Architecture de X Window

Christian Nguyen

Département d'informatique
Université de Toulon et du Var

Historique

1984 Origine du projet ATHENA (MIT),

1986 Version X10R4. HP et DEC décident de construire des stations “X”,

1987 Version X11R1,

1988 Version X11R2. Création du Consortium X : Apollo, Apple, ATT, DEC, HP, SUN, Tektronix, IBM, ...,

⋮

2010 Version X11R7.6, version actuelle.

Objectifs

- système de construction et de manipulation de fenêtres pour tout terminal « image »,
- indépendant du matériel,
- utilisable à travers un réseau d'ordinateurs hétérogènes,
- exécution simultanée d'applications différentes,
- utilisation de différents gestionnaires d'interfaces graphiques,
- manipulation d'une hiérarchie de fenêtres,
- outils performants pour la création de graphiques 2D, d'images et de textes,
- système ouvert et bien documenté,
- système extensible.

Le display

Un *display* `:0.0` comprend :

- une unité centrale,
- un ou plusieurs écrans (*screens*),
- un clavier,
- un dispositif de pointage (souris),
- un protocole de communication.

Le modèle client-serveur

Le client : c'est une application quelconque (dont le gestionnaire de fenêtre).

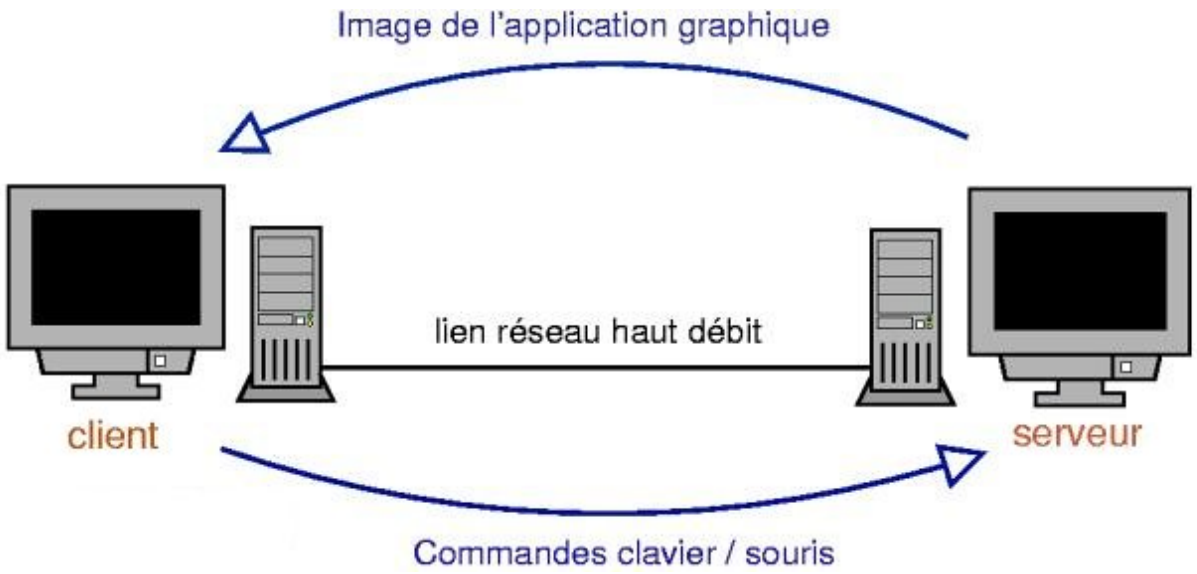
Le rôle du serveur : gérer les ressources locales de la machine (*display*) pour le compte d'un ou plusieurs clients :

- gère l'accès simultané de plusieurs clients au *display*,
- exécute les requêtes de dessin de base (segments, rectangles, ellipses),
- gère des structures de données complexes (CG, polices, curseurs, ...),
- interprète les messages émis par les clients (à travers le réseau),
- transmet des événements aux clients (à travers le réseau).

Le modèle client-serveur

Communication client-serveur : les applications communiquent avec le serveur au moyens d'appels à une bibliothèque de procédures écrites en C, la *Xlib*.

Identification d'une station : chaque station possède un nom. Un client entre en communication avec le serveur pour utiliser une station, ce client pouvant être sur une machine différente du serveur.



Le protocole X

Il existe quatre types de messages :

- requêtes du client (par exemple, créer une nouvelle fenêtre),
- réponse du serveur (à une requête),
- événement envoyé par le serveur (suite à une action de l'utilisateur),
- erreurs (fatales ou récupérables).

Le protocole X ne suppose aucun protocole réseau et ne dépend ni du réseau (RS232, Ethernet, Token Ring, ...), ni du système (Unix, VMS, ...).

Il est performant : un aller-retour entre 5 et 50 ms (à 200 Ko/s).

