dwm, un gestionnaire de fenêtres dynamique

- Objet : Découvrir et apprendre à installer puis à utiliser dwm
- Niveau requis :
  avisé
- Commentaires : *Bien que tous peuvent l'utiliser, dwm, de part le dénuement qui fait sa force, s'adresse d'avantage à des utilisateurs connaissant bien leur système et désireux d'en avoir un contrôle très fin.*
- Suivi :
  à-tester
  - Création par sogal 18/01/2015
  - Testé par <...> le <...>
- Commentaires sur le forum : Lien vers le forum concernant ce tuto [1)]

Introduction

Voici la très bonne et très officielle présentation de dwm, disponible sur son site:
http://dwm.suckless.org

dwm est un gestionnaire de fenêtres dynamique pour X. Il gère les fenêtres selon 3 agencements différents:

- tuilage : les fenêtres sont réparties entre une zone principale et une zone d'empilement. La zone principale contient la fenêtre dans laquelle vous travaillez, la zone d'empilement contenant les autres;

- monocle : les fenêtres sont toutes maximisées à la taille de l'écran (et se masquent donc l'une l'autre);

- flottant : dans cet agencement les fenêtres peuvent être redimensionnées et déplacées librement, c'est le comportement que l'on retrouve dans la plupart des autres gestionnaires et environnements de bureaux (XFCE, LXDE, etc...).

Notes : les fenêtres de dialogue sont toujours en mode flottant. De plus, dwm est bien un gestionnaire de fenêtres, il ne leur apporte aucune décoration, ni bordure avec titres et boutons “Fermer”, “Maximiser” ou “Réduire”.

dwm (et l'ensemble des outils du projet suckless.org) adhère à une philosophie minimaliste (voir http://suckless.org/philosophy). Il est écrit en C et sa configuration se fait dans un fichier écrit en C avant compilation. Cela peut paraître contraignant mais apporte une grande stabilité et la garantie que des options incompatibles ou erronées n'ont pas été indiquées (sans quoi une erreur empêcherait la compilation).

Ceux qui le souhaitent peuvent avoir un aperçu de dwm grâce aux captures d'écran des utilisateurs

disponibles sur la page dédiée du projet: http://dwm.suckless.org/screenshots/

Installation

Prérequis

L'idée de compiler soi-même un logiciel aussi important que son gestionnaire de fenêtres peut en effrayer certains mais nous verrons pas-à-pas comment procéder et il n'y a pas réellement de difficulté technique majeure.
Il suffit de savoir utiliser un terminal et de saisir les lignes de commandes indiquées ci-après. En effet, toute la configuration et l'installation seront faites via le terminal.

En revanche, vous aurez besoin d'un certain nombre d'outils pour compiler et des bibliothèques de développement X11.
Voici comment les installer:

```
apt-get install make gcc libx11-dev libxinerama-dev
```

Si vous souhaitez, au moment de l'installer, avoir la possibilité de créer un paquet .deb de votre configuration et vous assurer que votre installation sera prise en compte par le système de gestion des paquets de Debian, il peut être utile d'installer le paquet suivant:

```
apt-get install checkinstall
```

Récupération du code source

Bien que vous puissiez récupérer les sources sur le dépôt git du projet:

```
git clone git://git.suckless.org/dwm
```

dans ce tuto, nous allons préférer prendre les sources dans les dépôts Debian de manière à avoir les fichiers de contrôle et autre qui vont bien:

```
apt-get source dwm
```

Vous allez obtenir un dossier dwm-6.0 contenant les sources, nous allons y entrer:

```
cd dwm-6.0
```

```
ls
config.def.h  config.mk  debian  dwm.1  dwm.c  LICENSE  Makefile  README
```

Parmi ces fichiers, 2 seulement nous seront utiles, le config.mk et le config.def.h.

