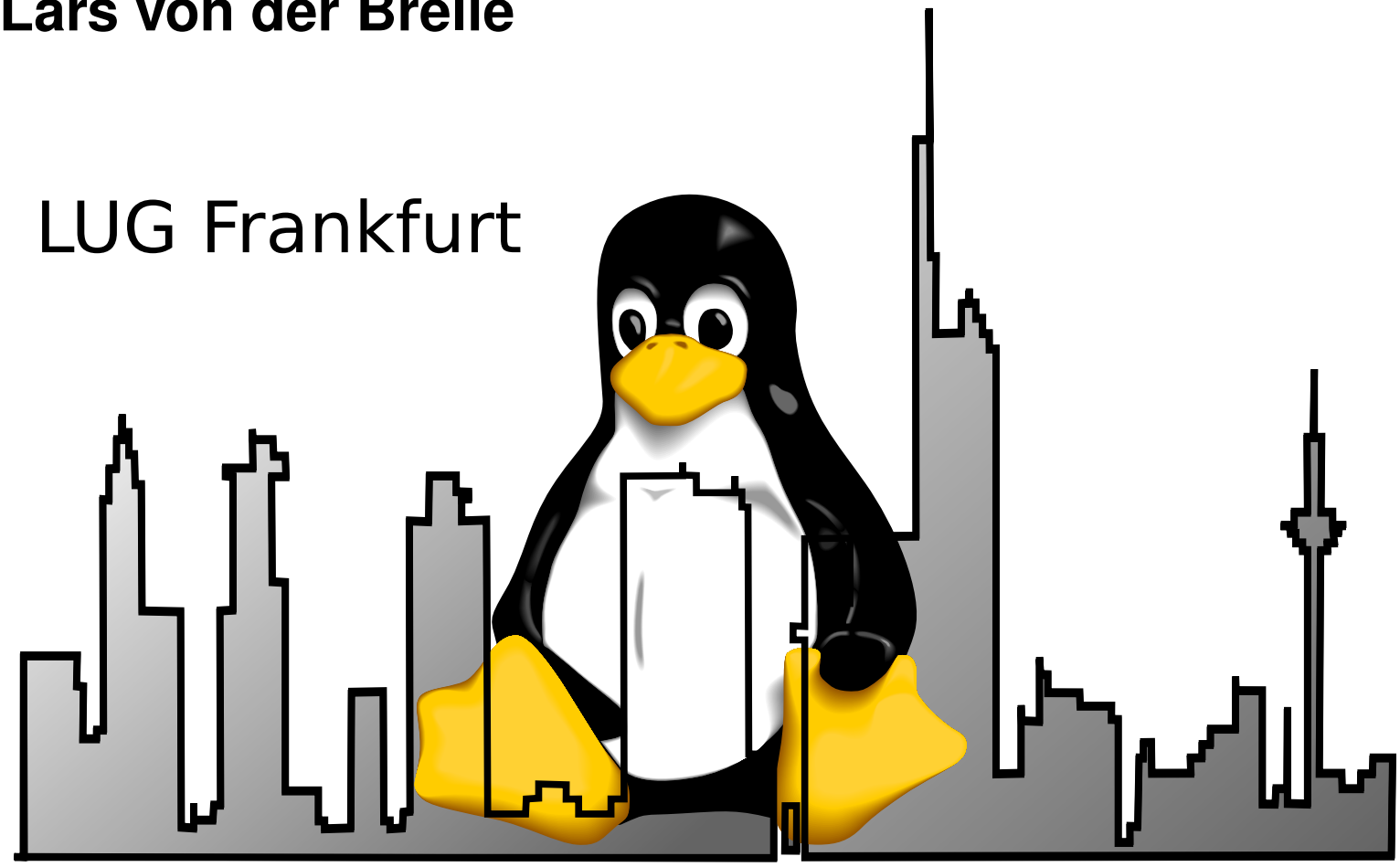
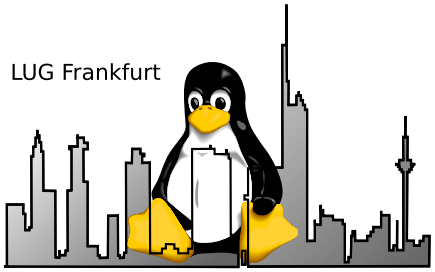
Fluxbox

Der wandlungsfähige und flexible underdog

Lars von der Brelie

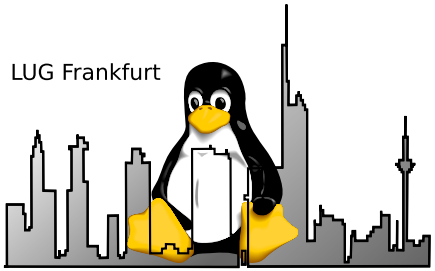
LUG Frankfurt





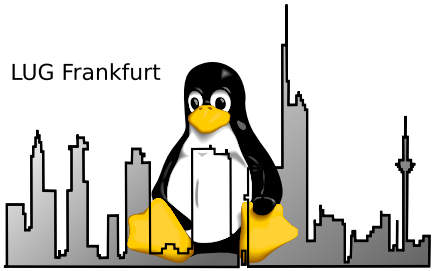
Übersicht

1. Persönliche Anmerkungen
2. Philosophie meine Werkzeuges
3. Windowmanager vs. Desktop environment
4. Geschichte
5. Installation
6. Konfiguration und Nutzung
7. Styles
8. Troubleshooting
9. Zusatzinformationen



Wer bin ich?

Name:	Lars von der Brelie
Herkunfts LUG:	LUG Frankfurt
Linux vita:	• Slackware (Heimserver im Studentenwohnheim) • Slackware (Desktop) • Debian (kurzes Intermezzo) • Arch Linux
Status:	interessierter Amateur



Philosophie meines Werkzeuges

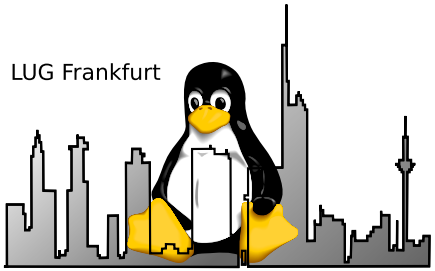
Vorbemerkung

Das hier vorgetragene ist meine eigene Meinung.

Im folgenden wird allerdings klar, dass jeder seine eigene Meinug zu den Sachen haben darf, soll und vielleicht auch haben muss.

Ich sehe viele Sachen einfach als Werkzeug. Für ein Werkzeug gibt es einen Einsatzzweck. In diesem funktioniert es gut und richtig.

Missbrauch ist hier doof und gängig.

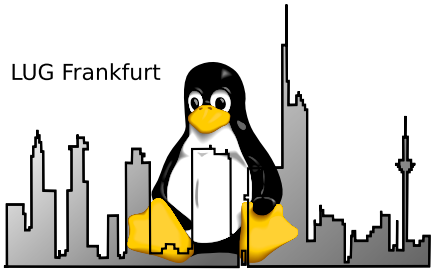


Philosophie meines Werkzeuges

Software als Werkzeug

Ein Werkzeug muss die Arbeit leichter, einfacher und schneller machen.
Es muss mich unterstützen.

Es darf mich auf keinen Fall ausbremsen oder gar behindern!

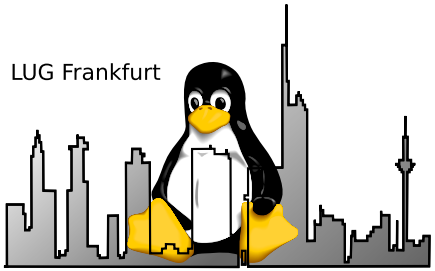


Philosophie meines Werkzeuges

Arbeitsweise

Der Computer, den ich auch als Werkzeug verstehe, muss also auch nach meiner logik und nach meiner Arbeitsweise arbeiten.

- Damit muss auch jedes Umfeld von jedem Benutzer individuell konfiguriert werden können und es muss demnach auch entsprechend gemacht werden.
- Das erfordert natürlich Eigenleistung
- Viele Menschen glauben offenkundig, dass sie mit einem so komplexen Gerät, wie einem Computer, ohne Grundwissen umgehen können.
- Eine Konfiguration ist niemals fertig. Die Anforderungen ändern sich stetig und es gibt neue Ideen und Ansätze, die eingepflegt werden können.



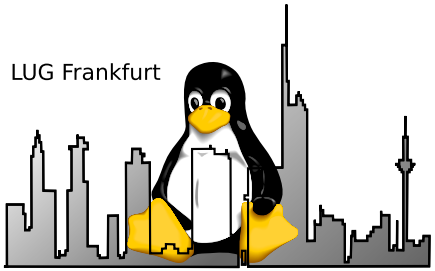
Philosophie meines Werkzeuges

Arbeitsweise

Warum Resourcen schonen und das eye-candy weglassen?

Weniger Wartezeit beim Arbeiten und keine Ablenkung durch schischi.

Schischi der hilft geht!

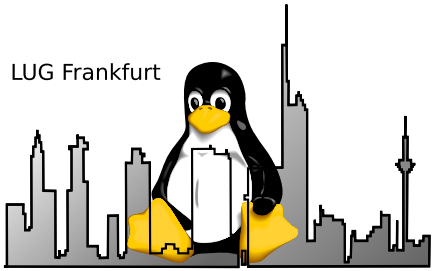


Philosophie meines Werkzeuges

Ergo

Man muss es selber machen. Durch die eingebrachte Arbeit und die Recherche steigt auch das eigene Kompetenzlevel.

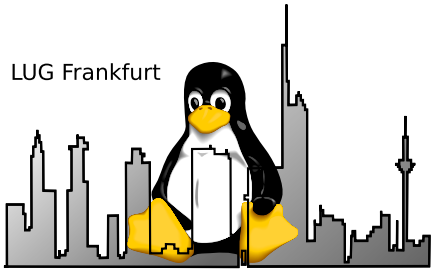
Da kann man nichts machen!