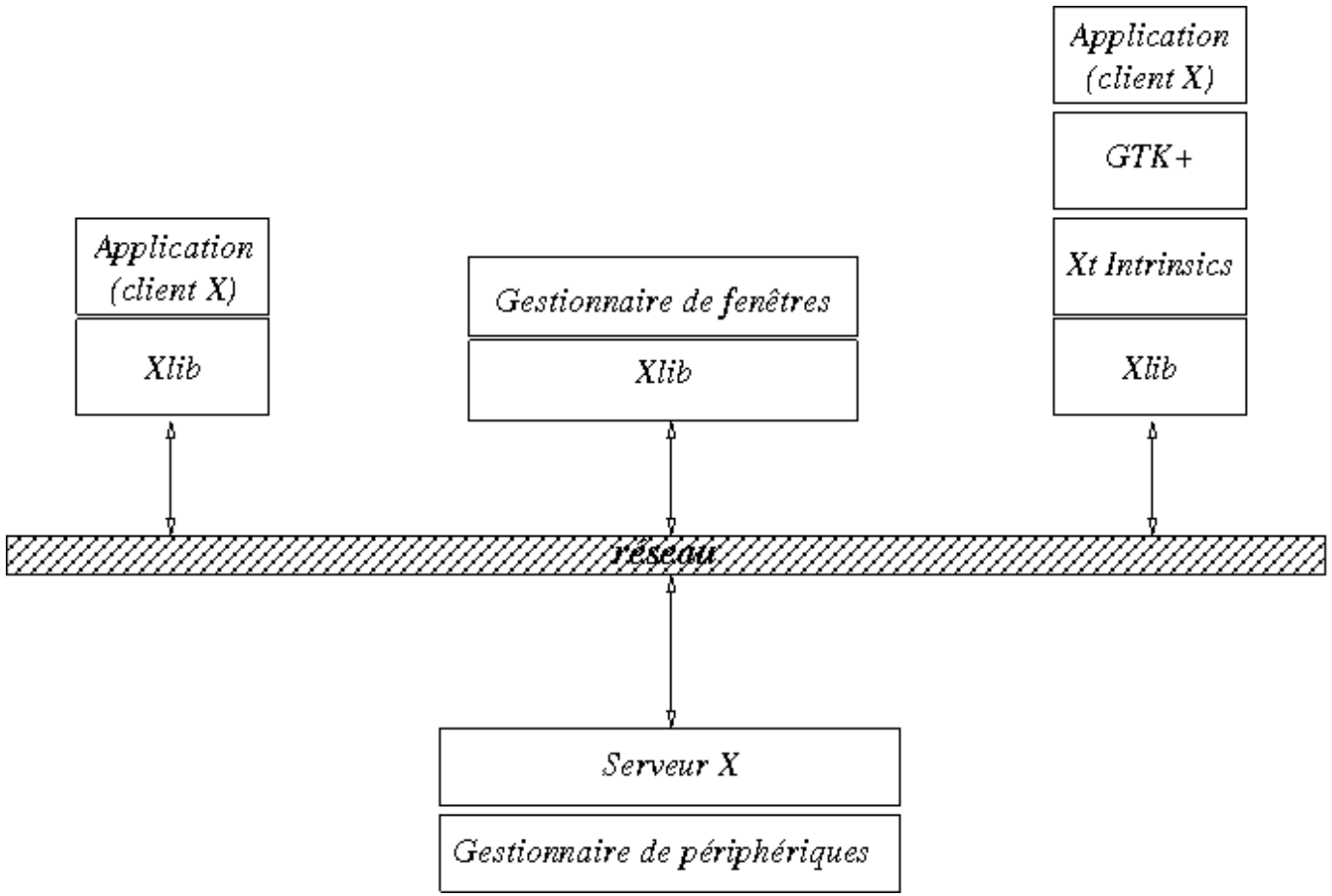
Gestion de requêtes

Ces performances imposent un fonctionnement asynchrone.

La Xlib sauvegarde les requêtes dans un buffer et envoie celles-ci dans un des trois cas suivants :

- une application a lancé une procédure de la Xlib qui attend un événement qui n'est pas encore arrivé,
- une requête nécessite une réponse immédiate,
- le client demande explicitement de vider le buffer.

