

# dzen2: the ultimate statusbar

**dzen2** est un logiciel de messages, notifications et menus pour X11. il a été conçu pour être léger, rapide et pour supporter les scripts en différents langages.

dzen2 supporte les polices XFT (en options), les couleurs et le formatage simple, les icônes et graphiques et les actions du clavier/souris. dzen2 est disponible dans la plupart des distributions GNU/Linux.

## Sommaire

dzen2: the ultimate statusbar.....	1
installation.....	1
configuration.....	2
options.....	2
affichage.....	2
arrêt.....	3
utilisations.....	3
option -l: slave window.....	3
option -e: events & actions.....	3
option -m: menu.....	5
option -u: update simultané.....	5
option de formats.....	6
exemples.....	7
exemples simples.....	7
conky dans dzen2.....	9

## installation

pour avoir les dernières options, je vous conseille d'installer la version **git**. pour l'obtenir:

```
$ git clone git://github.com/robm/dzen.git
```

installer les dépendances pour Debian:

```
# apt-get install libx11-dev libxft-dev libxinerama-dev
```

dzen2 s'installera par défaut dans `/usr/local/bin`. vous pouvez modifier le chemin d'installation en éditant le fichier `config.mk` situé dans le dossier `git`. vous pouvez en profiter pour activer le support xinerama et xpm en (dé)commentant les sections appropriées.

une fois votre `config.mk` prêt, vous pouvez commencer la procédure d'installation: placez-vous dans le répertoire:

```
$ cd dzen/
```

lancer la compilation et installer:

```
$ make
# make install
```

si vous désirez utiliser les "gadgets" (dbar, gdbar, gcpubar) permettant d'afficher des graphiques:

```
$ cd gadgets/
$ make
# make install
```

vous pouvez désormais tester dzen2 avec une commande simple:

```
$ echo "hi, i'm testing dzen2 :)" | dzen2 -p 10
```

## configuration

### options

dzen accepte différentes options, nous décriront plus tard le format appliqué à ces options:

options	arguments	description
-fg	#RRGGBB, color	couleur du texte
-bg		couleur du fond
-fn	police xft ou classic	type de police utilisé
-ta	l(ef), c(enter),	alignement du texte de la fenêtre "titre"
-sa	r(ight)	alignement du texte de la fenêtre "esclave"
-tw	x(pixels)	largeur de la fenêtre "titre"
-w		largeur de la fenêtre
-h		hauteur des lignes, par défaut: hauteur de la police +2px
-x		position horizontale depuis le bord gauche de l'écran
-y		position verticale depuis le bord haut de l'écran
-l	n(nombre)	nombre de lignes affichées dans la fenêtre "esclave"
-m	dzen2 en mode "menu" (voir section "menu")	
-e	événements et actions (voir section events & actions)	
-p	s(secondes)	la dernière valeur est persistente pendant 's' secondes (en option)
-u	mettre à jour les fenêtres "titre" et "esclave" en même temps	
-xs	n(uméro)	numéro de l'écran xinerama à utiliser
-v	version de dzen2	

### affichage

dzen2 affiche les fenêtres "titre" et "esclave" comme suit:

fenêtre titre
fenêtre esclave scrollable sur plusieurs lignes définies par "-l"

la première ligne passée à dzen2 est affichée dans la fenêtre titre, les autres lignes seront affichées

dans la zone esclave à moins que vous n'utilisiez l'option de format '^tw()' pour surpasser le comportement par défaut.

## arrêt

dzen2 suspend l'affichage:

- si le fichier envoyé à dzen2 se termine (EOF), sauf si l'option "-p" est utilisée:
  - '-p' sans argument fixe l'affichage pour toujours
  - '-p <n>' fixe l'affichage pour 'n' secondes
- lors d'un clic-droit sur dzen2 (par défaut)
- lors d'une pression sur [Esc] dans certaines conditions

## utilisations

notez que les options décrites ici sont celles par défaut. vous pouvez modifier ces comportements grâce à l'option -e (voir events & actions).

### option -l: slave window

utiliser l'option -l pour afficher plusieurs lignes au survol du pointeur sur la fenêtrés "titre" de dzen2. le nombre de ligne à afficher est passé en argument (-l 5 affiche 5 lignes).

lorsque le pointeur quitte la fenêtré "esclave", celle-ci est masquée, sauf si vous "fixez" la fenêtré avec un clic central.

les boutons 4 et 5 (molette de la souris) font défiler le contenu de la fenêtré "esclave" si le nombre de ligne dépasse la hauteur de la fenêtré.