Fichier config.mk

C'est dans ce fichier que nous pouvons modifier les options de compilation. Celle qui va nous intéresser principalement concerne le répertoire dans lequel nous allons installer dwm:

```
# Customize below to fit your system
# paths
PREFIX = /usr/local
MANPREFIX = ${PREFIX}/share/man
## La variable PREFIX par défaut va installer le binaire dwm dans
/usr/local/bin. Bien que je ne le recommande pas, si vous souhaitez
l'installer dans /usr/bin, il faut la modifier comme suit:
##PREFIX = /usr
```

Fichier config.def.h

Voici donc notre fichier de configuration, dans lequel nous allons modifier ou ajouter les options pour personnaliser notre dwm bien à nous.
Mais avant cela, il est nécessaire d'appliquer un patch modifiant les séquences de touches et les adaptant au clavier azerty. Le patch est récupérable à l'adresse suivante : http://dwm.suckless.org/patches/azertykey . Il s'applique comme suit (pour rappel, notre répertoire est toujours dwm-6.0) :

```
patch -p1 < ../dwm-azertykey.diff
```

Ce qui devrait vous renvoyer une confirmation de succès:

```
patching file config.def.h
Hunk #1 succeeded at 68 (offset 3 lines).
Hunk #2 succeeded at 102 (offset 3 lines).
```

01-2017 : il semble que le lien ci-dessus est mort. Vous pouvez tout de même utiliser votre dwm avec un clavier azerty en utilisant les codes pour tags ci-dessous (c'est-à-dire comme dans l'exemple de configuration qui suit:

```
    TAGKEYS(                        XK_ampersand,              0)
    TAGKEYS(                        XK_eacute,                 1)
    TAGKEYS(                        XK_quotedbl,               2)
    TAGKEYS(                        XK_apostrophe,             3)
    TAGKEYS(                        XK_parenleft,              4)
    TAGKEYS(                        XK_minus,                  5)
    TAGKEYS(                        XK_egrave,                 6)
    TAGKEYS(                        XK_underscore,             7)
    TAGKEYS(                        XK_ccedilla,               8)
```

Nous allons pouvoir passer à la configuration, voici une version commentée en français du

