RESSOURCES

Ressources: Définition

Une ressource est un champ de la structure définissant un widget. (Dans les langages orientés objet, on parle de membre ou d'attribut d'objet).

Exemple:

```
typedef struct  {
        Pixel           foreground;
        Dimension       width;
        Dimension       height;
        ...
} *LabelWidget;
```

Les ressources d'un widget ont toujours une valeur par défaut, mais on peut modifier cette valeur en de nombreux endroits, via des fichiers de ressources, la ligne de commande ou le code.
Les ressources par défaut sont gérées via une base de ressources et un ressource Manager au niveau du serveur X.

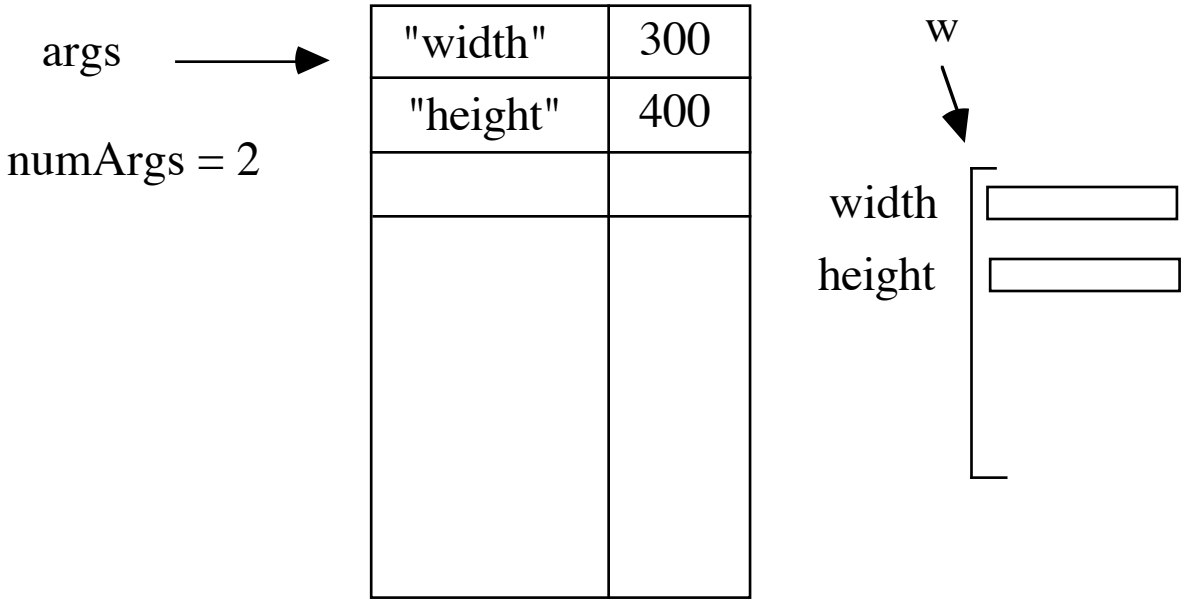
Ressources affectées à la création des widgets à l'aide d'une liste d'arguments (fonctions MOTIF)

```
typedef struct {
        String          name;
        XtArgVal        value;
}  Arg;

#define XtSetArg (a, n, v)  a.name = n; a.value = v
```

- Permet de modifier une structure Arg a

```
Widget create_button (Widget parent)
{       Arg         args[10];
        int         n = 0;

        XtSetArg (arg[n],XmNwidth, 300); n++;
        XtSetArg (arg[n],XmNheight, 400); n++;
        return XmCreatePushButton (parent, “button”, args, n);
}
```

Listes d'arguments: fonctionnement interne



Ressources: Modification après création

```
XtVaSetValues (    widget,
                   resourceName1, val1,
                   ...
                   resourceNameN, valN,
                   NULL)
```

• Modifie les paramètres d'un widget après sa création. (Ce n'est pas toujours possible).

```
Widget create_button (Widget parent)
{
    Widget widget;

    widget = XmCreatePushButton(parent, "button", NULL, 0);
    XtVaSetValues (widget,
                   XmNwidth, 300,
                   XmNheight, 400,
                   NULL);
    return widget;
}
```

Conversion explicite avec XtVaTypedArg

XtVaSetValues (widget,resourceName1, value1,
resourceName2, value2,
XtVaTypedArg, resourceName, resourceType, value, size,
...
resourceNameN, valueN, NULL)

- XtVaTypeArg permet de spécifier une valeur de ressource dans un autre type que le type interne de la ressource (en général on utilise le type String).

