウに移動し、そのキーを押せばわかります。これは右矢印キーを押したときの例です:

例 5.4. 右矢印キーに対する xev の出力

```
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: ""
```

興味を持っているのはキーの名前ですが、それは括弧の中に keysym と一緒に示されています。この例では (keysym 0xff53, Right) がそうで、名前は Right です。

特殊キー

便利なようにいくつかの特殊キーをあげておきます。これらは xev では 他のキーのモディファイアとしてでなく) キーを押した後すぐに表示されます。

キー	名前 (X11)
Control, Strg	Control
Alt	Mod1
Super, Meta, Win* Keys	Mod4

5.3. アクション

以下が現在 Fluxbox が提供するアクションです。キーバインドとして望まれているようなほとんどすべてのものがあります。設定ファイルではアクションのすぐ前に : がなければならないことに注意して下さい。

アクションは大文字小文字を区別します。

表 5.1. ナビゲーション

アクション	結果
Workspace	指定するワークスペースに移動。:Workspace1、

アクション	結果
	:Workspace2 などとします
NextTab	今のグループの次のタブに切替
PrevTab	今のグループの前のタブに切替
NextWindow N	次のウィンドウに移動. Note 1 を参照.
PrevWindow N	前のウィンドウに移動. Note 1 を参照.
NextWorkspace	次のワークスペースに移動
PrevWorkspace	前のワークスペースに移動
NextGroup, PrevGroup	次/前のウィンドウグループに切替
LeftWorkspace	PrevWorkspace と同じ
RightWorkspace	NextWorkspace と同じ

1. NextWindow / PrevWindow

NextWindow/PrevWindow は数値の引数を持っていますが、上の表内で説明するには少し複雑すぎますので、ここでどのような働きをするか示します。

整数値のパラメータで振舞いを設定します:

ビット値	オプション
0 か未指定	デフォルトの振舞い - スキップしない
1	後面のタブをスキップ
2	スタックウィンドウをスキップ
3	後面のタブ/スタックウィンドウをスキップ
4	シェードウィンドウをスキップ
5	後面のタブ/シェードウィンドウをスキップ
6	スタックウィンドウ/シェードウィンドウをスキップ
7	後面のタブ/スタックウィンドウ/シェードウィンドウをスキップ

御望みならオプション値を合計し、それが NextWindow/PrevWindow に対するパラメータとなります。また下の表からただ拾ってやることも可能です:

パラメータ	オプション
0	スキップしない
1	後面のタブをスキップ
2	スタックウィンドウをスキップ
3	後面のタブ/スタックウィンドウをスキップ
4	シェードウィンドウをスキップ
5	後面のタブ/シェードウィンドウをスキップ
6	スタック/シェードウィンドウをスキップ
7	後面のタブ/スタック/シェードウィンドウ

パラメータ	オプション
	をスキップ

表 5.2. ウィンドウ操作

アクション	結果
Close	ウィンドウを閉じる
KillWindow	xkill を呼んでそのウィンドウをクリックするのと同じです
Minimize	"iconify" としても知られていますが、ウィンドウをアイコン化します
ShadeWindow	ウィンドウを 'シェード' 状態にする、あるいはその状態から戻す
StickWindow	ウィンドウの 'sticky' 状態を切替
ToggleDecor	ウィンドウが枠、ボタン、タイトルバーを持つかどうかを切替
Raise	ウィンドウを最前面に出し、他のウィンドウの '上' にくるようにします
Lower	Raise の逆
NextTab, PrevTab	次/前のタブをアクティブに
MoveTabLeft, MoveTabRight	n 個分、左/右のタブをアクティブに
DetachClient	タブグループからクライアントを取り除きます

表 5.3. ウィンドウのサイズ変更

アクション	結果
MaximizeHorizontal	ウィンドウを水平方向に最大化
MaximizeVertical	ウィンドウを垂直方向に最大化
MaximizeWindow	ウィンドウを最大化
Resize	アクティブなウィンドウを指定した分リサイズします。例:resize -8 -8.
ResizeHorizontal	指定した分、水平方向にリサイズ
ResizeVertical	指定した分、垂直方向にリサイズ
ArrangeWindows	魔法を使ったように、ウィンドウをタイル状に並べます
ShowDesktop	すべてのウィンドウをアイコン化

注意

リサイズの "一単位" はアプリケーションによって異なるでしょう。xterm/aterm/Eterm などではリサイズの際に 1 pixel 分ではなく文字幅分だけリサイズします。

他のプログラムでは単純に 1 pixel 分だけリサイズします。

表 5.4. ウィンドウの移動

アクション	結果
SendToWorkspace	ウィンドウを指定のワークスペースに移動させます。例: :SendToWorkspace 1
Move	指定した値分移動
MoveLeft	そのまま
MoveRight	そのまま
MoveUp	そのまま
MoveDown	そのまま

表 5.5. その他

アクション	結果
AbortKeychain	複数のバインドのキーチェーンで、キーバインドを取り消す
ExecCommand	コマンド実行. 例: ExecCommand xmms -t.
RootMenu	ルートメニューを表示
WorkSpaceMenu	ワークスペースメニューを表示
Restart	fluxbox を再起動
Reconfigure	fluxbox を再設定し、設定を読み直させる。例えば、keys を変更したら再度読み込みますが、init と slitlist は読み込む前に書き込まれるでしょう
SetStyle	指定したファイルを読み込む
SetWorkspaceName	ワークスペース名を設定
SaveRC	リソースファイルを保存
Quit	fluxbox を終了

第6章 デスクトップの背景

Fluxbox は、Blackbox と同じように、`bsetroot` と `bsetbg` のただ二つのラッパーユーティリティを持っているだけです。それらがどのように動作するか見てみましょう。

