wmfs²

ce wiki ne présente qu'une partie des possibilités de WMFS² (mais suffit largement pour une première installation). j'ai le plaisir de contribuer au wiki officiel qui contient l'intégralité des options proposées par xorg62.
WMFS² est un gestionnaire de fenêtres en **tiling** écrit en **C** par xorg62, très léger et hautement configurable. c'est un logiciel libre distribué selon les termes de la licence BSD. il se pilote depuis un fichier de configuration en mode texte facilement compréhensible.
principales caractéristiques de wmfs²:

- lanceur console intégré (prompt) avec auto-complétion.
- zone de notification native.
- support Xinerama pour le multi-écran.
- respect des normes EWMH.
- possibilités d'éditer des règles pour les tags/clients.
- support complet de la souris.
- rechargement à la volée du wm après modification de la configuration (on-the-fly reloading).
- ajout dynamique de tags.
- barre de status avec support des barres de progression, des couleurs, des images (grâce à imlib2), et des scripts dynamiques.
- layout manuel: wmfs n'a pas de layout (entendez 'disposition') precis, l'agencement des clients se fait manuellement selon vos besoins et vos envies, grâce au clavier ou la souris.
- support pour les onglets (tabbing), déplacement et redimensionnement directionnels des clients.
- support des thèmes pour les barres de statut et les clients.
- support des ”@include” dans le fichier de configuration.

installation

compilation manuelle (dépôt git)

c’est la méthode d’installation recommandée. le code de wmfs change souvent et les paquets précompilés deviennent vite obsolètes.
pour installer wmfs², vous devrez obtenir quelques dépendances: **Xlib**, **Xinerama**, **Imlib2**, **pkg-config**, et bien sur **git** pour cloner le code source.

```
$ git clone git://github.com/xorg62/wmfs.git
$ cd wmfs
$ ./configure -h
$ ./configure
$ make
$ sudo make install
```

Pour mettre à jour en utilisant cette méthode:

```
$ git pull
$ ./configure
$ make clean
$ sudo make install
```

WMFS s'installe dans /usr/local/bin par défaut.
copiez la configuration par défaut dans votre ~/.config/wmfs si vous souhaitez la modifier, puis lancer wmfs depuis ~/.xinitrc via startx.

```
mkdir~/.config/wmfs
cp wmfsrc ~/.config/wmfs/
echo "exec wmfs" >> ~/.xinitrc
startx
```

sur Debian et dérivés

les utilisateurs Debian peuvent facilement installer depuis les sources:
installation des dépendances

```
sudo apt-get install git-core libimlib2-dev build-essential libxinerama-dev
libx11-dev
```

puis **installation depuis le dépôt git**

```
git clone git://github.com/xorg62/wmfs.git
cd wmfs/
./configure -h
./configure
make
sudo make install
```

lancement

cette section vous aidera à lancer WMFS avec startx ainsi que quelques programmes. notez que ce ne sont que quelques exemples.

lancement basique

dans votre $HOME/.xinitrc :

```
feh --bg-scale /path/to/my/background
# ou : nitrogen --restore
pcmanfm --desktop
```

```
# et d'autres commandes
# ck-launch-session : lance ConsoleKit, vous donne accès au montage des
volumes en tant qu'utilisateur
# depuis votre navigateur de fichier (pcmanfm, nautilus, thunar, ...), ou
halt/reboot votre machine sans être root.
# dbus-launch : utilisé par quelques applications.
exec ck-launch-session dbus-launch wmfs
# si vous n'avez pas besoin/envie, utilisez simplement :
# exec wmfs
```

lancement amélioré

ici, nous démarrons WMFS avant les applications ! car certaines applications ont besoin que wmfs soit lancé pour démarrer.
voici le contenu de votre $HOME/.xinitrc maintenant :

```
exec ck-launch-session dbus-launch wmfs &
wmpid=$! # ici nous sauvegardons le PID de wmfs
feh --bg-scale /path/to/your/wallpaper
pcmanfm --desktop
$HOME/.local/bin/status.sh
wait $wmpid # ici on attend la fin de wmfs WMFS, quand l'utilisateur lance
un : wmfs -c quit :(
```

configuration

edition du wmfsrc

WMFS se configure depuis son fichier principal en mode texte: **$XDG_CONFIG_HOME/wmfs/wmfsrc**. lors d'une nouvelle installation de wmfs, vous devez copier le fichier par défaut dans votre dossier de configuration (~/.config/wmfs/) depuis le fichier source (généralement /usr/local/etc/xdg/wmfs/wmfsrc). vous pouvez aussi trouver la dernière version de ce fichier sur le dépôt git.
pour modifier votre configuration, il suffit d'éditer ce fichier puis de relancer wmfs. cette section reprend et détaille le fichier wmfsrc par sections. pour une liste complète des fonctions uicb, voir plus bas.

@include

wmfsrc permet l'inclusion de fichier afin de scinder le fichier principal pour plus de lisibilité.

```
# Possible file inclusion:
```

```
@include ~/.config/wmfs/wmfs_themes
@include ~/.config/wmfs/wmfs_rules
```

[themes]

wmfs² intègre la possibilité d'utiliser un ou plusieurs thèmes. cette section configure le(s) thème(s).

```
# Multi theme section
[themes]

  [theme]
     # No name mean default
     # name = "default"

     font = "fixed"
     # Bars
     bars_width = 14
     bars_fg = "#AABBAA"
     bars_bg = "#223322"

     # Element tags
     tags_normal_fg = "#AABBAA"
     tags_normal_bg = "#223322"
     # tags_normal_statusline = ""
     tags_sel_fg    = "#223322"
     tags_sel_bg    = "#AABBAA"
     # tags_sel_statusline = ""
     tags_occupied_fg = "#AABBAA"
     tags_occupied_bg = "#445544"
     tags_occupied_statusline = "\R[0;0;3;3;#AABBAA]"
     tags_urgent_fg    = "#223322"
     tags_urgent_bg    = "#CC5544"
     # tags_urgent_statusline = ""
     tags_border_color = "#112211"
     tags_border_width = 1

     # Frame / Client
     client_normal_fg = "#AABBAA"
     client_normal_bg = "#223322"
     client_normal_statusline = "\s[3;9;#121212;x]
\s[2;8;#ff0000;x](1;client_close)"
     client_sel_fg = "#223322"
     client_sel_bg = "#AABBAA"
     client_sel_statusline = "\s[3;9;#121212;x]
\s[2;8;#ff0000;x](1;client_close)"
     #client_free_statusline = ""
     frame_bg = "#555555"
     client_titlebar_width = 12
```

```
    client_border_width = 1
  [/theme]

[/themes]
```