Windowmanager

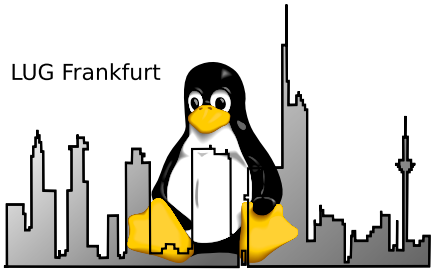
Ein Windowmanager verwaltet die geöffneten Fenster und stellt die Infrastruktur für die GUIs der Programme bereit.

Streng genommen stellt er kein Menü, keine Toolbar und auch sonst keine Automatismen bereit.



Desktop environment

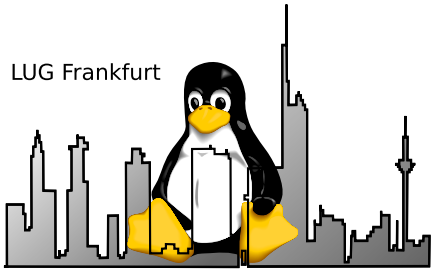
Ein Desktop environment stellt nun eine komplette Infrastruktur zur Verfügung. Mit Menü, mit Toolbar und mit einem eingebetteten Window manager zur Darstellung der GUIs. Eventuell auch noch verschiedene Hintergrunddienste.



Was ist Fluxbox?

Fluxbox ist ein Windowmanager und damit eigentlich nicht autark nutzbar.

Aber es hat eine Toolbar und ein Menü und es bringt alles mit, was man zur Nutzung braucht. Auch keine Desktop icons.

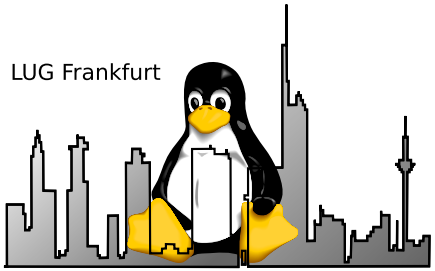


Was ist Fluxbox

Die Zusätze sind allerdings alle abschaltbar, sodass Fluxbox als reiner Windowmanager genutzt werden kann.

In Plasma oder Gnome.

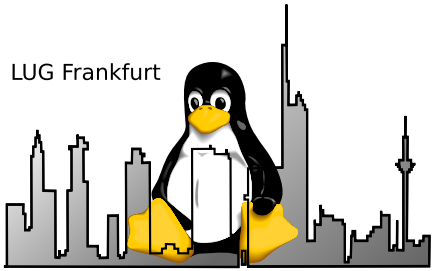
Viel Glück!



Kurze geschichtlicher Abriss

Fluxbox ist ein Fork des Quellcodes von Blackbox 0.61.1. Das war direkt am Anfang dieses Jahrtausends.

Fertig.

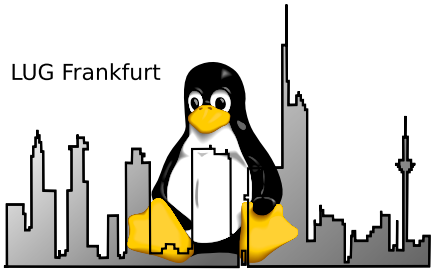


Installation

Die Installation ist normalerweise unkompliziert. Fluxbox ist in den Repositorien der Distributionen

- Slackware
- Arch Linux
- Ubuntu (siehe auch fluxbuntu)
- Fedora
- openSUSE
- Debian

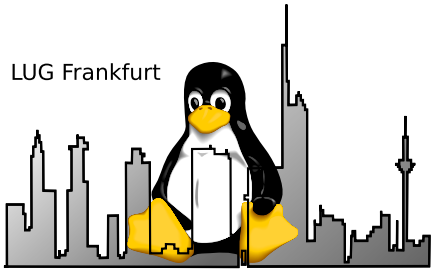
enthalten. Diese Liste ist nicht vollständig.



Installation

Ansonsten ist das Paket tatsächlich auch schnell selbst kompiliert.

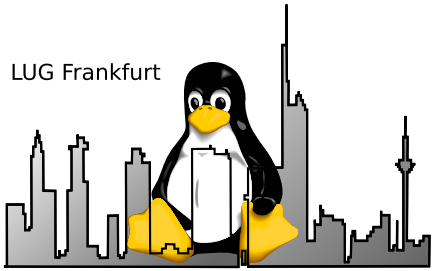
Fluxbox läuft leider nicht auf wayland.



Konfiguration und Nutzung

Alles ist Text

. . . und er wird immer lesbar sein.



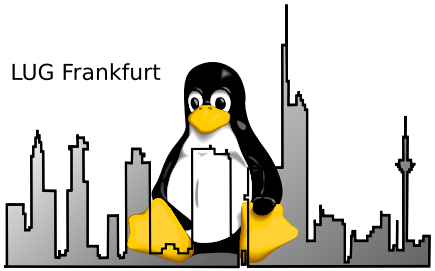
Konfiguration und Nutzung

Überblick

Die Konfiguration ist per default in `~/.fluxbox/` und `/usr/share/fluxbox/`.

Die Konfiguration erfolgt durch die Dateien:

- init
- startup
- menu
- apps
- slitlist
- windowmenu
- overlay



Konfiguration und Nutzung

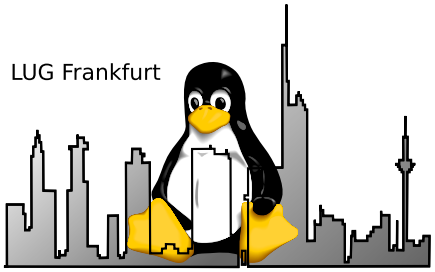
init

In den Dateien `~/.fluxbox/init` und `/usr/share/fluxbox/init` ist die Konfiguration für die Laufzeit gespeichert.

Sie ist auch während der Laufzeit per Hand änderbar, wird aber überschrieben, wenn anschließend mit einem der Menüs eine Einstellung geändert wird[1].

Einfach Speichern und sofort die Konfiguration neu einlesen.

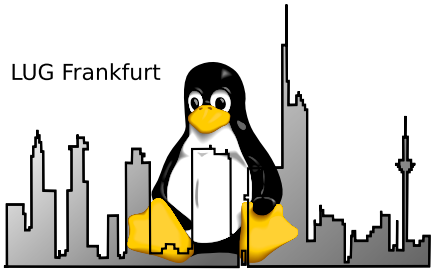
[1] Siehe den Teil über das Menü im folgenden



Konfiguration und Nutzung

init: Dateiauszug

```
...
session.screen0.window.focus.alpha: 255
session.screen0.window.unfocus.alpha: 255
session.screen0.systray.pinLeft:
session.screen0.systray.pinRight:
session.screen0.iconbar.mode: {static groups} (minimized=yes) (workspace)
session.screen0.iconbar.alignment: RelativeSmart
session.screen0.iconbar.iconWidth: 128
session.screen0.iconbar.iconTextPadding: 10
session.screen0.iconbar.usePixmap: false
session.screen0.menu.alpha: 255
session.screen0.tabs.intitlebar: true
...
```

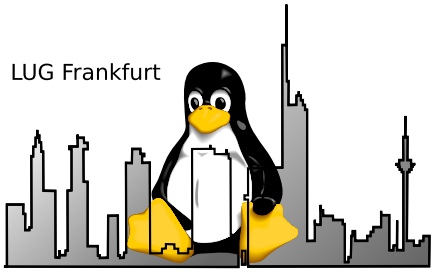


Konfiguration und Nutzung

startup

Durch die Dateien `~/.fluxbox/startup` und `/usr/share/fluxbox/startup` wird Fluxbox und alle Programme gestartet. Egal, ob die Programme vor oder nach dem Fluxboxstart gestartet werden müssen.

startup ist ein shell Skript. Die korrespondierende man page ist startfluxbox.

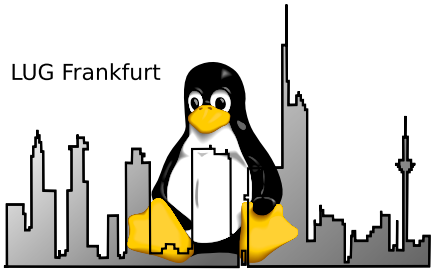


Konfiguration und Nutzung

startup

Es gibt mehrere Wege Programme beim Start mitzustarten. Der Weg über die Datei startup ist best practice und der empfohlene Weg.

Von einer Mischung mehrerer Methoden rät nicht nur der Author ausdrücklich ab.



Konfiguration und Nutzung

startup

Man achte dringend auf die & an den Zeilenenden

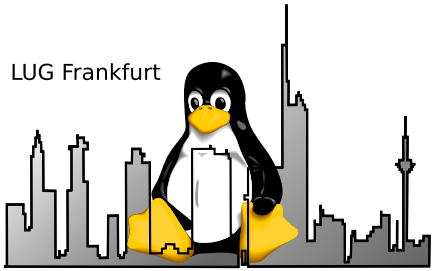
```
#!/bin/sh
#
# fluxbox startup-script:
#
# Lines starting with a '#' are ignored.

# Change your keymap:
xmodmap "$HOME/.Xmodmap"

# Applications you want to run with fluxbox.
# MAKE SURE THAT APPS THAT KEEP RUNNING HAVE AN & AT THE END.
#
# unclutter -idle 2 &
# wmnd &
# wmsmixer -w &
# idesk &

# And last but not least we start fluxbox.
# Because it is the last app you have to run it with exec before it.

exec fluxbox
# or if you want to keep a log:
# exec fluxbox -log "$fluxdir/log"
```



Konfiguration und Nutzung

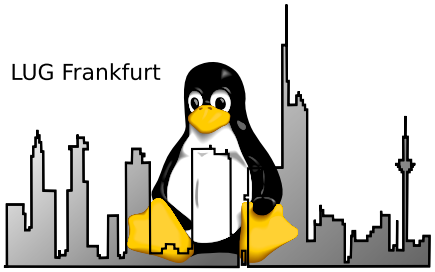
startup

Programme mitstarten:

```
...
# Applications you want to run with fluxbox.
# MAKE SURE THAT APPS THAT KEEP RUNNING HAVE AN & AT THE END.
#
# unclutter -idle 2 &
# wmnd &
# wmsmixer -w &
# idesk &

# And last but not least we start fluxbox.
# Because it is the last app you have to run it with exec before it.

exec fluxbox
...
```



Konfiguration und Nutzung

startup

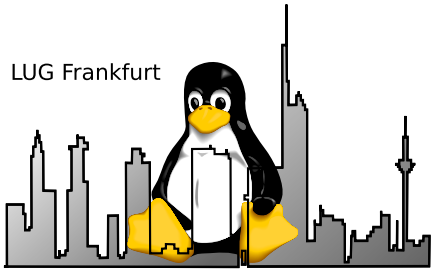
Programme nach fluxbox starten:

```
...
# And last but not least we start fluxbox.
# Because it is the last app you have to run it with exec before it.

exec fluxbox &
fbpid=$!

sleep 1
{
    # Applications you want to run after fluxbox has started
    ipager &
    gkrellm &
} &

wait $fbpid
```



Konfiguration und Nutzung

keys

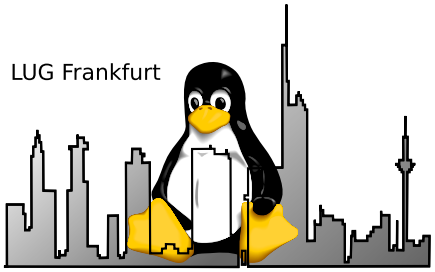
Die Dateien `~/.fluxbox/keys` und `/usr/share/fluxbox/keys` definiert die shortcuts.