`bsetroot` は大体 `xsetroot` と同じで、可能なかぎり簡単に単色、そしてまたグラデーションの背景を設定することができます。

`fbsetbg` は、適切な背景設定アプリケーションを探して、そのアプリケーションを使って背景を設定しようとするラッパーです。`fbsetbg` は単純に最初にみつけたアプリケーションを使うので、設定は必要ありません。

テーマの背景の上書き指定

Fluxbox では、`~/.fluxbox/init` ファイルに指定すれば、スタイルファイルの中で指定されている背景設定を無視することができます。

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

`~/.fluxbox/init` ファイルを変更するのを避けたいのなら、`fbsetbg` に `-l` オプションをつけると `~/.fluxbox/lastwallpaper` に使用した壁紙を保存し、次回はそれを読み込もうとします。この機能を利用するには `rootCommand` を次のようにします。

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

第7章 slit

もっともよくきかれる質問の一つが“slit って何?”というものです。実のところ、この文書を書いたとき、一日に10回も #fluxbox できかれるのを止め slit がツールバーの別名であるという神話を終りにするために、この節を FAQ の冒頭で差し示すようにしました。

slit は Fluxbox が Blackbox から継承した多くの部分の一つです。WindowMaker のドックアプリケーション (と withdrawn あるいはよりまれに swallowed と呼ばれるモードで動作する他のもの) を保持するように設計されています。そのようなアプリケーションはしばしば `-w` オプションを持ちますが、いくつかは [そのオプションなしでも] 自動的に withdrawn モードになります。

さてまず Fluxbox が slit を組込んでコンパイルされているか確かめましょう。たいていは大丈夫ですが、御存知のようにディストリビューションによってパッケージは異なります。もし、[スリット]なしの方がいいならコンパイル時に無効にすることもできます。けれども未使用の slit はまったく画面領域をとらずごくわずかなメモリを食うだけなので、通常は、スリットが御利用の他のアプリケーションと干渉するのでなければ無効にする必要はないということを気にとめておいて下さい。

ドックアプリケーションなら何でも実行できます (withdrawn モードとして知られています)。例えば `xmms` は `wmxmms` と一緒に配布されていますが、`wmxmms &` とすれば slit 内に表示されます。上で述べたようにいくつか (例: `gkrellm`) を slit に収めるのには `-w` オプションが必要です。

どこにいけばドックアプリケーションが見つかる?

はじめるのに一番いい場所は the Dockapp warehouse です。ドックアプリケーションの非常に大きなリポジトリです。Freshmeat で探すか、ディストリビューションをチェックすることもできます。

bbtools ページには沢山の Blackbox/Fluxbox ユーティリティがあり、ほとんどは slit 内で実行できます。

また KDE サポートが有効ならば KDE ドックアプレットも slit 内で実行できます。

Dockapps.Org はまったくドックアプリケーションだけに注力した新しいサイトです。

slit の振舞いをどうやって変える?

もちろん変えられます。単に slit の見える部分で右クリックしてオプションを選ぶだけです。ほとんどのオプションはタスクバーとちょうど同じです。違うのは Direction があることで、slit は Horizontal か Vertical 方向に配置できます。

configure メニューには Maximize Over Slit オプションというものもあり、slit を覆うようにウィンドウを最大化できます。

slit アプリケーションの順番を覚えさせたい!

Fluxbox 0.1.10a以降なら `slitlist` ファイルを使えば可能です。利用方法を少し説明してみます。

ドックアプリケーションの順番がこのファイル (デフォルトでは `~/ .fluxbox/slitlist`) に保存されます。slit にドックアプリケーションを読み込む際、その名前が前に動作していたものに一致すれば、前の順番を保持しようとします。

slit の希望の順番を決定する簡単な方法は以下のとおりです:

手順 7.1. ドックアプリケーションの順番づけ

1. ドックアプリケーションなしで Fluxbox を実行
2. 希望する順にドックアプリケーションを個々に起動
3. ドックアプリケーションを自動起動スクリプト、例えば `.xinitrc` か `.xsession` に記載. 順番は気にしなくて良い

この順番は、デフォルトでは `~/fluxbox/slitlist` に保存され、将来のセッションにわたって保持されます。

`slitlist` を手作業で編集するのも自由です。単純なウィンドウ名のリストで、一行毎にドックアプリケーションがなっています。Fluxbox が実行中でないときに編集するべきで、さもないと変更が上書きされてしまいます。

異なる slit リストファイルを選択することもできます。次の `init` ファイルのエントリの例ではパスを変更しています：

例 7.1. slit リスト指定

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

完全に順番づけを無効にするオプションはないことに注意して下さい。パッチ作者は順番を任意にすることにいかなるメリットもないと考えています。

第8章 ツールバー

ツールバーは Fluxbox が時計や実行中のプログラムのボタンなどの情報を表示するのに使う小さな領域です。

ツールバーは init ファイルとスタイルファイルで設定します。init 設定は通常ツールバーメニューで変更します。

ツールバーは init ファイルでオフにする(設定で非表示に: `session.screen0.toolbar.visible: false`) こともできます。

表示するツールはリソースファイル(通常 `.fluxbox/init`) 内の `toolbar.tools` エントリで設定します。

例 8.1. ツールバー、ツールの例

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

可能なツール: `workspacename`, `prevworkspace`, `nextworkspace`, `iconbar`, `systemtray`, `prevwindow`, `nextwindow`, `clock`

ツールバーの幅と透明度、レイヤーはツールバーメニューで設定できます。ツールバーメニューを表示するには、ツールバーの時計かワークスペース名の部分で右シングルクリックします。

アイコンバーの表示設定は、非表示、すべてのワークスペースのアイコンを表示、ワークスペースアイコンを表示、ワークスペース内のすべてのウィンドウを表示、すべてのワークスペースのすべてのウィンドウを表示、のどれかです。