config.def.h :

```
/* See LICENSE file for copyright and license details. */

/* appearance */
/* police de caractères à utiliser */
static const char font[]            = "-*-terminus-medium-r-*-*-16-*-*-*-*-
*-*-*";
/* couleur des bordures de fenêtres inactives : */
static const char normbordercolor[] = "#515151";
/* couleur d'arrière plan barre / tags inactifs / dwmstatus: */
static const char normbgcolor[]     = "#191919";
/* couleur de police des tags inactifs et de police de dwmstatus: */
static const char normfgcolor[]     = "#FFE400";
/* couleur des fenêtres actives: */
static const char selbordercolor[]  = "#D70751";
/* couleur d'arrière plan des tags actifs: */
static const char selbgcolor[]      = "#515151";
/* couleur de police des tags actifs: */
static const char selfgcolor[]      = "#33FF00";

static const unsigned int borderpx  = 1;        /* épaisseur en pixel des
bordures de fenêtres */
static const unsigned int snap      = 32;       /* snap pixel */
static const Bool showbar           = True;     /* afficher (True) ou pas
(False) la barre des tags */
static const Bool topbar            = True;     /* la barre des tags doit
être en haut (True) ou en bas (False) */

/* tagging */
/* édition des tags, vous mettez ce que vous voulez, chiffres, symbole ou
nom (très court de préférence) du tag */
static const char *tags[] = { "1", "2", "3", "4", "5", "6", "7", "8", "9" };

/* définition des règles: vous pouvez ici appliquer un agencement par défaut
à un logiciel particulier */
static const Rule rules[] = {
    /* class      instance    title       tags mask     isfloating   monitor
*/
    { "Gimp",     NULL,       NULL,       0,            True,        -1 },
    { "Firefox",  NULL,       NULL,       1 << 8,       False,       -1 },
};

/* layout(s) */
static const float mfact      = 0.55; /* facteur de taille de la zone
principale [0.05..0.95] */
static const int nmaster      = 1;    /* nombre de fenêtres dans la zone
principale */
static const Bool resizehints = True; /* True = respecter les indications de
taille lors du redimensionnement */
```

```
static const Layout layouts[] = {
    /* symbol     arrange function */
    { "[]=",      tile },    /* la première entrée sera celle par défaut */
    { "><>",      NULL },    /* NULL = aucune fonction d'agencement = mode
flottant */
    { "[M]",      monocle },
};

/* key définitions */
/* définition du modificateur: Mod1Mask = touche Alt ; Mod4Mask = touche
Super (ou windows) */
#define MODKEY Mod1Mask
#define TAGKEYS(KEY,TAG) \
    { MODKEY,                       KEY,      view,           {.ui = 1 <<
TAG} }, \
    { MODKEY|ControlMask,           KEY,      toggleview,     {.ui = 1 <<
TAG} }, \
    { MODKEY|ShiftMask,             KEY,      tag,            {.ui = 1 <<
TAG} }, \
    { MODKEY|ControlMask|ShiftMask, KEY,      toggletag,      {.ui = 1 <<
TAG} },

/* helper for spawning shell commands in the pre dwm-5.0 fashion */
/* cette fonction permet le lancement de commande shell pour démarrer un
programme */
/* comme vous le feriez dans un terminal */
#define SHCMD(cmd) { .v = (const char*[]){ "/bin/sh", "-c", cmd, NULL } }

/* commands */
/* commande permettant le lancement de dmenu */
static const char *dmenucmd[] = { "dmenu_run", "-fn", font, "-nb",
normbgcolor, "-nf", normfgcolor, "-sb", selbgcolor, "-sf", selfgcolor, NULL
};
/* commande d'exécution de l'émulateur de terminal par défaut du système */
static const char *termcmd[]  = { "x-terminal-emulator", NULL };

static Key keys[] = {
    /* modifier                     key        function        argument */
    { MODKEY,                       XK_p,      spawn,          {.v =
dmenucmd } },
    { MODKEY|ShiftMask,             XK_Return, spawn,          {.v = termcmd
} },
    { MODKEY,                       XK_b,      togglebar,      {0} },
    { MODKEY,                       XK_j,      focusstack,     {.i = +1 } },
    { MODKEY,                       XK_k,      focusstack,     {.i = -1 } },
    { MODKEY,                       XK_i,      incnmaster,     {.i = +1 } },
    { MODKEY,                       XK_d,      incnmaster,     {.i = -1 } },
    { MODKEY,                       XK_h,      setmfact,       {.f = -0.05}
},
    { MODKEY,                       XK_l,      setmfact,       {.f = +0.05}
},
```

```
    { MODKEY,                       XK_Return, zoom,           {0} },
    { MODKEY,                       XK_Tab,    view,           {0} },
    { MODKEY|ShiftMask,             XK_c,      killclient,     {0} },
    { MODKEY,                       XK_t,      setlayout,      {.v =
&layouts[0]} },
    { MODKEY,                       XK_f,      setlayout,      {.v =
&layouts[1]} },
    { MODKEY,                       XK_m,      setlayout,      {.v =
&layouts[2]} },
    { MODKEY,                       XK_space,  setlayout,      {0} },
    { MODKEY|ShiftMask,             XK_space,  togglefloating, {0} },
    { MODKEY,                       XK_agrave,      view,           {.ui =
~0 } },
    { MODKEY|ShiftMask,             XK_agrave,      tag,            {.ui =
~0 } },
    { MODKEY,                       XK_comma,  focusmon,       {.i = -1 } },
    { MODKEY,                       XK_semicolon, focusmon,    {.i = +1 } },
    { MODKEY|ShiftMask,             XK_comma,  tagmon,         {.i = -1 } },
    { MODKEY|ShiftMask,             XK_semicolon, tagmon,      {.i = +1 } },
    TAGKEYS(                        XK_ampersand,                   0)
    TAGKEYS(                        XK_eacute,                      1)
    TAGKEYS(                        XK_quotedbl,                    2)
    TAGKEYS(                        XK_apostrophe,                  3)
    TAGKEYS(                        XK_parenleft,                   4)
    TAGKEYS(                        XK_minus,                       5)
    TAGKEYS(                        XK_egrave,                      6)
    TAGKEYS(                        XK_underscore,                  7)
    TAGKEYS(                        XK_ccedilla,                    8)
    { MODKEY|ShiftMask,             XK_q,      quit,           {0} },
};

/* button definitions */
/* click can be ClkLtSymbol, ClkStatusText, ClkWinTitle, ClkClientWin, or
ClkRootWin */
static Button buttons[] = {
    /* click                event mask      button          function
argument */
    { ClkLtSymbol,          0,              Button1,        setlayout,
{0} },
    { ClkLtSymbol,          0,              Button3,        setlayout,
{.v = &layouts[2]} },
    { ClkWinTitle,          0,              Button2,        zoom,
{0} },
    { ClkStatusText,        0,              Button2,        spawn,
{.v = termcmd } },
    { ClkClientWin,         MODKEY,         Button1,        movemouse,
{0} },
    { ClkClientWin,         MODKEY,         Button2,        togglefloating,
{0} },
    { ClkClientWin,         MODKEY,         Button3,        resizemouse,
{0} },
```

```
    { ClkTagBar,            0,              Button1,        view,
{0} },
    { ClkTagBar,            0,              Button3,        toggleview,
{0} },
    { ClkTagBar,            MODKEY,         Button1,        tag,
{0} },
    { ClkTagBar,            MODKEY,         Button3,        toggletag,
{0} },
};
```

