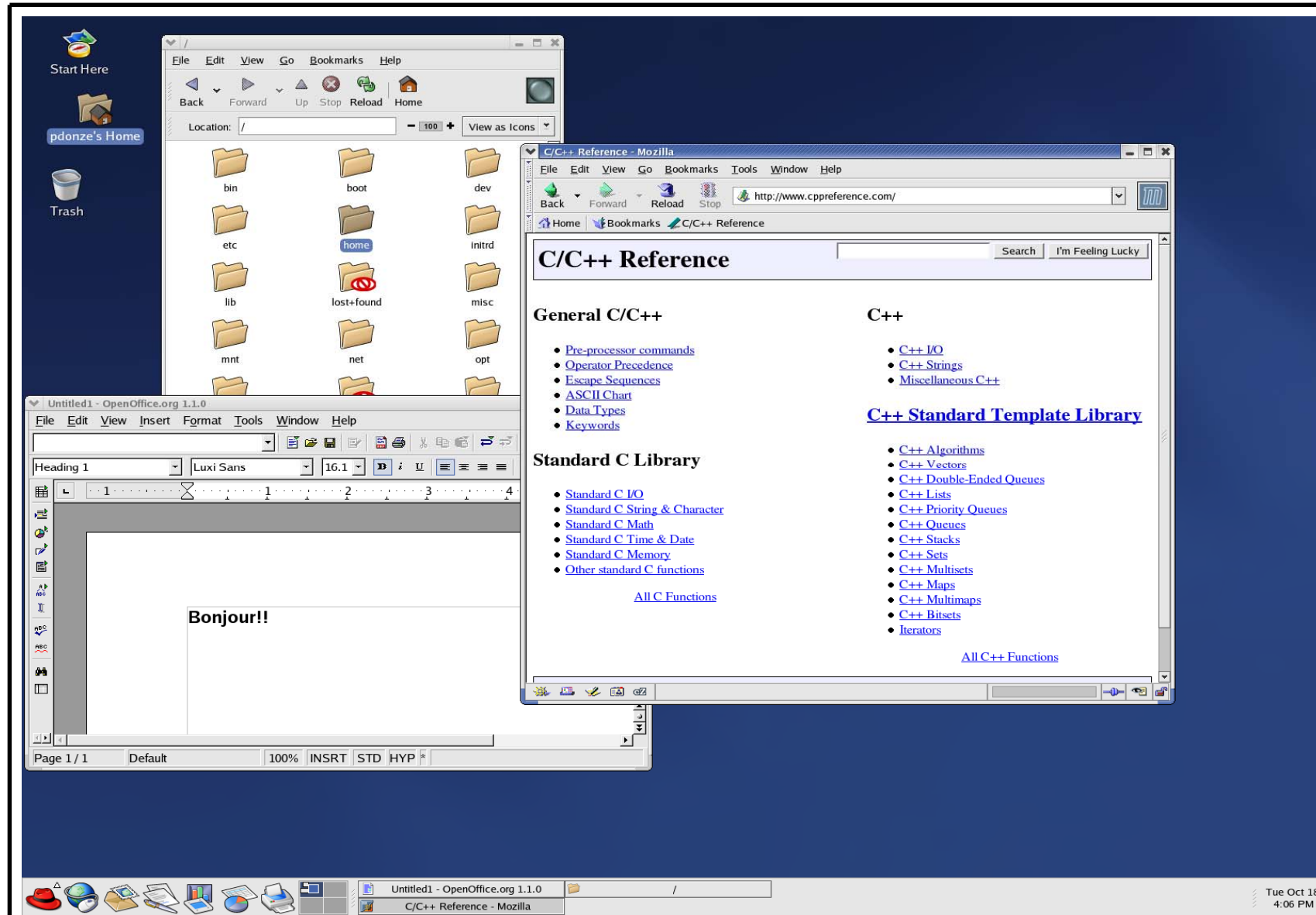


(0, 0)

