

Pimp

my

Linux



Les shells

Shell Bourne

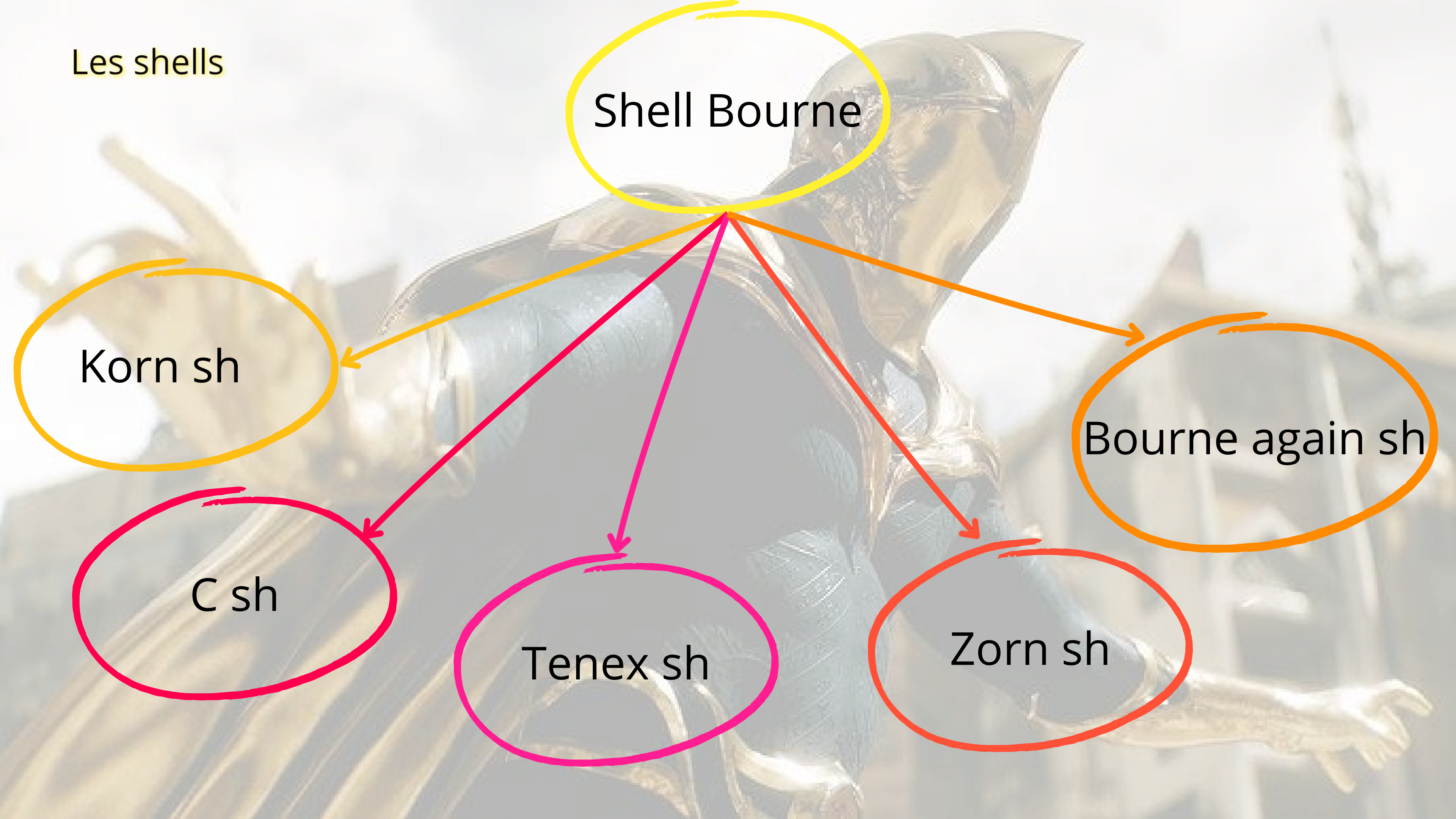
Korn sh

Bourne again sh

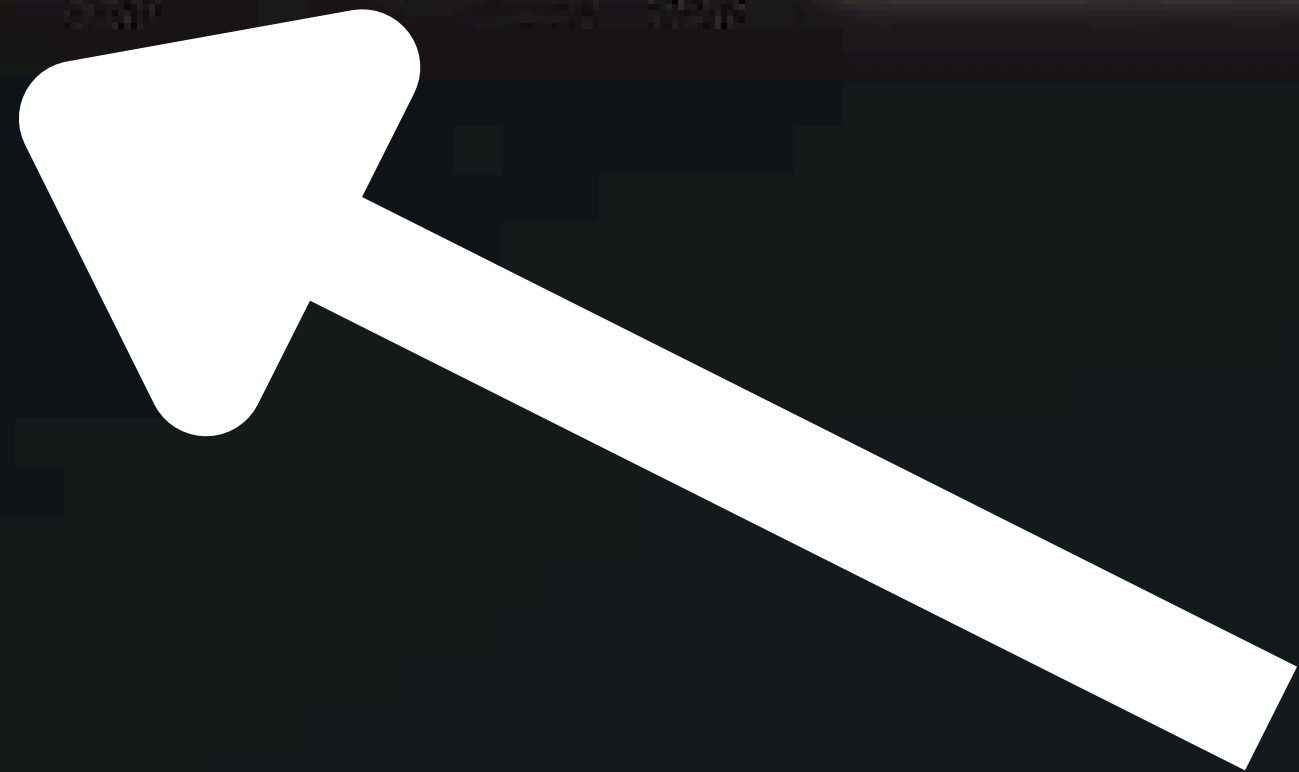
C sh

Tenex sh

Zorn sh



```
[ronin@arch hyprshot]$ █
```



Keske c finalement ça ?

Pk c moche comme ça

```
PS1='[\u005B\u005C \W]\$ '
```

```
PS1='[\u005B\u005C@ \W]\$ '
```

La syntaxe pour rendre sa jolie

- \n : Faire un retour à la ligne
- \r : Faire un retour en début de ligne (carriage return)
- \j : le nombre de jobs réalisé par le shell (correspond aux tâches exécutés en fond, grâce au ctrl+z)
- \d : La date au format : « Jour Mois Année »
- \t ou \T : le temps soit sous la forme 24h, hh:mm:ss ou sur 12h
- \A : La date actuelle sous le format HH:MM (24h)
- \u : Le nom de l'utilisateur courant.
- \v : La version du bash
- \! : Le numéro de la dernière commande exécuté.
- \w : l'arborescence de votre \$HOME jusqu'au dossier courant
- \W : le dossier courant

Exemple :

```
PS1='\u@\h (\d\t) - \j - \w\$'
```

```
ronin@arch (jeu. oct. 24 19:52:49) - 0 - ~$
```

Le gris c'est triste

Des couleurs

1er Méthode : \e

- \e[: début de la couleur
- 48;5;Xm : identifiant couleur background
- 38;5;Xm : identifiant couleur foreground
- \e[m : reset

Pour obtenir votre code couleur

```
color=0;
while [ $color -lt 255 ]; do
    echo -e "$color: \e[38;5;${color}mhello\e[48;5;${color}mworld\e[0m"
    ((color++));
done
```

2ème Méthode : tput

- **tput bold** – passe en gras
- **tput rev** – Inverse les couleurs
- **tput sgr0** – Remet tout à zéro
- **tput setaf {CODE}** - Couleur de police
- **tput setab {CODE}** – Couleur d'arrière plan