Une fois notre configuration faite, nous pouvons compiler et installer dwm avec la commande:

```
make clean install
```

Toutefois, comme mentionné plus haut, je recommande l'utilisation de checkinstall:

```
checkinstall -D
```

Ceci permet une utilisation interactive, vous autorisant notamment à changer le numéro de version pour un code de version personnel et permet de créer un paquet .deb de votre version personnalisé de dwm.

Lors de la première compilation, un fichier config.h est créé à partir du config.def.h. C'est ce config.h qui est pris en compte véritablement, pensez donc par la suite, soit à le modifier directement, soit à modifier le config.def.h puis, juste avant la compilation, à le copier sous le nom config.h.

Une fois installé, vous pouvez le tester en sélectionnant la session adéquate dans votre gestionnaire de session (gdm3, kdm, slim...) ou en ajoutant la ligne:

```
exec /usr/local/bin/dwm
```

à votre fichier ~/.xinitrc.

Utilisation

Comprendre la notion de tag (étiquettes):

dwm ne fonctionne pas sur le modèle des espaces de travail, mais sur le principe des étiquettes (tags) que vous attribuez à une ou plusieurs fenêtre(s). C'est par exemple ce que vous faites de façon statique dans le config.def.h, mais cela peut être fait de façon dynamique. Ainsi la combinaison de touche [Maj + MODKEY + é] attribuera le 2ème tag à la fenêtre active. Vous pouvez sélectionner les tags grâce à la combinaison MODKEY+touche_correspondante. Pour reprendre notre exemple, si vous faites MODKEY + é vous sélectionnez toutes les fenêtres ayant le tag “é” et retrouvez donc celle précédemment étiquetée. Le fonctionnement des tags est détaillé de fort belle manière ici: dwm, chez thuban.

