X-I

Xlib : textes

dessin

- Dessiner un texte
- Polices
- Caractéristiques d'un texte
- Nom des fontes
- Développements récents

saisie

- Saisir un caractère
- Les touches symboliques
- Lecture
- Foyer du clavier
- Localisation et internationalisation

X-I

Dessiner un texte

- **Toute fonction X de dessin de textes existe en deux versions:**
  - **pour les caractères codés sur 1 octet (la plupart);**
  - **pour les caractères codés sur 2 octets (16 bits): fonctions suffixées par 16.**
- **Dans la version X11R5, il y a des codages plus sophistiqués.**
- **Un texte est tracé dans une *fonte*. X n'impose pas de police (!) sur les fontes.**
- **Une fonte doit être *chargée* dans le serveur:**
  - **pour chaque serveur X, les fontes résident dans des répertoires particuliers d'une machine particulière;**
  - **il y a donc des problèmes de codage et de portabilité.**
- **Une fois chargée, la fonte est utilisée en l'intégrant dans un *contexte graphique*.**
- **Le dessin d'un texte se fait en invoquant un contexte graphique.**

X-I

- **Chargement d'une police:**

```
Font f;
f = XLoadFont(dpy, nom);
```

  - `nom`**, de type** `char *`**, désigne la police à charger.**
- **Inclusion dans un contexte graphique:**

```
XSetFont(dpy,ctx,f)
```

- **Ecriture d'une chaîne de caractères:**

```
XDrawString(dpy,fen,ctx,x,y, texte,longueur)
XDrawImageString(dpy,fen,ctx,x,y, texte,longueur)
```

  - `x,y` **sont les coordonnées du début du texte (voir plus loin);**
  - `texte` **de type** `char *` **est composé de** `longueur` **caractères;**
  - `XDrawString` **trace dans le contexte graphique (y compris filigrane);**
  - `XDrawImageString` **trace**
    en couleur d'encre, avec `GXcopy`**,** `FillSolid`**, et**
    un rectangle (voir plus loin) dans la couleur de fond (de remplissage) du contexte graphique.

X-I

Paramètres d'une police

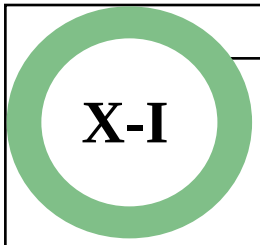
- **Les paramètres relatifs à une police sont contenus dans une structure de nom** `XFontStruct`**.**
- **On charge une police *et* la description de ses paramètres par:**

```
XFontStruct *s;
s = XLoadQueryFont(dpy, nom);
```

- **On peut récupérer séparément les informations sur une police** `f` **déjà chargée par:**

```
s = XQueryFont(dpy, f);
```

- **Les champs importants de la structure sont :**
  - `fid` **qui donne la fonte** `f`**;**
  - `ascent` **et** `descent` **qui donnent la cote haute et la cote basse;**
  - `min_bounds` **et** `max_bounds` **qui donnent des encadrements extrêmes pour les caractères.**



Structure X.lib

```
typedef struct {
    XExtData  *ext_data;            /* hook for extension to hang data */
    Font      fid;                  /* Font id for this font */
    unsigned  direction;            /* hint about direction the font is painted */
    unsigned  min_char_or_byte2;    /* first character */
    unsigned  max_char_or_byte2;    /* last character */
    unsigned  min_byte1;            /* first row that exists */
    unsigned  max_byte1;            /* last row that exists */
    Bool      all_chars_exist;      /* flag if all characters have non-zero size*/
    unsigned  default_char;         /* char to print for undefined character */
    int       n_properties;         /* how many properties there are */
    XFontProp   *properties;        /* pointer to array of additional properties*/
    XCharStruct  min_bounds;        /* minimum bounds over all existing char*/
    XCharStruct  max_bounds;        /* maximum bounds over all existing char*/
    XCharStruct  *per_char;         /* first_char to last_char information */
    int       ascent;               /* log. extent above baseline for spacing */
    int       descent;              /* log. descent below baseline for spacing */
} XFontStruct;
```