Exemple:

XtVaSetValues(widget, XmNwidth, 200,
XtVaTypedArg, XmNforeground, XmRString, “Blue”, 5,
XtVaTypedArg, XmNfontList,XmRString,“10x20”, 6 ,
NULL);

Ressources: Récupération des valeurs

```
XtVaGetValues (widget,      ressourceName1, &val1,
                                ...
                            ressourceNameN, &valN,
                            NULL)
```

- Récupère la valeur de n'importe quel paramètre d'un widget.

```
void print_size (Widget widget)
{
    Dimension    width, height;

    XtVaGetValues (widget,
                        XmNwidth, &width,
                        XmNheight, &height,
                        NULL);
    printf("width: %d height: %d", width, height);
}
```

Ressources

Une ressource peut être affectée à un widget de différentes façons et par différentes personnes.

- Un fichier ASCII de ressources propre à chaque utilisateur (.XDefaults par ex.)

- Des arguments dans la ligne de commande d'appel de l'application (options définies par l'utilisateur).

- Dans le code de l'application.

Ressources: Spécifier les valeurs

Fichier ASCII de spécification de ressources:

```
test*okbutton.fontList:          helvetica12b
test*okbutton.background:        Navy Blue
```

Ressources spécifiées dans la ligne de commande:

```
% test  -fn  times12  -bg  grey
```

Ressources spécifiées avec une liste d'arguments à la création :

```
CreateButton (Widget parent, XmFontList fp, Pixel bg)
{     Arg       args [2];
      int       n =0;

      XtSetArg (args[n], XmNfontList, fp); n++;
      XtSetArg (args[n], XmNbackground, bg); n++;
      XmCreatePushButton(parent, “okbutton”, args, n);
}
```

Ressources: syntaxe des fichiers de ressources

Spécification de ressources dans un fichier texte (ex.: .XDefaults)

1. Chaque champ décrit un widget (par son nom) ou une classe de widgets.

2. Chaque champ est séparé des autres par un point ou un astérisque.
   L'astérisque permet d'éluder un ensemble de widgets dans l'arborescence des instances de widgets du programme.

Exemple:

```
Test.form.okbutton.labelString:        OK
Test.form.XmPushButton.fontList:       12x24
Test*okbutton.labelString:             OK
Test*background:                       red
```

Ressources: Définition des options de commande

```
typedef struct {
        char            *option;        /* (abréviation de) l'option dans argv */
        char            *specifier;     /* specifieur de ressource */
        XrmOptionKind   argKind;        /* style d'option */
        XPointer        value;          /* valeur  si style  XrmoptionNoArg */
}       XrmOptionDescRec;
```

Cette structure permet d'affecter les ressources d'une application à partir de la ligne de commande, par exemple:

```
static XrmOptionDescRec     options [] = {
        {"-rv", "*reverse", XrmoptionNoArg,  "True"},
        {"-fn", "*XmPushButton*fontList",     XrmoptionSepArg, NULL},
};

toplevel = XtVaAppInitialize (&app, "Test", options, 2, &argc, argv, NULL, NULL);

% test -rv -fn times12
```

Les styles d'options

Les différents styles d'options reconnues sont:

XrmoptionNoArg	La valeur est dans le champ value.
XrmoptionIsArg	La valeur est dans le champ option. % foo -rv # valeur = -rv
XrmoptionStickyArg	La valeur est collée à l'option % foo -debug5 #valeur = 5
XrmoptionSepArg	La valeur est dans le prochain argument % foo -fg red #valeur = red
XrmoptionSkipArg	Ignore l'option et l'argument suivant
XrmoptionSkipLine	Ignore l'option et le reste de la ligne de commande