Clients et serveur X



Clients et serveur X

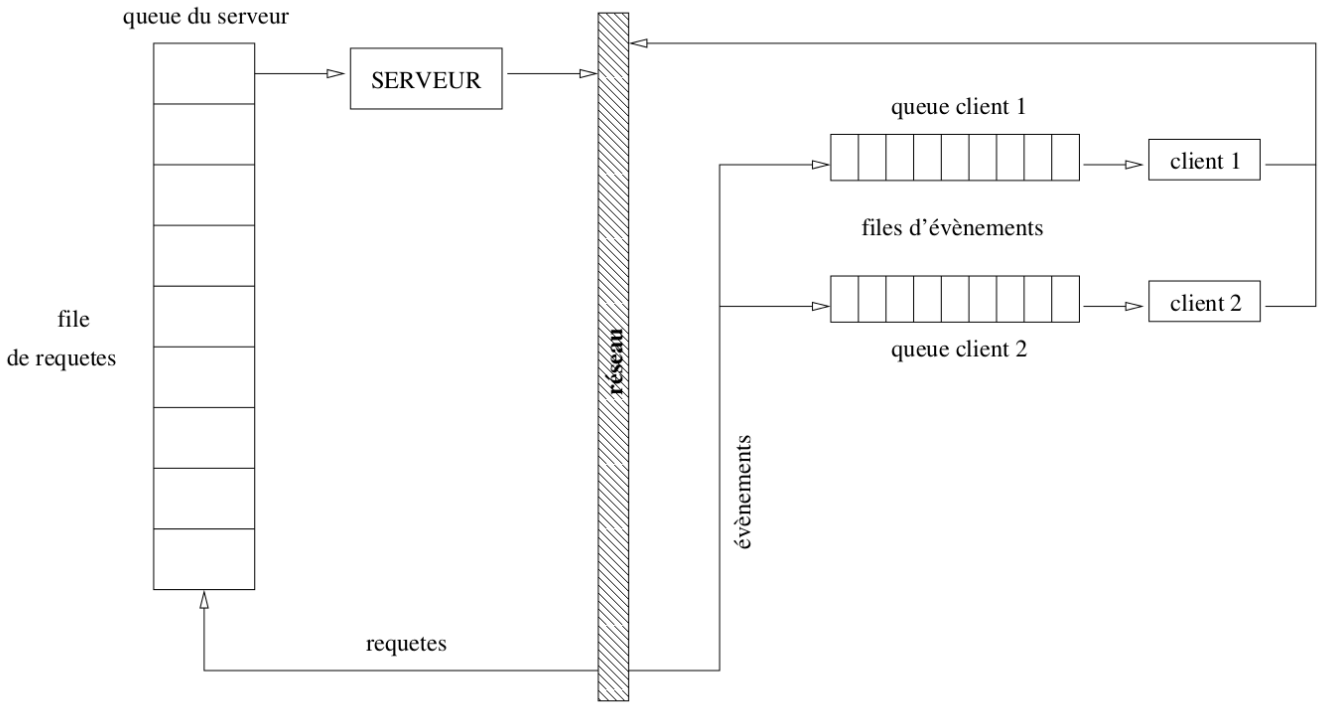
L'activation d'un client X comprend les huits étapes suivantes :

1. connexion avec le serveur (`XOpenDisplay`),
2. création d'une fenêtre mère (`XCreateWindow`),
3. création des attributs de cette fenêtre,
4. création du contexte graphique (CG : stocké sur le serveur, permet de réduire la quantité d'informations à transmettre, il peut y avoir plusieurs CG par serveur),
5. création des autres sous-fenêtres,
6. sélection des événements possibles dans chaque sous-fenêtres,
7. affichage des fenêtres (`XMapWindow`),
8. entrée dans une boucle d'attente d'événements (*event-driven*).

Clients et serveur X

Principes de fonctionnement :

- toute information échangée entre le serveur et un client prend la forme d'événements,
- un événement est transmis via le canal de communication,
- le serveur gère une file (FIFO) pour chaque client,
- tout événement est rattaché à une fenêtre.

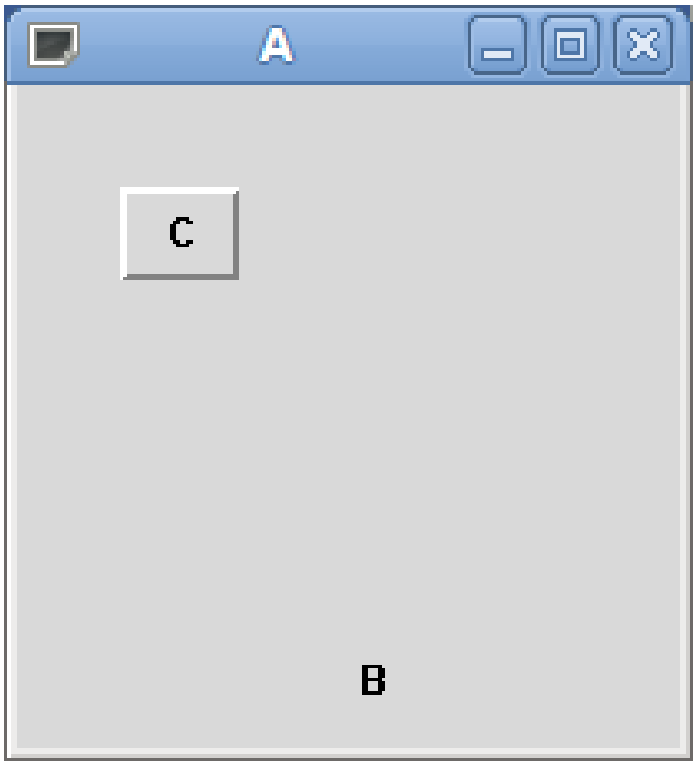


Les évenements

Types d'événements :

- événements matériels (*pointermotion, pointerbutton, keypressed, keyreleased*, ...),
- événements de changement (*enterwindow, leavewindow*, ...),
- événements géométriques (*createnotify, destroynotify, resizenotify, expose*, ...),
- événements du gestionnaire de fenêtres (*window manager : selectionrequest*, ...).