Interface X-Window

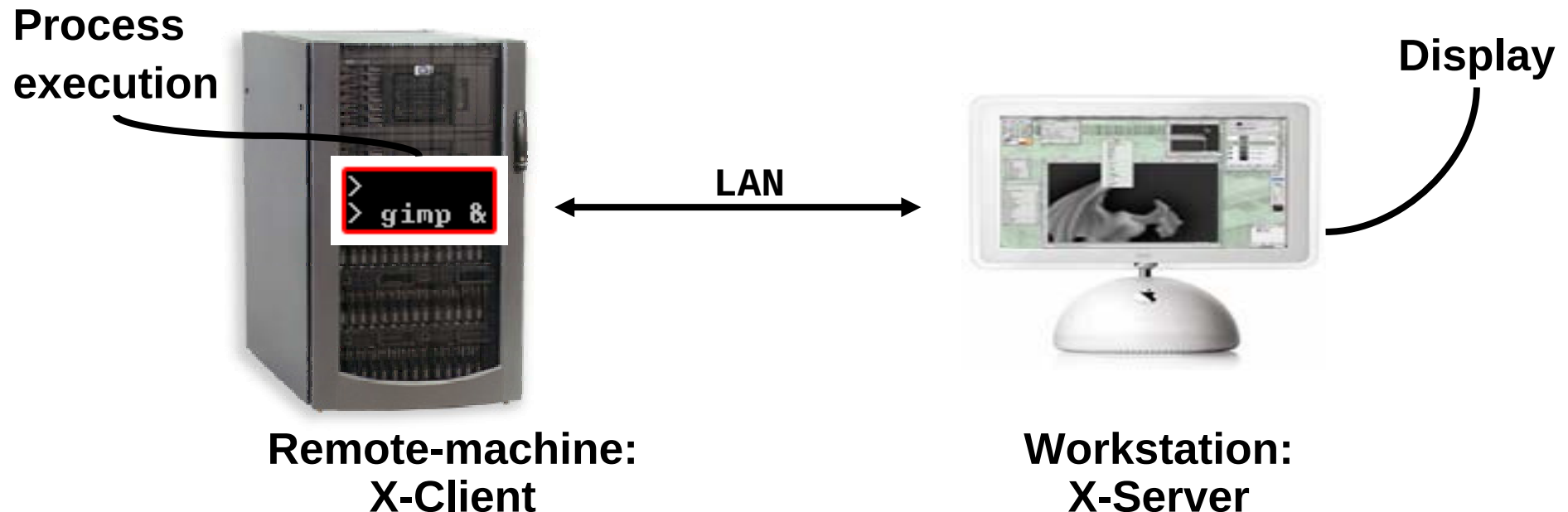
X



Y

Gestions de fenêtres sous Unix

Sous Unix, la gestion de l'interaction homme-machine (écran, souris, clavier) repose sur le système «X Window» (initulé X11 ou X). X repose sur un concept client-serveur. Des programmes graphiques peuvent s'exécuter sur une machine distante en utilisant le clavier et l'écran connectés à la machine locale.



Gestions de fenêtres sous Unix (2)

La communication avec le serveur X est offerte par la librairie Xlib qui met à disposition des fonctions de bas niveau:

- Gestion de fenêtres et événements**
- Gestion de contexte graphique (couleur courant, taille de lignes, police de caractères courante,...)**
- Fonctions de dessin primitives (points, lignes, texte,...)**

Pour utiliser des menus, dialogues, boutons, etc. sans les programmer soi-même, faire appel à des librairies de plus haut niveau telles que Tk, Motif, GTK etc...

Xlib

La programmation d'interface graphiques est dite « asynchrone » car le programme réagit sur les événements (touche clavier, déplacement de la souris, ...) .

La structure d'un programme « event-driven » comprend:

- 1. Initialisation (ouverture de fenêtres,...)**
- 2. Boucle infinie**
 - (i) Attendre un événement**
 - (ii) Traiter l'événement**
- 3. Terminaison (fermeture des fenêtres, libération des ressources, ...)**

Xlib example (1)

Exemple *x_simple.c*

Compiler : > gcc x_simple.c -o x -lX11 -L/usr/X11R6/lib

#include <stdlib.h>

#include <X11/Xlib.h> // Every Xlib program must include this library

int main(void)

{ // Open the display

Display *dpy = XOpenDisplay(NULL);

if (dpy == NULL) { puts("Could not connect to display"); return 1;

}

// Get black and white for the target display

int blackColor = BlackPixel(dpy, DefaultScreen(dpy));

int whiteColor = WhitePixel(dpy, DefaultScreen(dpy));

// Create the window descriptor

Window wdo = XCreateSimpleWindow(dpy, DefaultRootWindow(dpy), 0,
0,200, 100, 0, blackColor, blackColor);

// Selects the MapNotify events on which the program waits

XSelectInput(dpy, wdo, StructureNotifyMask);

Xlib example (2)

// Displays the window (maps from memory to display memory)

```
XMapWindow(dpy, wdo);
```

