X—WINDOW

Bibliographie sur Motif

O'Reilly & Associates

X Toolkit Intrinsics Programming Manual
vol. 4M, A. Nye & T. O'Reilly, 1992

X Toolkit Intrinsics Reference Manual
vol 5, D. Flanagan, 1992

Motif Programming Manual
vol 6a, D. Heller & P. Ferguson, 1993

→ **Motif Reference Manual**
vol 6b, P. Ferguson & D. Brennan, 1993

The X-Window system - Programming and applications with Xt
second edition, Douglas A. Young, PTR Prentice Hall, 1994.

Qu'est-ce que X-WINDOW ?

• Un système de multi-fenêtrage
  - indépendant du matériel
  - indépendant du logiciel

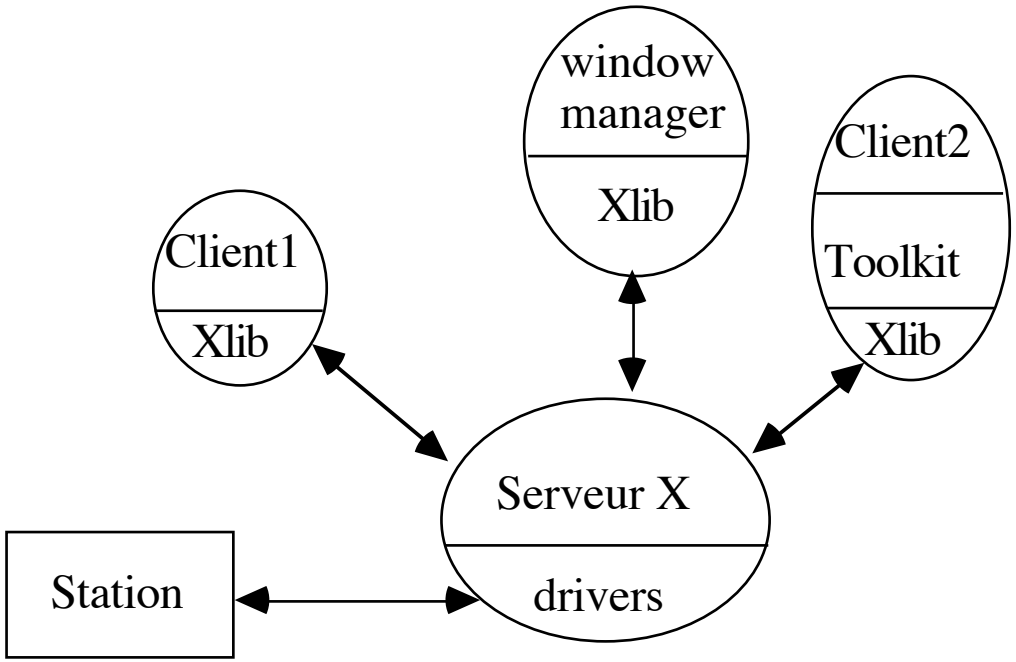
• X fonctionne sur de nombreuses machines et systèmes d'exploitation
  - Unix et variantes, VMS, MS-DOS, etc.
  - Sun, DEC, IBM, HP, PC, Terminaux X, ...

• X gère
  - le clavier et la souris
  - l'affichage sur un ou plusieurs écrans
    (textes, graphiques, images)
  - la communication avec les applications
    (utilisation en réseau)

Modèle Client-Serveur

Objectif:

Fonctionnement en réseau : afficher sur l'écran d'une même station des applications tournant sur différentes machines.



Un client particulier: le *window manager*

Le gestionnaire de fenêtres ou *window manager* est un client qui joue un rôle particulier.

Le *window manager* **s'occupe des fenêtres** qui sont **filles de la racine** et permet:

- de les déplacer sur la racine
- de les iconifier
- de changer leur taille
- de modifier l'ordre d'empilement entre les sœurs
  (passer devant ou derrière)
- d'afficher des menus proposant l'envoi de commandes

Son rôle est d'uniformiser l'apparence des fenêtres de plus haut niveau à l'écran et d'offrir à l'utilisateur un mode d'interaction uniforme pour les opérations de base sur les fenêtres.

Il rajoute souvent une barre de décoration comportant un titre et des icônes permettant de réaliser facilement les opérations de base (iconification, changement de taille).