Propagation des événements.



Hiérarchie de trois fenêtres A, B et C.

Les évenements

La configuration est la suivante :

C `event_mask = 0; do_not_propagate_mask = 0;`

B
```
event_mask = (KeyPressMask | PointerMotionMask);
do_not_propagate_mask = 0;
```

A
```
event_mask = (ButtonPressMask | ButtonPressRelease);
do_not_propagate_mask = 0;
```

1. aucun événement n'est pas pris en compte dans la fenêtre C mais il est propagé,
2. l'événement `ButtonPress` n'est pas pris en compte dans B mais il est propagé,
3. il est pris en compte dans A et mis dans la file d'attente.

Le gestionnaire de fenêtres

Le *window manager* est une application qui permet d'afficher, de déplacer, de détruire, d'icônifier les fenêtres, de les redimensionner et de les décorer. Il ne gère que les fenêtres de niveau 0 (*RootWindow*) et 1.

Le protocole X ne spécifie aucun style d'interface. Il offre simplement les possibilités standards suivantes :

- opération "couper-coller",
- sélection courante,
- transfert de sélection,
- communication.

Chaque utilisateur peut donc utiliser son propre style d'interface : mwm (OSF/Motif), CDE (Solaris), enlightenment, Xfce, ...

Les bibliothèques

Les *toolkits* sont des bibliothèques contenant des fonctions toutes faites servant à dessiner et à gérer des widgets de plus ou moins haut niveau.

Par exemple, *GTK+* est un toolkit.

Il est à noter que le window manager gère *toutes* les fenêtres indépendamment des applications, alors que les toolkits dépendent du programme.

Le bureau virtuel

Le *desktop* permet de manipuler les fichiers, par exemple *Gnome*.

Sans desktop, pas de bureau virtuel, pas d'icône, pas de poubelle, si ce n'est une fenêtre dans laquelle tourne un shell.

- un programme utilisant n'importe quel *toolkit* peut voir ses fenêtres décorées par n'importe quel *window manager* et ses fichiers gérés par n'importe quel *desktop*.

Une session

Le terme session désigne le cycle lancement, travail et sortie.

Le lancement de X Window dépend de `PATH` qui doit contenir le chemin d'accés au serveur X.

Il y a deux manières de lancer X :

1. exécuter le script `startx` au démarrage, qui se charge d'initialiser quelques variables et de passer les options adéquates au serveur X par `xinit`,
2. utiliser un gestionnaire de sessions (`xdm` pour *X daemon manager*) lancé dès l'amorçage du système et qui lance X sans aucun environnement.

Connaître son environnement, c'est identifier le serveur (`machine:display.screen`) et le gestionnaire de fenêtre et savoir comment se lancent les clients.

Une session

Que ce soit par le shell ou par `xdm`, le script approprié lit le fichier `/etc/X11/Xresources`, puis celui de l'utilisateur `$HOME/.Xresources` via la commande `xrdb`.

Il reconfigure ensuite le clavier si le fichier `$HOME/.Xmodmap` est présent via la commande `xmodmap`.

Enfin il contient à la fin de celui-ci une séquence de lancement de clients communs (qui peut dépendre du gestionnaire de fenêtres) puis propre à chaque utilisateur (définit dans le fichier `$HOME/.xsession` ou `$HOME/.Xclients`)

.xsession/.xinitrc

```
#!/bin/bash
# .xsession/.xinitrc
# choose a window manager
defaultwm=kde
#set the window manager to $1 if it was supplied
windowmgr=${1:-$defaultwm}
#start the respective window managers
case ${windowmgr} in
  kde|kwm|kdestart)
        WINDOWMANAGER=startkde
        ;;
  fvwm|fvwm2)
        WINDOWMANAGER=fvwm2
        ;;
   *)    WINDOWMANAGER=windowmgr   # default for unknown wm's
esac
```

.xsession/.xinitrc

```
# load resources
if [ -f /usr/X11R6/lib/X11/Xmodmap ]; then
    xmodmap /usr/X11R6/lib/X11/Xmodmap
fi
 if [ -f ~/.Xmodmap ]; then
    xmodmap ~/.Xmodmap
fi
 if [ -f ~/.Xdefaults ]; then
    xrdb -merge ~/.Xdefaults
fi
 if [ -f ~/.Xresources ]; then
    xrdb -merge ~/.Xresources
fi
# finally start the window manager
exec $WINDOWMANAGER
```

La sécurité

Il y a deux façons de restreindre l'accès à un serveur X.