X-I

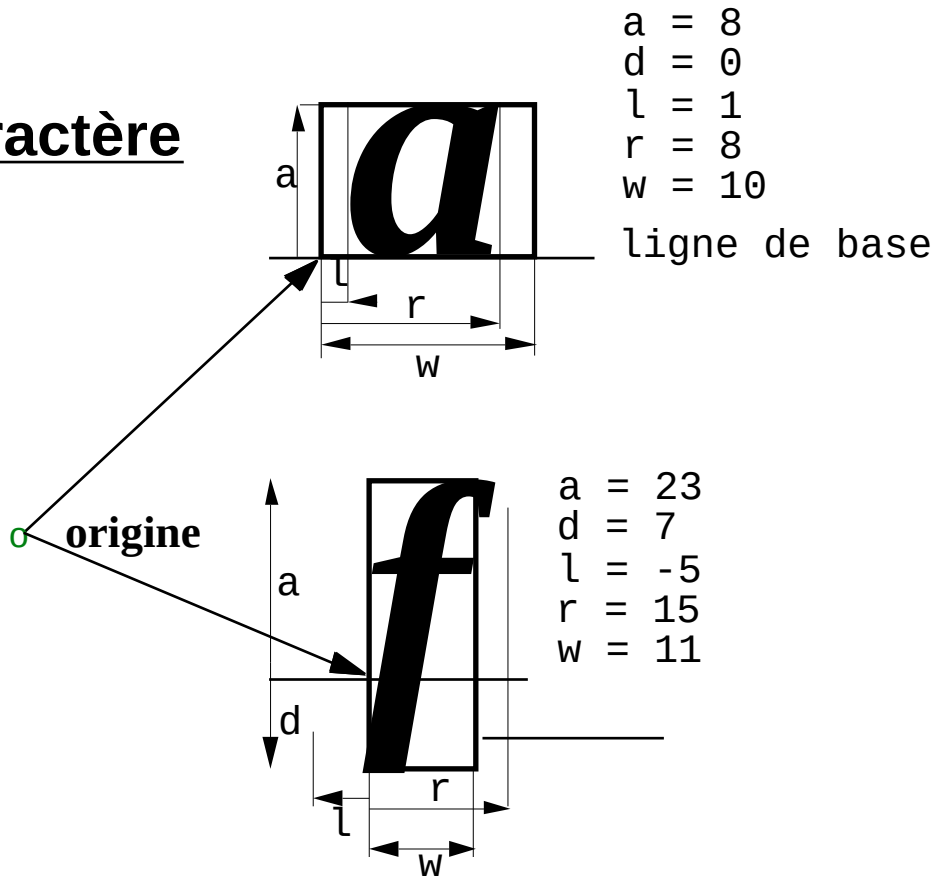
Paramètres d'un texte

- **Un texte est composé en enfilant ses caractères sur une *ligne de base*.**
- **Chaque *fonte* possède deux paramètres dont la somme donne la hauteur (le *corps*):**
  - ***cote haute*** **(`s->ascent`) : taille au dessus de la ligne de base;**
  - ***cote basse*** **(`s->descent`): taille en dessous de la ligne de base.**
- **Dans les fonctions `XDrawString` et `XDrawImageString`, `(x,y)` sont les coordonnées du début du texte sur la ligne de base.**
- **Dans `XDrawImageString`, le rectangle a**
  - **pour coin supérieur gauche** `(x, y - s->ascent);`
  - **pour hauteur** `s->ascent+s->descent.`
  - **pour largeur** `XTextWidth(s, texte, longueur).`