Les options standards

Des options de la ligne de commande sont reconnues en standard, par exemple :

OPTION	RESSOURCES AFFECTEES
-bg	*background
-bd	*borderColor
-fg	*foreground
-fn	*font
-geometry	*TopLevelShell.geometry
-title	*title
-display (ou –d)	*display
-synchronous	*synchronous

% programme -xrm "*fontList: 10x20" -geometry 500x250+10+30 -bg white

Ressources: Chargement

Algorithme (approximatif) pour charger les ressources lors de l'initialisation de la toolkit :

1. Charger /usr/lib/X11/app-defaults/<CLASS>
2. Charger $XAPPLRESDIR/<CLASS>
3. Charger les ressources de la propriété RESOURCE_MANAGER de la racine
   (affectée par la commande xrdb).
4. Si la propriété RESOURCE_MANAGER n'est pas définie,
   charger $HOME/.Xdefaults
5. Charger le fichier $XENVIRONMENT
6. Si $XENVIRONMENT est vide
   charger $HOME/.Xdefaults-<HOST>
7. Charger les options de la ligne de commande

Ressources de la classe CORE (héritées par tous les widgets)

Nom	Type	Défaut	Accès/Modification
XmNx	Position	0	CSG
XmNy	Position	0	CSG
XmNwidth	Dimension	dynamic	CSG
XmNheight	Dimension	dynamic	CSG
XmNdepth	int	dynamic	CG
XmNbackground	Pixel	White	CSG
XmNbackgroundPixmap	Pixmap	non spec.	CSG
XmNborderWidth	int	1	CSG
XmNborderColor	Pixel	Black	CSG
XmNborderPixmap	Pixmap	non spec.	CSG
XmNcolormap	Colormap	dynamic	CG
XmNsensitive	Boolean	True	CSG
XmNdestroyCallback	XtCallbackList	NULL	C
XmNtranslations	XtTranslations	dynamic	CSG

Accès/Modification: C = Create (Création) S = Set (Affectation) G = Get (Récupération)

Les ressources de contraintes

• Les ressources de contraintes (*constraint resources*) sont définies par la classe Constraint, superclasse de Manager (dans Xt)

• A la différence des autres ressources, **les ressources de contraintes s'appliquent aux descendants** du manager widget parent dans la hiérarchie des instances de widget

• Lorsqu'un widget est créé comme le fils d'un widget dont Constraint est une super classe, les ressources de contraintes définies par le père sont ajoutées à l'ensemble des ressources reconnues par le fils

• Par exemple, hauteur et largeur minimum et maximum, tailles et positions relatives

Les ressources de contraintes : exemple

• Un widget de la classe XmLabel fils d'un widget de la classe XmPanedWindow va posséder les ressources de la classe XmLabel (XmNlabelString, etc.) ainsi que les ressources de contraintes de XmPanedWindow (XmNpaneMinimum, XmNpaneMaximum):

```
Create_paned (Widget top)
{
      Widget paned, label;

      paned = XtVaCreateManagedWidget ("paned",
                        xmPanedWindowWidgetClass,  top, NULL);

      label = XtVaCreateManagedWidget ("label", xmLabelWidgetClass,
                  paned,
                  XmNlabelString, XmCreateSimple("hello world"),
                  XmNpaneMinimum, 100,
                  XmNpaneMaximum, 200, NULL);
}
```

Le widget XmForm: ressource de contraintes

• L'attachement d'un widget fils d'un XmForm est déterminé pour chaque côté du widget par les quatre ressources de contraintes définies dans le XmForm :

XmNtopAttachment	politique d'attachement du côté haut
XmNbottomAttachment	politique d'attachement du côté bas
XmNleftAttachment	politique d'attachement du côté gauche
XmNrightAttachment	politique d'attachement du côté droite