- **name** nom du thème: sera utiliser dans les sections suivantes.
- **font** police du theme, au format XLFD (X Logical Font Description). ex "-*-terminus-medium-*-*-*-12-*-*-*-*-*-*-*"
- **bars**
  - **bars_width** hauteur de la barre d'info en pixels.
  - **bars_fg** couleur du texte de la barre d'info.
  - **bars_bg** couleur du fond de la barre d'info.
- **tags**
  - **tags_normal_fg** couleur du texte des tags par defaut.
  - **tags_normal_bg** couleur du fond des tags par defaut.
  - **tags_normal_statusline** sequence du tag par defaut.
  - **tags_sel_fg** couleur du texte du tag sélectionné.
  - **tags_sel_bg** couleur du fond du tag sélectionné.
  - **tags_sel_statusline** sequence du tag sélectionné.
  - **tags_occupied_fg** couleur du texte des tags occupés.
  - **tags_occupied_bg** couleur du fond des tags occupés.
  - **tags_occupied_statusline** sequence des tags occupés.
  - **tags_urgent_fg** couleur du texte des tags 'urgents'.
  - **tags_urgent_bg** couleur du fond des tags 'urgents'.
  - **tags_urgent_statusline** sequence des tags 'urgents'.
  - **tags_border_color** couleur de la bordure des tags.
  - **tags_border_width** epaisseur de la bordure des tags en pixels.
- **clients**
  - **client_normal_fg** couleur du texte de la barre de titre des clients.
  - **client_normal_bg** couleur du fond de la barre de titre des clients.
  - **client_normal_statusline** sequence de la barre de titre des clients.
  - **client_sel_fg** couleur du texte de la barre de titre du client sélectionné.
  - **client_sel_bg** couleur du fond de la barre de titre du client sélectionné.
  - **client_sel_statusline** sequence de la barre de titre du client sélectionné.
  - **client_free_statusline** sequence de la barre de titre du client libre.
  - **frame_bg** couleur de la bordure des clients.
  - **client_titlebar_width** hauteur de la barre de titre en pixels.
  - **client_border_width** épaisseur des bordures des clients.

[bars]

wmfs² permet de créer plusieurs barre d'infos contenant chacune des informations spécifiques et ayant leur propre thème.

```
[bars]

  # Position:
  #
  # 0  Top
```

```
  # 1  Bottom
  # 2  Hide

  # Element type:
  #
  # t  Tags
  # s  Statustext (will take available space)
  # y  Systray (can be set only ONE time among all element)
  # l  Launcher (will be expended at launcher use)

  [bar]
     position = 0
     screen = 0
     elements = "tlsy"   # element order in bar
     theme = "default"
  [/bar]

# [bar]
#    position = 0
#    screen = 1
#    elements = "ts"
#    theme = "default"
# [/bar]

[/bars]
```

- **positon** placement de la barre sur l'écran. 0:haut, 1:bas, 2:masqué.
- **screen** écran où s'affiche la barre(commence à 0), metrre -1 pour afficher sur tous les écrans.
- **elements** ordre d'affichage des composants de la barre. t:*tags*, s:*statustext*, y:*systray*(ne faire apparaitre qu'une fois), l:*launcher*.
- **theme** thème utilisé pour la barre. doit correspondre à un thème existant.

[tags]

configuration des tags: nombre, nom, écran d'affichage, actions de la souris sur les boutons des tags.

```
[tags]

  # enable/disable tag wrapping navigation
  circular = false

  # Use no screen option or screen = -1 to set tag on each screen
  [tag]
      screen = -1
      name = "1"
      # statusline = ""
  [/tag]

  [tag] name = "2" [/tag]
```

```
  [tag] name = "3" [/tag]
  [tag] name = "4" [/tag]
  [tag] name = "5" [/tag]
  [tag] name = "6" [/tag]
  [tag] name = "7" [/tag]

  # Mousebinds associated to Tags element button
  [mouse] button = "1" func = "tag_click" [/mouse]
  [mouse] button = "4" func = "tag_next"  [/mouse]
  [mouse] button = "5" func = "tag_prev"  [/mouse]

[/tags]
```

- **circular** (dés)active la navigation circulaire entre les tags.
- **screen** écran d'affichage des tags: commenter ou mettre -1 pour afficher les tags sur tous les écrans.
- **name** nom affiché du tag.
- **statusline** affiche une statusline spécifique par tag (peut afficher toutes les séquences).
- **mousebinds** actions de la souris sur les boutons des tags: ici clic-gauche(bouton1) affiche le tag, scrollup(b4) affiche le tag suivant et scrolldown(b5) affiche le tag précédent.

[client]

configuration de l'apparence des clients et de leur comportement.

```
[client]

  # padding between clients (default:0)
  padding = 75

  # Give focus to new created client (default = false)
  autofocus = false

  theme = "default"
  key_modifier = "Super"

  # Focus type:
  #  enter : focus follow mouse (default)
  #  click : click to focus
  #  everything-else : disable mouse focus support
  focus = enter

  [mouse] button = "1" func = "client_focus_click"    [/mouse]
  [mouse] button = "1" func = "mouse_swap"            [/mouse]
  [mouse] button = "2" func = "mouse_tab"             [/mouse]
  [mouse] button = "3" func = "mouse_resize"          [/mouse]
  [mouse] button = "4" func = "client_focus_next_tab" [/mouse]
  [mouse] button = "5" func = "client_focus_prev_tab" [/mouse]
```

```
[/client]
```

- **padding** distance en pixels entre les clients.
- **autofocus** donne le focus au nouveau client (par défaut: false).
- **theme** thème appliqué aux clients par défaut.
- **key_modifier** touche de modification utilisée pour agir sur les clients. (Alt, Super, Control, Shift…)
- **focus** détermine le comportement de la souris: enter, le focus suit la souris – click, clic pour donner le focus – everything-else, désactive la souris pour le focus(le focus se donne grâce aux raccoucis clavier)
- **mousebinds** actions de la souris sur les clients. voir la liste des fonctions.

[rules]

les règles scpécifiques appliquées aux clients. pour identifier une application, utiliser xprop.

```
[rules]

  [rule]
      instance = "chromium"
      # class = ""
      # role   = ""
      # name   = ""
      # theme  = "default"

      tag    = 1  # 2nd tag
      screen = 0

      free       = false
      tab        = false
      ignore_tag = false
  [/rule]

[/rules]
```

- **instance** première partie de WM_CLASS.
- **class** seconde partie de WM_CLASS.
- **role** WM_WINDOW_ROLE
- **name** NET_WM_NAME
- **theme** thème appliqué au client.
- **tag** tag d'affichage du client (commence à 0).
- **screen** écran d'affichage du client.
- **free** le client est en mode libre (true/false).
- **tab** ouvre le client dans un onglet (true/false).
- **ignore_tag** tag le client avec tous les tags (aka "toujours visible").