X-I

Caractéristiques d'un caractère

- **Chaque caractère a une largeur (*chasse*).**
- **La partie noircissante du caractère est l'*œil*.**
- **Le rectangle englobant l'oeil est repéré par:**
  - **l'*approche gauche*** (`lbearing`)**;**
  - **l'*approche droite*** (`rbearing`)**;**
  - **les cotes haute et basse.**
- **Certaines de ces cotes peuvent être négatives, donc physiquement irréalisables, mais informatiquement significatives.**




```
typedef struct {
    short      lbearing;            /* origin to left edge of raster */
    short      rbearing;            /* origin to right edge of raster */
    short      width;               /* advance to next char's origin */
    short      ascent;              /* baseline to top edge of raster */
    short      descent;             /* baseline to bottom edge of raster */
    unsigned short attributes;       /* per char flags (not predefined) */
} XCharStruct;
```

X-I

Caractéristiques d'un texte

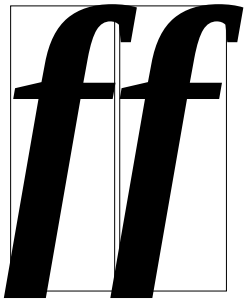
- **La *largeur* d'un texte, en pixels, est la somme des largeurs des caractères qui le composent. Elle s'obtient par:**

  `largeur = XTextWidth(s, texte,longueur);`

  `s` est de type `XFontStruct *`.

- **C'est cette largeur qui est utilisée dans `XDrawImageString`.**

- **Elle ne correspond donc pas nécessairement au plus petit rectangle englobant la chaîne tracée.**



- **Informations générales par**

```
XTextExtents(
    XFontStruct*   s          /* font_struct */,
    char*          chaine     /* string */,
    int            longueur   /* nchars */,
    int*           sens       /* direction_return */,
    int*           a          /* font_ascent_return */,
    int*           d          /* font_descent_return */,
    XCharStruct*   x          /* overall_return */
);
```

X-I

Exemple

```
XTextExtents(s, chaine, longueur, &sens, &asc, &desc, &x);
```

- **sens vaut** `FontLeftToRight` **ou** `FontRightToLeft`
- `asc` **est la cote hausse *de la fonte* (et non du texte), de même pour** `desc`
- `x` **contient les caractéristiques de la chaîne, comme pour un caractère; en particulier :**
  - `x.width` **est égal à** `XTextWidth`
  - `x.lbearing` **et** `x.rbearing` **sont les approches gauche et droite *de la chaîne***
  - `x.ascent` **et** `x.descent` **sont les cotes haute et basse de la chaîne.**

X-I

Plusieurs fontes

```
XDrawText(d, f, gc, x, y, items, n);
```

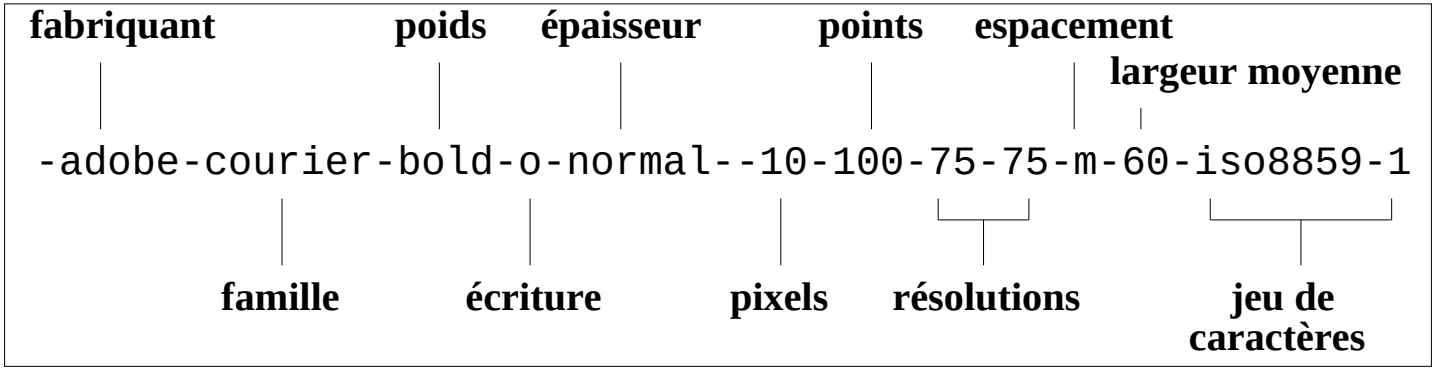
où `items` est un tableau de `n` objets de type `XTextItem`, permet le mélange plusieurs fontes dans la même requête