ツールバーの外観はスタイルファイルで設定します。

第9章 メニューの編集

さて、Fluxbox をインストールすると、デスクトップを右クリックしたときに現われるメニューという小粋なアプリケーションに気づくでしょう。しかしこれは使うアプリケーションを起動するように編集できないければたいして便利ではありません。この節ではメニューに関するすべての疑問に答えようと思います。

まず、Fluxbox には、fluxbox-generate_menu という便利なユーティリティがついています。これは、環境からブラウザや端末といった普通にインストールされている様々なプログラムのパスをみつけだし、メニューファイルを生成します。fluxbox-generate_menu の詳細については 項3.3. 「fluxbox-generate_menu」でふれています。

9.1. メニューファイルの位置の指定

メニューのデフォルトは `~/.fluxbox/menu` ですが、この設定は `init` ファイルで変更できます。例えば、

例 9.1. メニューファイルの設定

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

別のファイルを使いたいならただ `~/.fluxbox/menu` を編集するだけでよいです。しかしほとんどの人にとってはこのままでよいはずです。

9.2. 利用可能なコマンド

メニューはただのテキストファイルで、サブフォルダ作成、アプリケーション起動、ワークスペース制御、設定、X の終了などが可能です。メニューでは次のコマンドを使うことができます:

```
[begin] (MenuTitle)
[submenu] (SubMenuName) {SubMenuTitle}
[exec] (ApplicationName) {/path/to/program}
[include] (/path/to/menufile)
[end]
[nop] (-----)
[workspaces] (SubMenuName)
[stylesdir] (/path/to/stylesdir)
[config] (FluxboxConfiguration)
[reconfigure] (Reconfigure)
[restart] (Restart)
[exit] (Exit)
```

- ・ # で始まる行はコメントで、行の以降の部分は無視されます
- ・ [] 内のテキストは fluxbox のコマンドです
- ・ () 内のテキストはメニューに表示されるテキストです

- ・ {} 内のテキストはそのエントリがクリックされたときに開始されるコマンドです
- ・ <> でアイコンファイルを指定します。アイコンファイルは絶対パスで指定し、XPM 形式でなければなりません
- ・ include ディレクティブへのパラメータがディレクトリなら、そのパスのすべてのファイルが include されます

どんな場合でもすべてのエントリが必ず必要というわけではありません。例えば、[end] はアイコンについては無意味です。

サブメニューの入れ子の深さにはまったく制限はないことに注意して下さい。けれどもおそらく実用には一階層で十分なはずです。

[nop] - これによって望むならテキストか空の行を入れることができます。何も実行されずメニュー内で単に区切として働きます。

[reconfigure] - Fluxbox の設定をメニューで変えても終了するとその変更は失われてしまいます。恒久的な変更とするには変更後に reconfigure して init に書き込む必要があります。