**exemple:** une manière simple de comprendre comment la fenêtré esclave fonctionne (clic droit pour quitter dzen2):

```
cat ~/.bashrc | dzen2 -l 30 -p
```

### option -e: events & actions

dzen permet d'associer certains événements (clavier/souris) à des actions configurables.

syntaxe de l'option -e: -e 'event1=action1:option1:...option<n>,...,action<m>;...'

chaque événement peut être associé à plusieurs actions qui peuvent avoir plusieurs options (limité à 64 pour chaque mais modifiable depuis les sources, fichier action.h).

#### • exemple

```
-e 'button1=exec:xterm:firefox;entertitle=uncollapse;button3=exit'
```

#### • explication

- *button1=exec:xterm:firefox;* : au clic gauche, lancement de xterm et firefox. dans cet exemple, xterm et firefox sont les *options* de l'*action* exec pour l'*event* button1.
- *entertitle=uncollapse;* : au passage du pointeur, dzen affiche (uncollapse) la fenêtré esclave
- *button3=exit* : au clic droit, arrêt de dzen2.

## liste des événements

- onstart : l'action se lance au démarrage de dzen.
- onexit : l'action se lance juste avant de quitter dzen.
- onnewinput : l'action se lance si une nouvelle entrée est passée à la fenêtre esclave.
- button1 : l'action se lance lors du clic gauche
- button2 : l'action se lance lors du clic central
- button3 : l'action se lance lors du clic droit
- button4 : l'action se lance avec la molette vers le haut
- button5 : l'action se lance avec la molette vers le bas
- button6 : l'action se lance lors d'une pression sur le bouton 6 (sur certaines souris)
- button7 : l'action se lance lors d'une pression sur le bouton 7 (sur certaines souris)
- entertitle : l'action se lance lors du survol de la fenêtre titre
- leavetitle : l'action se lance lorsqu'on quitte la fenêtre titre
- enterslave : l'action se lance lors du survol de la fenêtre esclave
- leaveslave : l'action se lance lorsqu'on quitte la fenêtre esclave
- sigusr1 : l'action se lance à réception du signal SIGUSR1 ?
- sigusr2 : l'action se lance à réception du signal SIGUSR2 ?
- key\_KEYNAME : l'action se lance lors d'un événement clavier(\*)

## les événements clavier

toutes les touches peuvent entrainer une action avec l'option **-e**. la syntaxe utilisée est celle du fichier `/usr/include/X11/keysymdef.h` . par exemple, pour "e" `event = key_e` , pour "\_" `event = key_underscore`

## liste des actions

- `exec:command1:...:n` : exécute les options passées en argument
- `menuexec` : exécute l'entrée de menu sélectionnée
- `exit:retval` : quitte dzen et retourne la valeur 'retval'
- `print:str1:...:n` : écrit les options passées en argument dans STDOUT
- `menuprint` : écrit l'entrée de menu sélectionnée dans STDOUT
- `collapse` : enroule (masque) la fenêtre esclave
- `uncollapse` : déroule (affiche) la fenêtre esclave
- `togglecollapse` : (en/de)roule la fenêtre esclave
- `stick` : fixe la fenêtre esclave
- `unstick` : libère la fenêtre esclave
- `togglestick` : fixe/libère la fenêtre esclave
- `hide` : masque la fenêtre titre
- `unhide` : affiche la fenêtre titre
- `togglehide` : masque/affiche la fenêtre titre
- `raise` : passe la fenêtre au premier plan
- `lower` : passe la fenêtre au dernier plan
- `scrollhome` : affiche les premières lignes de la fenêtre esclave
- `scrollend` : affiche les dernières lignes de ma fenêtre esclave
- `scrollup:n` : fait défiler la fenêtre esclave de 'n' lignes vers le haut (n=1 par défaut)
- `scrolldown:n` : fait défiler la fenêtre esclave de 'n' lignes vers le bas (n=1 par défaut)
- `grabkeys` : active le support su clavier

- ungrabkeys : désactive le support du clavier
- grabmouse : active le support de la souris (nécessaire seulement pour certains gestionnaires de fenêtres tels que fluxbox)
- ungrabmouse : désactive le support de la souris (nécessaire seulement pour certains gestionnaires de fenêtres tels que fluxbox)