Les codes couleurs

de 0 à 15, ce sont les couleurs définies sur le terminal

256-color mode — foreground: ESC[38;5;#m background: ESC[48;5;#m

Standard colors

High-intensity colors

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
---	---	---	---	---	---	---	---	---	---	----	----	----	----	----	----

216 colors

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51
52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87
88	89	90	91	92	93	94	95	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120	121	122	123
124	125	126	127	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191	192	193	194	195
196	197	198	199	200	201	202	203	204	205	206	207	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223	224	225	226	227	228	229	230	231

Grayscale colors

232	233	234	235	236	237	238	239	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

Pour zsh

PS1:

- %m: nom de la machine
- %n: nom de l'utilisateur
- %c: nom du dossier dans lequel vous vous trouvez
- %d: chemin complet
- %~: chemin complet mais le home est contracté en ~
- %T: heure
- %D: date au format aa-mm-jj

1er méthode coloriage:

- C'est tout pareil

2ème méthode coloriage:

- C'est tout pareil

Les bonnes pratiques

Répartir sa configuration dans différents fichiers :

- .bash_alias : fichier pour les alias
- .bash_function : fichier pour les fonctions perso
- .bash_color : si vous avez bcp de couleurs

Nommer vos variables:

- fRED=\$(tput setaf 1)
- RESET=\$(tput sgr0)
- PS1="Vive\$fRED Tux\$RESET \ \$"

Syntaxe à mettre en début de .bashrc
OU .bashrc_custom OU .bash_profile

```
if [ -f file ]; then  
    . file  
fi
```