[launchers]

wmfs² dispose d'un prompt qui supporte l'autocompletion.

```
[launchers]

  # command can be an uicb function or an uicb function + extension (see
example)
  [launcher]
     name = "exec"
     prompt = "Run:"

     # Example of uicb + ext:
     #   command = "spawn xterm -e"
     command = "spawn"

     width = 150
  [/launcher]

[/launchers]
```

- **name** nom du lanceur. sera utilisé dans la section [keys].
- **prompt** texte affiché au début du lanceur.
- **command** type de commande utilisé par le lanceur. cette commande peut-être de type uicb (spawn) ou uicb + extension afin de créer des lanceurs personnalisés.

[keys]

les racourcis clavier sont définis dans cette section.

```
[keys]

  [key] mod = {"Super"} key = "Return" func = "spawn" cmd = "urxvt || xterm"
[/key]

  [key] mod = {"Control", "Alt"} key = "q" func = "quit" [/key]
  [key] mod = {"Control", "Alt"} key = "r" func = "reload" [/key]

  # Tag manipulation
  [key] mod = {"Super"} key = "F1" func = "tag_set" cmd = "0" [/key]
  [key] mod = {"Super"} key = "F2" func = "tag_set" cmd = "1" [/key]
  [key] mod = {"Super"} key = "F3" func = "tag_set" cmd = "2" [/key]
  [key] mod = {"Super"} key = "F4" func = "tag_set" cmd = "3" [/key]
  [key] mod = {"Super"} key = "F5" func = "tag_set" cmd = "4" [/key]
  [key] mod = {"Super"} key = "F6" func = "tag_set" cmd = "5" [/key]
  [key] mod = {"Super"} key = "F7" func = "tag_set" cmd = "6" [/key]
  [key] mod = {"Super"} key = "F8" func = "tag_set" cmd = "7" [/key]

  [key] mod = {"Super", "Shift"} key = "F1" func = "tag_client" cmd = "0"
[/key]
  [key] mod = {"Super", "Shift"} key = "F2" func = "tag_client" cmd = "1"
```

```
[/key]
  [key] mod = {"Super", "Shift"} key = "F3" func = "tag_client" cmd = "2"
[/key]
  [key] mod = {"Super", "Shift"} key = "F4" func = "tag_client" cmd = "3"
[/key]
  [key] mod = {"Super", "Shift"} key = "F5" func = "tag_client" cmd = "4"
[/key]
  [key] mod = {"Super", "Shift"} key = "F6" func = "tag_client" cmd = "5"
[/key]
  [key] mod = {"Super", "Shift"} key = "F7" func = "tag_client" cmd = "6"
[/key]
  [key] mod = {"Super", "Shift"} key = "F8" func = "tag_client" cmd = "7"
[/key]

  [key] mod = {"Super"} key = "minus" func = "tag_del" [/key]
  [key] mod = {"Super", "Shift"} key = "minus" func = "tag_new" [/key]

  # tag function: cmd = nameofthetag
  #[key] mod = {"Super"} key = "z" func = "tag" cmd = "2" [/key]

  [key] mod = {"Control"} key = "Right" func = "tag_next" [/key]
  [key] mod = {"Control"} key = "Left"  func = "tag_prev" [/key]

  [key] mod = {"Control"} key = "Up"  func = "screen_next" [/key]
  [key] mod = {"Control"} key = "Down" func = "screen_prev" [/key]

  [key] mod = {"Super"} key = "q" func = "client_close" [/key]

  # Focus next / prev client and next / prev tabbed client
  [key] mod = { "Alt" }            key = "Tab" func = "client_focus_next"
[/key]
  [key] mod = { "Alt", "Shift" }   key = "Tab" func = "client_focus_prev"
[/key]
  [key] mod = { "Super" }          key = "Tab" func = "client_focus_next_tab"
[/key]
  [key] mod = { "Super", "Shift" } key = "Tab" func = "client_focus_prev_tab"
[/key]

  # Focus next client with direction
  [key] mod = {"Alt"} key = "h" func = "client_focus_left"   [/key]
  [key] mod = {"Alt"} key = "l" func = "client_focus_right"  [/key]
  [key] mod = {"Alt"} key = "k" func = "client_focus_top"    [/key]
  [key] mod = {"Alt"} key = "j" func = "client_focus_bottom" [/key]

  # swap next client with direction:
  [key] mod = {"Control", "Shift"} key = "h" func = "client_swap_left"
[/key]
  [key] mod = {"Control", "Shift"} key = "l" func = "client_swap_right"
[/key]
  [key] mod = {"Control", "Shift"} key = "k" func = "client_swap_top"
[/key]
```

```
  [key] mod = {"Control", "Shift"} key = "j" func = "client_swap_bottom"
[/key]

  # Resize selected tiled client with direction
  [key] mod = {"Super"} key = "h" func = "client_resize_left"   cmd = "20"
[/key]
  [key] mod = {"Super"} key = "l" func = "client_resize_left"   cmd = "-20"
[/key]
  [key] mod = {"Super"} key = "k" func = "client_resize_top"    cmd = "20"
[/key]
  [key] mod = {"Super"} key = "j" func = "client_resize_top"    cmd = "-20"
[/key]
  [key] mod = {"Super", "Control"} key = "h" func = "client_resize_right"
cmd = "-20" [/key]
  [key] mod = {"Super", "Control"} key = "l" func = "client_resize_right"
cmd = "20" [/key]
  [key] mod = {"Super", "Control"} key = "k" func = "client_resize_bottom"
cmd = "-20" [/key]
  [key] mod = {"Super", "Control"} key = "j" func = "client_resize_bottom"
cmd = "20" [/key]

  # Tabbing command
  [key] mod = {"Alt", "Shift"} key = "h" func = "client_tab_left"   [/key]
  [key] mod = {"Alt", "Shift"} key = "l" func = "client_tab_right"  [/key]
  [key] mod = {"Alt", "Shift"} key = "k" func = "client_tab_top"    [/key]
  [key] mod = {"Alt", "Shift"} key = "j" func = "client_tab_bottom" [/key]
  [key] mod = {"Alt", "Shift"} key = "u" func = "client_untab" [/key]

  # Layout manipulation
  [key] mod = {"Super"}          key = "m" func = "layout_vmirror" [/key]
  [key] mod = {"Super", "Shift"} key = "m" func = "layout_hmirror" [/key]
  [key] mod = {"Super"}          key = "r" func = "layout_rotate_right"
[/key]
  [key] mod = {"Super", "Shift"} key = "r" func = "layout_rotate_left" [/key]

  [key] mod = {"Control", "Super", "Alt"} key = "h" func =
"layout_integrate_left" [/key]
  [key] mod = {"Control", "Super", "Alt"} key = "j" func =
"layout_integrate_bottom" [/key]
  [key] mod = {"Control", "Super", "Alt"} key = "k" func =
"layout_integrate_top" [/key]
  [key] mod = {"Control", "Super", "Alt"} key = "l" func =
"layout_integrate_right" [/key]

  # Layout set historic travelling function (TESTING)
  [key] mod = {"Super"} key = "o" func = "layout_prev_set" [/key]
  [key] mod = {"Super", "Shift"} key = "o" func = "layout_next_set" [/key]

  # Toggle client free/tile
  [key] mod = {"Super"} key = "f" func = "client_toggle_free" [/key]
```

```
  # Toggle client ignore_tag
  [key] mod = {"Super","Shift"} key = "f" func = "client_toggle_ignore_tag"
[/key]

  # Launcher
  [key] mod = {"Super"} key = "p" func = "launcher" cmd = "exec" [/key]

[/keys]
```

la syntaxe de la section [keys]: chaque raccourcis est encadré par [key] ... [/key]