Liste des raccourcis clavier :

RACCOURCIS CLAVIER	***RESULTAT***
MODKEY+Maj+↵ Entrée	ouvre un émulateur de terminal
MODKEY + B	affiche/cache la barre des tags
MODKEY + T	passer en agencement tuilage (tiling)
MODKEY + F	passer en agencement flottant
MODKEY + M	passer en agencement monocle, la fenêtre principale est maximisée
MODKEY + Espace	passe de l'agencement courant à celui précédemment utilisé
MODKEY + J	passe le focus à la fenêtre suivante
MODKEY + K	passe le focus à la fenêtre précédente
MODKEY + H	réduit la taille de la zone principale
MODKEY + L	augmente la taille de la zone principale
MODKEY + ↵ Entrée	envoie la fenêtre sélectionnée vers la zone principale (ou l'envoie dans la pile si elle y est déjà)
MODKEY + Maj+C	ferme la fenêtre sélectionnée
MODKEY + Maj+Espace	bascule la fenêtre sélectionnée entre les agencements tuilage et flottant
MODKEY + Tab ⇆	sélectionne la vue précédente
MODKEY + Maj + [1..n]	donne à la fenêtre l'étiquette 1..n
MODKEY + Maj+	donner toutes les étiquettes à la fenêtre
MODKEY + Ctrl+Maj + [1..n]	ajoute/retire l'étiquette numéro 1..n à la fenêtre sélectionnée, pratique si on veut donner une nouvelle étiquette à une fenêtre sans la faire disparaître. Équivalent à MODKEY + Maj puis MODKEY + Ctrl+N
MODKEY + [1..n]	voir toutes les fenêtres dont l'étiquette est 1..n
MODKEY +	voir toutes les fenêtres, quelle que soient leurs étiquettes
MODKEY + Ctrl + [1..n]	ajoute/retire de la vue toutes les fenêtres avec l'étiquette 1..n
MODKEY + ,	change d'écran, si il y en a plusieurs
MODKEY + Maj+Q	quitte dwm (et votre session X par la même occasion)
MODKEY + D	diminuer le nombre de fenêtres dans la zone principale
MODKEY + I	augmenter le nombre de fenêtres dans la zone principale

dwm et la souris :

Bien qu'étant conçu pour être piloté intégralement au clavier, il est possible d'utiliser la souris dans dwm:

RACCOURCIS	***RESULTAT***
Clic-gauche sur symbole de l'agencement	bascule entre agencements précédents
Clic-milieu sur titre fenêtre	bascule des fenêtres depuis/vers la zone principale
Clic-milieu sur zone de status	ouvre un nouvel émulateur de terminal
Clic-gauche + MODKEY sur fenêtre	déplacer la fenêtre
Clic-droit + MODKEY sur fenêtre	redimensionner la fenêtre
Clic-milieu + MODKEY sur fenêtre	basculer la fenêtre en agencement flottant
Clic-gauche sur étiquette	voir les fenêtres possédant l'étiquette
Clic-droit sur étiquette	basculer entre les vues sélectionnées
Clic-gauche + MODKEY	attribuer l'étiquette cliquée à la fenêtre active

RACCOURCIS	***RESULTAT***
Clic-droit + MODKEY	basculer entre les étiquettes attribuées à une fenêtre

Astuces:

Attribution d'un tag à un programme:

Si vous souhaitez affecter une étiquette particulière à un programme de façon statique, cela est possible grâce aux règles de fenêtres qui se trouvent dans la partie du config.def.h

static const Rule rules[]

Une règle se définit sur le modèle suivant:

```
{ "Classe_de_la_fenêtre",    Instance,    Titre,    Masque_d'étiquette,
Flottante?,    Num_écran },
```

dans lequel la classe, l'instance et le titre sont récupérables en lançant l'utilitaire xprop et en cliquant sur la fenêtre en question. La plupart du temps la classe suffit et instance et titre ont pour valeur NULL.
Le masque d'étiquette répond à une logique particulière: 0 représente toutes les étiquettes, 1 « 2 représente la 3ème, 1 « 0 la 1ère, 1 « 7 la 8ème, etc... Le paramètre Flottante (isfloating dans le texte) prend pour valeur True ou False selon si la fenêtre doit nativement être flottante ou pas. Enfin Num_écran est le numéro de l'écran sur lequel s'affichera la fenêtre. “-1” représente l'écran actif.