• Les ressources de contraintes

XmNtopWidget	XmNtopOffset	XmNtopPosition
XmNbottomWidget	XmNbottomOffset	XmNbottomPosition
XmNleftWidget	XmNleftOffset	XmNleftPosition
XmNrightWidget	XmNrightOffset	XmNrightPosition

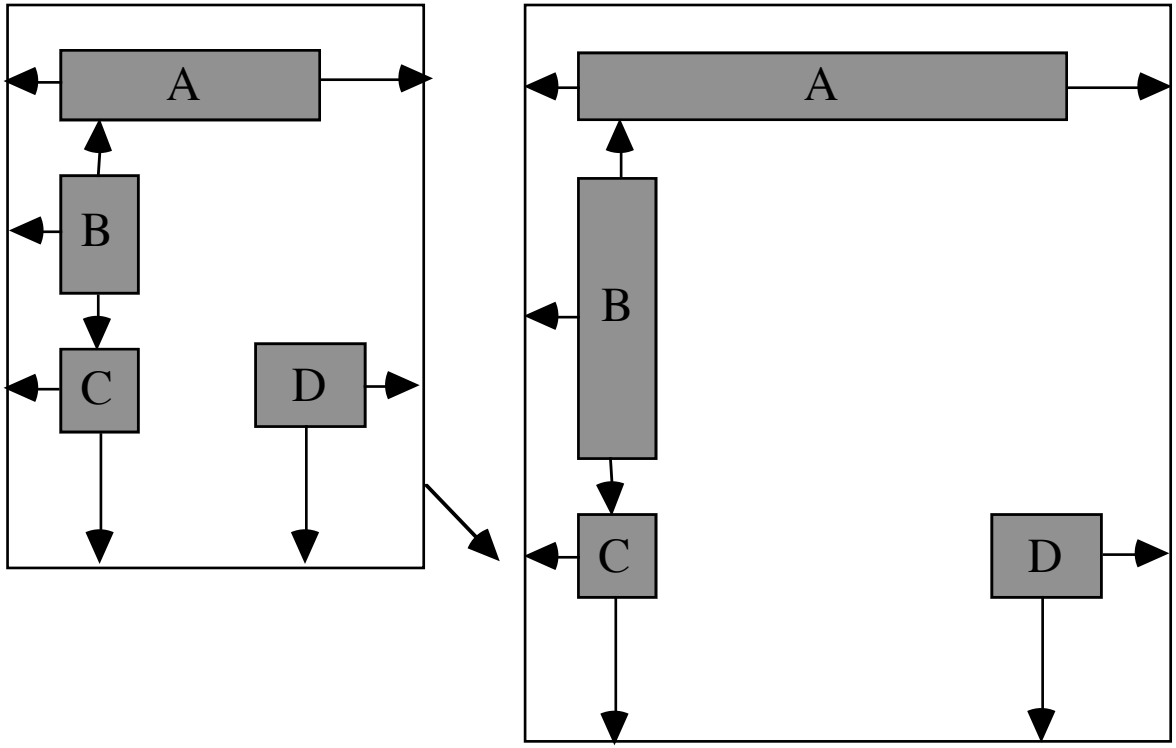
sont utilisées (éventuellement) pour compléter l'information relative à l'attachement d'un côté donné.

Les fils d'un widget XmForm: ressources de contraintes héritées

Le tableau ci-dessous présente les différentes valeurs des politiques d'attachement attribuables à un côté, et indique les ressources obligatoires (**+**), optionnelles (?) ou sans effet ( ) que l'on peut (ou doit) ajouter pour compléter l'information pour cette valeur d'attachement.