Die manpage ist fluxbox-keys.

Das Format eines Eintrages ist:

```
[Modifizierer] Taste :'Kommando' [Argumente ...]
```

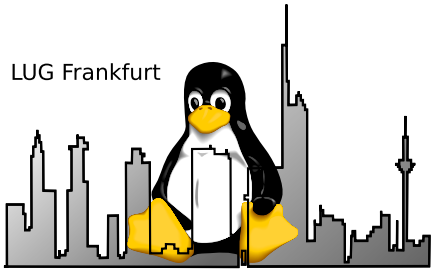
Im Zweifel bekommt man die Keycodes und X-Symbole mit dem Programm `xev`. Tasten ummappen kann man in der Datei `.Xmodmap`. Das passiert damit in der Konfiguration vom X-Server.



Konfiguration und Nutzung

Kommentare

Kommentare werden am Zeilenbeginn mit # oder ! eingeleitet.

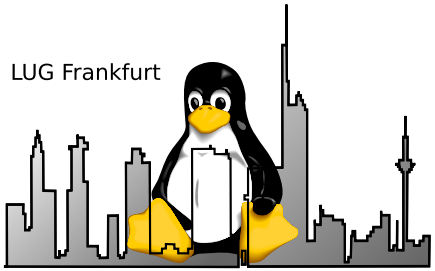


Konfiguration und Nutzung

keys: Kommandos

Die Kommandos werden in verschiedene Gruppen kategorisiert. Es sind wirklich viele Kommandos!

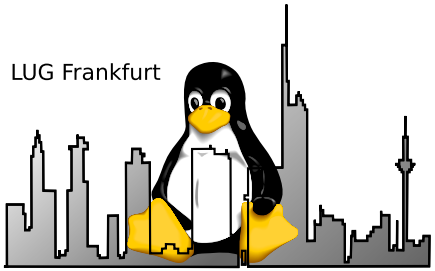
- Mouse commands
- Window commands
- Workspace commands
- Menu commands
- Window manager commands
- Special commands



Konfiguration und Nutzung

keys: Kommandos

Da das einfach viel zu viele Kommandos für hier sind, seien die Kategorien kurz erklärt und einige bemerkenswerte herausgepickt. Siehe hier dringend die manpage fluxbox-keys.

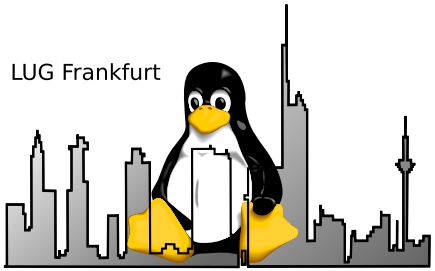


Konfiguration und Nutzung

keys: Kommandos: mouse commands

Diese Kommandos werden an die mousebuttons gebunden.

Also kann die Interaktion mit der mouse auch in allen Spezialitäten, wie Chaining, Makros, Switchings, KeyModes, etc. genutzt werden!

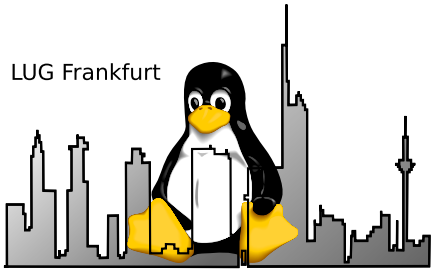


Konfiguration und Nutzung

keys: Kommandos: window commands

Diese Kommandos werden ausschließlich das aktuell fokussierte Window gesendet.

Mit den Ausnahmen `onWindow` modifizierer und dem `ForEach` Kommando.



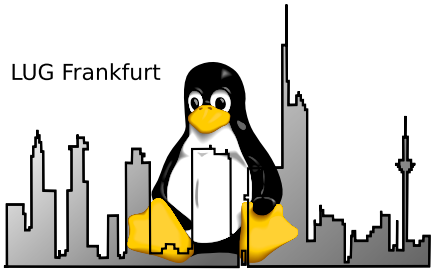
Konfiguration und Nutzung

keys: Kommandos: workspace commands

Diese Kommandos wirken sich auf den workspace / desktop aus.

Hier werden die `ArrangeWindows*` Kommandos und das `Unclutter` Kommando hervorgehoben. Da kommen wir beim pseudoTiling noch drauf.

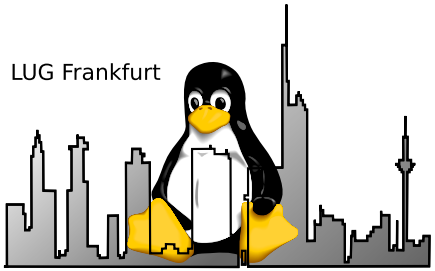
Interessant ist auch `CloseAllWindows`. Dieses Kommando schließt alle windows auf allen desktops.



Konfiguration und Nutzung

keys: Kommandos: menu commands

Hier stehen nur ein paar übersichtliche Befehle. Das explodiert einem dann später bei den Menüs ins Gesicht.



Konfiguration und Nutzung

keys: Kommandos: window manager commands

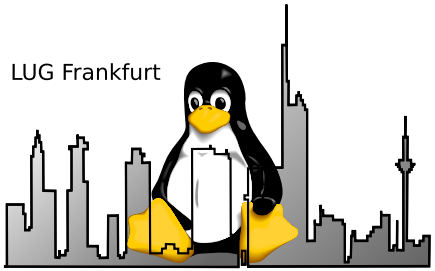
Hier sind die Kommandos zur Steuerung des window managers enthalten. Also:

Restart : Naja . . .

Quit/Exit : siehe restart

Exec : Die Programmausführung von Programmen. Der Aufruf wird dann an `$SHELL` geschickt, oder `/bin/sh` wenn die Variable `$SHELL` nicht gesetzt ist.

Die Ersetzungen in der bash (z. B.: ~ → /home/*user*/) funktionieren hier!

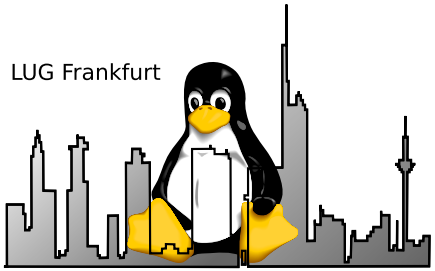


Konfiguration und Nutzung

keys: Kommandos: window manager commands

SetEnv : Setzt environment variables.

SetResourceValue : Setzt eine Resource, die normalerweise in der Datei `init` gespeichert wird.



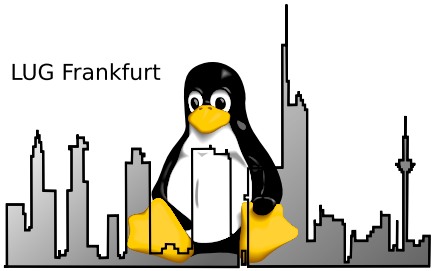
Konfiguration und Nutzung

keys: Kommandos: special commands

Hier wird es spannend!

MacroCmd: Mehr als ein Kommando gleichzeitig absetzen

```
1 Mod4 M :MacroCmd {MoveTo 0 0} {ResizeTo 1280 800}
```

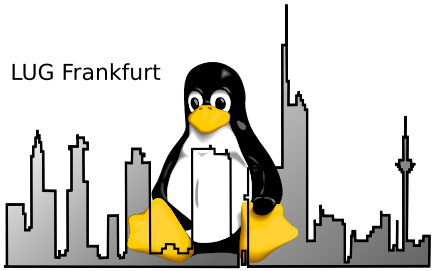


Konfiguration und Nutzung

keys: Kommandos: special commands

Delay: Das Kommando zeitversetzt absetzen

```
1 Mod4 L :Delay {SetLayer Dock} [1000]
```

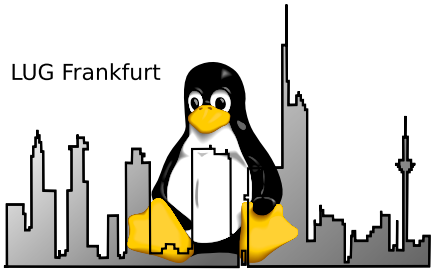


Konfiguration und Nutzung

keys: Kommandos: special commands

ToggleCmd: Führt die angegebenen Kommandos zyklisch aus

```
1 Mod4 t :ToggleCmd {Exec xinput disable 13} {Exec xinput enable 13}
```