La première ne concerne que les machines sur lesquelles tournent les clients. Elle ne distingue pas plusieurs utilisateurs d'une même machine. Elle est donc très rudimentaire et elle est mise en œuvre par le client `xhost`.

La seconde concerne vraiment les utilisateurs et offre plusieurs niveaux de sécurité. Elle utilise le fichier `.Xauthority` qui est manipulé par la commande `xauth`.

Autorisation basée sur les machines

Le serveur X gère une liste de machines à partir desquelles les clients sont autorisés à se connecter au serveur X, manipulée par le client `xhost` :

- `xhost` : affiche la liste des machines.
- `xhost +` : mécanisme d'autorisation désactivé, tous les clients sont autorisés à se connecter.
- `xhost -` : mécanisme d'autorisation activé, seuls les clients dans la liste sont autorisés à se connecter.
- `xhost +machine` : ajoute la machine à la liste.
- `xhost -machine` : supprime la machine de la liste.

Autorisation basée sur l'authentification

Méthode basée sur l'authentification des clients qui doivent s'identifier en se connectant au serveur X.

Offre 4 modes de fonctionnement, dont certains donnent un niveau de sécurité élevé même s'ils sont lourds à mettre en œuvre.

Les certificats d'authorisation sont stockés dans le fichier `.Xauthority` qui est manipulé par la commande xauth (protégé en lecture).

.Xauthority

Contient des clés, qui ne sont pas codées en clair, permettant de se connecter. Le client `xauth` permet d'afficher son contenu, d'en extraire des clés et d'en ajouter :

- `xauth list` : liste le contenu du fichier .Xauthority de manière compréhensible.
- `xauth [n]extract file display` : extrait du fichier .Xauthority et place dans le fichier `file` les clés correspondantes au serveur `display`.
- `xauth [n]merge file` : ajoute au fichier .Xauthority les clés contenues dans le fichier `file`.

Autorisation basée sur l'authentification

Le serveur X, lancé avec l'option `-auth`, mémorise les clés du fichier passé en paramètre et n'accepte que les connexions des clients proposant une clé identique à l'une des clés mémorisées.

Les clients utilisent également le fichier `$HOME/.Xauthority` : la connexion d'un client lancé par un autre utilisateur sera acceptée si cet utilisateur a les clés du serveur X dans son propre `$HOME/.Xauthority` (possibilité quasi nulle si on change de clé aléatoirement à chaque lancement du serveur).

Par contre si on veut volontairement autoriser un autre utilisateur à dialoguer avec son serveur X, il faut lui transmettre les clés par la commande :

```
xauth extract - $DISPLAY | xauth -f ~user/.Xauthority merge -
```

Cas particulier

Lancer un client sur une autre machine sur laquelle on a pas le même HOME directory, la connection sera refusée car le client n'aura pas le même .Xauthority que le serveur.

Deux solutions : toujours utiliser la même clé, ou transmettre à chaque fois la clé par une commande du type :

```
xauth extract - $DISPLAY | ssh otherhost xauth merge -
```

Les ressources de X

Toutes les informations associées aux fenêtres, aux pixmaps, aux polices de caractères, aux tables de couleurs et aux contextes graphiques sont stockées au niveau du serveur.

Les clients peuvent y référer via un identificateur, unique par ressource, sous la forme d'un entier sur 32 bits (dont les 3 bits de poids fort sont à 0).

Chaque client a son propre ensemble d'identifiants, dans lequel il peut « piocher » pour créer de nouvelles ressources.

Ces identifiants sont uniques au niveau du serveur. Un client peut donc accéder à n'importe quelle ressource via son identificateur.

Exemple : le programme Ghostview crée une fenêtre, stocke son identifiant dans une variable d'environnement, puis lance le programme Ghostscript ; ce dernier affiche des données PostScript dans cette fenêtre.

Les ressources de X

Le mécanisme des ressources permet de modeler une interface selon les envies et les besoins de l'utilisateur.

Elles utilisent pour cela les notions de classe et d'instance (commande `xprop`, champ `WM_CLASS`).

Paramétrage en ligne de commande ou par le fichier de ressources .Xresources. Exemple :

```
emacs -geometry 80x25+0+0&
Emacs.geometry : 80x25+0+0
```

Prise en compte en redémarrant X ou bien en utilisant le client xrdb.

Les ressources de X

Chaque élément d'interface (bouton, zone de texte, ...) dispose d'un nom de classe, d'un nom d'instance et d'un ensemble de ressources associées.