XmN<côté>Attachment	XmN<côté>Offset	XmN<côté>Widget	XmN<côté>Position
XmATTACH_FORM	?		
XmATTACH_OPPOSITE_FORM	?		
XmATTACH_WIDGET	?	**+**	
XmATTACH_OPPOSITE_WIDGET	?	**+**	
XmATTACH_NONE			
XmATTACH_SELF			**+**
XmATTACH_POSITION	?		**+**

Politique XmATTACH_FORM et XmATTACH_WIDGET : exemple



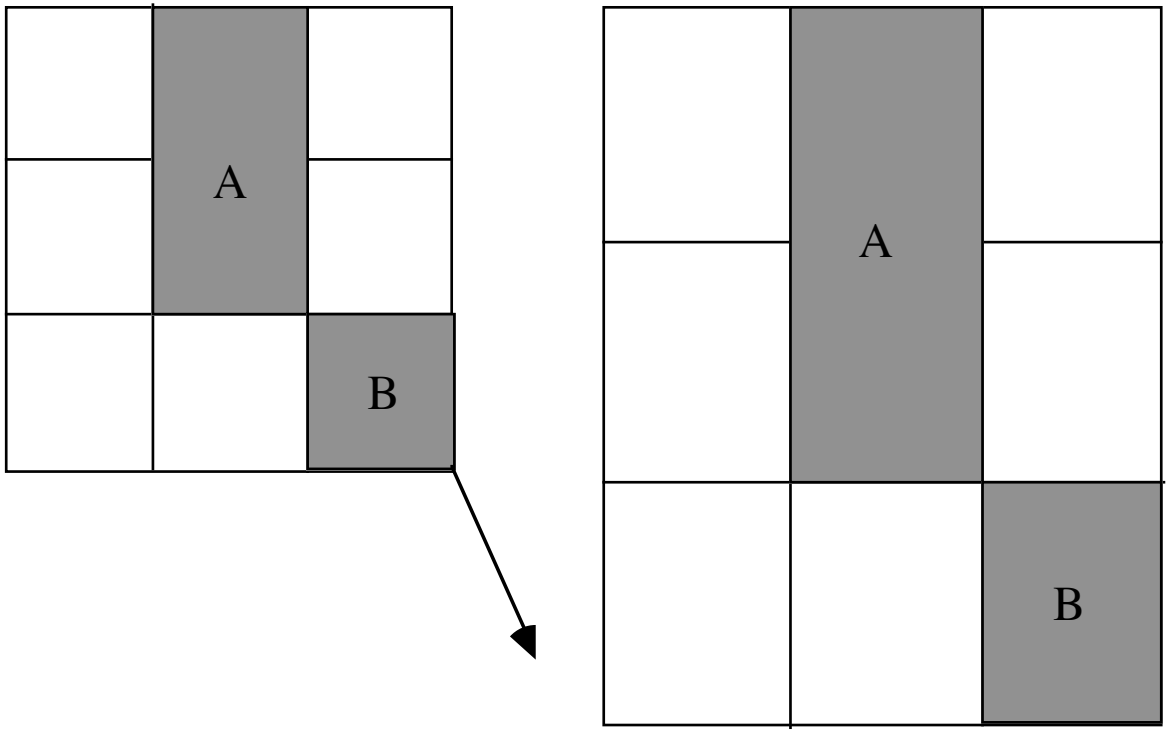
Widget A: XmNleftAttachment et XmRightAttachment: XmATTACH_FORM
XmNleftOffset: 10
XmNrightOffset: 20

Widget B: XmNtopAttachment et XmNbottomAttachment: XmATTACH_WIDGET
XmNtopWidget: Widget A
XmNtopOffset: 10
XmNbottomWidget: Widget C
XmNbottomOffset: 10
XmNleftAttachment: XmATTACH_FORM
XmNleftOffset: 10

Widget C: XmNleftAttachment et XmNbottomAttachment: XmATTACH_FORM
XmNleftOffset: 10
XmNbottomOffset: 30

Widget D: XmNrightAttachment et XmNbottomAttachment: XmATTACH_FORM
XmNrightOffset: 10
XmNbottomOffset: 30

Politique XmATTACH_POSITION : exemple



Le widget XmForm: XmATTACH_POSITION

La ressource XmNfractionBase du XmForm contrôle le nombre de lignes et de colonnes d'une grille. Par défaut, c'est la valeur 100.