WAIT



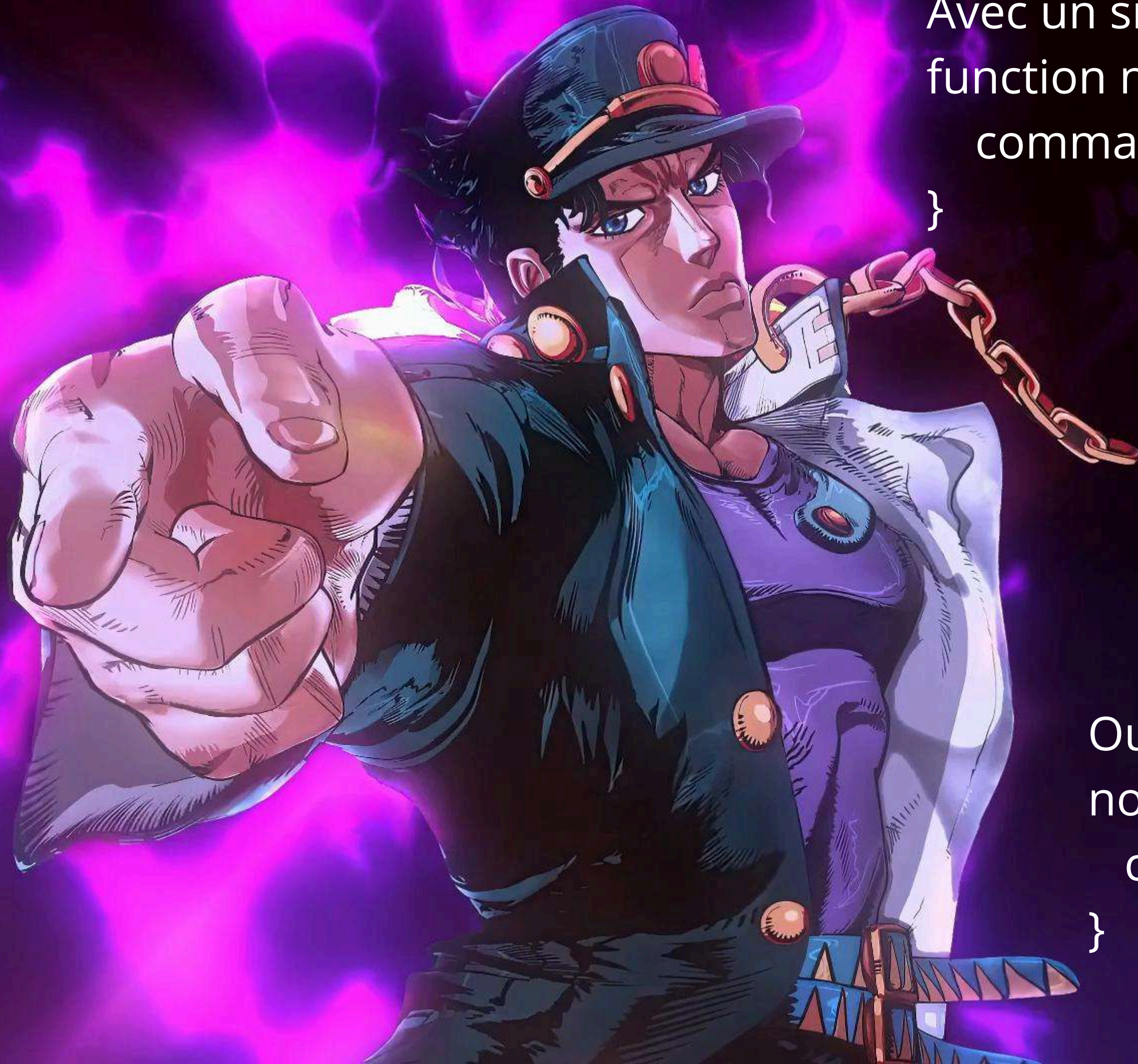


DES FONCTIONS ??

Et oui

Avec un simple:
function nom() {
 commands;
}

Ou juste
nom() {
 commands;
}



Des exemples

Une fonction pour récupérer la branche courante git

```
function parse_git_branch() {  
    git branch 2> /dev/null | sed -e '/^[^*]/d' -e 's/* \(.*\)/(\1)/'  
}
```

ou

```
function git_ref {  
    rf=$(git symbolic-ref -q HEAD || git name-rev --name-only --always HEAD)  
    rf=${rf#refs/heads/}  
    echo $rf  
}
```

```
PS1='${fBORDEAU}\u${fSAND}@${fFOREST}\h \W ${fBLUE}$(parse_git_branch)${fINPUT}\$ '
```

```
ronin@arch ~ $
```

```
ronin@arch bdf-unifont (master)$
```

Moto Moto



Fonction pour ouvrir un nouveau terminal

```
new(){
  case "$1" in
    tab) gnome-terminal --tab;;
    window) gnome-terminal --window;;
  esac
}
```