**les événements par défaut** si aucune option n'est passée en arguments, dzen observe les comportements par défaut suivant:

- fenêtre titre seule:

```
-e 'button3=exit:13'
```

- fenêtre titre + esclave multi-lignes ou en mode menu:

```
-e 'entertitle=uncollapse,grabkeys;
enterslave=grabkeys;leaveslave=collapse,ungrabkeys;
button1=menuexec;button2=toggletick;button3=exit:13;
button4=scrollup;button5=scrolldown;
key_Escape=ungrabkeys,exit'
```

- fenêtre en mode menu horizontal:

```
-e 'enterslave=grabkeys;leaveslave=ungrabkeys;
button4=scrollup;button5=scrolldown;
key_Left=scrollup;key_Right=scrolldown;
button1=menuexec;button3=exit:13
key_Escape=ungrabkeys,exit'
```

⚠ note: si vous définissez des options différentes, dzen ne tiendra aucun compte des valeurs par défaut: vous devrez spécifier **toutes** les options.

## option -m: menu

dzen propose deux modes de menu: vertical et horizontal. pour activer ces modes, ajouter 'v'(vertical) ou 'h'(horizontal) à l'option '-m'. le menu est vertical par défaut.

- menu vertical (les deux expressions sont équivalentes):

```
dzen2 -p -l 4 -m < file
dzen2 -p -l 4 -m v < file
```

- menu horizontal:

```
dzen2 -p -l 4 -m h < file
```

toutes les actions commençant par "menu" sont appliquées à l'entrée de menu sélectionné.

le mode de menu fonctionne de paire avec l'option '-l <n>'

le menu horizontal n'a pas de fenêtre titre, les actions passées à la fenêtre titre seront ignorées.

## option -u: update simultané

### deprécié

cette option permet la mise à jour simultané entre la fenêtre titre et la fenêtre esclave.

un exemple sera plus explicite: imaginonsque nous voulions afficher l'heure dans la fenêtre titre et

une sortie de log dans le fenêtre esclave:

```
while true; do
  date          # output goes to the title window
  dmesg | tail -n 10 # output goes to the slave window
  sleep 1
done | dzen2 -l 10 -u
```

pour que cela fonctionne, il est important de spécifier le nombre exacte de ligne du fichier lu avec l'option '-l'.

## option de formats

cette option permet d'appliquer dynamiquement un format particulier à l'affichage de dzen2.

dzen2 accepte les formats suivants:

### • couleurs

- `^fg(color)`: couleur du texte/icones
- `^fg()`: retour à la couleur par défaut
- `^bg(color)`: couleur du fond
- `^bg()`: retour à la couleur par défaut

### • graphiques

- `^i(chemin de l'image)`: affiche une image au format xbm ou xpm(en option)
- `^r(largeurxhauteur)`: affiche un rectangle aux dimensions spécifiées
- `^ro(largeurxhauteur)`: affiche un rectangle en contour aux dimensions spécifiées
- `^c('r' rayon)`: affiche un cercle de rayon 'r'
- `^co('r' rayon)`: affiche un cercle en contour de rayon 'r'

### • positions `^p=position relative / ^pa=position absolue`

- `^p(+/-X)`: déplace la prochaine entrée à droite(+) ou gauche(-) de X pixels
- `^p(+/-X;+/-Y)`: déplace la prochaine entrée à droite(+) ou gauche(-) de X pixels et vers le haut(+) ou le bas(-) de Y pixels.
- `^pa(args)`: même arguments que précédemment mais X et Y sont des valeurs absolues.

### • interactions

- `^ca(BTN, CMD) ... ^ca()`: utilisé pour définir une zone cliquable dans la fenêtre titre
  - BTN détermine le bouton de souris actif (1=gauche, 2=central, 3=droit ...).
  - CMD détermine la commande à lancer lors d'un clic sur la zone concernée.
  - ... : représente un texte affiché par dzen
  - `^ca()`: sans arguments, ferme la zone cliquable

### • exemple:

```
^ca(1,mpc toggle)^ca(3,mpc stop)[^ca(4,mpc prev)^ca(5,mpc next)mpd^ca()^ca()|
^ca(4,mpc volume +3)^ca(5,mpc volume -3)vol^ca()^ca()|^ca()^ca()
```

cette commande affichera “[mpd | vol]” où

- le bloc entier [...] répond au clic droit et gauche pour lecture/stop
  - **mpd** répond à la molette pour naviguer dans la playlist mpd
  - **vol** répond à la molette pour augmenter/diminuer le volume
- ### • commandes d'actions
- voir la section “events & actions” pour une explication détaillée des commandes.
- `^togglecollapse()`
  - `^collapse()`
  - `^uncollapse()`