Form: XmNfractionBase: 3

Widget A: XmNtopAttachment, XmNbottomAttachment, XmNleftAttachment et XmNrightAttachment : XmATTACH_POSITION
XmNtopPosition: 0
XmNleftPosition: 1
XmNrightPosition: 2
XmNbottomPosition: 2

Widget B: XmNtopAttachment, XmNbottomAttachment, XmNleftAttachment, et XmNrightAttachment: XmATTACH_POSITION
XmNtopPosition: 2
XmNleftPosition: 2
XmNrightPosition: 3
XmNbottomPosition: 3

Les chaînes composées MOTIF : type XmString

La toolkit MOTIF propose des fonctions spéciales de création de chaînes.

```
XmString  XmStringCreateSimple (text)      /*  obsolète   */
XmString  XmStringCreateLocalized (text)
      char*       text;
```

Exemple:

```
XmString str;
str = XmStringCreateLocalized ("Chaine composee");
```

Pour libérer une chaîne composée :

```
XmStringFree (str);
```

XmString : création et manipulation

```
fontList: -*-helvetica-medium-r-normal-*-12-*-*-*-*-*-iso8859-1, \
 -*-helvetica-medium-o-normal-*-12-*-*-*-*-*-iso8859-1=italic, \
 -*-helvetica-bold-r-normal-*-12-*-*-*-*-*-iso8859-1=bold
```

Les fontes 2 et 3 sont dites "taguées" respectivement "italic" et "bold". La première est la fonte par défaut. On peut ensuite utiliser les arguments tag de type char* pour créer des chaînes composées utilisant ces fontes.

```
XmString      XmStringCreate(char *text, char *tag)
XmString      XmStringCreateLocalized(char *text)
XmString      XmStringCreateLtoR(char *text, char *tag)

Boolean  XmStringCompare(XmString str1, XmString str2)
Boolean  XmStringEmpty(XmString str1)
```

Les chaînes composées MOTIF: segments

L'intérêt des chaînes composées est d'avoir des messages utilisant différentes fontes.

Une chaîne composée est constituée de segments dans différentes polices.

Chaque segment comporte:

- un texte
- un jeu de caractères (associé à une fonte).

Pour créer un segment on utilise la fonction

```
XmString      XmStringSegmentCreate (string, charset, direction, separator)
              char*                 string;
              XmStringCharset       charset;
              XmStringDirection     direction;
              Boolean               separator;
```

Les chaînes composées MOTIF: utilisation

Pour avoir la chaîne composée: “ceci est *penché* ” avec penché en italique:

```
XmString            s1, s2, string;
s1 = XmStringSegmentCreate (“ceci est “, “normal”,
                    XmSTRING_DIRECTION_L_TO_R, False);
s2 = XmStringSegmentCreate (“penché“, “italique”,
                    XmSTRING_DIRECTION_L_TO_R, True);
string = XmStringConcat (s1, s2);
XmStringFree (s1);
XmStringFree (s2);
```

Pour obtenir l’effet souhaité, il suffit d’associer la fonte helvetica12 aux caractères normaux et la fonte helvetica12i aux caractères italiques:

```
test*fontList:        helvetica12=normal, helvetica12i=italique
```

Les chaînes composées MOTIF: utilisation

Boolean XmStringGetLtoR (XmString str,
XmStringCharSet tag, char** text)

Cette fonction permet de récupérer le texte (en char *) associé à une chaîne dans une fonte donnée.

On peut appeler cette fonction avec XmFONTLIST_DEFAULT_TAG pour deuxième argument. Dans ce cas, la fonte correspondra à la fonte par défaut, i.e. celle utilisée pour la création avec XmStringCreateLocalized (ou XmStringCreateSimple).

Pour le cas général, si on veut récupérer le texte associé à une XmString, il faut utiliser un XmStringContext et la fonction XmStringGetNextSegment (cf. programme donné en cours).