- **mod** touche(s) de modification. ex:{“Control”, “Alt”}
- **key** touche du clavier à presser.
- **func** fonction uicb à lancer.
- **cmd** si *func* = *spawn*, détermine la commande externe à lancer. ex:func = “spawn” cmd = “cream-browser”

statusbar

la barre d’info de wmfs affiche les tags et la zone de notification (systray), mais elle peut aussi afficher bien d’autres choses grâce à la commande **wmfs -c status**.
c’est ainsi que l’on peut afficher le résultat d’un **script bash**, d’un **conky** ou une simple **commande**.
de plus, la barre de statut supporte les *barres de progression*, de *position*, les *graphiques*, les *couleurs*, les *rectangles*, les *images* et les *thèmes*. et tout ceci par écran et par barre.
la façon la plus simple d’afficher des informations, est par le biais d’un script: **status.sh** à lancer au démarrage de votre session.
un exemple basic du status.sh:

```
#!/bin/sh
#WMFS status.sh example file
TIMING=10
statustext()
{
     wmfs -c status "default `date`"
}
while true;
do
statustext
    sleep $TIMING
done
```

- utilisation simple:wmfs -c status "<barname> TEXTE visible dans la barre 'barname'"
- pour les couleurs, wmfs supporte le rgb:wmfs -c status "<barname> ^s[<position>;<couleur>;<texte>]"
- voici la syntaxe pour dessiner des rectangles:wmfs -c status "<barname> ^R[<position>;<dimensions>;<couleur>]"
- comment afficher des images (supporté par imlib2):wmfs -c status "<barname> ^i[<position>;<dimensions>;<imagepath>]"

- comment afficher une barre de progression:wmfs -c status "<barname> ^p[<position>;<dimensions>;<border>;<value>;<valuemax>;<bgcolor>;<fgcolor>]"
- comment afficher une barre de position:wmfs -c status "<barname> ^P[<position>;<dimensions>;<curser>;<value>;<valuemax>;<bgcolor>;<fgcolor>]"
- afficher un graphique:wmfs -c status "<barname> ^g[<position>;<dimensions>;<value>;<valuemax>;<bgcolor>;<fgcolor>;<name>]"

mousebinds

les séquences sont des zones cliquables grâce au code **(<key>;<uicb-function>)** ou **(<key>;<spawn>;<command>)** exemple:

```
wmfs -c status "<barname> ^R[<position>;<dimensions>;<color>](1;spawn;urxvt -e ranger)"
```

affichera un rectangle qui lancera ranger dans urxvtc lors d'un clic gauche (bouton1).

popup status

vous pouvez afficher des fenêtres surgissantes depuis la statusbar lors d'un clic avec le code **(<key>;status_surface;<position>,<dimension>,<color> <datas>)** note: l'utilisation de l'argument "position" est optionnel. si il n'est pas renseigné, le popup se place sous le pointeur.

syntaxe acceptée

- **position**: "left/right" (position relative) ou "x;y" (position absolue)
- **dimension**: "ww;hh" pour largeur;hauteur du rectangle ou de l'image, pour afficher l'image à son format d'origine, mettre "0;0".
- **couleur**: "#rrggbb"
- **imagepath**: chemin de l'image à afficher
- **border**: épaisseur de la bordure de la barre de progression
- **curser**: taille du curseur de position dans les barres de position
- **value**: variable assignée à la barre de progression
- **valuemax**: valeur maximale de la variable assignée à la barre de progression

note:

- "^" peut être remplacé par "\" dans les séquences.
- pour afficher un "]", vous devrez placer un "\" devant.

- si vous utilisez la sequence ^p[...], et si w < h , la barre de progression sera verticale.

intégration dans le wmfsrc

le format classique **wmfs -c status "<barname><datas>"** implique que la barre d'affichage soit spécifiée dans votre wmfsrc dans la section [bars] sous la forme:

```
[bar]
    name = "barname"
    position = 0
    screen = 0
    elements = "tlsy"   # element order in bar
    theme = "default"
[/bar]
```

à adapter à votre configuration.

exemple simple


```
#!/bin/bash
wmfs -c status "testbar ^s[left;#ff0000; red-alignleft-text]\
^s[130;10;#00ff00;greentext]
^i[190;1;0;0;/home/arp/.config/wmfs/icons/sound.png]\
^R[210;2;40;10;#0000ff] ^s[217;11;#000000;text]"
```

notez que la **[bar] name = "testbar" [/bar]** doit exister dans votre wmfsrc.

scripts complets

- script pour une barre de progression :