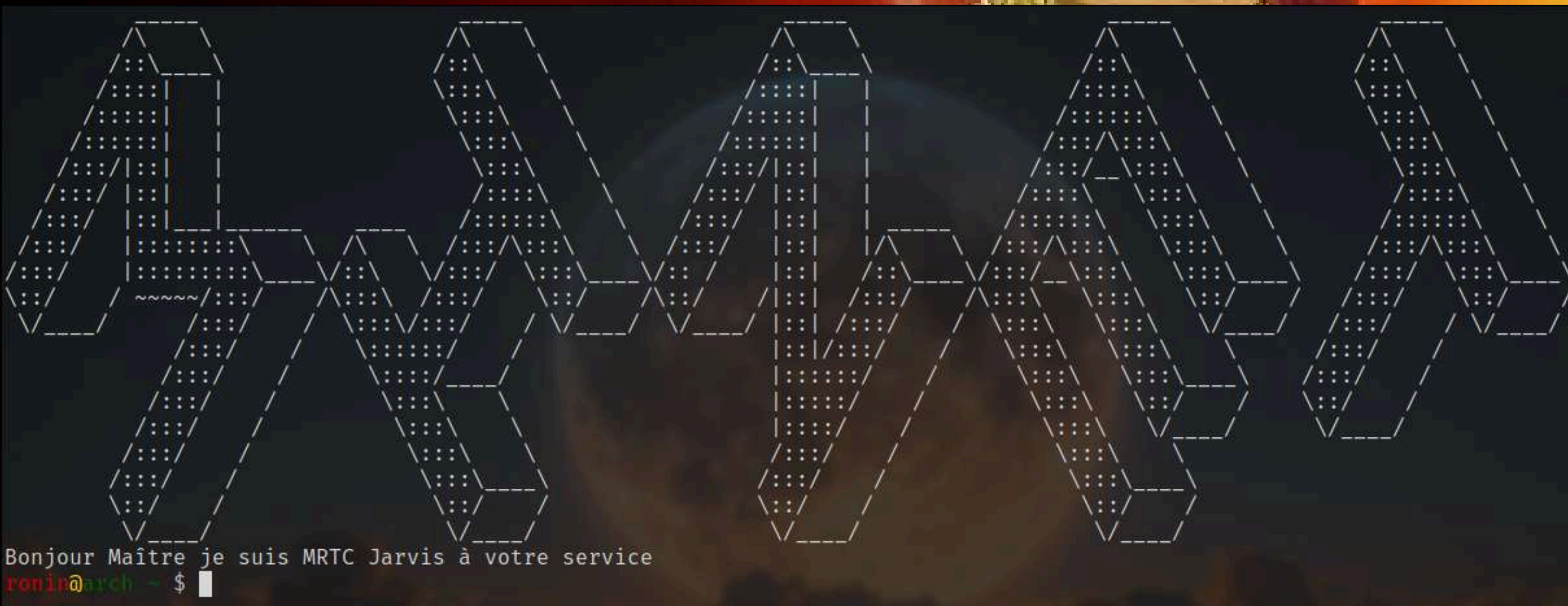
Fonction pour remonter son arborescence plus rapidement

```
function up() {
  ups=""
  for i in $(seq 1 $1)
  do
    ups=$ups"../"
  done
  cd $ups
}
```

Fonction pour changer le titre du terminal

```
function set-title() {
  if [[ -z "$ORIG" ]]; then
    ORIG=$PS1
  fi
  TITLE="\[\\e]2;$*\[\\a\"
  PS1=${ORIG}${TITLE}
}
```

Dernière petite chose



ASCII ART

Générateur de texte sympa

```
cat<<'EOF'  
#Votre truc  
EOF
```

Et



dans tout ça ?

Les classiques

Dans les paramètres



Apparence



Bureau Ubuntu

Barre des tâches "Dock"

Dock

Masquer automatiquement le dock

Le dock se cache quand une fenêtre le recouvre



Mode du panneau

Le dock va jusqu'aux bords de l'écran



Taille d'icône



Afficher sur

Écran principal (1) ▾

Position à l'écran

Bas ▾

Configurer le comportement du dock



Les classiques

Dans les paramètres



Apparence



Bureau Ubuntu

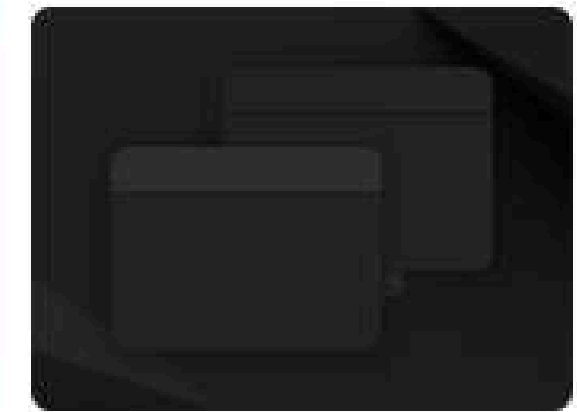


Fond d'écran

Style



Par défaut



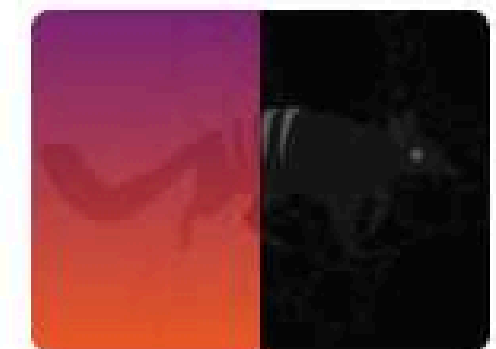
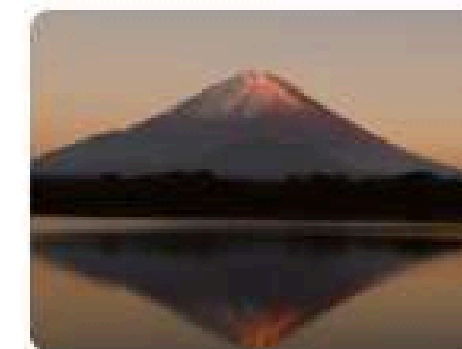
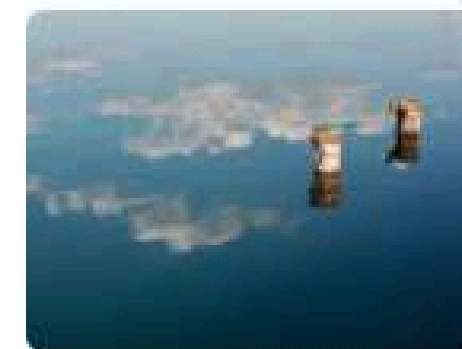
Sombre

Couleur



Arrière-plan

+ Ajouter une image...