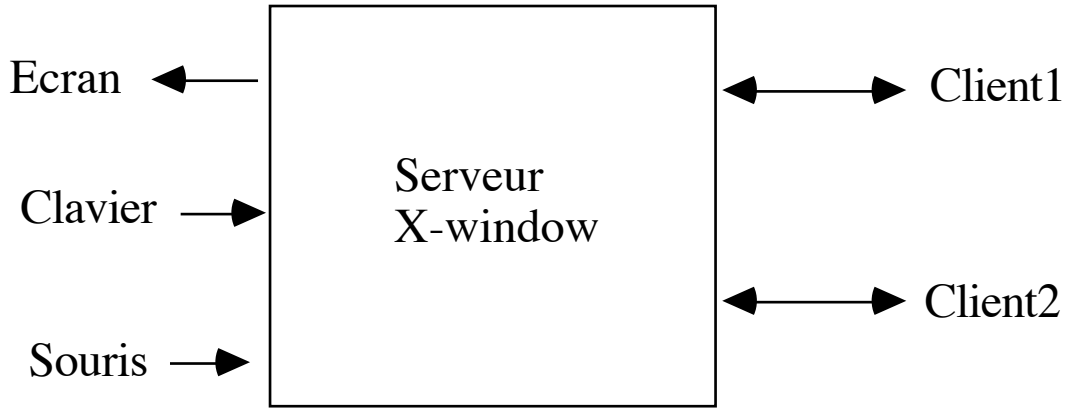
ex: olwm, mwm, twm.

Le serveur graphique X et la bibliothèque Xlib

C'est un intermédiaire entre

- les applications de l'utilisateur

- les ressources de la station utilisée
  (clavier, souris, écran(s), communication)



La bibliothèque Xlib

Les requêtes destinées au serveur sont exprimées en protocole X.

Le protocole X est efficace mais peu pratique d'emploi.

La bibliothèque Xlib est une interface au protocole X qui permet de communiquer avec le serveur de manière plus simple.

Xlib est un sur-ensemble du protocole X et elle assure la gestion:

- des fenêtres
- des événements
- des contextes graphiques
- des fontes
- des régions, des images et des couleurs.

Terminologie

Un *display* est un dispositif d'affichage comportant
CPU + clavier + souris + écran(s)

Un *screen* est un écran physique

Le serveur <=> *display*

Sous Unix:
La variable d'environnement DISPLAY est utilisée pour la connection au serveur X.

format: <machine>:<serveur>.<écran>

exemple: setenv DISPLAY brest:0.0 (csh)
setenv DISPLAY unix:0.0

Autorisation de connexion

Pour afficher sur l'écran de la machine *plougastel* à partir de la machine *toto*, il faut affecter la variable d'environnement du shell csh tournant sur *toto* par

```
setenv DISPLAY plougastel:0.0
```

Mais cela ne suffit pas. Il faut que le serveur X de *plougastel* autorise cette connection
Pour cela faire au préalable sur la machine *plougastel*:

```
xhost +toto
```

qui autorise les connections de la machine *toto* sur *plougastel*. On peut aussi restreindre les connections à un seul utilisateur:

```
xhost +user@toto
```

Requêtes

Les clients effectuent des requêtes au serveur par l'intermédiaire de la Xlib.

Le serveur tourne de manière asynchrone par rapport aux clients. Les requêtes sont effectuées dans l'ordre où elles ont été demandées mais pas de manière immédiate.

Les requêtes sont empilées par la Xlib avant l'envoi au serveur et empilées dans le serveur avant traitement.