- ^togglestick()
- ^stick()
- ^unstick()
- ^togglehide()
- ^hide()
- ^unhide()
- ^raise()
- ^lower()
- ^scrollhome()
- ^scrollend()
- ^exit()
- **autres**
  - ^tw(): affiche le texte uniquement dans la fenêtre titre. cette option peut générer des conflits, à utiliser une seul fois par ligne et en début d'entrée.
  - ^cs(): vide la fenêtre esclave, cette option doit être seule sur la ligne
  - ^ib(VALUE): appliquer (ou pas) la couleur de fond définie par ^bg(color). "VALUE"=0:appliquer la couleur, =1:ne pas applique la couleur.

**ces commandes peuvent être placées à n'importe quel endroit et avec toutes les combinaisons dans l'entrée soumise à dzen2.**

**les couleurs peuvent être spécifiées avec leur nom (red, green, darkorange...) ou leur valeur rrggbb (#fffaa)**

## exemples

### exemples simples

voici une liste d'exemples de codes et leur affichage illustrant les sections précédentes. j'utilise le fichier ~/.Xresources ce qui permet de définir la police, la couleur du texte et la couleur du fond par défaut:

```
! dzen2 -----
dzen2.font:  *_fixed-_*_*_*_*_*_10-70-_*_*_*_*_*_*_*
dzen2.foreground: #C7C7C7
dzen2.background: #222222
```

notez que les options passées en ligne de commande écrasent les options Xresources.

- texte couleur

```
echo "^fg(red)I'm red text ^p(+20)^fg(blue)I am blue" | dzen2 -w 300 -p
```



- texte et fond couleur

```
echo "^bg(#ffaata)The ^fg(yellow)text to ^bg(blue)^fg(orange)colorize" | dzen2 -w 300 -p
```



- afficher ""

```
echo "^fg(grey70)Some text containing ^^ characters" | dzen2 -w 300 -p
```

```
Some text containing ^ characters
```

- images xbm (pack d'icônes dispo [ici](#))

```
echo "^\i(dzicons/mail.xbm) I am an envelope ^fg(yellow)and  
^\i(dzicons/bat_full_02.xbm) I'm a battery." | dzen2 -w 300 -p
```

```
✉ I am an envelope and 🦇 I'm a battery.
```

- rectangles

```
echo "6x4 rectangle ^r(6x4) ^fg(red)12x8 ^r(12x8) ^fg(yellow)and finally 100x15  
^r(100x15)" | dzen2 -w 300 -p
```

```
6x4 rectangle ■ 12x8 ■ and finally 100x15
```

- rectangles et positions

```
echo "^\i(1)^fg(red)^ro(100x15)^p(-  
98)^fg(blue)^r(20x10)^fg(orange)^p(3)^r(40x10)^p(4)^fg(darkgreen)^co(12)^p(2)^c  
(10)" | dzen2 -x 20 -y 20 -w 120 -p
```



- messages pendant 10sec

```
(echo "This is a message"; sleep 10) | dzen2 -bg darkred -fg grey80
```

```
This is a message
```

- affiche "header" et un message multi-lignes

```
(echo Header; cal; sleep 20) | dzen2 -w 300 -l 8
```

```
Header  
January 2012  
Su Mo Tu We Th Fr Sa  
1 2 3 4 5 6 7  
8 9 10 11 12 13 14  
15 16 17 18 19 20 21  
22 23 24 25 26 27 28  
29 30 31
```

- affiche "header" et un message dynamique multi-lignes

```
(echo Header; while true; do echo test$((i++)); sleep 1; done) | dzen2 -w 300  
-l 12
```

```
Header  
test0  
test1  
test2  
test3  
test4  
test5  
test6
```

- menu simple:

```
dzen2 -w 200 -l 4 -p -m < dzenmenu.txt
```

où dzenmenu.txt se présente comme suit(chaque entrée après "Menu" correspond à une commande)

```
Menu  
urxvtc  
rox-filer
```

```
firefox
geany
```



- menu horizontal sans fichier menu

```
echo -e "xterm\xclock\xeyes\xfontsel" | dzen2 -w 300 -l 4 -m h -p
```