Les classiques

Dans les paramètres



Clavier

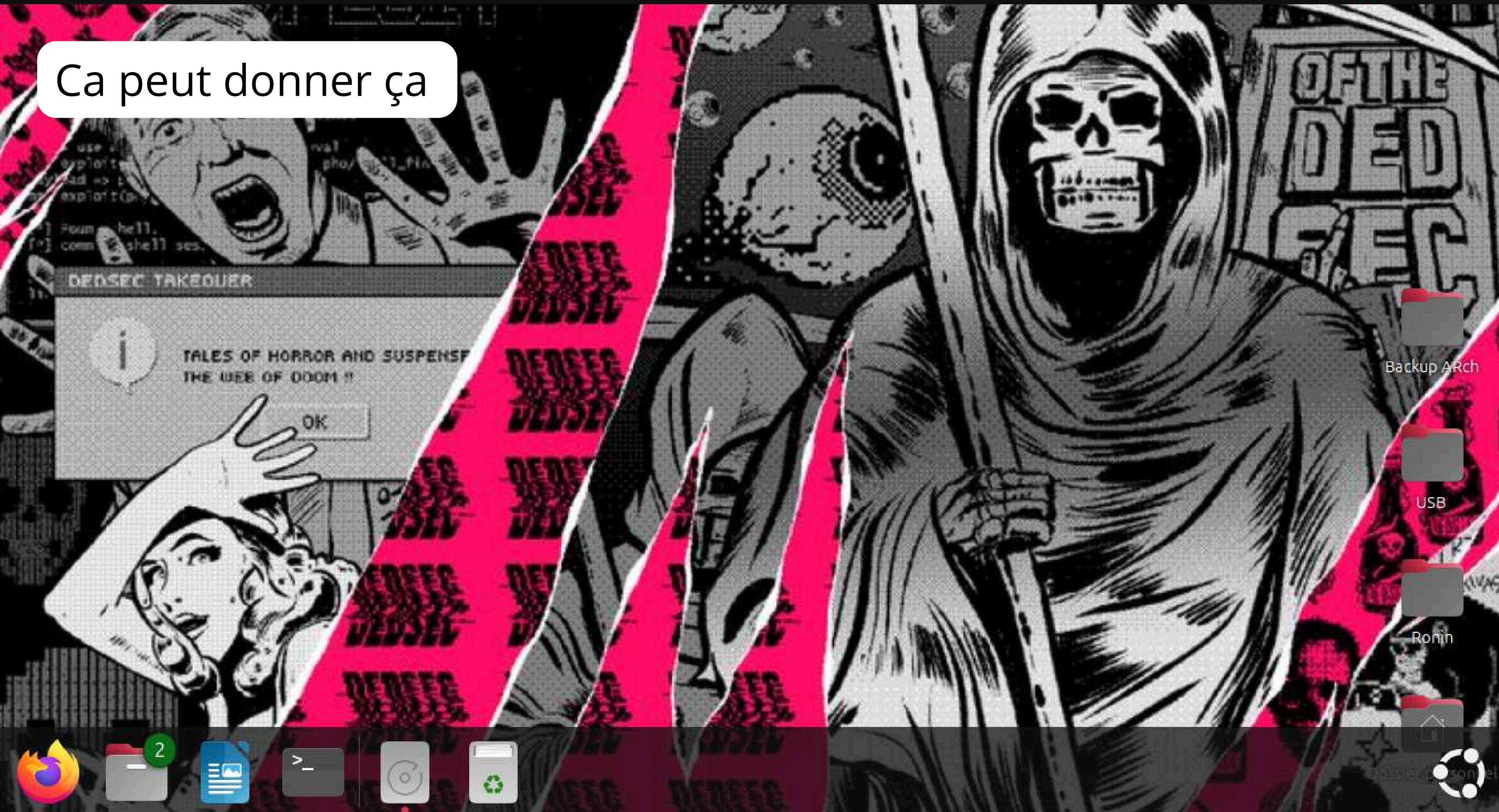
Raccourcis clavier