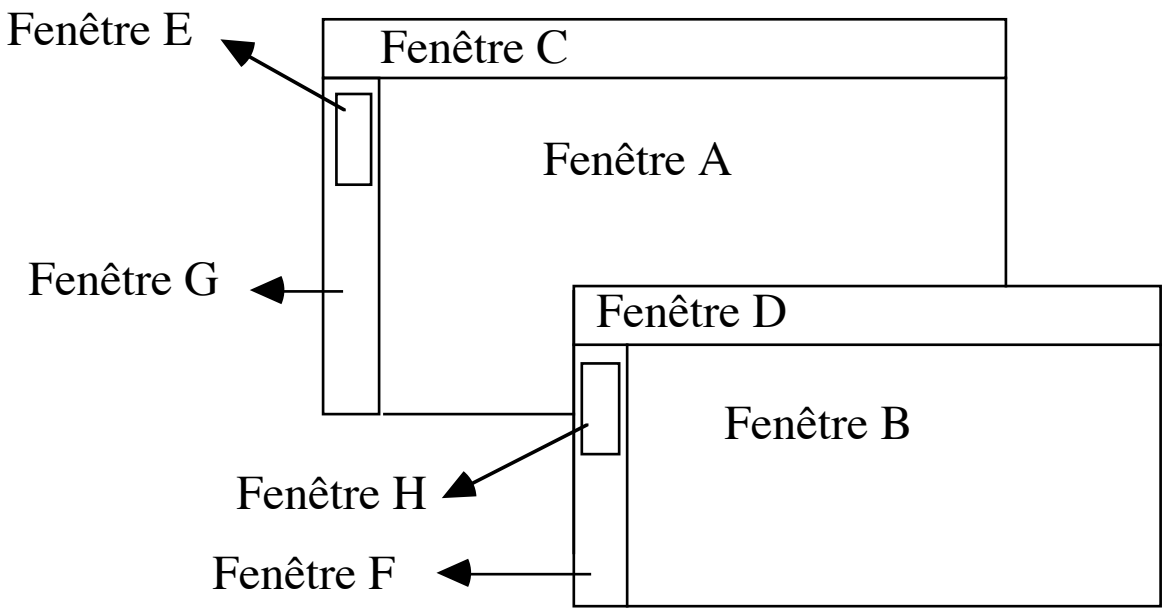
On peut demander un déroulement synchrone mais c'est très coûteux et on n'utilise cette fonctionnalité que pour debuguer.

Certaines requêtes peuvent être synchronisées .

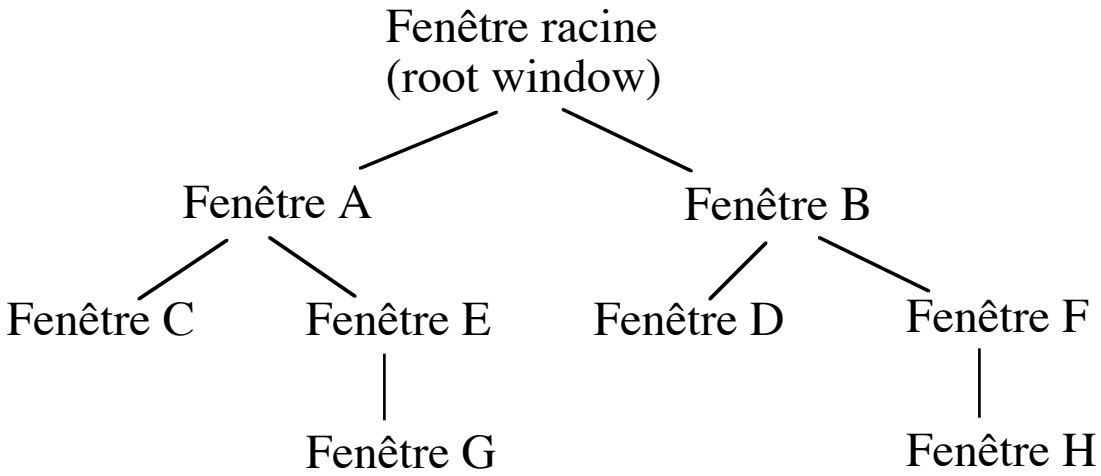
Les fenêtres

Les fenêtres sont les structures de base du serveur. Elles sont rectangulaires.

Les fenêtres sont organisées en hiérarchie. La fenêtre racine s'appelle la *root window*.
Toute fenêtre a une mère (ou parent) exceptée la racine.