Modification de la configuration par défaut d'un widget particulier de nom `mon_widget` :

```
Client.widget_parent1.widget_parent2.mon_widget.ressource : valeur
```

Le symbole "*" permet de remplacer un ou plusieurs widgets :

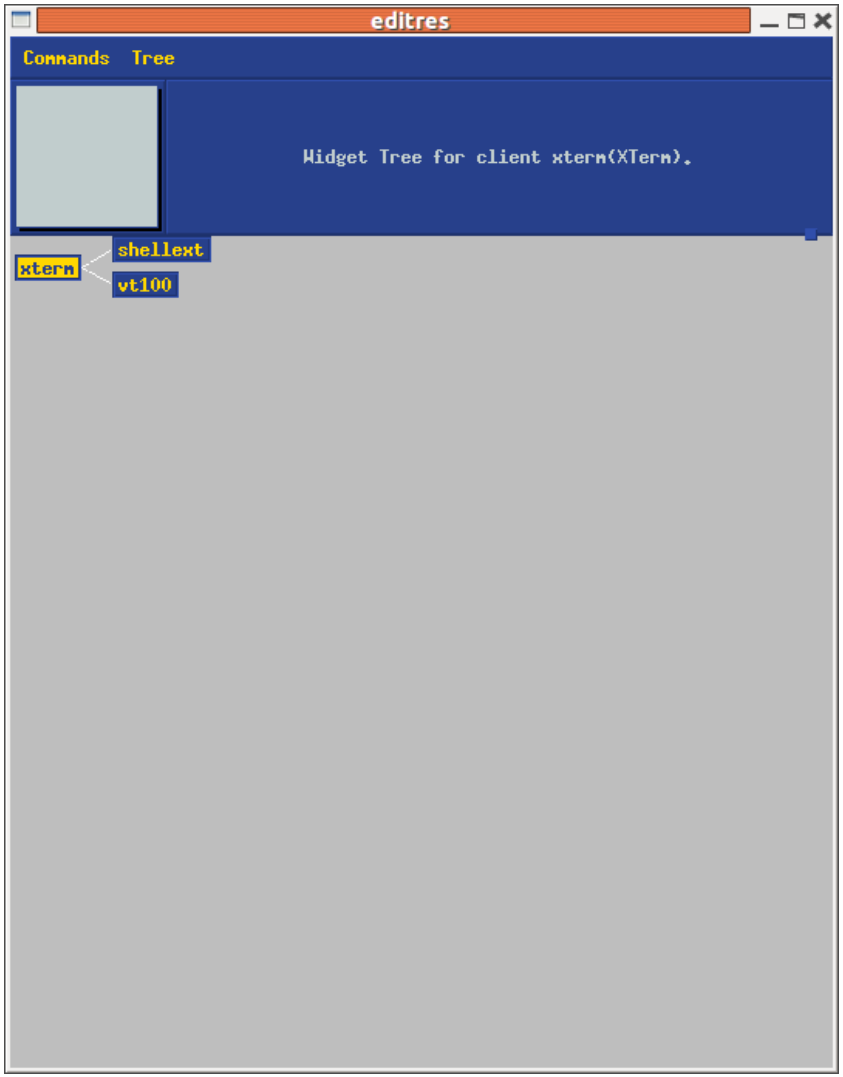
```
Client*mon_widget.ressource : valeur
```