Trois *styles* *differents*

```
typedef struct {
  char *     chars;      /* la chaîne */
  int        nchars;     /* la longueur de la chaîne */
  int        delta;      /* l'espacement avant cet item */
  Font       font;       /* la fonte */
}XTextItem;
```

X-I

Noms des fontes

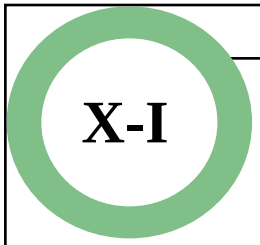


- **X introduit un système de description de fontes appelé XLFD (X Logical Font Description).**
- **Chaque fonte est décrite par une chaîne de caractères à 12 champs séparés par des tirets.**
  - **famille :** `courier times palatino`
  - **poids :** `medium`**,** `bold`
  - **écriture :** `r` **(romain),** `i` **(italique),** `o` **(oblique)**
  - **épaisseur :** `normal`**,** `condensed`**,..**
  - **pixels : taille en pixels d'écran**
  - **points : taille en 10e de points**
  - **résolution (donnée en point par pouce) :** `75` **ou** `100`
  - **espacement :** `m` **(mono = fixe) ou** `p` **(proportionnel)**
  - **largeur moyenne : en 10e de pixel**
  - **jeu de caractères : nom de la norme ISO**

X-I

Utilitaires

- ***Choix du nom*** **d'une fonte à travers** `xfontsel` **(conforme aux conventions XLFD: *XLogical Font Description*)**
  - **Le symbole * remplace n'importe quel suite de caractères et ? remplace un caractère quelconque. Choisir une fonte avec les menus associés aux champs. Ex:** `*times-medium-r-*-120*`
  - **Lorsqu'il y a plusieurs fontes répondant aux spécifications, l'utilitaire choisit le premier dans la liste.**
  - **Conseil: donner explicitement, au moins la famille, le poids, l'écriture et la taille en points.**
  - **Faire un copier-coller pour récupérer le nom.**
- ***Visualisation*** **d'une fonte par xfd (*X font displayer*) qui affiche l'ensemble des caractères disponibles dans une fonte ainsi que leurs caractéristiques géométriques.**
- **Liste des noms des fontes disponibles dans une spécification donnée:**
  xlsfonts "*times-medium-r-*"



Exemples de noms

- **Toute partie inutile peut être omise, en utilisant les notations usuelles :**
  - **? pour un caractère**
  - *** pour une chaîne (y compris des tirets)**
- **Il existe des abbréviations pour des polices courantes.**
- **NB. Ces noms correspondent à des polices qui figurent ou ne figurent pas dans les répertoires de police de la machine associée au serveur.**
- **Exemples:**

```
-bigelow & holmes-*-bold-i-*-serif-*-360*
*courier-bold-o-*-100*
-schumacher-clean-medium-r-normal--8-80*
```

```
10x20
12x24
5x8
6x10
6x12
6x13
6x9
7x13
7x14
8x13
8x16
9x15
9x15b
9x15bold
courr08
courr10
fixed
helvb14
helvb18
helvb24
k14
kana14
lucidav2rt12
olglyph10
pellucidaserif12
timr24
```

X-I

Développements X11R5

- **Serveur de fontes:**
  - **un nouveau format de stockage de polices PCF (Portable Compiled Font) permet de rendre les applications indépendantes des architectures.**
  - **Il se substituera progressivement au format actuel SNF (Server Normal Format).**
  - **Le format BDF (Bitmap Distribution Format) est en ASCII. Il permet donc l'échange et le transfert.**
- **Polices vectorielles:**
  - **Une extension du XLFD permet d'englober les polices vectorielles: pour ces polices, les champs de taille en pixels, en dixième de point et la largeur moyenne valent 0.**
- **Format de textes plus compliqués et fonctions assorties :**