[restart] - これは Fluxbox だけの再起動でありシステム全体の再起動という意味ではないということ、必ず理解しておいて下さい。

例 9.2. ~/.fluxbox/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] (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
```

あるいは Fluxbox に同梱されている、より完全な menu の例を調べてみて下さい。

第10章 テーマ

10.1. テーマの基本

この節は Justin Rebelo (通称 demerol) の貢献によるものです。

スタイルとは何で、どう機能するのか?

スタイルは基本的には Fluxbox のテーマです。単純な ASCII テキストファイルであり、ウィンドウマネージャの各コンポーネントについてどのような見栄えにするか、Fluxbox に指示しています。通常は、`~/fluxbox/styles` と、インストール方法に依存して変わる、システムの Fluxbox 共有ディレクトリ内に置かれています。

どうやって独自のスタイルをつくるか?

手始めに、スタイルを好きなテキストエディタ (私は [訳者も ;-])vim を御勧めします) で開いて、どう構造化、組織化されているかを見てください。ただそうするだけでほとんど答えが出てしまうことでしょう。

スタイルの構造

スタイルは、それぞれ独自のサブディレクティブを持つ、いくつかの主要なコンポーネント (toolbar, menu そして window) で構成されています。window.* ディレクティブはウィンドウ枠の、window.tab.* はウィンドウタブの、menu.* は右クリックしたときにデスクトップに現れるポップアップメニューの、toolbar.* は画面の上か下に表示されるバーの外観をコントロールします。slit (他の WM ではドック、wharf などとも呼ばれます) はもしそのスタイルを明示的に指定していなければ toolbar 設定でコントロールします。

slit の外観を変えるにはどうしますか?

通常、slit はツールバーと同じオプションを使います。ほとんどの場合これはまあまあよく機能するはずですが、スタイルを明示的に指定したいなら、次の三つのスタイルディレクティブを使うことができます:

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

これらのコマンドは、slit を装飾する際に、ちょうどメニューやウィンドウなどの場合と同じように機能します。

背景画像/色を設定できますか?

スタイルファイル内のどこかに、rootCommand で始まり、スタイルの背景色または画像を設定するコマンドが続いている行があるかもしれません。Fluxbox には、背景色(グラデーションも含む)を設定するプログラム bsetroot と、背景画像(壁紙)を設定するプログラム fbsetbg が含まれています。

スタイルにメモやコメントを追加できますか?

もちろん。ただ行をハッシュ (#) か感嘆符 (!)、あるいは C++ スタイルのコメント (//) で始めてやる

だけです。

まだもっと質問があるのですが...

Fluxbox と一緒に提供されているスタイルを見たり異なる設定を試したりすれば、きっと答をみつけられるはずです。それでもなおわからなかったら OPN の #fluxbox に join してみてください。私のニックは demerol です。

10.2. より高度なテーマ編集

このセクションの大半は Fluxbox 0.1.13 の man ページからもってきた、あるいは多大な影響を受けています。通常この種の事柄の決定的な権威は man ページですが、この文書はスタイル作成をはじめたばかりの人をもっと[強く]啓蒙するようなものとなっています。

スタイルの仕組みがどう機能するかを理解するには、少し X11 のリソースの機能について知ってみるのがいいです。

X11 リソースはキーと値で構成されています。キーはピリオド(.)で区切られたいくつかのより小さなキー(ときどき子として参照されます)で構成されています。キーには、ワイルドカードの役目を果たすアスタリスク(*)を含めることもでき、その場合その一行でいくつかのキーにマッチすることを意味しています。これは一、二色をベースとするスタイルで便利です。

Fluxbox では、ツールバー、メニュー、ウィンドウ装飾の三つの主要コンポーネントを調整できます。slit は自動的にツールバーからスタイルを継承しますが、必要なら別にスタイルを指定することもできます。ウィンドウをドラッグする際に x-y 位置に表示される小ウィンドウは、そのスタイルをウィンドウタイトルバーから借ります。

以下は基本構文を説明するためのいくつかの簡単な例です:

例 10.1. ツールバーの時計のスタイル

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

ツールバーの時計の色リソースをgreenに設定します。別の例:

例 10.2. メニュースタイル

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

メニューとそのすべての子の色リソースを rgb:3/4/5 に設定しています。色名の説明は X11 man ページを参照して下さい。これは menu.title.color と menu.frame.color にも適用されます。そして次の例:

例 10.3. フォントスタイル

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

すべてのキーのフォントリソースを一度にこのフォント名に設定しています。システムにインストールされたフォントの情報については `xfonsetl`, `gfontsel` あるいは `xlsfonts` のようなプログラムを使うことができます。

Fluxbox をこんなに華やかにしているのは、テキストを動的に描画する能力です。テキストの記述は直接キーに指定され適用されるはずですが、例えば、

例 10.4. テキスチャスタイル

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

心配しないで、これらのディレクティブがどう機能するかについては[きちんと]説明します。テキストの記述は次のように最大五つのフィールドからなります:

表 10.1. テキスチャディレクティブ

ディレクティブ	説明
Flat / Raised / Sunken / Tiled	flat, raised あるいは sunken の外観をコンポーネントに与えます。Tiled は pixmap だけに作用し、スケールしません。
Gradient / Solid	単色かグラデーションテキストを描画します
Horizontal / Vertical / Diagonal / Crossdiagonal / Pipecross / Elliptic / Rectangle / Pyramid	これらテキストタイプの内から一つ選択します。これらも Gradient が指定されているときだけ機能します。
Interlaced	テキストを織り混ぜ(一行毎に暗くする)ます。このオプションはグラデーションテキストでもっともよく使われていますが、Blackbox 0.60.3 (そしてすなわちすべてのバージョンの Fluxbox) から solid テキスチャでも使えるようになっています。
Bevel1 / Bevel2	使用する bevel (面取り) の種類。bevel1 はデフォルトの設定で、陰影は画像の端につきます。bevel2 はその代替で、陰影は画像の端から 1 pixel 分内側につきます。

テキスト記述からは離れますが、オプション ParentRelative もまた利用でき、コンポーネントをその親の一部として見せることができます。

すべてのグラデーションテキストは color と colorTo リソースの二色の組み合わせからなります。Solid モードで interlace を使っているなら、その interlace 色をみつけるのに colorTo リソースが使われます。

完全なコンポーネントとそれぞれについて指定可能な値のリストは 付録 C. テーマリファンレンスにあります。

リストは長大ですが、独自のスタイルをつくるなら、多数のキーを一つのコマンドで簡単に設定できることを覚えておいて下さい。例えば次のように。

例 10.5. 典型的な簡略スタイル

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

付録 A. .xinitrc / .xsession の設定

Verin の貢献による。

xinit の位置づけ

ウィンドウマネージャは mozilla や gimp、xterm などと同じように、単に X11 のアプリケーションの一つにすぎません。X11 に馴染みのない多くの人は、X11 がウィンドウマネージャを実行し、ウィンドウマネージャがプログラムを実行していると思いこんでしまっているようですが、それは正しくありません。適切に設定されていればすべてのアプリケーションを X11 で実行し、ウィンドウマネージャを kill して他のウィンドウマネージャを起動できます。

X11 が本当に動かしているのは .xinitrc あるいは .xsession スクリプトであり、他のプログラムはそれらから実行されています。X11 が起動するとこれらのスクリプトの内どちらかが実行され、スクリプトが終了すると X11 は終了します。重要なので繰り返させて下さい: .xinitrc が終了すると X も終了します。ウィンドウマネージャの終了時ではありません。

スクリプトレイアウト

さてまず何について知っているかはっきりさせましょう。シェルではコマンドを入力すると、それが完了し終了するまで何もできません。 .xinitrc または .xsession スクリプトについてももちろん同じことが言えます。起動して実行していく間、(ほとんどの X11 アプリケーションでそうであるように) あるプログラムが実行に長時間かかるようであれば、それが終了するまで [スクリプトは] 一時停止します。

理想をいえばスクリプトが停止するのは一箇所に留めるべきです。そして普通停止するのは一番最後にしたいと思うでしょう。だからこの停止場所にくるまでに X11 で動かしたいプログラムがあるなら、バックグラウンドで動かす (& を行の終りにおく) べきです。例えば他のものに加えて xclock を動かしたいなら、停止場所の前に次の行をおきます:

```
xclock &
```

さて次は exec です。これについては、沢山のソースでスクリプトにウィンドウマネージャを追加するための方法として推奨しています。しかし本当のことをいうと、スクリプトの最後にウィンドウマネージャを書けば exec なしでもちょうどそこでうまくとまるので、必ずしも必須というわけではありません。

ではなぜ exec するのでしょうか? 沢山のウィンドウマネージャをスクリプトに書いておいてその中の一つを使うとしましょう。exec を選択したものの前に置き、一番上にもってきます。なぜならそれが exec の意味するものだからです。

「自身をこのプログラムで置き換える、すなわちそれを起動し、終了時には即座に自身を終了させる。」

exec wmaker 行を exec enlightenment 行の前に置けば、wmaker が終了するとスクリプトは決して次の行に進みません。

必須ではないといった意味はわかりますか? ウィンドウマネージャ群の行をコメントアウトするだけで、ちょうど同じように動作するからです。

その他の方法

代替手段としては、ウィンドウマネージャを最初に起動し、プロセス ID を環境変数に保持しておくこともできます:

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

バックグラウンドで実行 (&) し、プロセス ID (\$!) を変数 (wmpid) に保持します。それから停止場所で wait できます:

```
wait $wmpid
```

あるいは、ただバックグラウンドに送ることで gkrellm のようないつも使いたいプログラムを停止させることもできます。しかし気をつけなければならないのは、それを終了させると X11 セッションも終了してしまうということです。

私はドックアプリケーションやツールを起動する前に、ウィンドウマネージャを起動しておくのが好きなので、wait を使っています。また他のものの前に DPMS、スクリーンセーバー、フォントパス (私はシステム全体用にフォントをインストールしません) などの X11 サーバ設定を変更する方を好むからでもあります。そしてすべてが済むと、主にディスプレイマネージャを動かしているために、フォントパスを掃除しておきたい (いつもフォントパスを再初期化するのはいいことではありません) のです。

例 A.1. .xinitrc

```
# turn off screen blanking and turn on energy star features
# (スクリーンを暗くする機能をオフにし、energy star 機能をオンに)
xset s off
xset dpms 600 60 60

# add my optional fonts to the font path
# (フォントパスにオプションのフォントを追加)
xset +fp "$X_FONTPATH"
xset fp rehash

# export the current environment, in case it needs to be debugged
# (環境設定をデバッグ用に出力)
env > ~/.xenv

# window manager
# (ウィンドウマネージャ)
fluxbox & wmpid=$!

bbrun &
wmCalClock &
wmxmms &

# HANG POINT - wait for window manager to exit
# (停止点 - ウィンドウマネージャの終了を待つ)
wait $wmpid

# restore the x fontpath
# (X フォントパスを復活)
xset fp default
```

付録 B. よくきかれる質問と回答 (FAQ)

B.1.

slit って何?

slit についてまず知っておかなければならないのはそれが Fluxbox のタスクバーではないということです。

slit はドックアプリケーションがドックできる場所です。この文書では完全に一章さいて slit について説明しています: 章 7. slit。slit が何であるか、またどのように機能するかについて質問する前にその章を読んで下さい。

B.2.

決まった順番で slit にドックアプリケーションを入れる方法はある?

Fluxbox 0.1.10 以降なら可能です。章 7. slit に説明があります。

B.3.

ツールバーの時刻表示の書式はどうやって変えるの?

init の次の行を変更してみてください:

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

書式の説明は man ページの strftime (3) にあります。

B.4.

~/fluxbox/init を変更しても上書きされてしまう。

0.1.8-bugfix2 以前のバグです。バグ報告の前に最新版にアップグレードして下さい。

B.5.

タブはどんな風に動くの?

章 4. タブ を参照のこと。

B.6.

アンチエイリアスにするとフォントが大きくなっちゃうよ! どう直したらいい?

Xft の登場で、フォントは[今までとは]異なるやり方で処理されるようになりました。手始めにテーマ(スタイル)ファイル内でこれを指定してみてください:

```
*.font: Verdana:size=7
```

もちろん他のフォントやサイズも使えますが、上の例のようにすればきっとメニューが使えるようになるはずです。利用可能なフォントについては次の質問を読んで下さい。

B.7.

Snap のような古いフォントだと AA できないよ。どうしたらいい?

Fluxbox の AA 機能を有効にするとすぐに文字の描画は FreeType2 に頼るようになります。AA を使わなければ X11 生来のフォント描画機能が使用されます。FreeType2 は X11 のサポートするすべてのフォント形式をサポートしていないので、Fluxbox のフォントの選択も限られてしまいます。この文書を書いている時点では FreeType2 は以下の形式をサポートしています:

- ・ TrueType ファイル(.ttf)とそのコレクション(.ttc)
- ・ Type 1 フォントファイル (ASCII .pfa とバイナリ .pfb 両方)
- ・ Type 1 複数マスタフォント
- ・ Type 1 CID-keyed フォント
- ・ OpenType/CFE (.otf) フォント
- ・ CFF/Type 2 フォント
- ・ Adobe CEF フォント (.cef)
- ・ Windows FNT/FON ビットマップフォント

一方 FreeType 1 は、外部ライブラリによる GX および OTF フォントのサポートはあるものの、自身がサポートするのは TrueType だけでした。詳細については FreeType のサイトに情報があります。

- B.8. Artwiz フォントについてずっと耳にしつづけているのだけど、これは何?

付録 D. Fluxbox で Artwiz フォントを使うには に説明があります。

Artwiz フォントが好きだけど端末での見栄えが気に入らないなら Linux Font Project で入手できる LFP フォントパックを試してみてください。LFP 固定幅フォント (端末用) と LFP 可変幅フォント (端末以外) の二組あります。固定幅フォントは Linux コンソールでも使えます。

- B.9. どうやって背景を設定するの?

章 6. デスクトップの背景 で説明しています。

- B.10. テーマを変えると背景が変なのに変ってしまうんだけど。

章 6. デスクトップの背景 に解決策があります。

- B.11. 既存の .blackboxrc を Fluxbox で使うことはできる?

多分使えますが、タイトルバーとキーグラブのために何行か追加するのを忘れないで下さい。より賢く blackboxrc のシンボリックと ~/.fluxbox/init をどうにかして一緒に使うこともできるでしょう。

- B.12. Fluxbox 起動時にアプリケーションを自動的に起動させるにはどうするの?

付録 A. `.xinitrc` / `.xsession` の設定 を参照。

- B.13. Blackbox のスタイル(テーマ)は Fluxbox で使える?

ええ。100% 互換です。保証はできませんが Waimea と Openbox でも使えるはずです。それら二つのプロジェクトではそのような約束をしているのを見たことはありませんが、Fluxbox では Blackbox のスタイルとの互換性を保つのは目標の一つとなっています。

- B.14. `.xinitrc` / `.xsession` はどう設定したらいい?

付録 A. `.xinitrc` / `.xsession` の設定を参照のこと。

- B.15. KDE はサポートしてる?

ええ、`configure` オプション `--enable-kde` を使って下さい。これで KDE トレイアイコンが `slit` に表示されるようになります。

- B.16. GNOME はサポートしてる?

ええ、`configure` オプション `--enable-gnome` を使って下さい。これで GNOME ヒントが有効になります。Fluxbox 0.1.12 以降ではデフォルトでそうになっています。

- B.17. BBtools が再起動後もスタイル設定を使ってません。

単純に `~/.blackboxrc` を `~/.fluxbox/init` にリンクして下さい、例えば次のように:

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

- B.18. タブがいくつかのスタイルで汚なくなってしまうけどどう直せばいい?

0.1.14 以前のバージョン

タブを(もっと)綺麗にするには、希望のスタイル(テーマ)にいくつか余分なエントリを追加する必要があります。しかし Fluxbox はそれ自身が適切な色/スタイルをタブに設定する能力を持っていることに注意して下さい。(それでもなお)見栄えを細かく制御したいと望むなら次の数行を追加してもよいでしょう:

例 B.1. スタイルファイルでのタブのカスタマイズ