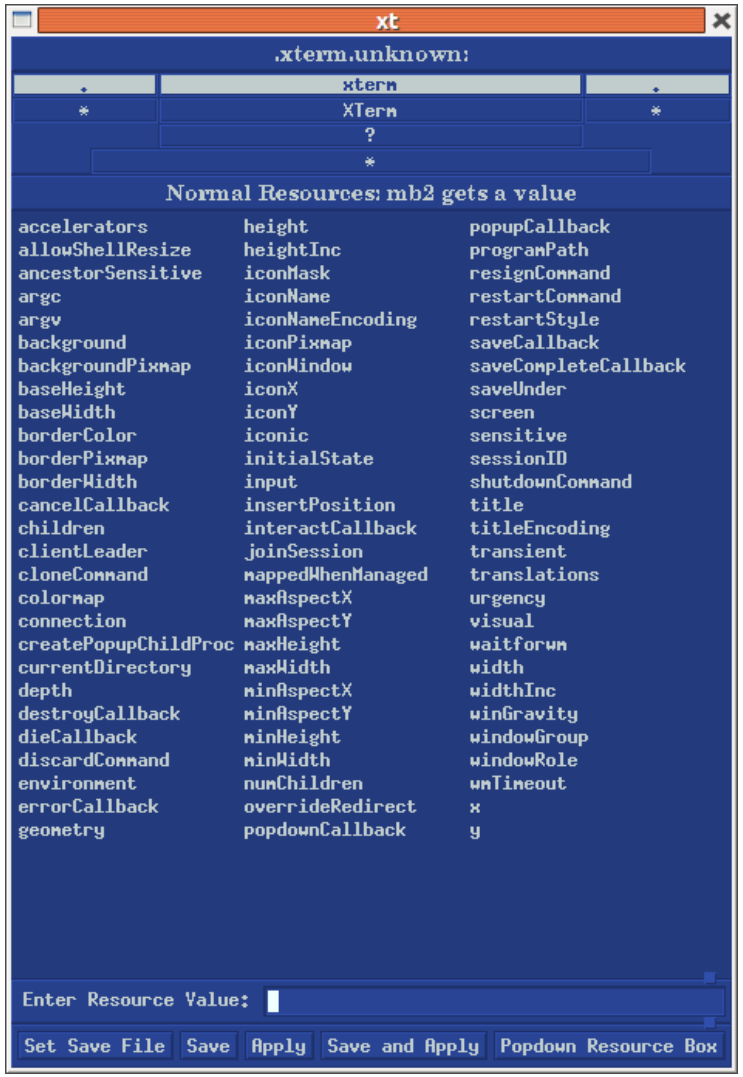
Les ressources de X

L'arborescence d'une application n'est pas directement accessible par l'utilisateur. Pour permettre de récupérer cette arborescence, X intègre un protocole de communication entre clients qui donne aussi la possibilité de modifier interactivement les ressouces.

Le client `editres` donne les moyens d'exploiter ce protocole pour configurer interactivement une application, il permet à la fois de découvrir l'arborescence des widgets servant d'interface à une application, mais aussi de modifier les ressources associées.

Le client `editres` (pour le client `xterm`).





Le clavier et la souris

Les touches physiques du clavier sont codées par des entiers appelés `keycode` qui ne peuvent pas être changés.

À ces touches physiques sont associées des touches symboliques qui représentent des actions, les `keysym` (ou `keysymname`).

Le client `xmodmap` permet de gérer ces associations.

Les noms symboliques des touches logiques sont définis dans le fichier `keysymdef.h` (le préfixe `XK_` doit être supprimé).

Modificateurs

Ces touches particulières (*modifiers* en anglais) peuvent être enfoncées en même temps qu'une autre touche pour modifier son action.

Ces modificateurs sont au nombre de 8 et portent les noms `Shift`, `Lock`, `Control`, `Mod1`, `Mod2`, `Mod3`, `Mod4` et `Mod5`.

Le modificateur `Lock` correspond à la touche « Caps Lock ». Le modificateur `Mod1` est généralement associé aux touches « Meta » ou « Alt ».

Evènements associés

Enfoncer ou relâcher une touche provoque l'envoi d'un événement à l'application qui possède le focus.

Cet événement contient entre autre, le keycode de la touche, le keysym associé et les modificateurs enfoncés en même temps (sous forme d'un octet avec un bit par modificateur).

Les touches des modificateurs provoquent également l'envoi d'événements.

Appui sur les touches Shift et z :

```
KeyPress event, serial 24, synthetic NO, window 0x2400001,
    root 0xae, subw 0x2400002, time 2701669, (31,37), root:(35,651),
    state 0x0, keycode 50 (keysym 0xffe1, Shift_L), same_screen YES,
    XLookupString gives 0 bytes:  ""

KeyPress event, serial 24, synthetic NO, window 0x2400001,
    root 0xae, subw 0x2400002, time 2702141, (31,37), root:(35,651),
    state 0x1, keycode 52 (keysym 0x5a, Z), same_screen YES,
    XLookupString gives 1 bytes:  "Z"

KeyRelease event, serial 24, synthetic NO, window 0x2400001,
    root 0xae, subw 0x2400002, time 2702221, (31,37), root:(35,651),
    state 0x1, keycode 52 (keysym 0x5a, Z), same_screen YES,
    XLookupString gives 1 bytes:  "Z"

KeyRelease event, serial 24, synthetic NO, window 0x2400001,
    root 0xae, subw 0x2400002, time 2702501, (31,37), root:(35,651),
    state 0x1, keycode 50 (keysym 0xffe1, Shift_L), same_screen YES,
    XLookupString gives 0 bytes:  ""
```

Tables d'associations

Deux tables, une pour les associations entre keycode et keysym et une autre pour les modificateurs.

La table des associations touches physiques/touches symboliques donne pour chaque keycode d'une touche, les keysym (au plus 8) qui lui sont associés.

Dans l'ordre, signification de la touche :

1. sans modificateur,
2. avec Shift,
3. avec Mode_Switch (touche AltGr - PC ou option - Mac),
4. avec Shift et Mode_Switch,

...