Voir et personnaliser les raccourcis

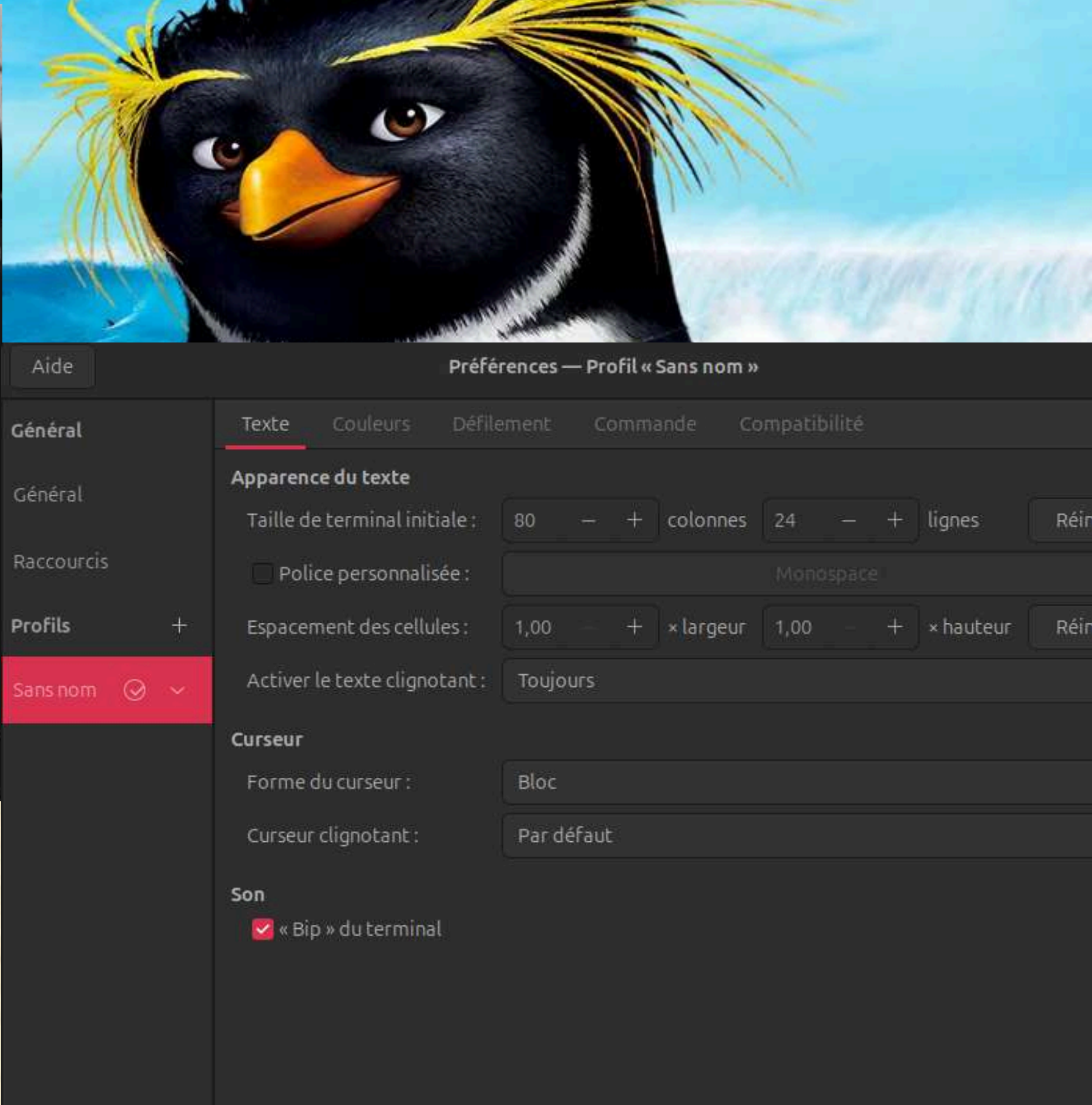
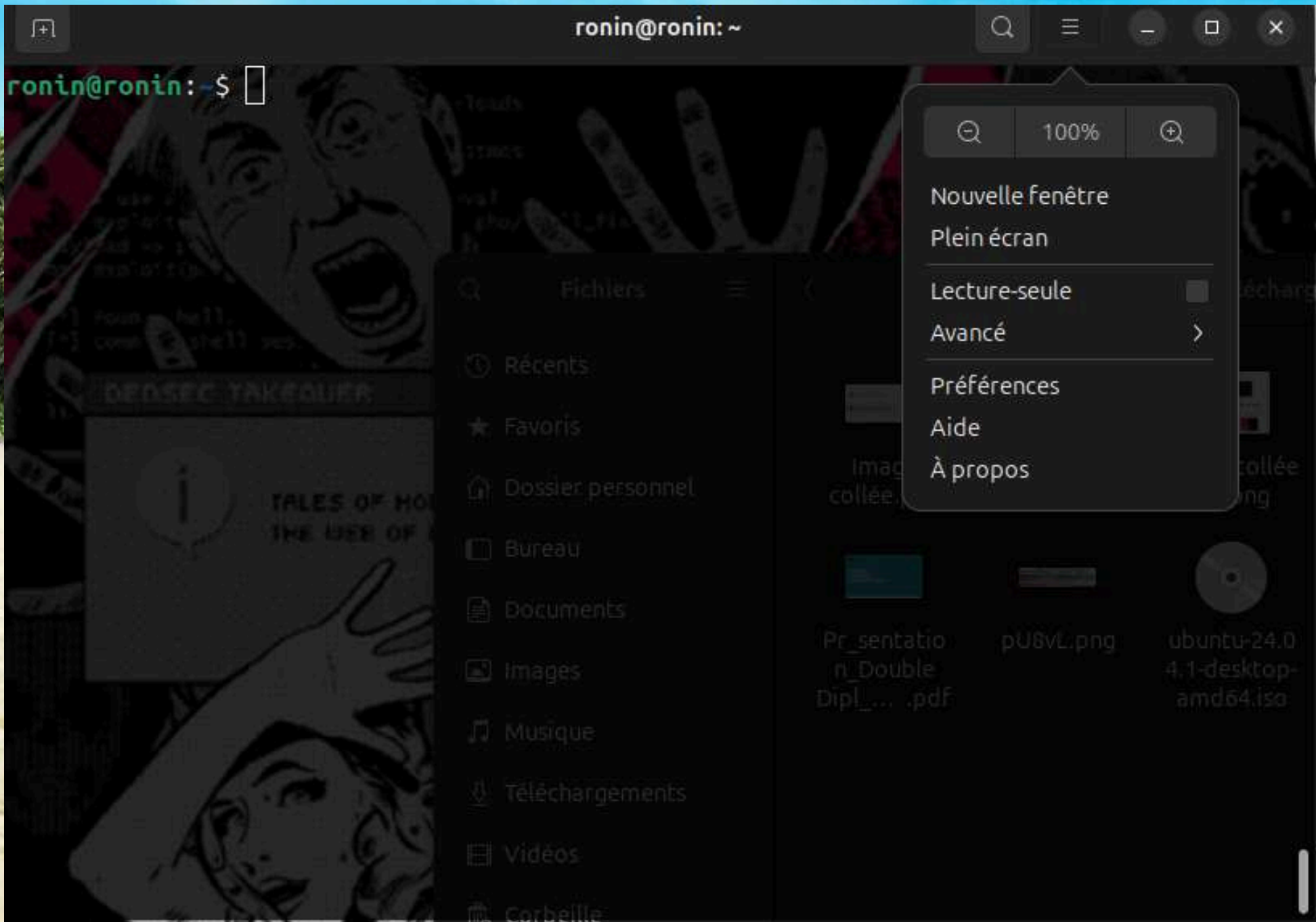