Exemple:

```
static const Rule rules[] = {
/* class        instance    title       tags mask     isfloating   monitor
*/
{ "Dwb",              NULL,       NULL,       1 << 1,       False,       -1
},
{ "Iceweasel",        NULL,       NULL,       1 << 1,       False,       -1
},
{ "Gimp",             NULL,       NULL,       1 << 3,       False,       -1
},
{ "Inkscape",         NULL,       NULL,       1 << 3,       False,       -1
},
};
```

Toutes les fenêtres ayant pour classe Dwb s'ouvriront en ayant l'étiquette 2.

Ajout commande shell et de son raccourci clavier:

Il est bien entendu possible, et indispensable d'ailleurs, de paramétrer vos propres raccourcis clavier pour exécuter un programme, un script etc... En voici la syntaxe:

```
  { ControlMask|Mod1Mask,          XK_i,           spawn,
```

```
SHCMD("dwb") },
  { ControlMask|Mod1Mask,          XK_m,          spawn,          SHCMD("st
-e mutt -y") },
  { ControlMask|Mod1Mask,          XK_p,          spawn,          SHCMD("st
-e mcabber") },
  { ControlMask|Mod1Mask,          XK_g,          spawn,
SHCMD("gimp") },
```

On voit que dans un premier temps on définit les touches modificatrices utilisées (ici Ctrl+Alt) associées à une touche du clavier (notée XK_et-sa-lettre). En bout de ligne, on voit que l'on utilise la fonction de dwm appelé SHCMD pour lancer le programme voulu. En somme, Ctrl+Alt+I ouvre dwb, Ctrl+Alt+M ouvre mutt dans l'émulateur de terminal st, Ctrl+Alt+P ouvre mcabber dans st également et enfin Ctrl+Alt+G lance Gimp.

Si vous souhaitez utiliser une touche non alpha-numérique (ex: ImpÉc), il faut récupérer son code avec l'utilitaire xev et le mettre à la place de XK_ et de mettre 0 à la place des modificateurs (en effet, ces touches spéciales s'emploient souvent seule).

```
  { 0,                             0xff61,         spawn,
SHCMD("scrot -z") },
```

Dans cet exemple, j'utilise la touche ImpÉc (PrtScr), pour faire une capture d'écran à l'aide de scrot. Sur mon clavier, le code de cette touche est 0xff61.

Note sur la définition de l'émulateur de terminal:

Le terminal à utiliser se définit dans le config.def.h grâce à la ligne:

```
/* commande d'exécution de l'émulateur de terminal par défaut du système */
static const char *termcmd[]  = { "x-terminal-emulator", NULL };
```

Si vous utilisez Debian, je vous recommande de laisser x-terminal-emulator. Cela vous permettra par la suite de choisir le terminal par défaut grâce au système d'"alternatives" de Debian (commande root update-alternatives –config x-terminal-emulator). Vous y gagnerez en souplesse et ne serez plus obligé de modifier la configuration de dwm et de le compiler à chaque changement.

Ajout commande static pour st tabbed:

Pour les utilisateurs de l'émulateur de terminal st et de tabbed, une interface générique à onglet, il est intéressant de définir une constante permettant de lancer l'un dans l'autre directement, sans utiliser la fonction interne SHCMD. Cela nous économise le lancement d'un processus de /bin/sh supplémentaire. Il suffit de rajouter cette ligne en dessous de celle définissant l'émulateur de terminal :

```
static const char *tabbedst[] = { "tabbed", "-c", "-r", "2", "st", "-w",
"''", NULL };
```

Modification du volume avec molette de la souris:

Il est possible de modifier le volume sonore simplement en faisant jouer la molette de la souris dans la zone de statuts de la barre de dwm. Pour cela, ajoutez les lignes suivantes à la fin du fichier de configuration, dans la section “buttons définitions”:

```
   { ClkStatusText,        0,              Button4,        spawn,
SHCMD("amixer -q set Master 2dB+ unmute") },
   { ClkStatusText,        0,              Button5,        spawn,
SHCMD("amixer -q set Master 2dB- unmute") },
```