xmodmap -pk

```
...
50     0xffe1 (Shift_L) 0x0000 (NoSymbol) 0xffe1 (Shift_L)
51     0x002a (asterisk) 0x00b5 (mu) 0x002a (asterisk) 0x00b5 (mu) 0xfe50
52     0x0077 (w) 0x0057 (W) 0x0077 (w) 0x0057 (W) 0x00ab (guillemotleft)
53     0x0078 (x) 0x0058 (X) 0x0078 (x) 0x0058 (X) 0x00bb (guillemotright)
54     0x0063 (c) 0x0043 (C) 0x0063 (c) 0x0043 (C) 0x00a9 (copyright) 0x00
55     0x0076 (v) 0x0056 (V) 0x0076 (v) 0x0056 (V) 0x100202f (U202F) 0x08f
56     0x0062 (b) 0x0042 (B) 0x0062 (b) 0x0042 (B) 0x08fe (downarrow) 0x08
57     0x006e (n) 0x004e (N) 0x006e (n) 0x004e (N) 0x00ac (notsign) 0x08fd
58     0x002c (comma) 0x003f (question) 0x002c (comma) 0x003f (question) 0
59     0x003b (semicolon) 0x002e (period) 0x003b (semicolon) 0x002e (perio
60     0x003a (colon) 0x002f (slash) 0x003a (colon) 0x002f (slash) 0x00f7
61     0x0021 (exclam) 0x00a7 (section) 0x0021 (exclam) 0x00a7 (section) 0
62     0xffe2 (Shift_R) 0x0000 (NoSymbol) 0xffe2 (Shift_R)
63     0xffaa (KP_Multiply) 0x10022c5 (U22C5) 0xffaa (KP_Multiply) 0x10022
64     0xffe9 (Alt_L) 0xffe7 (Meta_L) 0xffe9 (Alt_L) 0xffe7 (Meta_L)
65     0x0020 (space) 0x0020 (space) 0x0020 (space) 0x0020 (space) 0x0020
66     0xffe5 (Caps_Lock) 0x0000 (NoSymbol) 0xffe5 (Caps_Lock)
...
```

Table d'associations des modificateurs

Elle donne pour chacun des 8 modificateurs la liste des touches symboliques qui jouent le rôle de ce modificateur (xmodmap -pm) :

```
shift       Shift_L (0x32),  Shift_R (0x3e)
lock
control     Control_L (0x25),  Control_R (0x6d)
mod1        Alt_L (0x40),  Alt_R (0x71)
mod2        Num_Lock (0x4d)
mod3        Mode_switch (0x73),  Mode_switch (0x74)
mod4
mod5        Scroll_Lock (0x4e)
```

Aucune touche n'étant affectée au modificateur `lock`, ceci signifie que la touche « Caps lock » est sans effet.

Boutons de la souris

Lorsque la souris a plusieurs boutons, on peut changer les associations entre les boutons physiques de la souris et les événements envoyés aux application.

L'expression `pointer` de la commande `xmodmap` permet de changer les associations des boutons de la souris :

pointer = num1, num2 ...

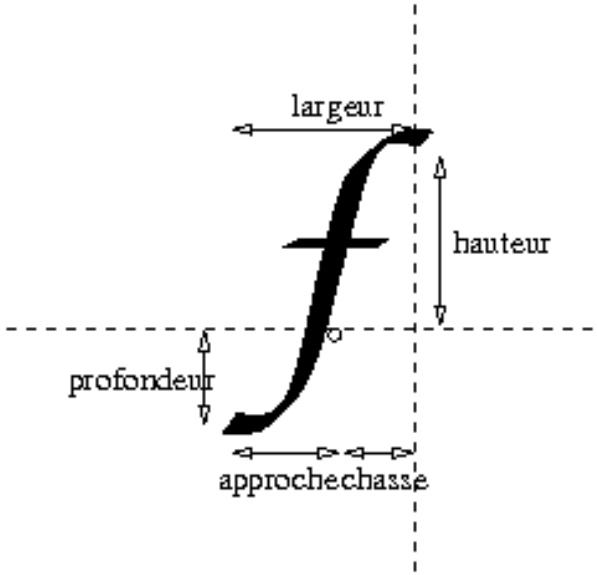
avec num1, num2, ... les noms symboliques associés aux boutons physiques 1, 2, ... de la souris.

Exemple : inversion de l'ordre des boutons pour gaucher

```
xmodmap -e "pointer = 3 2 1"
```

Les polices

Un caractère est définit par 5 paramètres qui sont : hauteur, largeur, profondeur, approche et chasse.



Pour assurer la portabilité de X Window, il a fallu décrire de façon très précise les caractères disponibles, par exemple :

```
-adobe-courier-bold-r-normal-11-80-100-100-m-60-iso8559-1
```

Vous pouvez obtenir la liste des fontes disponibles en utilisant le programme xlsfonts.

Les polices

`-adobe-courier-bold-r-normal-11-80-100-100-m-60-iso8559-1`

- compagnie propriétaire de la police
- jeu de caractères ou famille
- poids (largeur du trait) gras ou moyen
- pente (`r` pour romain, `i` pour italique, `o` pour oblique).
- nombre moyen de caractères par ligne (condensé, étroit, ...)
- dimension en pixel (0 : scalable)
- dimension en 10ème de point d'imprimerie [1] ($\times 10$)
- résolution verticale et horizontale (DPI)
- interligne (espacement) proportionnel ou constant (`p`, `m`)
- largeur moyenne en 10ème de point d'imprimerie ($\times 10$)
- jeu de caractères (standard international)

1. 1/72ème de pouce, soit 0,353 mm