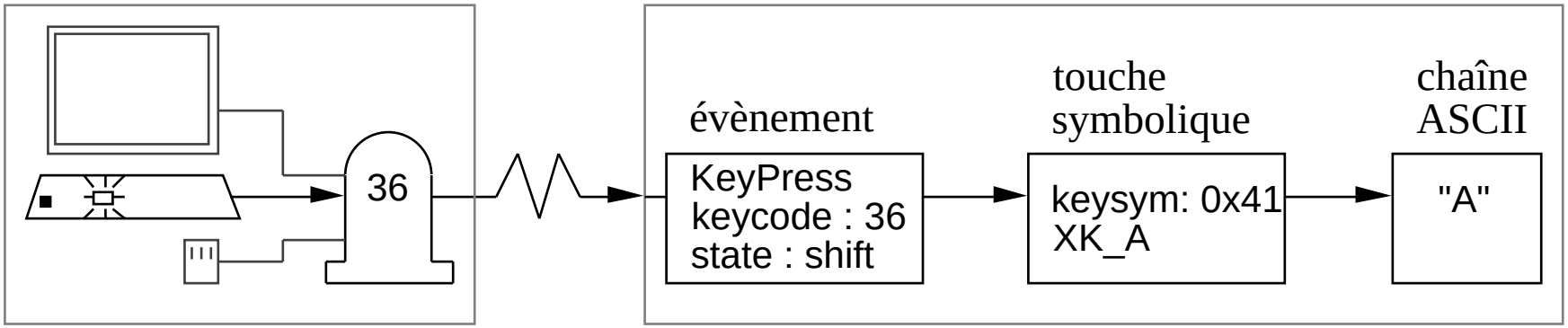
`XwcDrawString()`	**"wide characters"**
`XmbDrawString()`	**"multi-byte characters"**

Saisie de textes

- Saisir un caractère
- Les touches symboliques
- Lecture
- Foyer du clavier
- Localisation et internationalisation

X-I

La chaîne de traitement

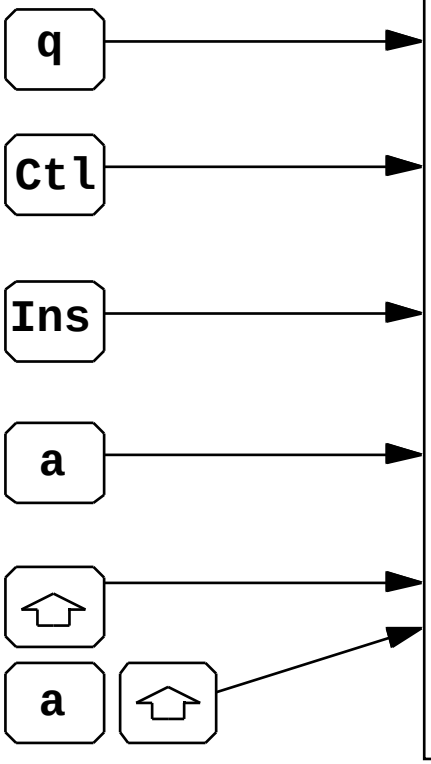


Entre le moment où l'on enfonce ou relâche une touche et l'utilisation du caractère tapé (si c'est un caractère), les transformations suivantes ont lieu:

- **Transformation de l'impulsion en un *numéro de touche* (`keycode` : entier entre 8 et 255).**
- **Génération (par le serveur) d'un évènement `KeyPress` ou `KeyRelease` contenant**
  - **Le numéro de la touche,**
  - **Un masque pour les touches d'*altération* présentes (`Shift`, `Ctrl`).**
- **Transcription (par le client) de ces données en *touche symbolique* (`keysym`).**
- **Transcription (par le client) de la touche symbolique en chaîne ASCII (éventuellement vide).**