wmfs-status-progressbar.sh#!/bin/bash

```
memu(){
   memu="$(free -m | sed -n 's|^-.*:[ \t]*\([0-9]*\) .*|\1|gp')"
   echo "$memu"
}
memt(){
   memt="$(free -m | sed -n 's|^M.*:[ \t]*\([0-9]*\) .*|\1|gp')"
   echo "$memt"
}
cpu(){
```

```
    cpu="$(eval $(awk '/^cpu /{print "previdle="  "; prevtotal=" #!/bin/bash   memu(){ memu="
$(free -m | sed -n 's|^-.*:[ \t]*\([0-9]*\) .*|\1|gp')" echo "$memu" } memt(){ memt="$(free -m | sed
-n 's|^M.*:[ \t]*\([0-9]*\) .*|\1|gp')" echo "$memt" } cpu(){ cpu="$(eval $(awk '/^cpu /{print
"previdle=" $5 "; prevtotal=" $2+$3+$4+$5 }' /proc/stat); sleep 0.4; eval $(awk '/^cpu /{print
"idle=" $5 "; total=" $2+$3+$4+$5 }' /proc/stat); intervaltotal=$((total-${prevtotal:-0})); echo "$((
100*( (intervaltotal) - ($idle-${previdle:-0}) ) / (intervaltotal) ))")" echo "$cpu" }   TIMING=1
statustext() { wmfs -c status "bartest ^p[200;0;300;6;1;$(cpu)
;100;#AABBAA;#445544]^p[200;7;300;6;1;$(memu);$(memt);#AABBAA;#445544]" }   while true;
do statustext sleep $TIMING done++ }' /proc/stat); sleep 0.4;
            eval $(awk '/^cpu /{print "idle="  "; total=" #!/bin/bash   memu(){ memu="$(free -m | sed
-n 's|^-.*:[ \t]*\([0-9]*\) .*|\1|gp')" echo "$memu" } memt(){ memt="$(free -m | sed -n 's|^M.*:[
\t]*\([0-9]*\) .*|\1|gp')" echo "$memt" } cpu(){ cpu="$(eval $(awk '/^cpu /{print "previdle=" $5 ";
prevtotal=" $2+$3+$4+$5 }' /proc/stat); sleep 0.4; eval $(awk '/^cpu /{print "idle=" $5 "; total="
$2+$3+$4+$5 }' /proc/stat); intervaltotal=$((total-${prevtotal:-0})); echo "$((100*( (intervaltotal) -
($idle-${previdle:-0}) ) / (intervaltotal) ))")" echo "$cpu" }   TIMING=1   statustext() { wmfs -c status
"bartest ^p[200;0;300;6;1;$(cpu);100;#AABBAA;#445544]^p[200;7;300;6;1;$(memu);$(memt)
;#AABBAA;#445544]" }   while true; do statustext sleep $TIMING done++ }' /proc/stat);
            intervaltotal=$((total-${prevtotal:-0}));
            echo "$((100*( (intervaltotal) - ($idle-${previdle:-0}) ) / (intervaltotal) ))")"
    echo "$cpu"
}

TIMING=1

statustext()
{
     wmfs -c status "bartest ^p[200;0;300;6;1;$(cpu);100;#AABBAA;#445544]^p[200;7;300;6;1;
$(memu);$(memt);#AABBAA;#445544]"
}

while true;
do
    statustext
    sleep $TIMING
done
```

- barre de progression verticale :

status-vprogressbar.sh

```
#!/bin/bash

memu(){
    memu="$(free -m | sed -n 's|^-.*:[ \t]*\([0-9]*\) .*|\1|gp')"
    echo "$memu"
}
memt(){
    memt="$(free -m | sed -n 's|^M.*:[ \t]*\([0-9]*\) .*|\1|gp')"
    echo "$memt"
}
cpu(){
    cpu="$(eval $(awk '/^cpu /{print "previdle="  "; prevtotal=" #!/bin/bash   memu(){ memu="
$(free -m | sed -n 's|^-.*:[ \t]*\([0-9]*\) .*|\1|gp')" echo "$memu" } memt(){ memt="$(free -m | sed
-n 's|^M.*:[ \t]*\([0-9]*\) .*|\1|gp')" echo "$memt" } cpu(){ cpu="$(eval $(awk '/^cpu /{print
```

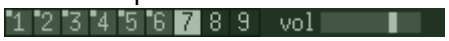
```
"previdle=" $5 "; prevtotal=" $2+$3+$4+$5 }' /proc/stat); sleep 0.4; eval $(awk '/^cpu /{print "idle=" $5 "; total=" $2+$3+$4+$5 }' /proc/stat); intervaltotal=$((total-${prevtotal:-0})); echo "$(( 100*( (intervaltotal) - ($idle-${previdle:-0}) ) / (intervaltotal) ))")" echo "$cpu" }  TIMING=1 statustext() { wmfs -c status "bartest ^s[80;10;#AABBAA;cpu] ^p[100;0;8;12;0;$(cpu) ;100;#445544;#AABBAA](1;spawn;urxvtc -e htop)\  ^s[120;10;#AABBAA;mem] ^p[140;0;8;12;0; $(memu);$(memt);#445544;#AABBAA]" }  while true; do statustext sleep $TIMING done++ }' /proc/stat); sleep 0.4;
        eval $(awk '/^cpu /{print "idle="  "; total=" #!/bin/bash  memu(){ memu="$(free -m | sed -n 's|^-.*:[ \t]*\([0-9]*\) .*|\1|gp')" echo "$memu" } memt(){ memt="$(free -m | sed -n 's|^M.*:[ \t]*\([0-9]*\) .*|\1|gp')" echo "$memt" } cpu(){ cpu="$(eval $(awk '/^cpu /{print "previdle=" $5 "; prevtotal=" $2+$3+$4+$5 }' /proc/stat); sleep 0.4; eval $(awk '/^cpu /{print "idle=" $5 "; total=" $2+$3+$4+$5 }' /proc/stat); intervaltotal=$((total-${prevtotal:-0})); echo "$((100*( (intervaltotal) - ($idle-${previdle:-0}) ) / (intervaltotal) ))")" echo "$cpu" }  TIMING=1  statustext() { wmfs -c status "bartest ^s[80;10;#AABBAA;cpu] ^p[100;0;8;12;0;$(cpu);100;#445544;#AABBAA](1;spawn;urxvtc -e htop)\  ^s[120;10;#AABBAA;mem] ^p[140;0;8;12;0;$(memu);$(memt);#445544;#AABBAA]" } while true; do statustext sleep $TIMING done++ }' /proc/stat);
        intervaltotal=$((total-${prevtotal:-0}));
        echo "$((100*( (intervaltotal) - ($idle-${previdle:-0}) ) / (intervaltotal) ))")"
  echo "$cpu"
}

TIMING=1

statustext()
{
   wmfs -c status "bartest ^s[80;10;#AABBAA;cpu] ^p[100;0;8;12;0;$(cpu)
;100;#445544;#AABBAA](1;spawn;urxvtc -e htop)\
   ^s[120;10;#AABBAA;mem] ^p[140;0;8;12;0;$(memu);$(memt);#445544;#AABBAA]"
}

while true;
do
  statustext
  sleep $TIMING
done
```

- barre de position :



status-positionbar.sh

```
#!/bin/bash

volpcm(){
  volpcm="$(amixer get PCM | tail -1 | sed 's/.*\[\([0-9]*%\)\].*/\1/')"
    if [ "$volpcm" == "100%" ]; then
      pcm="100"
    else
      pcm="$(echo $volpcm | cut -c1-2)"
    fi
  echo "^s[30;11;#AABBAA;vol ]^P[50;2;50;10;4;$pcm;100;#445544;#AABBAA]"
}

TIMING=1
```

```
statustext()
{
    wmfs -c status "bartest $(volpcm)"
}

while true;
do
   statustext
   sleep $TIMING
done
```