Changer l'opacité des fenêtres avec la souris (ou le pad)

Il faudra pour que cela fonctionne que le logiciel **transset** soit installé. Il est disponible dans le paquet **x11-apps**.

```
    { ClkClientWin,         MODKEY|ControlMask,   Button4,  spawn,
SHCMD("transset -a --inc 0.1") },
    { ClkClientWin,         MODKEY|ControlMask,   Button5,  spawn,
SHCMD("transset -a --dec 0.1") },
```

Ainsi en maintenant les touches Super (si c'est celle que vous avez choisi en tant que MODKEY) et Control et en scrollant la roulette de la souris au dessus d'une fenêtre, vous pouvez augmenter ou diminuer l'opacité de celle-ci.

Présentation dwmstatus:

dwmstatus est un petit programme permettant d'afficher toutes sortes d'informations utiles dans la zone de statut de la barre. La version téléchargeable sur le site du projet suckless n'est qu'un squelette vous permettant d'en faire ce que bon vous semble. Voici le code de celui que j'utilise, il est constitué d'exemples trouvés ici et là et adapté à mon ordinateur:

```
/* Made by sogal, last update: 20141224
**
** Compile with:
** gcc -Wall -pedantic -std=c99 -lX11 status.c
*/
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <time.h>
#include <X11/Xlib.h>
#include <stdarg.h>
#include <string.h>
#include <strings.h>
#include <sys/sysinfo.h>
#include <sys/types.h>
```

```
#include <sys/stat.h>
#include <dirent.h>
#include <locale.h>
#include <fcntl.h>
#include <errno.h>
#include <math.h>

static Display *dpy;

void setstatus(char *str) {
    XStoreName(dpy, DefaultRootWindow(dpy), str);
    XSync(dpy, False);
}

char *
smprintf(char *fmt, ...)
{
  va_list fmtargs;
  char *buf = NULL;

  va_start(fmtargs, fmt);
  if (vasprintf(&buf, fmt, fmtargs) == -1){
    fprintf(stderr, "malloc vasprintf\n");
    exit(1);
  }
  va_end(fmtargs);

  return buf;
}

char*
runcmd(char* cmd) {
  FILE* fp = popen(cmd, "r");
  if (fp == NULL) return NULL;
  char ln[50];
  fgets(ln, sizeof(ln)-1, fp);
  pclose(fp);
  ln[strlen(ln)-1]='\0';
  return smprintf("%s", ln);
}

char *
loadavg(void)
{
    double avgs[3];

    if (getloadavg(avgs, 3) < 0) {
        perror("getloadavg");
        exit(1);
    }
```

```
    return smprintf("%.2f %.2f %.2f", avgs[0], avgs[1], avgs[2]);
}

char *getdatetime() {
    char *buf;
    time_t result;
    struct tm *resulttm;

    if((buf = malloc(sizeof(char)*65)) == NULL) {
        fprintf(stderr, "Cannot allocate memory for buf.\n");
        exit(1);
    }
    result = time(NULL);
    resulttm = localtime(&result);
    if(resulttm == NULL) {
        fprintf(stderr, "Error getting localtime.\n");
        exit(1);
    }
    if(!strftime(buf, sizeof(char)*65-1, "%d %h %H:%M:%S", resulttm)) {
        fprintf(stderr, "strftime is 0.\n");
        exit(1);
    }

    return buf;
}

int getbattery() {
    FILE *fd;
    int energy_now, energy_full;

    fd = fopen("/sys/class/power_supply/BAT0/energy_now", "r");
    if(fd == NULL) {
        fprintf(stderr, "Error opening energy_now.\n");
        return -1;
    }
    fscanf(fd, "%d", &energy_now);
    fclose(fd);


    fd = fopen("/sys/class/power_supply/BAT0/energy_full", "r");
    if(fd == NULL) {
        fprintf(stderr, "Error opening energy_full.\n");
        return -1;
    }
    fscanf(fd, "%d", &energy_full);
    fclose(fd);

    return (((float)energy_now / (float)energy_full) * 100);
    }

int
```

```
getvolume() {
  int volume;
  sscanf(runcmd("amixer | grep -A 6 Master | grep 'Playback'\
| grep -o '[0-9%]*%'"), "%i%%", &volume);
  return volume;

}


int main(void) {
    char *status;
    char *avgs = NULL;
    char *datetime;
    int bat0;
    int volume = 0;


    if (!(dpy = XOpenDisplay(NULL))) {
        fprintf(stderr, "Cannot open display.\n");
        return 1;
    }

    if((status = malloc(200)) == NULL)
        exit(1);


    for (;;sleep(1)) {
        avgs = loadavg();
        datetime = getdatetime();
        bat0 = getbattery();
        volume = getvolume();
        snprintf(status, 200, "%s | %s | Ω %d%% | ♫ %d%%", avgs, datetime,
bat0, volume);

        free(avgs);
        free(datetime);
        setstatus(status);
    }

    free(status);
    XCloseDisplay(dpy);

    return 0;
}
```