Super = Bouton windows

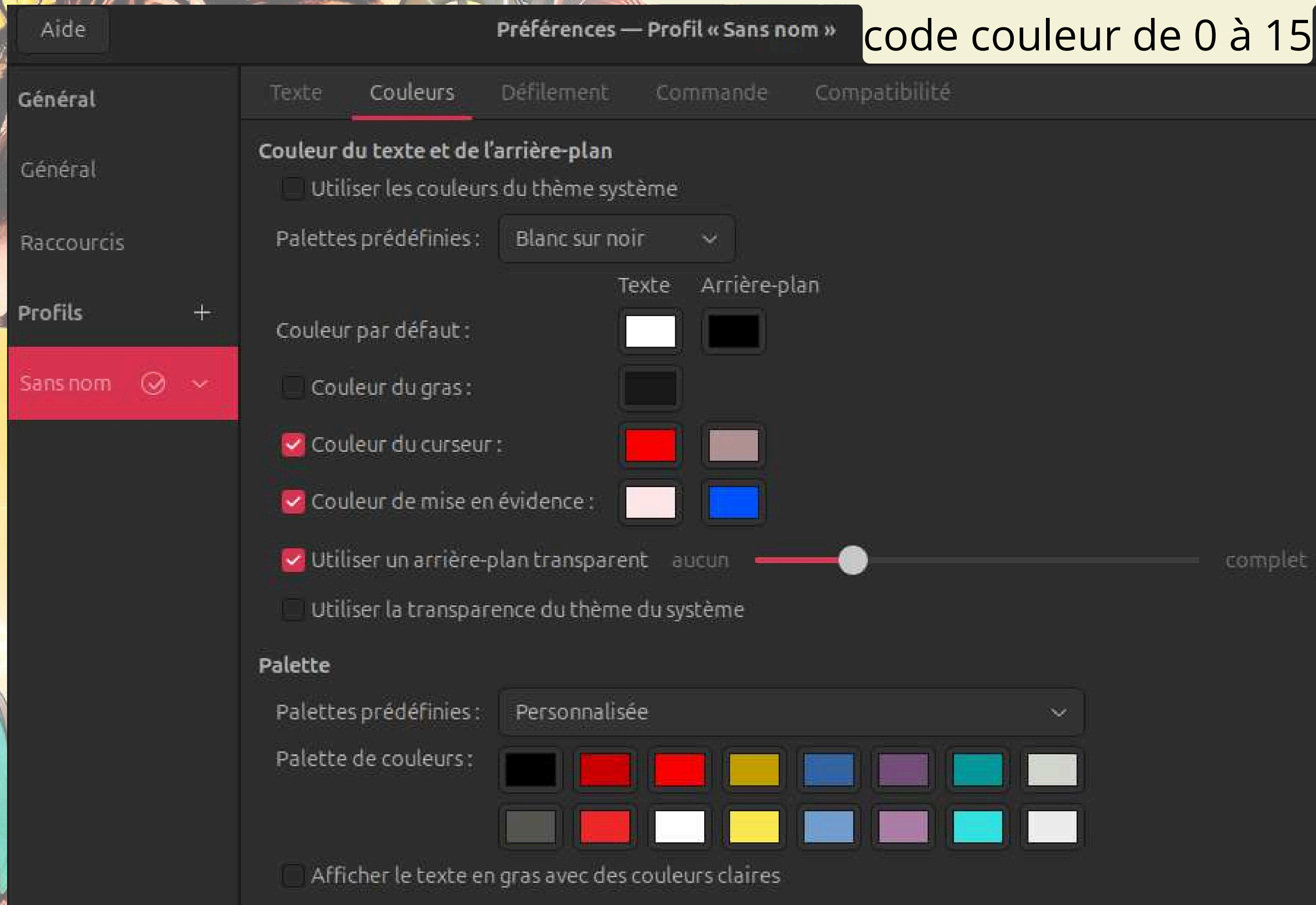
Ca peut donner ça



Le retour du terminal



Les 16 couleurs sont celles identifiées avec le code couleur de 0 à 15



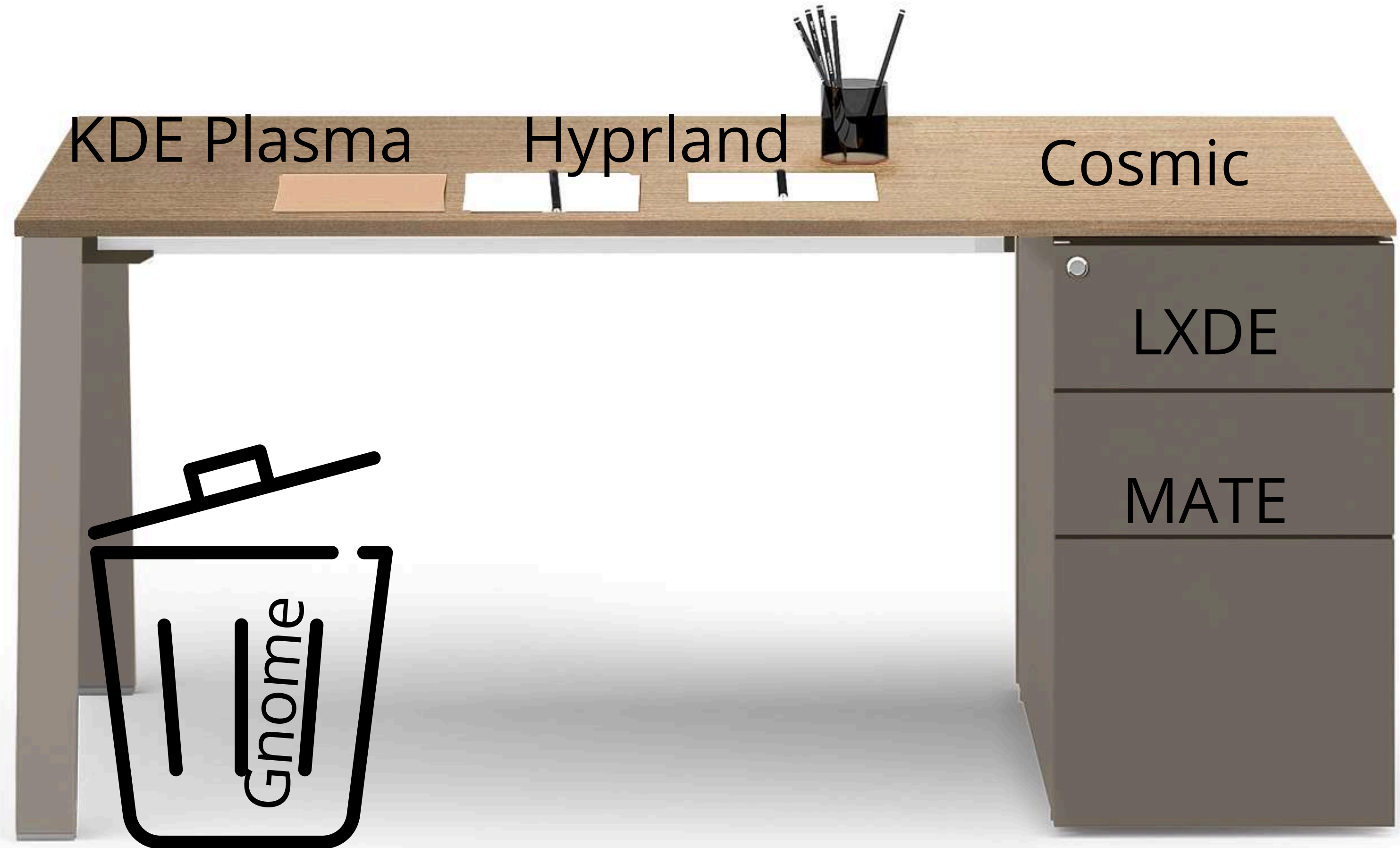
CHOTTO MATTE



C'est quoi



Les environnements de bureau - DE



ça je gère

ça je gère

ça je gère

```
.14.0
-oSSSSSSSO. :SSSSSSSO.
de (12.0pt)
:oSSSSSSSS/ oSSSSO+++.
) i7-4510U (4) @ 3.10 GHz
/oSSSSSSSSS/ +SSSSO00/-
840M
`/oSSSSSO+/: - -:/+oSSSSO+-
ULT Integrated Graphics Controller @ 1.10 GHz [Integrated]
`+SSO+:-` `.-/+oso:
69 GiB (40%)
`++:. `-/+/
iB (0%)
`/
456.48 GiB (9%) - ext4
159.49.19/24
0% [AC Connected]

ronin@arch ~ $
```

```
WM: Hyprland (Waylan)
Cursor: Adwaita
Terminal: alacritty 0
Terminal Font: FiraC
CPU: Intel(R) Core(TM)
GPU 1: NVIDIA GeForce
GPU 2: Intel Haswell
Memory: 3.16 GiB / 7)
Swap: 0 B / 1024.00 )
Disk (/): 41.69 GiB 4
Local IP (eno1): 1574
Battery (Primary): 1]
Locale: fr_FR.UTF-8

ronin@arch ~ $
```

```
ronin@arch ~ $
ronin@arch ~ $
ronin@arch ~ $
ronin@arch ~ $
ronin@arch ~ $
ronin@arch ~ $