X-I

Exemple



Evènement	état	keycode	keysym	caractères
KeyPress	0x0	29	XK_q	"q"
KeyRelease	0x0	29	XK_q	"q"
KeyPress	0x0	28	XK_Control_L	""
KeyRelease	0x4	28	XK_Control_L	""
KeyPress	0x0	111	XK_Insert	""
KeyRelease	0x0	111	XK_Insert	""
KeyPress	0x0	36	XK_a	"a"
KeyRelease	0x0	36	XK_a	"a"
KeyPress	0x0	97	XK_Shift_R	""
KeyPress	0x1	36	XK_A	"A"
KeyRelease	0x1	36	XK_A	"A"
KeyRelease	0x1	97	XK_Shift_R	""

X-I

Champ xkey

- **La structure X correspondant à un évènement est un champ de la structure** `XEvent`**:**
- **La structure** `XKeyEvent` **a la forme :**

```
typedef union _XEvent {
  int type;
   ...
   XKeyEvent xkey;
   ...
} XEvent;
```

```
typedef struct {
   int type;        /* of event */
   ...
   Window window;   /* "event" window it is reported relative to */
   int x, y;        /* pointer x, y coordinates in event window */
   ...
   unsigned int state;    /* key or button mask */
   unsigned int keycode;  /* detail */
   ...
} XKeyEvent;
typedef XKeyEvent XKeyPressedEvent;
typedef XKeyEvent XKeyReleasedEvent;
```

X-I

touches d'altération

- L'état (`state`) ne reflète pas exactement l'état des touches d'altération:
  - Il n'y a plus distinction entre shifts gauche et droit.
- La correspondance varie selon le matériel du serveur. On obtient les informations par `xmodmap`. On a par exemple:

```
xmodmap:  up to 2 keys per modifier, (keycodes in parentheses):

shift       Shift_L (0x1a),  Shift_R (0x61)
lock        Caps_Lock (0x19)
control     Control_L (0x1c),  Control_R (0x60)
mod1        Alt_L (0x21)
mod2
mod3
mod4
mod5
```

- Certains claviers ont des touches spécifiques (`Meta`)...