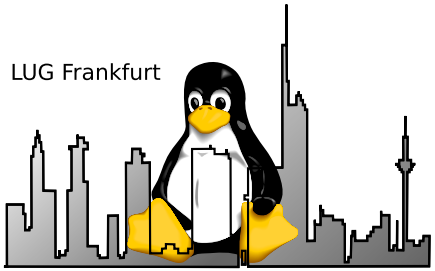


Konfiguration und Nutzung

keys: Kommandos: special commands

ToggleCmd: Führt die angegebenen Kommandos zyklisch aus

```
1 Mod4 t :ToggleCmd {Exec xinput disable 13} {Exec xinput enable 13}
```

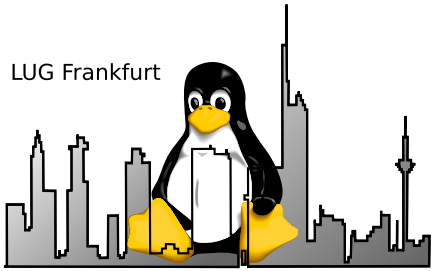


Konfiguration und Nutzung

keys: Kommandos: special commands

CustomMenu: Öffnet ein angegebenes Menü aus einer eigenen Datei

```
1 Menu space :CustomMenu ~/.fluxbox/.streng_geheimes_menue
```



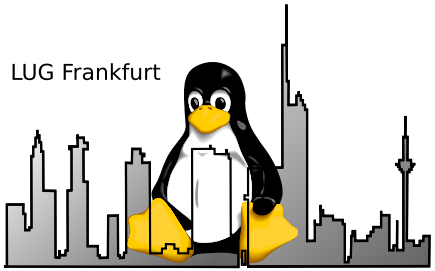
Konfiguration und Nutzung

keys: Kommandos: special commands

KeyMode: Aktiviert und deaktiviert den benannten KeyMode. Das beenden dieses Modus erfolgt per `ESC` (default) oder per zugewiesener Tastenkombination.

```
1 Control F1 :KeyMode Neuer_Modus
```

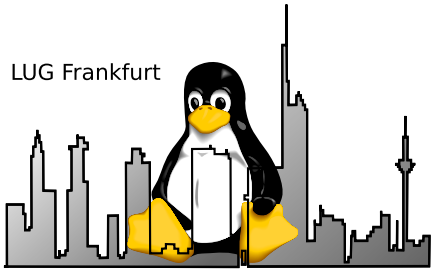
Dazu später mehr.



Konfiguration und Nutzung

keys: Kommandos: special commands

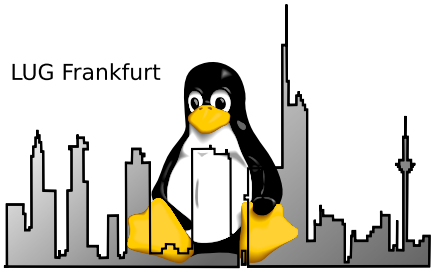
ForEach: Führt ein Kommando auf allen Fenstern aus. Es sei denn sie werden gefiltert.



Konfiguration und Nutzung

keys: Kommandos: special commands

If: Führt ein Kommando basierend auf einer Bedingung aus.

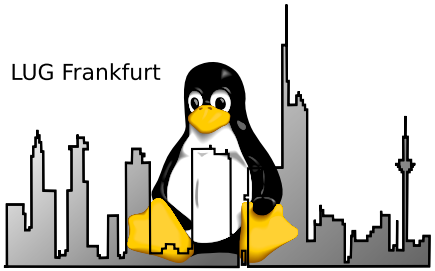


Konfiguration und Nutzung

keys: Kommandos: special commands

BindKey: Hiermit kann mit Fluxbox Command on-the-fly ein Shortcut angelegt werden.

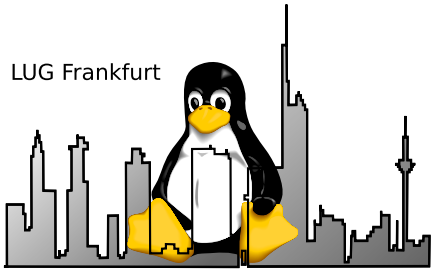
Bei der direkten Eingabe der Befehle durch den Menüpunkt `Fluxbox command` im Konfigurationsmenü.



Konfiguration und Nutzung

Nutzung der Kurztasten

Ab hier kommen jetzt ein paar Nutzungsfälle.



Konfiguration und Nutzung

keys: einfacher Befehl

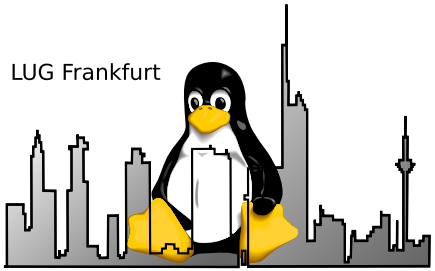
Modifiezierer sind Tasten, wie Control, Shift, Alt. In der X-eigenen Nomenklatur (Mod1, Mod2, . . .)

Man kann viele Modifizierer mittels Leerschlag getrennt angeben.

Beispiel: Alt + F4

```
Mod4 F4 :Close
```

Schließt die fokussierte Anwendung.



Konfiguration und Nutzung

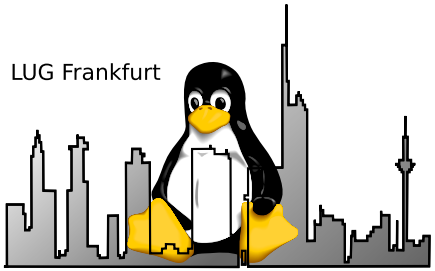
keys: chaining

Kurztasten können auch verkettet werden (hallooo Emacs!).

Beispiel: Ctrl + x Ctrl + c

```
Control x Control c :Quit
```

beendet Fluxbox.



Konfiguration und Nutzung

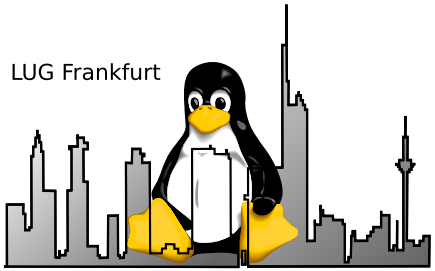
keys: keymodes

Es ist möglich ein spezifisches shortcut-setting zu aktiveren oder zu deaktiveren. Das erfolgt im laufenden Betrieb.

Aufgebaut ist die Zeile so:

```
keymode: modifizierer tasten :'Kommando' [Argumente]
```

keymode ist der Name des Modus. Alle Zeilen mit dem Modusnamen werden dann bei der Modusaktivierung aktiviert und sind dann durch die Aktivierung aktiv.

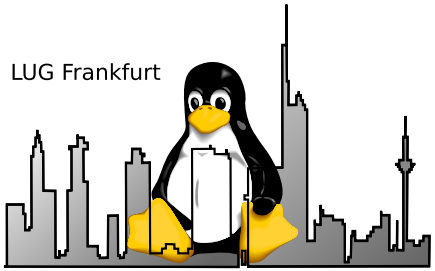


Konfiguration und Nutzung

keys: keymodes

Die Aktivierung und Deaktiverung erfolgt mit dem KeyMode Kommando. Während der Aktivierung werden nur noch diese Tastenkürzel genutzt und alle anderen ignoriert.

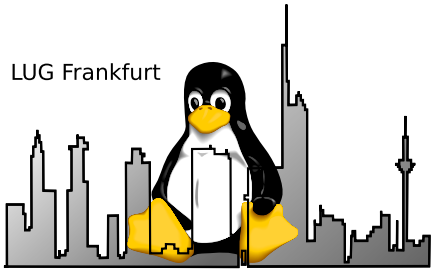
```
1 Mode_Ende: F1 :Close
2 Mode_Ende: F2 :Quit
```



Konfiguration und Nutzung

keys: keymodes

Das De-/Aktivieren erfolgt dann, wie vorgenannt, mit dem Kommando KeyMode, das auch in dieser Datei belegt wird, wie vorgenannt.

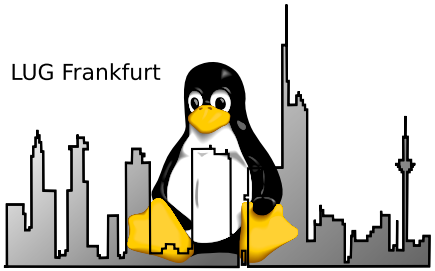


Konfiguration und Nutzung

keys: MacroCmd

Mit diesen Makros kann man mit einem Tastendruck mehrere Befehle absetzten.

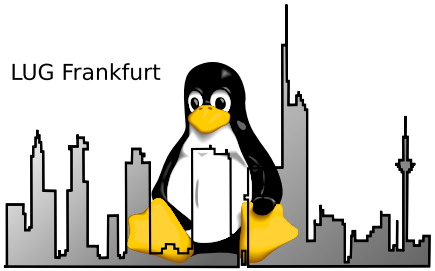
```
1 MacroCmd: {MoveTo 0 0} {ResizeTo 1280 800}
```



Konfiguration und Nutzung

keys: Pseudo tiling

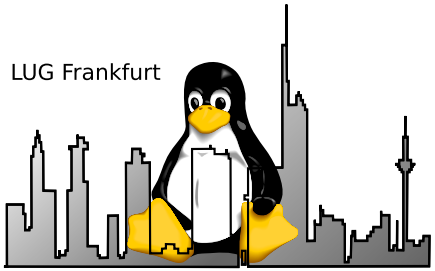
Fluxbox ermöglicht es, die offenen Windows auf Tastendruck zu arrangieren.



Konfiguration und Nutzung

keys: Pseudo tiling: Unclutter

Das Kommando Unclutter arrangiert die offenen Windows mit ein möglichst wenig Überlappung, ohne ihre größe zu verändern.