ronin@arch ~ $
```

```
ronin@arch ~ $
```

ce bordel je gère

Cas Hyprland

Fichier de conf only

```
ronin@arch hypr $ ls
conf  hyprland.conf  hyprlock.conf  hyrpaper.conf
ronin@arch hypr $ ls conf/
animations.conf  binds.conf  decoration.conf  general.conf  input.conf  misc.conf  windowrules.conf
autostart.conf  cursor.conf  environments.conf  gestures.conf  layouts.conf  monitor.conf
ronin@arch hypr $ ls ../waybar/
config  icon  modules.json  style.css
ronin@arch hypr $
```

et c'est tout



**Il existe plein de packets pour faire la
même chose
A vous de trouver le vôtre**



A vous de vous amuser avec votre PC

Bonus sur les fonctions

Les arguments sont les mêmes que pour un fichier:

- \$# : le nombre d'arguments
- \$1 ... \$n : l'argument en position n
- \$0 : le nom de la fonction

Petit bonus :

- \$* et @\$: représente tout les arguments
- "\$*" et "\$@" sont différents
 - "\$*" renvoie une unique chaîne de caractères "\$1 \$2 \$n"
 - "\$@" renvoie les arguments un par un "\$1" "\$2" ... "\$n"

JOTARO KUJO
JOJO NOKIMYOU NA BOUKEN