X-I

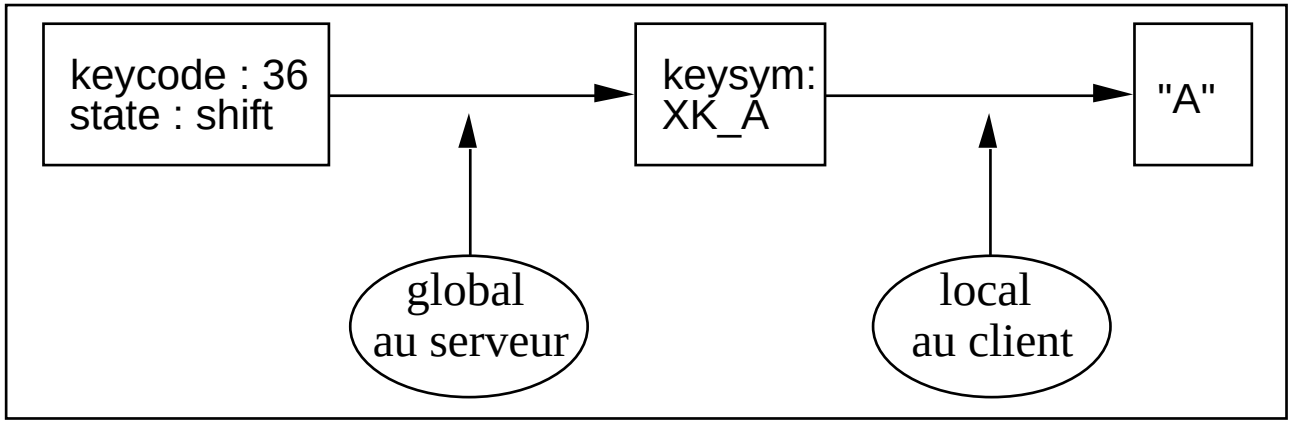
Touches symboliques

- **Le système X définit de très nombreuses *touches symboliques*. Elles commencent par `XK_` et sont contenues dans les fichiers `<keysym.h>`(version courte) et `<keysymdef.h>` (version longue).**
- **Chaque *serveur* possède une *table de correspondance* ("keyboard mapping") entre les numéros de touches (+ altéra–teurs) de son clavier et les touches symboliques réalisables.**
- **La table se présente sous la forme d'une liste de touches symboliques pour chaque numéro de touche.**
- **Le choix de la touche symbolique dans la liste d'un numéro dépend du masque des altérateurs.**

```
...
keycode  28 = Control_L
keycode  29 = q Q
keycode  30 = 1 exclam
...
keycode  35 = s S
keycode  36 = a A
keycode  37 = w W
keycode  97 = Shift_R
keycode  98 = Return
...
keycode 100 = backslash bar
...
keycode 102 = F12
keycode 103 = Break
...
keycode 110 = BackSpace
keycode 111 = Insert
keycode 112 =
keycode 113 = KP_1 End
...
```

X-I

“Keyboardmapping”



La table de correspondance entre clavier et touches symboliques est *globale au serveur*.

- **Elle est chargée par chaque client à l’ouverture dans la structure `Display`.**
- **Elle peut être modifiée par un client, à l’aide de la fonction `XChangeKeyboardMapping`, et cette modification devient globale au serveur.**
- **Le serveur envoie dans ce cas d’office *à tous les clients* un évènement `MappingNotify`. NB: cet évènement n’est pas masquable !**
- **En réponse à cet évènement, un client qui saisit du texte doit rafraîchir sa copie de la table de correspondance en incluant, dans sa boucle principale,**

```
case MappingNotify :
  XRefreshKeyboardMapping(&evmt);
  break;
```

X-I

Evènements en caractères

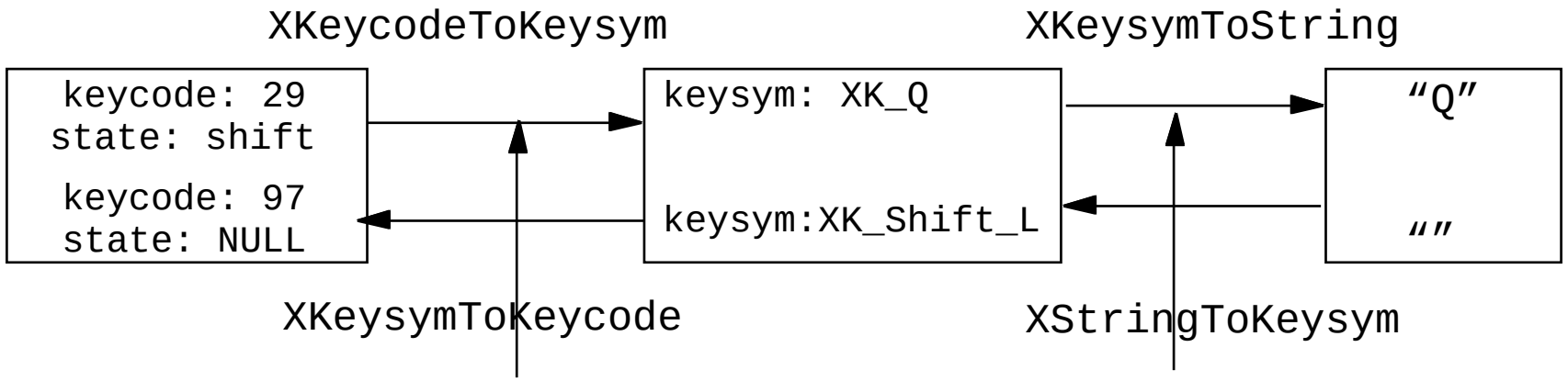
La correspondance entre touches symboliques et chaînes de caractères est *locale au client.*

```
int XLookupString(e, tampon, taille, k, statut);
  XKeyEvent *e;              évènement utilisé
  char *tampon;              chaîne de caractères retournée
  int taille;                taille du tampon
  Keysym *k;                 touche symbolique obtenue
  XComposeStatus *statut;    inutilisé jusqu'à maintenant
```

- **La fonction `XLookupString` intègre le passage de l'évènement à la chaîne ASCII.**
- **Elle retourne le nombre de caractères lus.**

X-I

Les 3 apparences d'un caractère



- **Modifiable *globalement*, c'est-à-dire pour tous les clients, par l'utilitaire** `xmodmap`**. Les clients reçoivent un événement** `MappingNotify` **et mettent à jour la nouvelle conversion par un appel à la fonction** **`XRefreshKeyboardMapping`**.