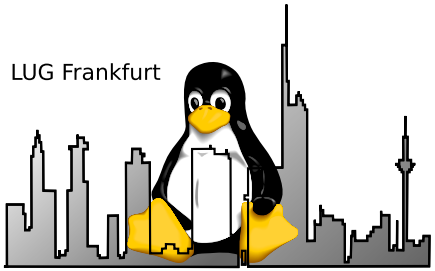


Konfiguration und Nutzung

keys: Pseudo tiling: ArrangeWindows*

Mit ArrangeWindows, ArrangeWindowsVertical, ArrangeWindowsHorizontal, ArrangeWindowsStackLeft, ArrangeWindowsStackRight, ArrangeWindowsStackTop und ArrangeWindowsStackBottom werden die offenen Windows auf dem Bildschirm angeordnet. Dabei wird ihre Größe geändert.

```
1 Mod4 left :ArrangewindowsStackRight
```



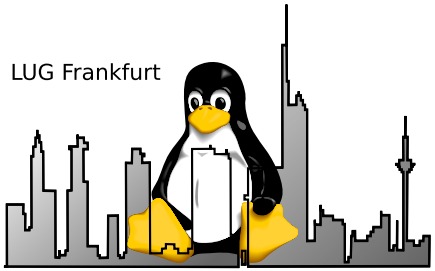
Menü

Mein Menü

Das ist mein Menü, es gibt viele Menüs aber das ist meines. . . .

 [2]

[2] https://i.ytimg.com/vi/WIJzgiJQGhw/maxresdefault.jpg

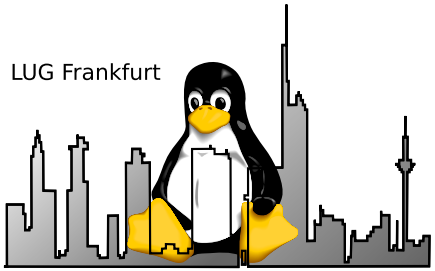


Menü

It’s full of menues!

Es gibt mehrere Menüs in Fluxbox:

- Window Menü (~/.fluxbox/windowmenu)
- root Menü (~/.fluxbox/menu)
- workspace Menü
- Die aufrufbaren Menüs durch Shortcuts



Menü

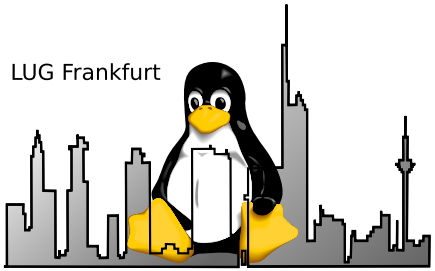
Windowmenu

Das Windowmenu wird in

`~/.fluxbox/windowmenu` und

`/usr/share/fluxbox/windowmenu`

gespeichert.

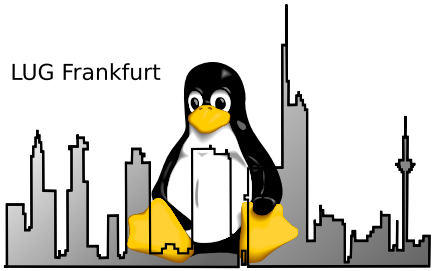


Menü

Windowmenu

Das Windowmenu ist das Menü das erscheint, wenn ein rechts-click auf die Titelzeile ausgeführt wird.

Damit wird das Verhalten der Windows gesteuert.

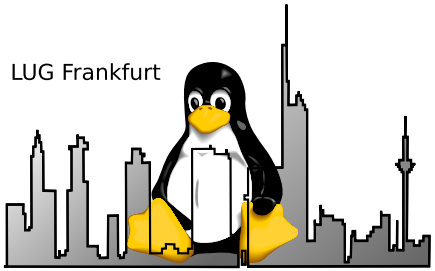


Menü

Windowmenu

Das windowmenu darf allerdings auch alle der im folgenden Kommandos enthalten, die auch das root-Menü enthalten darf.

Wer kann schon seinen Rechner über das Kontextmenü seines Programmes runterfahren? ;-)

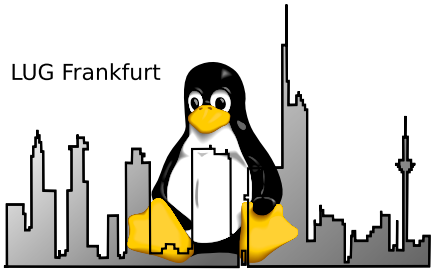


Menü

Workspace Menü

Das workspacemenu ist ein Menü, das erscheint, wenn man auf dem Desktop einen mittleren mouseclick vollführt.

Hier kann man die verschiedenen Workspaces verwalten und auch Windows auf anderen Workspaces direkt fokussieren.



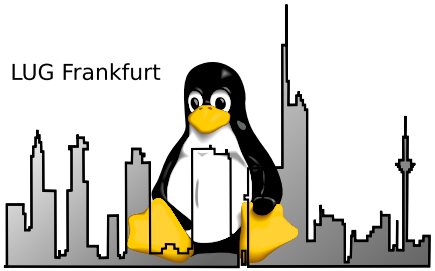
Menü

root Menü

Es gibt in modernen Oberflächen eine Suchfunktion in deren Menüs.

Warum?

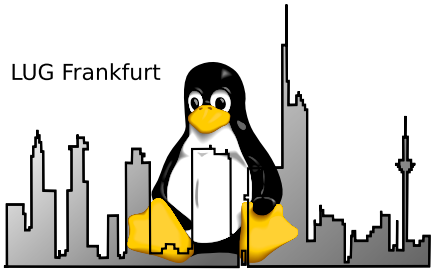
Wenn ein Menü so groß und komplex ist, dass eine Suche benötigt wird, ist das Menü aufgrund dessen Größe und Komplexität nutzlos (IMHO).



Menü

Änderungen ausrollen

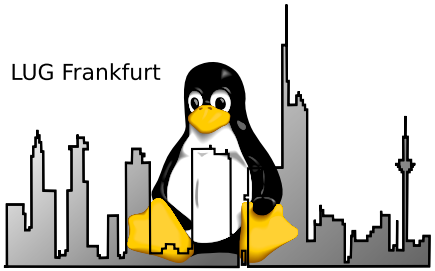
Die Änderungen in der Dateien des root-Menüs wird nach jedem Speichern neu gelesen. Es muß also nicht die Konfiguration neu geladen werden.



Menü

root-Menü

Dieses Menü ist das Menü, dass erscheint, wenn man auf dem Desktop einen Rechtsclick macht, oder mit einem shortcut (siehe vorne).

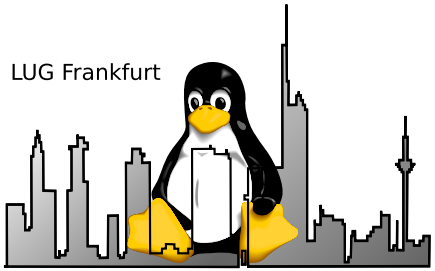


Menü

root-Menü

Das Menü ist die ganze Zeit im Fluss.

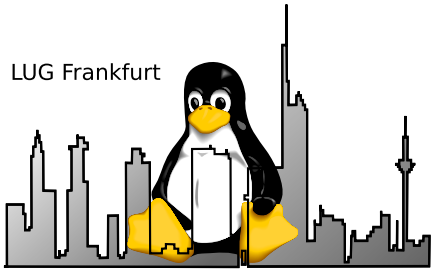
Es ändern sich mit der Zeit die Anforderungen und die Programme.



Menü

root-Menü

Das root-Menü ist in den Dateien `~/.fluxbox/menu` und `/usr/share/fluxbox/menu` gespeichert.

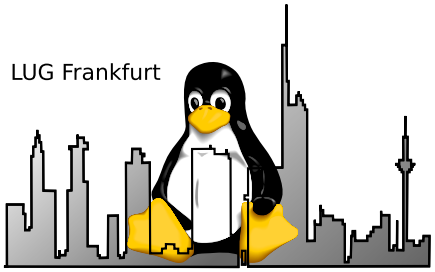


Menü

root-Menü

Wenn jetzt aber doch ein fertig generiertes Menü genutzt werden soll?

`fluxbox-generate-menu`!

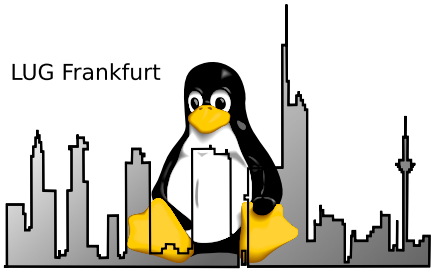


Menü

root-Menü: Aufbau

Das minimale, nicht sinnhafte Grundgerüst für ein Menü ist:

```
1 [begin] (Menue Name)
2 [end]
```

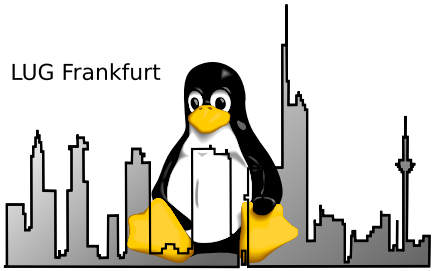


Menü

root-Menü: Aufbau: Kommentare

Es können natürlich Kommentare in die Menüdatei geschrieben werden. Das funktioniert mit den Zeichen # oder %. Dann ist diese Zeile ein Kommentar bis zum Zeilenende.

Hurra!

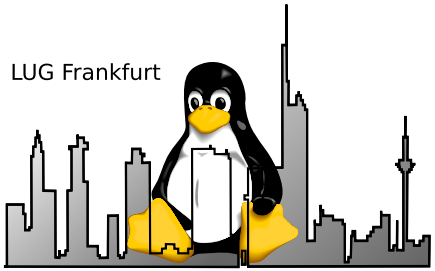


Menü

root-Menü: Aufbau

Ein Eintrag in das Menü ist also grundlegend auf folgende Weise aufgebaut:

```
1 [tag] (label) {command} <'icon'>
```

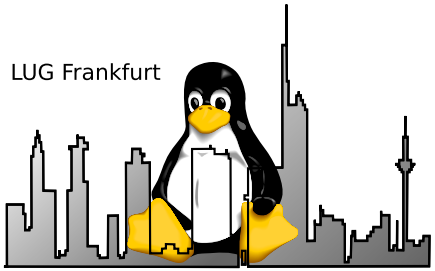


Menü

root-Menü: Instruktionen

Ein tag kann:

- die Struktur regeln,
- eine Anwendung öffnen, einen Befehl ausführen oder
- Eine Fluxbox-Funktion ausführen

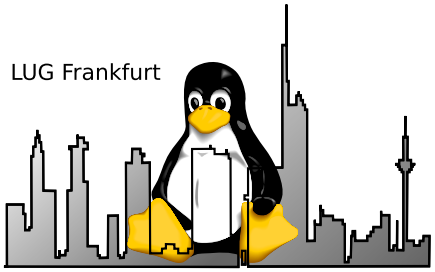


Menü

root-Menü: Instruktionen

Die Struktur-tags sind:

begin	Startet einen Block
submenu	Startet ein Untermenü
end	Beenden einen Block oder ein Untermenü
encoding	Startet ein Encoding
endencoding	Beendet ein Encoding
separator	Malt ’nen Strich
nop	Macht nix!
include	inkludiert eine weitere Datei!