- script pour afficher un cpugraph :

status-graph.sh

```
#!/bin/bash

memu(){
   memu="$(free -m | sed -n 's|^-.*:[ \t]*\([0-9]*\) .*|\1|gp')"
   echo "$memu"
}
memt(){
   memt="$(free -m | sed -n 's|^M.*:[ \t]*\([0-9]*\) .*|\1|gp')"
   echo "$memt"
}
cpu(){
   cpu="$(eval $(awk '/^cpu /{print "previdle="  "; prevtotal=" #!/bin/bash   memu(){ memu="
$(free -m | sed -n 's|^-.*:[ \t]*\([0-9]*\) .*|\1|gp')" echo "$memu" } memt(){ memt="$(free -m | sed
-n 's|^M.*:[ \t]*\([0-9]*\) .*|\1|gp')" echo "$memt" } cpu(){ cpu="$(eval $(awk '/^cpu /{print
"previdle=" $5 "; prevtotal=" $2+$3+$4+$5 }' /proc/stat); sleep 0.4; eval $(awk '/^cpu /{print
"idle=" $5 "; total=" $2+$3+$4+$5 }' /proc/stat); intervaltotal=$((total-${prevtotal:-0})); echo "$((
100*( (intervaltotal) - ($idle-${previdle:-0}) ) / (intervaltotal) ))")" echo "$cpu" }   TIMING=1
statustext() { wmfs -c status "bartest ^g[200;0;300;12;$(cpu)
;100;#445544;#AABBAA;cpugraph](1;spawn;urxvtc -e htop)" }   while true; do statustext sleep
$TIMING done++ }' /proc/stat); sleep 0.4;
            eval $(awk '/^cpu /{print "idle="  "; total=" #!/bin/bash   memu(){ memu="$(free -m | sed
-n 's|^-.*:[ \t]*\([0-9]*\) .*|\1|gp')" echo "$memu" } memt(){ memt="$(free -m | sed -n 's|^M.*:[
\t]*\([0-9]*\) .*|\1|gp')" echo "$memt" } cpu(){ cpu="$(eval $(awk '/^cpu /{print "previdle=" $5 ";
prevtotal=" $2+$3+$4+$5 }' /proc/stat); sleep 0.4; eval $(awk '/^cpu /{print "idle=" $5 "; total="
$2+$3+$4+$5 }' /proc/stat); intervaltotal=$((total-${prevtotal:-0})); echo "$((100*( (intervaltotal) -
($idle-${previdle:-0}) ) / (intervaltotal) ))")" echo "$cpu" }   TIMING=1   statustext() { wmfs -c status
"bartest ^g[200;0;300;12;$(cpu);100;#445544;#AABBAA;cpugraph](1;spawn;urxvtc -e htop)" }
while true; do statustext sleep $TIMING done++ }' /proc/stat);
            intervaltotal=$((total-${prevtotal:-0}));
            echo "$((100*( (intervaltotal) - ($idle-${previdle:-0}) ) / (intervaltotal) ))")"
   echo "$cpu"
}

TIMING=1

statustext()
{
    wmfs -c status "bartest ^g[200;0;300;12;$(cpu)
;100;#445544;#AABBAA;cpugraph](1;spawn;urxvtc -e htop)"
```

```
}

while true;
do
   statustext
   sleep $TIMING
done
```

- popups statusbar :

status-surface.sh
```
#!/bin/bash

# membar
membar(){
   memu="$(free -m | sed -n 's|^-.*:[ \t]*\([0-9]*\) .*|\1|gp')"
   memt="$(free -m | sed -n 's|^M.*:[ \t]*\([0-9]*\) .*|\1|gp')"
   echo "^s[65;10;#AABBAA;mem ](1;status_surface;150,16,110,14,#AABBAA ^s[left;#445544;
$memu Mb/$memt Mb]) ^p[90;1;80;10;0;$memu;$memt;#445544;#AABBAA]"
}

TIMING=1

statustext()
{
    wmfs -c status "bartest $(membar)"
}

while true;
do
   statustext
   sleep $TIMING
done
```

- script bash :