Il affiche la charge système, la date et l'heure, le niveau de batterie et le volume.

Il est également possible d'obtenir quelque chose de similaire sans avoir recours au C mais simplement avec un script shell:

Afin d'afficher des informations diverses dans la barre de dwm (la date, le nombre de mails...), la

procédure est extrêmement simple. En effet, dwm affiche ce que vous passez à la commande **xsetroot**. Exemple :

```
xsetroot -name "Coucou"
```

Bien sûr, vous ce qui vous intéresse c'est la date ou autres choses. Il suffit alors de lancer cette commande toute les secondes. Pour cela, avant de lancer dwm, exécutez un script shell de ce type :

```
while true; do
    xsetroot -name "$(date) $(uptime | sed 's/.*,//')"
    sleep 1
done &
```

Voici un autre exemple plus complet (qui affiche la chanson jouée, le niveau de batterie, le volume sonore et l'heure):

```
#!/bin/sh

    mus() {
        if [ "$(mpc |grep playing)" ]; then
            INFO="$(mpc |head -n1)"
            L="$(mpc |grep -Eo '[0-9]+:[0-9]+/[0-9]+:[0-9]+')"
            MUS="[ ♫ $INFO - $L] ";
        else
            MUS="";
        fi
    }

    bat() {
        BAT="[B : $(acpi -b | awk 'sub(/,/,"") {print $3, $4}')]"
    }
    vol() {
        VOL="$(amixer -c0 sget Master |grep -Eo '[0-9]+%')"
    }
    getdate() {
        DATE="$(date '+%d/%m/%y %H:%M')"
    }

    while true; do
        mus
        bat
        vol
        getdate
        xsetroot -name "$MUS $BAT [Vol:$VOL] [$DATE]"
        sleep 2s
    done
```

Merci à Thuban pour cette remarque et pour cet exemple.

Conclusion:

dwm est un excellent gestionnaire de fenêtre, je l'utilise au quotidien et l'apprécie particulièrement. La gestion des fenêtres est très bonne, grâce aux commandes clavier, il est extrêmement simple de les déplacer, de les étiquetter, de les rappeler, de les basculer d'un écran à l'autre. dwm est très sobre en ressources et d'une stabilité à toute épreuve. Sa mise en place est un peu délicate et peut rebuter toutefois (configuration pré-compilation, manipulation de fichiers source en C, dénuement apparent). J'espère que ce tutoriel et les sources dont il s'inspire saura vous aider à l'apprivoiser.

— *Sogal 'Seb' Punx 18/01/2015 17:24*

Sources: http://dwm.suckless.org/tutorial
http://yeuxdelibad.net/Logiciel-libre/Suckless/dwm/index.html http://dwm.suckless.org/dwmstatus/

[1)]

N'hésitez pas à y faire part de vos remarques, succès, améliorations ou échecs !