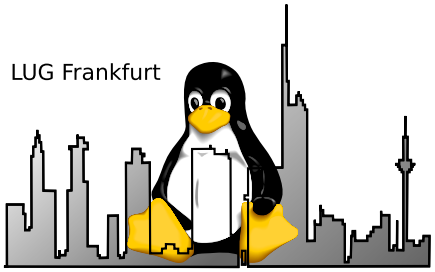
Menü

root-Menü: Instruktionen

Zum Starten von Anwendungen oder Befehlen nutzt man [exec].

Es wäre ja möglich den folgenden Eintrag in das Menü einzutragen:

```
1 [exec] (Menue anpassen) {gvim ~/.fluxbox/menu}
```



Menü I

root-Menü: Instruktionen

Die tags für die Fluxbox Funktionen sind:

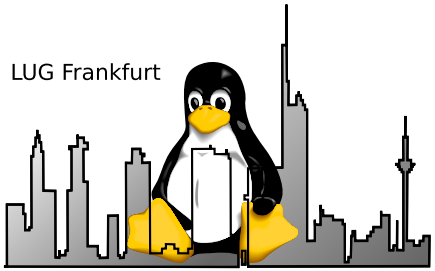
config erzeugt ein Untermenü mit verschiedenen Konfigurationsparametern (Fokussteuerung, Slit, Toolbar, . . .)

reconfig lädt die Style- und Menüdateien neu und übernimmt die Änderungen.

restart beendet Fluxbox und startet es neu. Oder ein anderes Programm, das hier angegeben wird.

exit beendet fluxbox

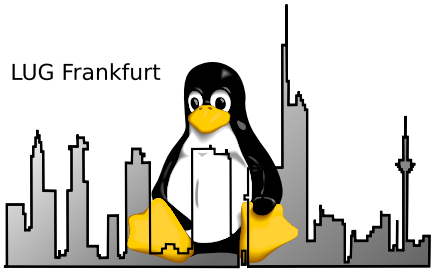
style übernimmt ein einzelnes theme-File.



Menü II

root-Menü: Instruktionen

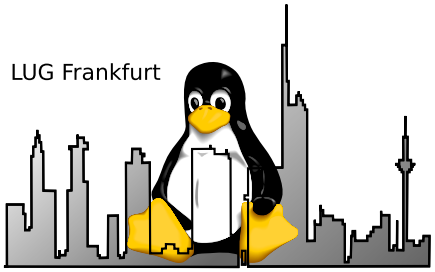
stylesmenu	erzeugt ein Menu mit allen theme-Files aus dem gegebenen Verzeichnis.
stylesdir	erzeugt ein Untermenu mit allen theme-files aus dem gegebenen Verzeichnis.
wallpapers	erzeugt einen Eintrag mit dem Inhalt des gegebenen Verzeichnisses und setzt die Wahl als Hintergrund.
workspaces	erzeugt einen Eintrag zum Öffnen des Workspaces-Menüs
command	Hier kann command jeder gültige command aus den Kurztasten sein!



Menü

root-Menü: Beispiel

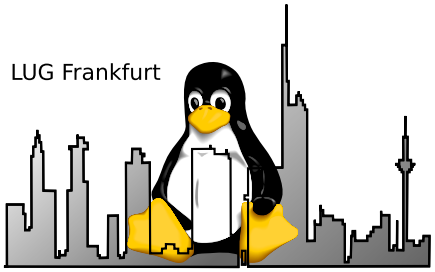
```
1  [begin] (Menu)
2  [encoding] {UTF–8}
3  [exec] (Notiz) {xpad ––new}
4
5  [separator]
6  [submenu] (Office)
7          [exec] (claws–mail) {claws–mail}
8          [exec] (LibreOffice) {soffice}
9          [nop] (Verzeichnisse)
10         [exec] (Projekte) {pcmanfm ~/Projekte/}
11         [exec](Vorlagen) {pcmanfm ~/Vorlagen/}
12 [end]
```



Menü

root-Menü: Beispiel

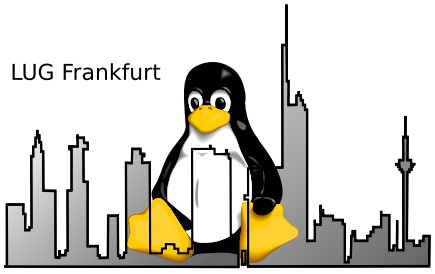
```
13 [submenu] (Grafik)
14     [exec] (Inkscape) {inkscape}
15     [exec] (gimp) {gimp}
16     [submenu] (Screenshots)
17         [exec] (jpg) {import /tmp/screenshot.jpg && display /tmp/screenshot.jpg}
18         [exec] (jpg 50%) {import /tmp/screenshot.jpg && display –resize 50% ↙
               /tmp/screenshot.jpg}
19     [end]
20 [end]
```



Menü

root-Menü: Beispiel

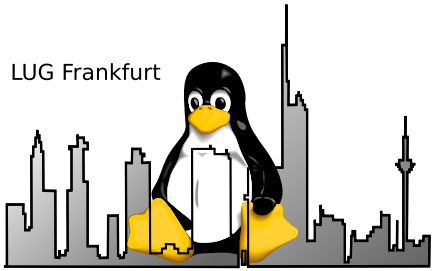
```
21 [submenu] (Konfiguration)
22     [config] (Konfiguration)
23     [submenu] (Styles)
24         [stylesdir] (/usr/share/fluxbox/styles)
25         [stylesdir] (~/.fluxbox/styles)
26     [end]
27     [submenu] (Wallpapers)
28         [exec] (Schwarz) {fbsetroot –solid black}
29         [exec] (Teal) {fbsetroot –solid Teal}
30         [exec] (Stahlblau) {fbsetroot –solid SteelBlue}
31         [separator]
32         [wallpapers] (/usr/share/fluxbox/backgrounds) {fbsetbg}
33         [wallpapers] (~/.fluxbox/backgrounds) {fbsetbg}
34     [end]
```



Menü

root-Menü: Beispiel

```
35      [separator]
36      [restart] (Restart Fluxbox)
37      [reconfig] (Reload config)
38      [exec] (Reload .Xresources) {xrdb –load $HOME/.Xresources}
39      [exit] (Exit)
40 [end]
41
42 [separator]
43 [exec] (Suspend) {systemctl suspend}
44 [exec] (Reboot) {systemctl reboot}
45 [exec] (Shutdown) {systemctl poweroff}
46 [endencoding]
47 [end]
```

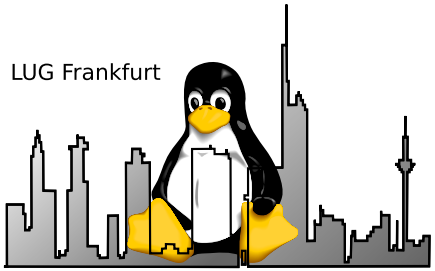


Menü

root-Menü: Anwendung

Das Menü kann auch mit der Tastatur genutzt werden.

In dem man es öffnet und und einfach anfängt zu tippen. Der Cursor springt dann auf den mutmaßlich am besten passenden Eintrag.

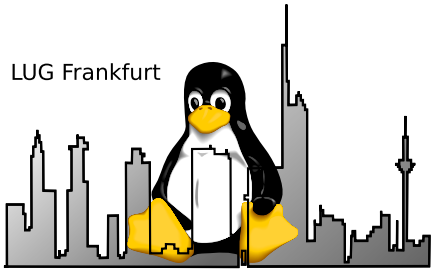


Menü

root-Menü: Dynamische Menüs

Durch das Kommando include kann man externe Dateien laden.

Dadurch sind dynamische Menüs möglich, die sich den wechselnden Gegebenheiten anpassen, indem sie von Skripten, Programmen und Regeln angelegt und beschrieben werden.

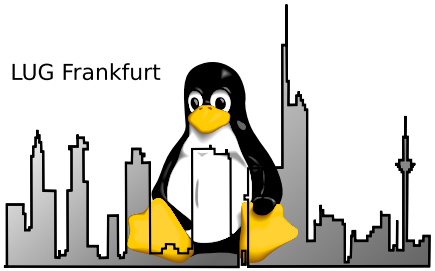


Menü

root-Menü: Dynamische Menüs

Ideen sind:

- eine Zusammenfassung der installierten Spiele von Steam™, GOG.com™, eigen installierten Spielen und Browsergames in einem einzigen Unterverzeichnis
- dynamischen Einbindung von zu mounteten Laufwerken. Wie dedizierten Speicherkarten, NAS-Speicherlaufwerken, USB-Sticks. Auch nach der Verbindung mit WLAN.
- Automatische Einbindung von Starteinträgen für installierte, virtuelle Maschinen
- Automatisches Einbinden der Playlists für den Audio-/Videoplayer
- . . .

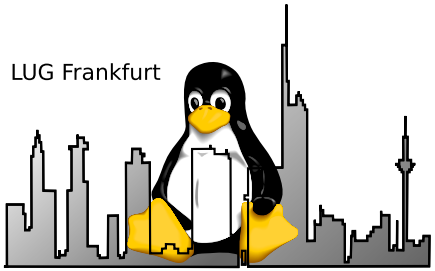


Menü

slit menu

Das slit menu öffnet sich beim rechts-click auf die slit.

Was die slit ist, wird später noch klar.