- **Modifiable *localement*, c'est-à-dire pour le seul client qui en fait la demande, par exemple, affecter la chaine** `"beta"` **à la touche** `b` **accompagnée de la touche majuscule gauche:**

```
KeySym md[1] = {XK_Shift_L};
XRebindKeysym(dpy, XK_b, md,
    1, "beta", strlen("beta"));
```

X-I

Foyer du clavier

- **La fenêtre qui reçoit les caractères entrées au clavier est le foyer du clavier (“input focus”).**
- **Le gestionnaire de fenêtre peut modifier la détermination de cette fenêtre.**
- **La convention explicite est la suivante :**
  - **A tout moment, une fenêtre possède le *foyer du clavier* (“input focus”), *indépendamment* de la position du pointeur de la souris.**
  - **Seule cette fenêtre et ses descendantes peuvent recevoir des évènements clavier.**
  - **Par défaut, la fenêtre racine est le foyer, et c’est le pointeur qui contrôle quelle fenêtre reçoit l’évènement.**
- **Les gestionnaires de fenêtres permettent de fixer la fenêtre foyer à autre chose que la racine (**`keyboardFocusPolicy`**).**
- **Lorsque le foyer est changé, des évènements particuliers sont engendrés que l’on peut sélectionner par** `FocusChangeMask`.
- **On change explicitement le foyer d’entrée par:**

```
XSetInputFocus(dpy,fen,mode,date);
```

  - **où** `fen` **est le le nouveau foyer,** `date` **(de type** `Time`**,** `CurrentTime` **convient) est le moment de prise d’effet, et**
  - `mode` **est** `RevertToParent`**,** `RevertToPointerRoot`**,** `RevertToNone`**, et indique la fenêtre qui hérite du foyer d’entrée si** `fen` **devient invisible.**

X-I

Internationalisation et localisation

- ***Objectif*** **: Utiliser un logiciel dans plus d'une langue (plus d'un pays) sans le réécrire.**
- ***Exigences*** **:**
  - **Lire et écrire dans la langue du pays;**
  - **Dialogues, menus, aides également dans la langue du pays;**
  - **Formats de nombres, monétaires, dates.**
- ***Internationalisation*** **: procédé qui consiste à écrire un code "générique", indépendant du pays, que l'utilisateur ou l'administrateur système profile.**
- ***Localisation*** **: faire les adaptations nécessaires (variables d'environnement, fichiers, etc) pour adapter un programme internationalisé à un pays ou une langue donnée.**
- **En X, l'internationalisation est fixée dans la norme `Xi18n` (ainsi nommée parce qu'il y a 18 lettres entre l'initiale et la finale du mot internationalisation). La norme s'appuie sur le concept de `locale` en C ANSI.**

X-I

Méthodes d'entrée

- **La saisie de textes rencontre de nombreux problèmes:**
  - **obtenir du clavier des caractères qui ne sont pas représentés (è par exemple);**
  - **entrer un texte dans une écriture idéographique (des dizaines de milliers de symboles) ou phonétique.**
- **Les *méthodes d'entrée* sont un procédé proposé par le Consortium X avec la version 5. Il a trois aspect majeurs:**
  - **l'entrée de texte peut être assurée par un serveur séparé;**
  - **la saisie de caractères est asynchrone (`XLookupString` revoie "en cours" tant que la lecture d'une entité n'est pas terminée);**
  - **la saisie de textes peut être interactive, avec consultation de l'utilisateur pour lever des ambiguïtés.**
- **Le mécanisme est complexe et est répandu au niveau des widgets.**