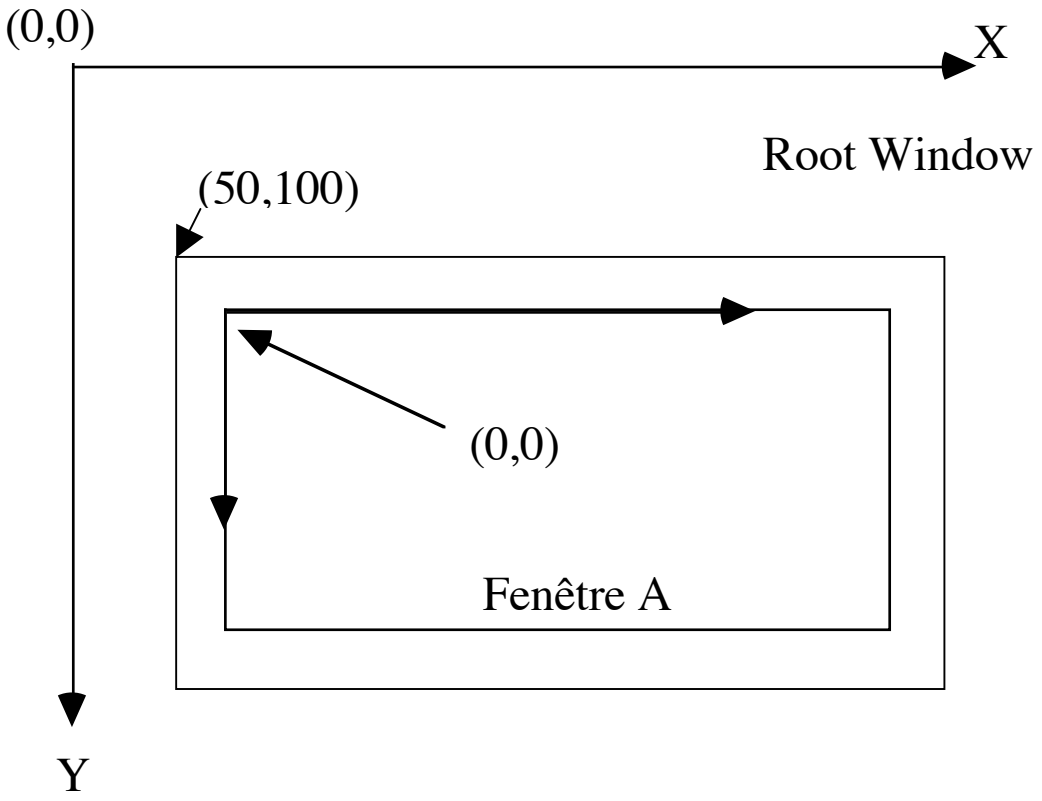


Hiérarchie ou arborescence des fenêtres



Le système de coordonnées des fenêtres

Une fenêtre est repérée par sa largeur, sa hauteur et la position de son coin supérieur gauche par rapport au coin supérieur gauche de sa mère.



Contenu des fenêtres

Le contenu d'une fenêtre n'apparaît graphiquement que dans le cadre (*clip*) de sa mère. On dit qu'il est encadré par sa mère. Lorsqu'on bouge une fenêtre, son contenu et ses fenêtres filles sont déplacés par le serveur.

Mais par défaut le contenu d'une fenêtre n'est pas mémorisé par le serveur.

Si le contenu est affiché puis rendu invisible, ce sera à l'application de prendre en charge le réaffichage. Néanmoins, certains serveurs font du *backing store*, et dans ce cas peuvent mémoriser le contenu de certaines fenêtres.

Le serveur informe cependant les applications que le contenu d'une fenêtre doit être réaffiché par l'intermédiaire d'un événement de type *Expose*.

Les toolkits réaffichent le contenu des objets graphiques de manière transparente.

Les événements

Le serveur communique avec les applications en leur envoyant des événements. Ce sont des structures de données d'informations qui correspondent à divers cas prototypés.

Serveur X = boucle infinie qui

- reçoit les requêtes des clients
- effectue les requêtes demandées par les clients
- détecte des actions (clic souris, touche clavier, réaffichage, requêtes, etc.)
- envoie les événements correspondants aux applications qui le souhaitent

Client X = boucle infinie qui

- récupère les événements qui l'intéressent
- exécute en retour d'autres requêtes ou appelle des fonctions appropriées

On appelle ce type de programmation la programmation dirigée par événements.
Attention: les événements récupérés par les applications sont également récupérés de manière asynchrone par rapport à l'action qu'il représente.

Sélection des événements

Une application qui souhaite recevoir des événements particuliers doit le déclarer au serveur.

Cette sélection s'effectue au niveau de chaque fenêtre à l'aide de masques prédéfinis. Ainsi, par exemple :

ButtonPressMask	sélectionne l'enfoncement d’un bouton de souris
ButtonReleaseMask	sélectionne le relâchement d’un bouton de souris
PointerMotionMask	sélectionne les mouvements de souris
ButtonMotionMask	sélectionne les mouvements souris bouton enfoncé (drag)
Button<i>MotionMask	sélectionne les mouvements de souris bouton<i> enfoncé
KeyPressMask	sélectionne l'enfoncement d'une touche du clavier
KeyReleaseMask	sélectionne le relâchement d'une touche
ExposureMask	sélectionne un besoin de réaffichage

Il y en a beaucoup d'autres. En outre certains événements ne sont pas sélectionnés par masques, mais envoyés systématiquement à toute application.