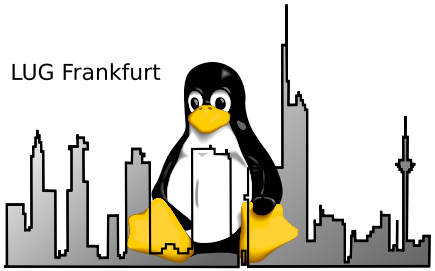
apps

Aufbau

In der Datei `apps` wird der Start und die initiale Einstellung von Programmen beschrieben.

Die Datei wird in sections aufgeteilt, anhand derer die Konfiguration gelesen wird.

Schauen wir uns hier ein paar alltägliche Beispiele an.

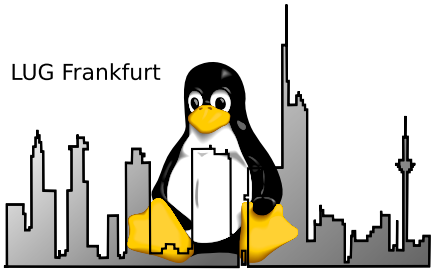


apps

Beispiel 1

Ein alltäglicher Eintrag für ein ganz normales Programm, das schon lange Zeit ziemlich entspannt auf dieser Festplatte liegt.

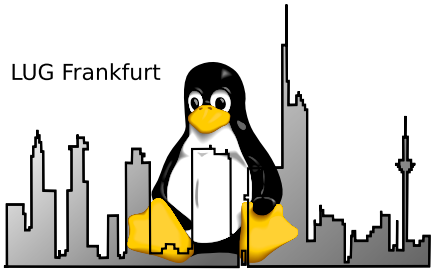
Wir wollen, dass das Programm immer in der gleichen, vorgegebenen Größe gestartet wird.



apps

Beispiel 1

```
1 [app] (name=xterm)
2       [Dimensions] {28% 55%}
3 [end]
```



apps

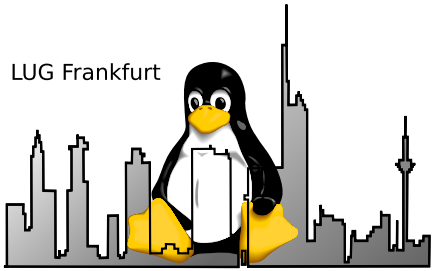
Beispiel 2

Da wir auch mit anderen Menschen kommunizieren müssen, haben wir das Programm `pidgin` installiert.

Dieses Programm soll:

- rechts unten positioniert
- in einer definierten Größe
- ohne Dekorationen erscheinen
- nicht in der Taskleiste auftauchen
- niemals vom System den Fokus kriegen

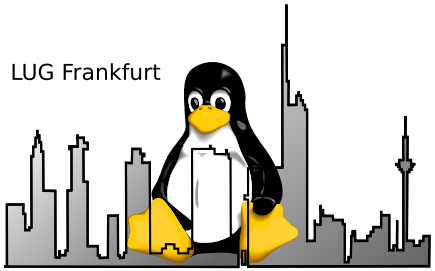
Die Fenster für die Chats sollen aber ganz normal gehandhabt werden.



apps

Beispiel 2

```
1 [app] (class=Pidgin) (role=buddy_list)
2       [Dimensions] {250 650}
3       [Position] (LOWERRIGHT) {50 70}
4       [Deco] {NONE}
5       [Hidden] {yes}
6       [Layer] {12}
7 [end]
```

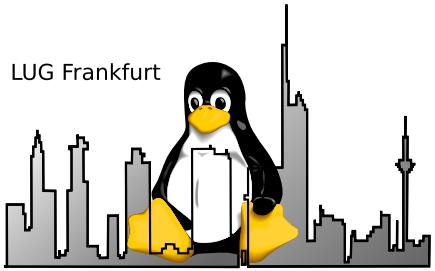


apps

Beispiel 3

Bei jedem Systemstart möchten wir, dass ein Browser (firefox), ein RSS-Feedreader (newsboat) und ein E-Mail Programm (claws-mail) geöffnet wird. Der Programmaufruf steht natürlich in der Datei startup.

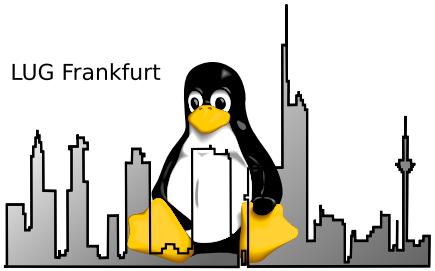
Diese Programme sollen aber:



apps

Beispiel 3

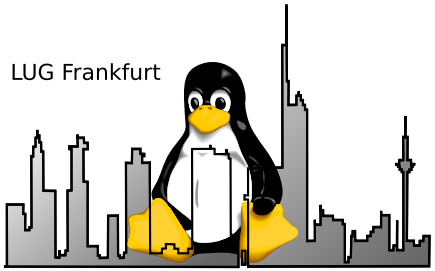
- auf dem zweiten Bildschirm
- auf dem workspace Nr. 2
- mittig im Bildschirm positioniert
- in einer definierten Größe
- zusammen gruppiert in tabs sein
- der Browser soll aber nur in der ersten Instanz so gruppiert werden



apps

Beispiel 3

```
1 [group]
2       [app] (class=firefox) {1}
3       [app] (name=newsboat)
4       [app] (class=claws-mail)
5       [Head] {2}
6       [Workspace] {2}
7       [Dimensions] {1280 1024}
8       [Position] (CENTER) {0 0}
9 [end]
```



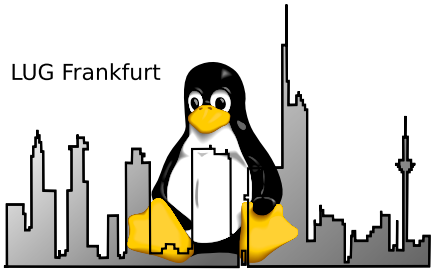
slitlist

Die slitlist ist ein Container für DockApps.

Ähnlich den Widgets können hier auch in einen expliziten Rahmen kleine Programme gestartet werden, die auf dem Desktop erscheinen.

Mancheiner kennt noch die bbtools oder die window maker dockapps.

Konfiguriert wird die slit in der Datei `/.fluxbox/slitlist`, auch die Reihenfolge der DockApps und gestartet werden sie in der Datei `startup`. Oft mit dem Parameter „-w“.

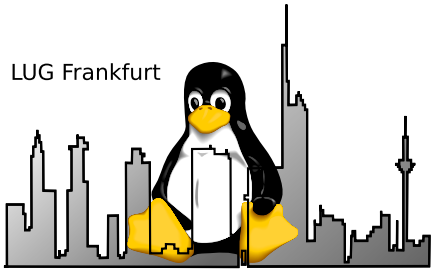


overlay

In der Datei `overlay` stehen style-Eigenschaften.

Diese Überschreiben immer die Eigenschaften, die im style definiert werden.

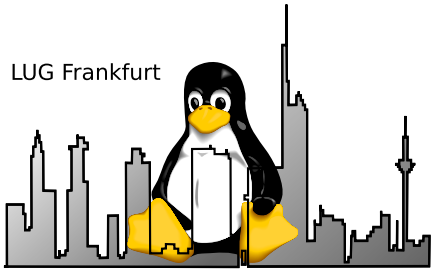
Die Formatierung ist die gleiche, wie in den style-files. dazu später mehr.



Tabbing

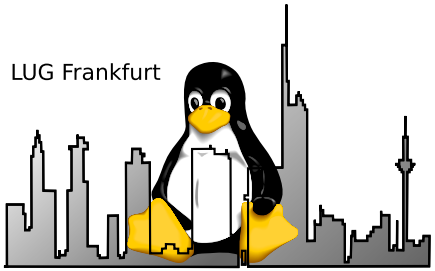
In Fluxbox können mehrere Programmfenster mit Tabs zusammengezogen werden.

Das ist sehr nützlich.



Style

Natürlich kann man die Fluxbox auch gestalterisch an seine ureigenen Bedürfnisse anpassen.

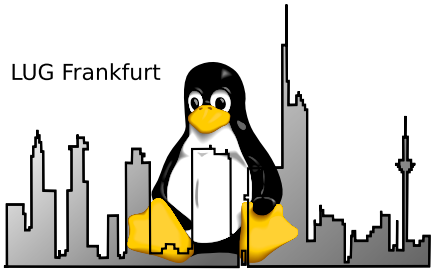


Styles

Dateien

Die Styles stehen in den Dateien /.fluxbox/styles/* und /usr/share/fluxbox/styles/*

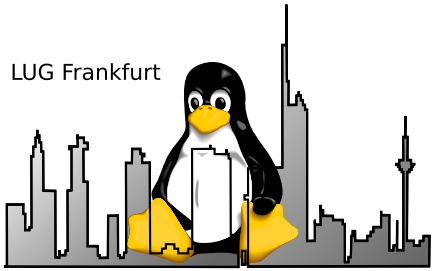
Es sei denn man sagt der Fluxbox was anderes.



Styles

Dateien

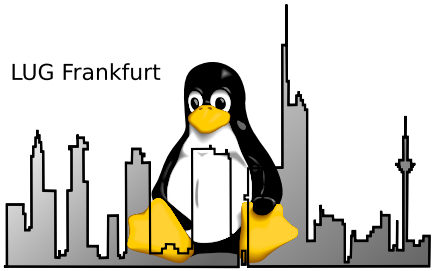
Bei einem Style ohne eigene Grafiken



Styles

Dateien

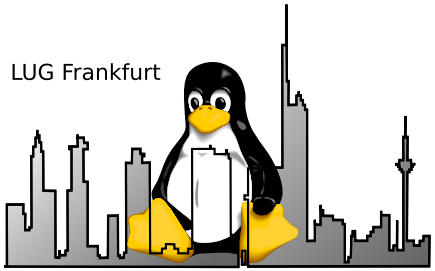
Ja, das geht!