## conky dans dzen2



dzen2 va lire les informations délivrées par conky comme pour la barre de statut de [dwm](#). le fichier de configuration du conky présenté dans le screenshot ressemble à celui utilisé pour dwm mais ajoute les options pour dzen:

```
out_to_x no
out_to_console yes
update_interval 1.0
total_run_times 0
use_spacer none

TEXT
^i(/home/arp/.arp_setups/dzicons/cal.xbm) ^fg(\#ccc){time %d/%m}^fg()
^i(/home/arp/.arp_setups/dzicons/clock.xbm) ${time %I:%M}\
^fg(\#ff4500)^i(/home/arp/.arp_setups/dzicons/cpu.xbm) ^fg(\#ccc){cpu}%\
^fg(\#ee2c2c)^i(/home/arp/.arp_setups/dzicons/temp.xbm) ^fg(\#ccc){ibm_temps
0}°C\
^fg(\#87ceeb)^i(/home/arp/.arp_setups/dzicons/monitor.xbm) ^fg(\#ccc){loadavg}\
^fg(\#00bfff)^i(/home/arp/.arp_setups/dzicons/mem.xbm) ^fg(\#ccc){memperc% >
$mem\
^fg(\#ffd700)^i(/home/arp/.arp_setups/dzicons/home.xbm) ^fg(\#ccc){
{fs_used_perc /}%\
${if_mounted /media/arp500} ^fg(\#000)^i(/home/arp/.arp_setups/dzicons/usb.xbm)
^fg(\#ccc){fs_used_perc /media/arp500/}%${endif}${if_mounted /media/lacie300}
^fg(\#1e90ff)^i(/home/arp/.arp_setups/dzicons/usb.xbm) ^fg(\#ccc){
{fs_used_perc /media/lacie300/}%${endif}\
^fg(){if_match ${battery_percent BAT0}
>=26}^fg(\#7cfc00)^i(/home/arp/.arp_setups/dzicons/bat_full_01.xbm){endif}\
${if_match ${battery_percent BAT0} <=25}
^fg(\#CC0000)^i(/home/arp/.arp_setups/dzicons/bat_low_01.xbm)^fg(){endif}
^fg(\#ccc){battery_percent}%\
^fg(){if_up eth0} ^i(/home/arp/.arp_setups/dzicons/net_wired.xbm)\
^fg(\#00cd00)^i(/home/arp/.arp_setups/dzicons/net_up_02.xbm) ^fg(\#ccc){
{upspeedf eth0}\
^fg(\#ffa500)^i(/home/arp/.arp_setups/dzicons/net_down_02.xbm) ^fg(\#ccc){
{downspeedf eth0}\
^fg() ${endif}\
${if_match "$ibm_volume" == "mute" }^fg(\#000000)
^i(/home/arp/.arp_setups/dzicons/spkr_04.xbm)^fg(\#ccc) mute ${else}^fg(\#FFDF1D)
^i(/home/arp/.arp_setups/dzicons/spkr_01.xbm)^fg(\#ccc) ${ibm_volume}/14${endif}
```

**note** l'emploi du "\n" en fin de ligne permet l'annulation du saut de ligne dans conky et aide à une meilleure lisibilité du fichier.

comme vous pouvez le constater, les options passées à conky sont "encadrées" par les options passées à dzen2:

- ^fg(): pour la couleur du texte/de l'image
- ^i() : pour l'adresse de l'image à afficher

les images doivent être au format \*xbm (facilement éditables avec the Gimp). une archive contenant 67 icônes est [disponible ici](#).

ce fichier conky est appelé par dzen2 grâce à un script (que vous prendrez soin de rajouter à votre fichier ~autostart selon votre configuration):

```
#!/bin/sh
RC="$HOME/.conkyrc_dzen"
FG="white"
BG="#404240"
ALIGN="left"
WIDTH="744"
HEIGHT="12"
FONT="-*-terminus-medium-*-*-*12-*-*-*-*-*-*"
XPOS="30"
YPOS="756"

conky -d -c $RC | dzen2 -fg $FG -bg $BG -ta $ALIGN -w $WIDTH -h $HEIGHT -x $XPOS
-y $YPOS -fn $FONT -dock &
exit 0
```

sources du script: [minull conky](#) par [xeNULL](#).

---

contributeur: [arpinux](#)

sources: [README](#) officiel + [wiki dzen](#)(eng).