status-bash.sh
```
#!/bin/bash

#colors
clear="#C7C7C7"
grey="#7D7D7D"
# mem section
memu(){
   memu="$(free -m | sed -n 's|^-.*:[ \t]*\([0-9]*\) .*|\1|gp')"
   echo "^s[right;$grey;mem]^s[right;$clear;$memu]"
}
memt(){
   memt="$(free -m | sed -n 's|^M.*:[ \t]*\([0-9]*\) .*|\1|gp')"
   echo "^s[right;$clear;/$memt  ]"
}
# cpu section
cpu(){
   cpu="$(eval $(awk '/^cpu /{print "previdle="  "; prevtotal=" #!/bin/bash   #colors clear=
```

```
"#C7C7C7" grey="#7D7D7D" # mem section memu(){ memu="$(free -m | sed -n 's|^-.*:[ \t]*\([0-9]*\) .*|\1|gp')" echo "^s[right;$grey;mem]^s[right;$clear;$memu]" } memt(){ memt="$(free -m | sed -n 's|^M.*:[ \t]*\([0-9]*\) .*|\1|gp')" echo "^s[right;$clear;/$memt ]" } # cpu section cpu(){ cpu="$(eval $(awk '/^cpu /{print "previdle=" $5 "; prevtotal=" $2+$3+$4+$5 }' /proc/stat); sleep 0.4; eval $(awk '/^cpu /{print "idle=" $5 "; total=" $2+$3+$4+$5 }' /proc/stat); intervaltotal=$((total-${prevtotal:-0})); echo "$((100*( (intervaltotal) - ($idle-${previdle:-0}) ) / (intervaltotal) ))")" echo "^s[right;$grey;cpu]^s[right;$clear;$cpu% ](1;spawn;urxvt -e htop)" } # hdd section hdd(){ hdd="$(df -h|grep sda3|awk '{print $5}')" echo "^s[right;$grey;hdd]^s[right;$clear;$hdd ](1;spawn;urxvt -e ncdu)" } # temperatures tempcpu(){ tempcpu="$(cat /sys/devices/platform/thinkpad_hwmon/temp1_input | awk '{print $1/1000}')" echo "^s[right;$grey;tmp]^s[right;$clear;$tempcpu°]" } temphdd(){ temphdd="$(cat /sys/devices/platform/thinkpad_hwmon/temp3_input | awk '{print $1/1000}')" echo "^s[right;$clear;$temphdd°]" } tempgpu(){ tempgpu="$(cat /sys/devices/platform/thinkpad_hwmon/temp2_input | awk '{print $1/1000}')" echo "^s[right;$clear;$tempgpu° ]" } # date/time section dte(){ dte="$(date +"%a %d/%m")" echo "^s[right;$grey;$dte·]" } tme(){ tme="$(date +"%H:%M")" echo "^s[right;$clear;$tme ]" } # internet section int(){ int="$("$HOME/bin/speed-wmfs.sh")" echo "^s[right;$grey;net] ^s[right;$clear;$int ](1;spawn;urxvt -e net-monitor)" } # power pwr(){ pwrsta="$(cat /sys/class/power_supply/BAT0/status | cut -c 1)" pwrperc="$(awk 'sub(/,/,"") {print $4}' <(acpi -b) | cut -d , -f 1 $1)" if [ "$pwrsta" == "F" ]; then pwr="F" else pwr="$pwrsta·$pwrperc" fi echo "^s[right;$grey;bat]^s[right;$clear;$pwr ]" }   TIMING=1   statustext() { wmfs -c status "testbar $(pwr) $(cpu) $(memu)$(memt) $(hdd) $(int) $(tempcpu) $(temphdd) $(tempgpu) $(dte) $(tme)" } while true; do statustext sleep $TIMING done++ }' /proc/stat); sleep 0.4;
        eval $(awk '/^cpu /{print "idle="  "; total=" #!/bin/bash   #colors clear="#C7C7C7" grey="#7D7D7D" # mem section memu(){ memu="$(free -m | sed -n 's|^-.*:[ \t]*\([0-9]*\) .*|\1|gp')" echo "^s[right;$grey;mem]^s[right;$clear;$memu]" } memt(){ memt="$(free -m | sed -n 's|^M.*:[ \t]*\([0-9]*\) .*|\1|gp')" echo "^s[right;$clear;/$memt ]" } # cpu section cpu(){ cpu="$(eval $(awk '/^cpu /{print "previdle=" $5 "; prevtotal=" $2+$3+$4+$5 }' /proc/stat); sleep 0.4; eval $(awk '/^cpu /{print "idle=" $5 "; total=" $2+$3+$4+$5 }' /proc/stat); intervaltotal=$((total-${prevtotal:-0})); echo "$((100*( (intervaltotal) - ($idle-${previdle:-0}) ) / (intervaltotal) ))")" echo "^s[right;$grey;cpu]^s[right;$clear;$cpu% ](1;spawn;urxvt -e htop)" } # hdd section hdd(){ hdd="$(df -h|grep sda3|awk '{print $5}')" echo "^s[right;$grey;hdd]^s[right;$clear;$hdd ](1;spawn;urxvt -e ncdu)" } # temperatures tempcpu(){ tempcpu="$(cat /sys/devices/platform/thinkpad_hwmon/temp1_input | awk '{print $1/1000}')" echo "^s[right;$grey;tmp]^s[right;$clear;$tempcpu°]" } temphdd(){ temphdd="$(cat /sys/devices/platform/thinkpad_hwmon/temp3_input | awk '{print $1/1000}')" echo "^s[right;$clear;$temphdd°]" } tempgpu(){ tempgpu="$(cat /sys/devices/platform/thinkpad_hwmon/temp2_input | awk '{print $1/1000}')" echo "^s[right;$clear;$tempgpu° ]" } # date/time section dte(){ dte="$(date +"%a %d/%m")" echo "^s[right;$grey;$dte·]" } tme(){ tme="$(date +"%H:%M")" echo "^s[right;$clear;$tme ]" } # internet section int(){ int="$("$HOME/bin/speed-wmfs.sh")" echo "^s[right;$grey;net] ^s[right;$clear;$int ](1;spawn;urxvt -e net-monitor)" } # power pwr(){ pwrsta="$(cat /sys/class/power_supply/BAT0/status | cut -c 1)" pwrperc="$(awk 'sub(/,/,"") {print $4}' <(acpi -b) | cut -d , -f 1 $1)" if [ "$pwrsta" == "F" ]; then pwr="F" else pwr="$pwrsta·$pwrperc" fi echo "^s[right;$grey;bat]^s[right;$clear;$pwr ]" }   TIMING=1   statustext() { wmfs -c status "testbar $(pwr) $(cpu) $(memu)$(memt) $(hdd) $(int) $(tempcpu) $(temphdd) $(tempgpu) $(dte) $(tme)" } while true; do statustext sleep $TIMING done++ }' /proc/stat);
        intervaltotal=$((total-${prevtotal:-0}));
        echo "$((100*( (intervaltotal) - ($idle-${previdle:-0}) ) / (intervaltotal) ))")"
        echo "^s[right;$grey;cpu]^s[right;$clear;$cpu%  ](1;spawn;urxvt -e htop)"
}
# hdd section
hdd(){
```

```
    hdd="$(df -h|grep sda3|awk '{print }')"
    echo "^s[right;$grey;hdd]^s[right;$clear;$hdd  ](1;spawn;urxvt -e ncdu)"
}
# temperatures
tempcpu(){
    tempcpu="$(cat /sys/devices/platform/thinkpad_hwmon/temp1_input | awk '{print  class="code
bash"/1000}')"
    echo "^s[right;$grey;tmp]^s[right;$clear;$tempcpu°]"
}
temphdd(){
    temphdd="$(cat /sys/devices/platform/thinkpad_hwmon/temp3_input | awk '{print  class="code
bash"/1000}')"
    echo "^s[right;$clear;$temphdd°]"
}
tempgpu(){
    tempgpu="$(cat /sys/devices/platform/thinkpad_hwmon/temp2_input | awk '{print  class="code
bash"/1000}')"
    echo "^s[right;$clear;$tempgpu°  ]"
}
# date/time section
dte(){
    dte="$(date +"%a %d/%m")"
    echo "^s[right;$grey;$dte·]"
}
tme(){
    tme="$(date +"%H:%M")"
    echo "^s[right;$clear;$tme ]"
}
# internet section
int(){
    int="$("$HOME/bin/speed-wmfs.sh")"
    echo "^s[right;$grey;net] ^s[right;$clear;$int  ](1;spawn;urxvt -e net-monitor)"
}
# power
pwr(){
    pwrsta="$(cat /sys/class/power_supply/BAT0/status | cut -c 1)"
    pwrperc="$(awk 'sub(/,/,"") {print }' <(acpi -b) | cut -d , -f 1  class="code bash")"
    if [ "$pwrsta" == "F" ]; then
        pwr="F"
    else
        pwr="$pwrsta·$pwrperc"
    fi
    echo "^s[right;$grey;bat]^s[right;$clear;$pwr  ]"
}

TIMING=1

statustext()
{
     wmfs -c status "testbar $(pwr) $(cpu) $(memu)$(memt) $(hdd) $(int) $(tempcpu) $(temphdd)
$(tempgpu) $(dte) $(tme)"
```