Styles

Dateien

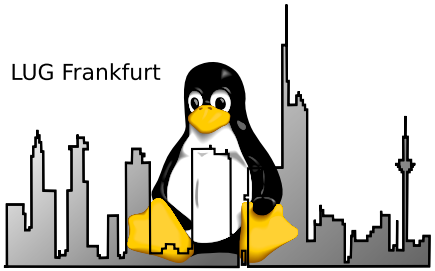
stehen einfach die entsprechenden Anweisungen in einer Textdatei, deren Name als Label für den Eintrag im Menü genutzt wird.



Styles

Dateien

Alternativ wird der Verzeichnisname als Label genutzt und in der Datei [name]/theme.cfg stehen die Anweisungen zum bunten Gestalten der Oberfläche.

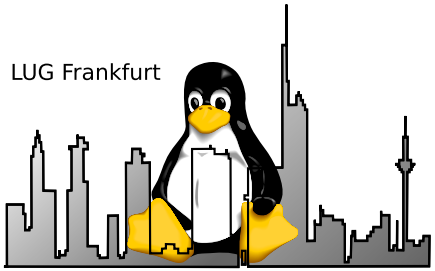


Styles

Grafiken

Es ist selbstredend auch möglich, einen Style mit eigenen bunten Grafiken zu verzieren. Diese sind dann im Verzeichnis [name]/pixmaps/* untergebracht.

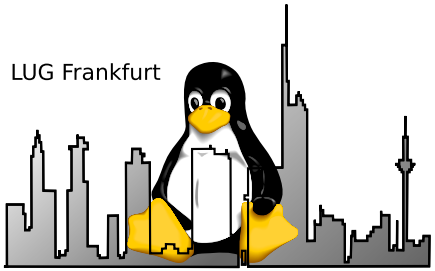
Das Format ist X PixMap (*.xpm) und es ist auch ein Textformat.



Styles

Grafiken

Welche und wieviel Grafiken man benutzt entscheidet jeder selbst und deklariert auch jeder selbst in der Datei `theme.cfg`

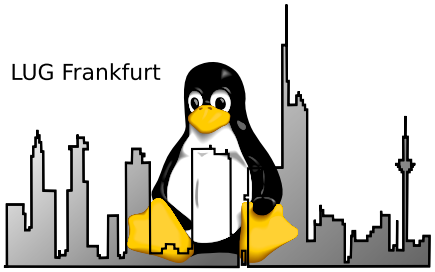


Styles

Wallpaper

Per default liegen die Wallpaper in dem Verzeichnis ./fluxbox/backgrounds/

Bildschirmhintergründe können selbsverständlich in den Style-Dateien definiert und im style-Verzeichnis hinterlegt werden.

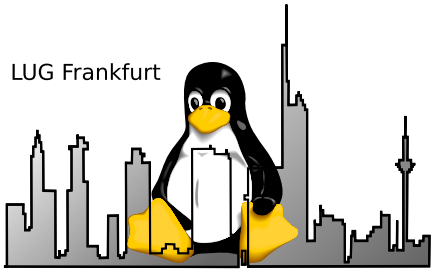


Styles

Wallpaper

Die Style-Datei kann aber auch über ein Programm, wie `feh` überschrieben werden. Das passiert dann durch einen entsprechenden Aufruf in der Datei overlay.

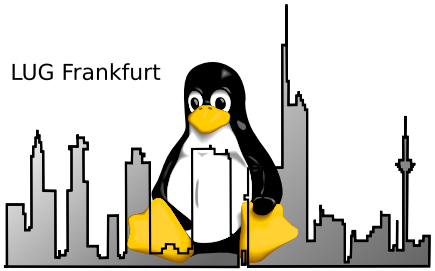
Es ist auch möglich, das Wallpaper per Skript in der Datei `startup` zu setzen



Styles

Wallpaper

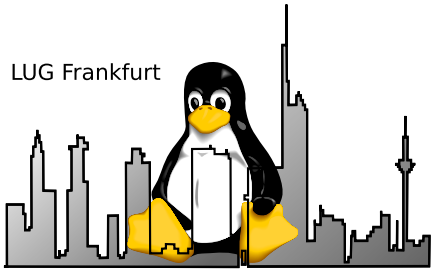
Das Gegenteil von gut gemacht ist gut gemeint.



Styles

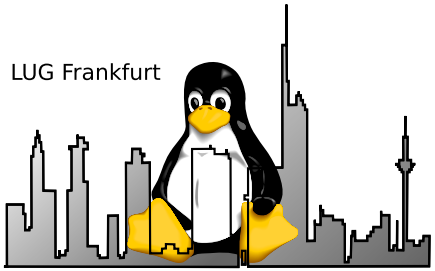
Wallpaper

Ein eigenes Wallpaper für jeden Desktop kann man auch machen. Der Command ChangeWorkspace, den man in der Datei `keys` definieren kann, kann das.



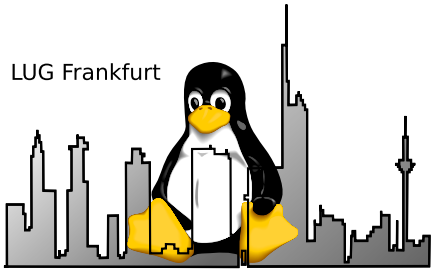
HiDPI Bildschirme

Sollten keine eigenen Grafiken verwandt werden, werden die generischen Grafiken entsprechend der Größenvorgabe in der Style-Datei skaliert.



HiDPI Bildschirme

Eigene Grafiken müssen vor der Benutzung also auf die entsprechend sinnvolle Größe skaliert werden.

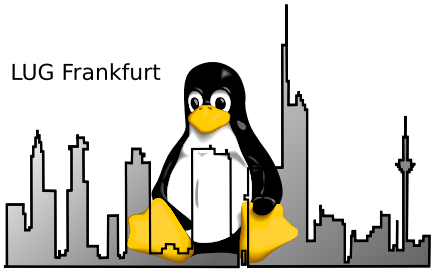


HiDPI Bildschirme

xrandr

Ein diskretes Upscaling durch xrandr kann auch hier und dort die richtigen Akzente setzen.

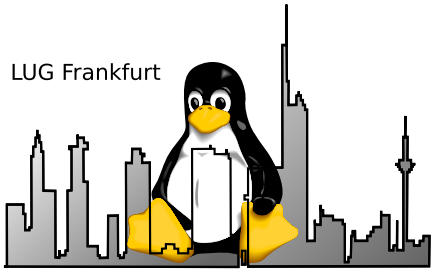
Aber nicht jedes Programm ist hier ein bedingungsloser Teamplayer.



Troubleshooting

Konsolenausgabe

Natürlich ist es bei Problemen immer eine gute Idee die entsprechenden Programme aus einer Konsole oder einem Terminalemulator zu starten und sich die Ausgaben anzusehen. Logisch!



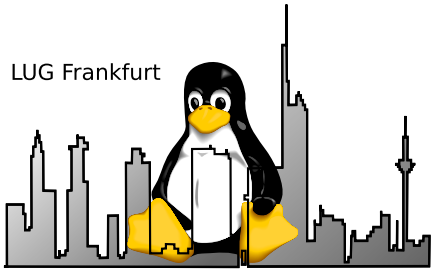
Troubleshooting

Logging

Zur qualifizierten Fehlersuche kann man in Fluxbox ein logging einschalten.

Zu diesem Zwecke gibt der interessierte Nutzer in der Datei startup dem Fluxbox-Aufruf den entsprechenden Parameter und den Namen der log-Datei mit:

```
1 exec fluxbox –log "~/.fluxbox/log"
```

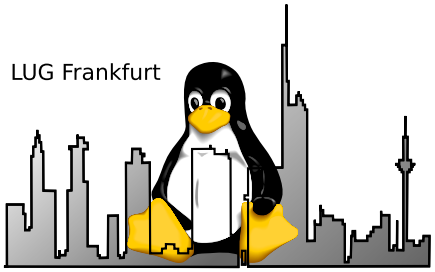


Troubleshooting

Signals

Fluxbox reagiert auf zwei Signale

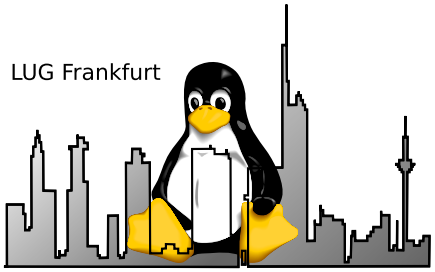
- SIGUSR1 Startet Fluxbox neu
- SIGUSR2 Forciert das neu laden der Konfiguration



man pages

- fluxbox
- startfluxbox
- fluxbox-keys
- fluxbox-menu
- fluxbox-apps
- fluxbox-style
- fluxbox-remote

- fbrun
- fbsetbg
- xprop
- regex
- wmctrl
- xev
- xmodmap
- feh
- xrandr



Links I

Hier sind links dargestellt

FraLUG: `https://www.lugfrankfurt.de`

Fluxbox: `http://www.fluxbox.org/`

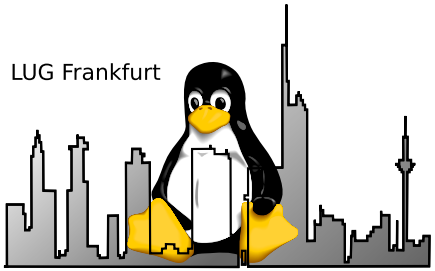
Fluxbox auf Github: `https://github.com/fluxbox/fluxbox`

Fluxbox auf ArchWiki: `https://wiki.archlinux.org/title/Fluxbox`

Fluxbox auf Sourceforge:
`https://sourceforge.net/projects/fluxbox/`

Fluxbox auf Wikipedia: `https://de.wikipedia.org/wiki/Fluxbox`

Fluxbox Themes: `https://www.box-look.org/browse?cat=139&ord=latest`



Links II

Hier sind links dargestellt

Windowmanager auf ArchWiki:

`https://wiki.archlinux.org/title/window_manager`

Desktop environment auf Wikipedia:

`https://de.wikipedia.org/wiki/Desktop-Umgebung`

Dockapps: `https://www.dockapps.net/`

X PixMap auf Wikipedia: `https://de.wikipedia.org/wiki/X_PixMap`

Danke fürs Mitnehmen und Tschüß