// Waits for the MapNotify event specifying that window is on display

```
for(;;) { XEvent e;XNextEvent(dpy, &e);  
    if (e.type == MapNotify) break;  
}
```

// Creates a "Graphics Context"

```
GC gc = XCreateGC(dpy, wdo, 0, NULL);
```

// Tell the GC we draw using the white color

```
XSetForeground(dpy, gc, whiteColor);
```

// Draw the line of coo given as parameters

```
XDrawLine(dpy, wdo, gc, 10, 60, 180, 20);
```

// Send the "DrawLine" request to the display server,

```
XFlush(dpy);
```

// Wait for 5 seconds to verify that the line has been displayed

```
sleep(5);
```

```
return 0;
```

```
}
```

Xlib example (3)

```
Display *dpy = XOpenDisplay(NULL);
```

Ouvre une connexion vers le serveur X (qui peut être sur une machine distante) et l' « écran » par défaut. Le descripteur de la connexion *dpy* est passé à tous les appels Xlib.

```
int blackColor = BlackPixel(dpy, DefaultScreen(dpy));
```

```
int whiteColor = WhitePixel(dpy, DefaultScreen(dpy));
```

Determine la représentation des couleurs noir et blancs, dont la représentation dépend du type de l'affichage (n/b, niveaux de gris, 256 couleurs, 24bit RGB, ...)

```
Window wdo = XCreateSimpleWindow(dpy, DefaultRootWindow(dpy), 0, 0,  
                                200, 100, 0, whiteColor, blackColor);
```

Crée une fenêtre principale (=comme enfant du fenêtre racine) à la position (0/0), de taille 200 x 100 pixel, avec un canvas de 0 pixels en blanc et avec la couleur de fond noire.

```
XSelectInput(dpy, wdo, StructureNotifyMask);
```

Demande que notre application reçoit des événements concernant le changement de la « structure » de la fenêtre comme la création et l'apparition.

```
XMapWindow(dpy, wdo);
```

Demande l'apparition de la fenêtre sur le serveur.

Xlib example (4)

```
for(;;) { XEvent e;XNextEvent(dpy, &e);  
    if (e.type == MapNotify) break;  
}
```

Cette boucle assure que l'on ne continue que si le fenêtre est apparue à l'écran. Les autres événements sont ignorés.

```
GC gc = XCreateGC(dpy, wdo, 0, NULL);
```

Crée un contexte graphique pour notre fenêtre avec les valeurs par défauts.

```
XSetForeground(dpy, gc, whiteColor);
```

Défini la couleur "blanc" pour le "foreground"

```
XDrawLine(dpy, wdo, gc, 10, 60, 180, 20);
```

Dessine une ligne du pixel (10, 60) à (180, 20).

```
XFlush(dpy);
```

Demande l'affichage de tous les commandes pas encore exécutées.

lspgraph (1)

La librairie lspgraph (utilisée pendant les exercices) permet l'affichage de points et de lignes sans devoir s'occuper de X11 et de la gestion d'événements. Elle introduit un système de coordonnée « mathématique » (axe X vers la droite, Y vers le haut, coordonnées réelles).

Fonctions pour gérer la fenêtre:

Créer (et ouvrir) la fenêtre:

```
void createWindow(const char *name, int width, int height);
```

Fermer la fenêtre:

```
void closeWindow(void);
```

Rafraichir le contenu de la fenêtre: (=forcer l'affichage du contenu)

```
void flushWindow(void);
```

Effacer le contenu de la fenêtre:

```
void clearWindow(void);
```

lspgraph (2)

Functions pour modifier le contenu de la fenêtre:

Definition des limites du systèmes de coordonnée représenté dans la fenêtre:

```
void setAxis(double xmin, double xmax, double ymin, double ymax);
```

Definition de la couleur courante: (valeurs entre 0 et 1)

```
void setColor(float red, float green, float blue);
```

Taille des lignes:

```
void setLineWidth(int width);
```

Dessiner un point:

```
void drawPoint(double x, double y);
```

Dessiner une ligne:

```
void drawLine(double sx, double sy, double ex, double ey);
```

Dessiner une chaîne de caractères:

```
void drawString(double x, double y, const char *str);
```

Afficher les axes:

```
void drawAxis(void);
```

lspgraph (3)

Functions pour modifier le contenu de la fenêtre:

Afficher un bitmap:

```
void drawBitmapRGBX(void *buf, int width, int height);
```

```
void drawBitmapRGB(void *rgb_buf, int width, int height);
```

```
void drawBitmapGray(void *buf, int width, int height);
```