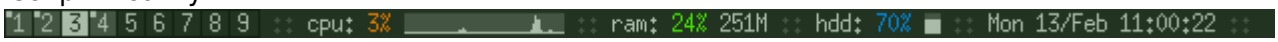
```
}

while true;
do
   statustext
   sleep $TIMING
done   le script net :   speed-wmfs.sh #!/bin/bash
# speed.sh
# sources <https://bitbucket.org/jasonwryan/workstation/src/d2045f97201e/scripts/speed.sh>

RXB=$(cat /sys/class/net/eth0/statistics/rx_bytes)
TXB=$(cat /sys/class/net/eth0/statistics/tx_bytes)
sleep 2
RXBN=$(cat /sys/class/net/eth0/statistics/rx_bytes)
TXBN=$(cat /sys/class/net/eth0/statistics/tx_bytes)
RXDIF=$(echo $((RXBN - RXB)) )
TXDIF=$(echo $((TXBN - TXB)) )

echo "$((RXDIF / 1024 / 2))k/s·$((TXDIF / 1024 / 2))k/s"
```

- script + conky :



se lance avec la commande suivante au démarrage de votre session:conky -c ~/.conkyrc_wmfs | while true; read line; do wmfs -c status "testbar $line"; done & le status_conkyrc : wmfs-conkyrc

```
##############################################
#  Settings
##############################################
background no
out_to_x no
out_to_console yes
update_interval 2
total_run_times 0
uppercase no
short_units yes
use_spacer none
if_up_strictness address
##############################################
#  Output
##############################################
TEXT
^s[left;\#445544; :: ]\
^s[left;\#AABBAA;cpu:] ^s[left;\#E07C00; ${cpu}% ]
^g[left;80;10;${cpu};100;\#445544;\#AABBAA;ckycpu]\
^s[left;\#445544; :: ]\
^s[left;\#AABBAA;ram:] ^s[left;\#43E000; $memperc%] ^s[left;\#AABBAA; $mem]\
^s[left;\#445544; :: ]\
^s[left;\#AABBAA;hdd:] ^s[left;\#008BE0; ${fs_used_perc /}% ] ^p[left;8;10;0;${fs_used_perc
/};100;\#445544;\#AABBAA;ckyhdd]\
^s[left;\#445544; :: ]\
^s[left;\#AABBAA;${time %a %d/%b %I:%M:%S}] ^s[left;\#445544; :: ]
```

liste des fonctions UICB

pour **User Interface Call Backs** aka appels d'interface utilisateur.
exemples d'utilisation:

- dans votre wmfsrc: func = "tag_next" ou func = "spawn" cmd = "urxvt -e vim"
- dans votre status.sh: wmfs -c status "<barname> ^s[<position>;<color>;next](1;tag_next)"
- dans votre terminal: wmfs -c tag_next

système

```
  spawn: lance une commande. ex: func = "spawn" cmd = "urxvtc -e screen
irssi".
  quit: quitte wmfs.
  reload: recharge la configuration de wmfs.
```

infobar

```
  infobar_toggle_hide : affiche/masque une barre d'info. ex: func =
"infobar_toggle_hide" cmd = "default" masque/affiche la barre "default".
```

tags

```
  tag_set : affiche le tag par n°. ex: func = "tag_set" cmd = "0" affiche le
tag n°0.
  tag : affiche le tag par nom. ex: func = "tag" cmd = "web" affiche le tag
nommé 'web'.
  tag_next : affiche le tag suivant.
  tag_prev : affiche le tag précédent.
  tag_client : tag le client. ex func = "tag_client" cmd = "2" tag le client
avec le n°2.
  tag_client_and_set : tag le client et affiche le tag.
  tag_move_client_next : tag le client avec le tag suivant.
  tag_move_client_prev : tag le client avec le tag précédent.
  tag_click : affiche le tag lors d'un clic-gauche de souris.
  tag_new : ajoute un tag.
```

```
  tag_del : enlève un tag.
```

layouts

```
  layout_vmirror : organise le pavage en miroir vertical.
  layout_hmirror : organise le pavage en miroir horizontal.
  layout_rotate_left : rotation du pavage dans le sens anti-horaire.
  layout_rotate_right : rotation du pavage dans le sens horaire.
  layout_prev_set : revenir au précédent layout.
  layout_next_set : aller au prochain layout.
  layout_integrate_left : intègre la client dans la zone de celui de gauche.
  layout_integrate_right : intègre la client dans la zone de celui de droite.
  layout_integrate_top : intègre la client dans la zone de celui du haut.
  layout_integrate_bottom : intègre la client dans la zone de celui du bas.
```

clients

```
  client_close : ferme le client.
  client_resize_right : redimensionne le client en partant du coté droit. ex:
"client_resize_right" cmd = "-20" réduit le client de 20 pixels depuis le
bord droit.
  client_resize_left : redimensionne le client en partant du coté gauche. ex
"client_resize_left" cmd = "20" agrandit le client de 20 pixels depuis le
bord gauche.
  client_resize_top : redimensionne le client en partant du haut.
  client_resize_bottom : redimensionne le client en partant du bas.
  client_focus_right : donne le focus au client de droite.
  client_focus_left : donne le focus au client de gauche.
  client_focus_top : donne le focus au client du haut.
  client_focus_bottom : donne le focus au client du bas.
  client_tab_right : déplace le client dans un onglet du client de droite.
  client_tab_left : déplace le client dans un onglet du client de gauche.
  client_tab_top : déplace le client dans un onglet du client du haut.
  client_tab_bottom : déplace le client dans un onglet du client du bas.
  client_swap_right : échange le client avec le client de droite.
  client_swap_left : échange le client avec le client de gauche.
  client_swap_top : échange le client avec le client du haut.
  client_swap_bottom : échange le client avec le client du bas.
  client_focus_next : donne le focus au client suivant.
  client_focus_prev : donne le focus au client précédent.
  client_swap_next : échange le client avec le client suivant.
  client_swap_prev : échange le client avec le client précédent.
  client_untab : libère le client de l’onglet.
  client_focus_next_tab : donne le focus à l’onglet suivant.
  client_focus_prev_tab : donne le focus à l’onglet précédent.
```

```
  client_focus_click : donne le focus au client lors d'un clic.
  client_toggle_free : libère/attache un client.
  client_toggle_ignore_tag : active/désactive client visible sur tous les
tags.
  client_tab_next_opened : ouvre le client dans un onglet.
```

status

```
  status : affiche le texte passé en argument dans la statusbar.
  status_surface : affiche une surface pouvant contenir des séquences.
```

mouse

```
  mouse_resize : redimensionne le client avec la souris.
  mouse_move : déplace le client avec la souris.
  mouse_swap : échange les clients avec la souris.
  mouse_tab : inclut le client dans un onglet avec la souris.
```

screen

```
  screen_next : aller à l'écran suivant.
  screen_prev : aller à l'écran précédent.
  screen_move_client_next : déplace le client vers l'écran suivant.
  screen_move_client_prev : déplace le client vers l'écran précédent.
```

launcher

```
  launcher : lanceur intégré aka prompt. ex: func = "launcher" cmd = "exec"
affiche le lanceur "exec".
```