```
! -- 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
```

さてそれではすべてについてこれを行うにはどうしたらいいでしょう? 同じことがテーマの他の部分にも言えるので、Blackbox テーマを前に作ったことがあるならきっとわかるはずです (作ったことがないなら 章 10. テーマ が役立つでしょう)。

また、これら余分なエントリがあっても Blackbox はちゃんと動くので、追加して失なうものはないことに注意して下さい。

B.19. アイコンを Fluxbox のデスクトップに置くにはどうしたらいい?

Fluxbox には、今のところ一緒に配布してはいませんが、この種の機能のための独自の協調プログラム fbdesk があります。また Rox Desktop, idesk, xdesk などの代替もあります。

B.20. Fluxbox 0.9.6 が遅いんだけど...

~/ .xinitrc の fluxbox を exec する行の前に以下の行を追加してみてください:

```
export LC_ALL=C
```

これはより新しい RedHat できっと役立つはずです。

付録 C. テーマリファンレンス

C.1. テーマディレクティブ

以下は利用可能なディレクティブと、対応する割り当て可能な値の完全なリストです。ディレクティブの詳細については 章 10. テーマ を参照下さい。

表 C.1. ツールバースタイル

toolbar	Texture
toolbar.height	Number
toolbar.color	Color
toolbar.colorTo	Color
ボタン	
toolbar.button	Texture か ParentRelative
toolbar.button.color	Color
toolbar.button.colorTo	Color
押してないときの矢印ボタンの色	
toolbar.button.picColor	Color
押したときのボタン	
toolbar.button.pressed	Texture (例えばSunken) か ParentRelative
toolbar.button.pressed.color	Color
toolbar.button.pressed.colorTo	Color
押したときの矢印ボタンの色	
toolbar.button.pressed.picColor	Color
ツールバーラベル	
toolbar.label	Texture か ParentRelative
toolbar.label.color	Color
toolbar.label.colorTo	Color
toolbar.label.textColor	Color
ワークスペースラベル	
toolbar.workspace	Texture か ParentRelative
toolbar.workspace.pixmap	Pixmap
toolbar.workspace.color	Color
toolbar.workspace.colorTo	Color
toolbar.workspace.textColor	Color
toolbar.workspace.font	Font
ウィンドウラベル	
toolbar.windowLabel	Texture か ParentRelative
toolbar.windowLabel.color	Color
toolbar.windowLabel.colorTo	Color
toolbar.windowLabel.textColor	Color

時計	
toolbar.clock	Texture か ParentRelative
toolbar.clock.pixmap	Pixmap
toolbar.clock.color	Color
toolbar.clock.colorTo	Color
toolbar.clock.textColor	Color
toolbar.clock.font	Font
空のときのアイコンバー	
toolbar.iconbar.empty	Texture か ParentRelative
toolbar.iconbar.empty.pixmap	Pixmap
toolbar.iconbar.empty.color	Color
toolbar.iconbar.empty.colorTo	Color
toolbar.iconbar.empty	Texture or ParentRelative
フォーカス/非フォーカス時のアイコンバー	
toolbar.iconbar.focused	Texture か 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
テキスト	
toolbar.justify	center, left, または right
toolbar.font	Font

表 C.2. Menu styles

タイトル	
menu.title	Texture
menu.title.color	Color
menu.title.colorTo	Color
menu.title.textColor	Color
menu.title.font	Font
menu.title.justify	center, left, または right
フレーム	
menu.frame	Texture
menu.frame.color	Color
menu.frame.colorTo	Color
menu.frame.textColor	Color
menu.frame.disableColor	Color
menu.frame.font	Font
menu.frame.justify	center, left, または right

サブメニュー 鋳

menu.bullet	empty, triangle, square, または diamond
menu.bullet.position	right または left
menu.submenu.pixmap	Pixmap
強調表示されたアイテム	
menu.hilite	Texture (例えばRaised)
menu.hilite.color	Color
menu.hilite.colorTo	Color
menu.hilite.textColor	Color
menu.selected.pixmap	Pixmap
menu.unselected.pixmap	Pixmap

表 C.3. ウィンドウスタイル

タイトル	
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
ラベル	
window.label.focus	Texture か ParentRelative
window.label.focus.color	Color
window.label.focus.colorTo	Color
window.label.focus.textColor	Color
window.label.unfocus	Texture か ParentRelative
window.label.unfocus.color	Color
window.label.unfocus.colorTo	Color
window.label.unfocus.textColor	Color

表 C.4. ハンドルバースタイル

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

表 C.5. リサイズグリップスタイル

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

表 C.6. ウィンドウボタンスタイル

window.button.focus	Texture か ParentRelative
window.button.focus.color	Color
window.button.focus.colorTo	Color
window.button.focus.picColor	Color
window.button.unfocus	Texture か 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

表 C.7. ウィンドウフレームスタイル

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

表 C.8. タブスタイル

window.tab.justify	right, left または 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

表 C.9. ウィンドウラベル上のフォントスタイル

window.font	Font
window.justify	center, left, または right

表 C.10. その他のスタイル

すべての部品に描かれる縁	
borderWidth	Number of pixels
borderColor	Color
bevelWidth	Number of pixels
handleWidth	Number of pixels
frameWidth	Number of pixels
スタイルが読み込まれるとき実行されるコマンド	
rootCommand	シェルコマンド、例えばfbsetbg nicepiccy.jpg
旧 Blackbox 0.51.x のリソース	
menuFont	Font
titleFont	Font

付録 D. Fluxbox で Artwiz フォントを使うには

はじめに

いわゆる Artwiz フォントは、自身を Artwiz と呼んでいる人物が(驚くべきことに)たった一人で作成しました。フォントは、Oliwier Ptak (aleczapka) のプロジェクトページ (artwiz-aleczapka - Artwiz font revisited) から gtk2/kde3 アプリケーション互換のもの [訳注: fontconfig を介して利用できるように改良されている] か、旧形式のものを Han のアーカイブ からダウンロードできます。

Mandrake の RPM を持っているなら、そのフォントもその中に含まれているので手作業でインストールする必要はありません [訳注: また Debian でもパッケージ(xfonts-artwiz) が存在するようです]。そうではない方のために、ここで、インストール方法について説明しましょう。artwiz フォントのインストールには、システム共有と特定のユーザのみの利用の二つの方法があります。

システム共有インストール

システム上のすべてのユーザ向けにフォントをインストールするには、アーカイブを /tmp/ [訳注: かどこか他の適当な場所] にダウンロードして、[訳注: root (スーパーユーザー) になって] 次のようにします:

```
# 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
```

[訳注: chkfontpath はおそらく Red Hat などに固有のフォントを管理するコマンドでしょう。このセクション自体、日本語グリフが含まれるはずもない Artwiz フォントについてのものですからあまり深くはふれませんが、疑問点などがあれば JF 等にある、フォント関連の日本語の文書を参照した方がよいでしょう。なお参考までに前述の artwiz-aleczapka パッケージに含まれる README の INSTALL の節も簡潔でわかりやすく良いと思います。]

そしてフォントサーバを再起動します [訳注: xfs (X フォントサーバ) を使っていない場合は必要ありません]。ディストリビューションによってはシステム共有フォントの位置が異なる、例えば /usr/share/fonts であるかもしれないことに注意して下さい。しかし上述のディレクトリは常識的なものであるはずです。

ユーザインストール

一人のユーザのためだけにフォントをインストールしたいということなら話はずっと簡単になります。ホームディレクトリにアーカイブをダウンロードし、次のようにします:

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

.xinitrc か .xsession ファイル (どのように X11 を起動しているかに依ります) を編集して、他のプログラムを呼び出す前に次の行を挿入します:

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

それから X11 を(再)起動し、xlsfonts か xfontsel でフォントが認識されているかどうか確認します。

不具合

artwiz フォントは時にあなたのロカール設定と衝突を起すことがあります。その場合きちんと動作するようにするには、以下の行を .xinitrc か .xsession の冒頭に追加する必要があるかもしれません:

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

[訳注: 最初の一行は何だかよくわかりませんが必要ないはずです。ロカールの設定は LC_ALL、LC_各カテゴリ(例: MESSAGES)、LC_* のどれも未設定の場合に最終的に LANG、という順に評価されます (つまり、よく言う LANG をまず設定しろというのは厳密には嘘です)。ちなみに当然のことながら LC_ALL が C なら日本語の入力や表示に多大な悪影響が出ますので、完全に英語環境で使うのならともかく、この設定はお勧めしかねます。]

これはロカールに対するものですので、もし他のフォントにまつわる問題やロカール関連の問題があるようなら再度上述の行を削除して下さい。以下に aleczapka が示唆してくれた別解を示します:

ロカール機能を有効にしつつ、Artwiz フォントを Fluxbox で使えるようにする方法は次のとおりです。

ロカール設定の修正

解決方法は非常に単純なものです。必要なのは fonts.alias (と/または fonts.dir) を修正することだけです。

またこれによって他のアプリケーションに関連する問題 (例 Evolution と UTF-8) も修正できます。最初にまず適切な fonts.dir ファイルをつくらなければなりません。これは Artwiz フォントをインストールしたディレクトリに置かれているはずですが、もしなければそのディレクトリに移動して、mkfontdir を実行します。

このファイルの構文は簡単です。最初の行はディレクトリ内のフォントの数を示しています。その後続くすべての行は次のような形式となっています:

```
font_filename fontname
```

fonts.dir ファイルの内容の例です:

例 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
```

フォントが簡略表記されているエントリ (ここではフォント Nu, Drift, Cure そして Aqui)だけに注目します。問題はそれらが完全な X11 フォント名を欠いていることです。

ファイルを修正します:

```
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
```

最後に iso646 ではないエンコーディングのフォントを使うために fonts.alias ファイルを修正します。

構文は フォント別名 フォント名となっていて、例えば artwiz フォントを ISO-8859-2 エンコーディングで使うには次のような別名を定義します (すべて一行に収める必要があります)。

```
-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
```

私の fonts.dir と fonts.alias ファイルを参考までに。ISO-8859-1, ISO-8859-2 そして iso10646-1 (UTF-8) をサポートしています。上述のように設定できれば環境変数 LC_* を C あるいは POSIX に変更する必要はありません。そして Fluxbox だけでなくすべてのアプリケーションが "can't convert character set" といったメッセージを出すことなく、しかるべき動作をするはずで

付録 E. デバッグ

一般的な情報

デバッグは習熟可能なスキルです。良いバグ報告のつくる方法を説明する文書は非常に沢山あります。バグは逃げないしバグに殺されたりはしないので、行動に移す前にそれらを読んで下さい。むしろ私達の方がそのバグを殺したい、あなたの助けが欲しいと思っています :-) 十分準備できたら戦場におもむきましょう。Bugzilla Bug Reporting HOWTO (同和訳) と Simon Tatham's How to Report Bugs Effectively という二つの素晴らしい文書があります。

私たちは、いくつかの非公式パッチについては、その性質からすべてをサポートすることはできません。つまり、Fluxbox がクラッシュしたら、適用したパッチなしでもクラッシュするのか確かめるべきです。もしディストリビューションのメンテナがパッチを適用しているのであれば、その問題について彼(女)に話してみてください。

もしデバッグについてこのガイドが提供する以上の助けが必要ななら、irc.freenode.net の #fluxbox に join して、そこの誰かに手助けを求めることもできます。そこにはいつも沢山の開発者と能力のある人達がいるので、問題を解決できるかもしれません。

Fluxbox 固有の事項

Fluxbox の出力メッセージ

他のアプリケーションと同様に、Fluxbox は起動時からすべてのメッセージをコンソールに出力していきます。問題は、通常すべての出力は text コンソールに出力されるということです。Fluxbox を起動している行を `exec xterm` (かまたは好みの仮想端末) に変えて X11 を起動し、Fluxbox をその端末から起動してみれば、驚くなかれ、Fluxbox のすべてのメッセージを簡単に見ることができます。

必要な情報

沢山のことを知りたいので、すべてが揃っていることを確かめて下さい。

- ・ OS / ディストリビューションとそのバージョン
- ・ Fluxbox のバージョン。CVS 版ならその日付
- ・ いつそれが起きたか? 何をしていたか? 再現できるか?
- ・ `~/fluxbox/init` の設定内容

コアダンプについてはどうするか

Fluxbox がコアダンプしたら次のようにします: OPN のチャンネル #fluxbox に join して fluxgen にコアダンプしたと告げて下さい。彼が求めるすべての情報を与えて下さい。また彼はおそらく次のようにするように伝えてくるでしょう。非常に沢山の作業となりますが、一般常識といくらかの Unix の経験がある人なら誰でもできます。ああそれから GNU デバッガ gdb が必要です。

Fluxbox の再構築

ええよく読んでますね。きちんとデバッグするには Fluxbox をデバッグシンボル付きで再構築しなければなりません。

通常と同じようにビルドしますが、make 実行時に次のオプションを追加します:

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

公式サイトあるいはディストリビューションから取得したソース RPM を使うなら次のようにします:

```
$ su
# rpm -ivh fluxbox-0.1.11.lmdk.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
```

[訳注: SRPM は必ずしも root でインストールする必要はありません。むしろ訳者は以下のようにできるかぎりユーザ権限で行う方をおすすめします。

まず ~/.rpmmacros を準備し、パスやディレクトリの設定をします:

```
$ cd ~
$ pwd
/home/foo
$ cat ~/.rpmmacros
%_topdir      /home/foo/rpm
%_tmppath     /tmp
$ mkdir -p ~/rpm/{BUILD,RPMS/{athlon,i{3,4,5,6}86,noarch},SOURCES,SPECS,SRPMS}

$ rpm -Uvh fluxbox-0.1.11.lmdk.src.rpm
$ env DEBUG=true rpm -ba ~/rpm/SPECS/fluxbox.spec
$ sudo rpm -Uvh --force ~/rpm/RPMS/i686/fluxbox*
...
```

(sudo については Sudo Main Page などを参照のこと):訳注終]

(そこでコアダンプを取得できるように) Fluxbox ディレクトリに移動します。シェルはコアファイルを無効にする小粋な機能を持っているので、確実にコアファイルを取得できるようにします: 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
```

X11 を起動し、デバッグしましょう。Fluxbox がコアを吐く動作をさせてデバッグ開始です:

```
$ gdb fluxbox core
```

gdb 内で次のように入力します (最初の部分は gdb のプロンプトなので入力しないで下さい:):

(gdb) where

一杯出力されるでしょう。fluxgen が知りたいと思うのは # ではじまるすべての出力です。

さて、そのログをメールに貼り付け、~/fluxbox/ 以下の四つの設定ファイルも添えて、fluxgen に送ります。

コアとソースコードディレクトリはまだ消さないで置いて下さい。fluxgen は二三別の質問をしたいかもしれないし、そのときそれらが必要となります。fluxgen が明確に求めないかぎりコアを送らないで下さい。なぜならそれはシステム固有のもので、おそらくその必要はなく、かつ容量が大きくなりがちですから。