# 技術角度看輕量級桌面系統

**Jim Huang**（黃敬群 / “jserv”）
http://blog.linux.org.tw/jserv/

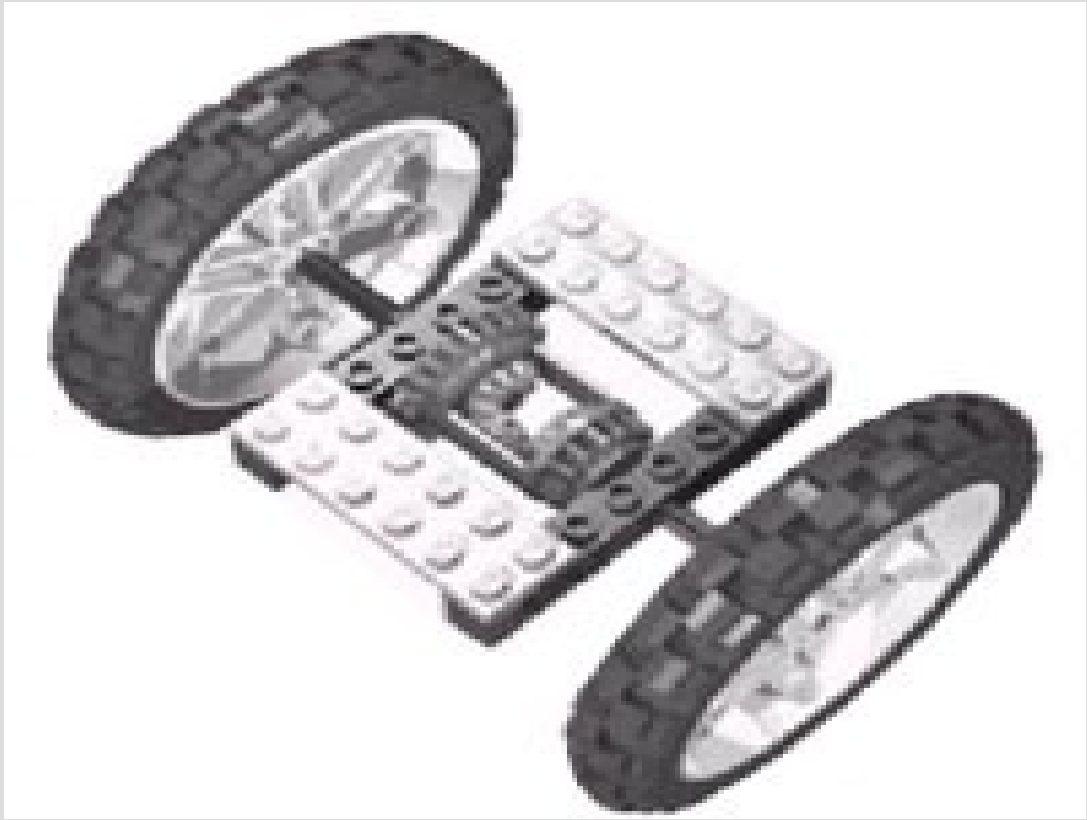
Oct 28, 2006

# 等等 ... 爲何要輕量級桌面？

# 想想自身 ...

- 硬體以 Moore's Law（IC 可容納的電晶體數目，約每隔 18 個月便會增加一倍，性能也將提升一倍） 進步
- 然而 ...
  - 「時代在變、技術在變，**苦苦等待的時間依然不減**」
- 環保 綠色工業

# 所以我們又要發明新輪子？！



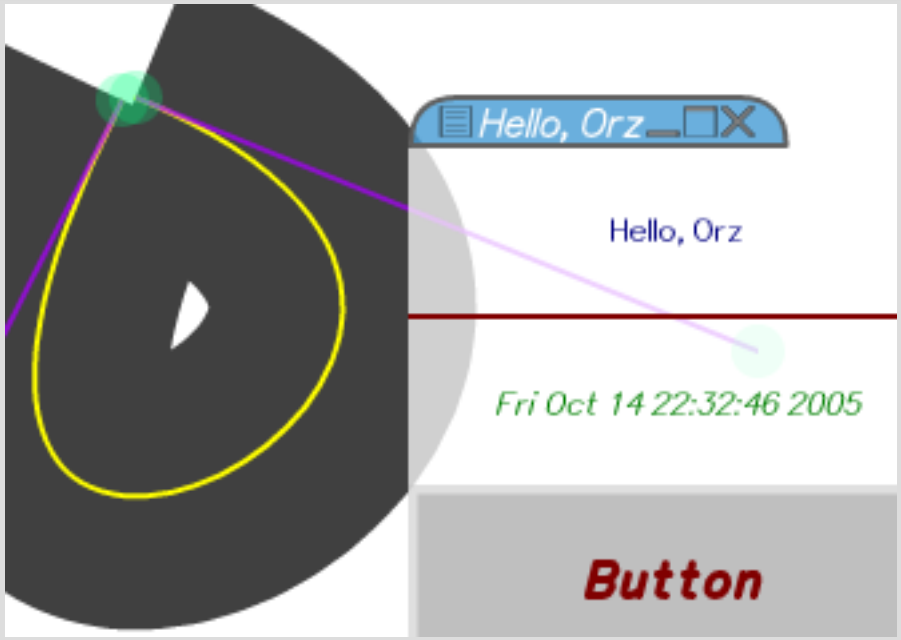
重新設計一個大怪物，也無濟於事

# 思維方式

- 體驗具體而微的桌面系統之設計
- 檢視與分析既有系統
  - Linux Kernel 的改進
  - 架構於 X Window System 的 KDE 與 GNOME
- 捲起袖子，動手作！

# Xorz/Embedded

- Window system
- GUI Framework
- Window Manager
- RGB/ARGB windows
- Composited
- stroke-based font




「沒錢買硬碟，所以只好把程式寫得小巧」

# 眞正走入桌面應用的 Linux 2.6

- 不只是效能改進
  - 引入針對桌面應用的新機制
- kevent (Kernel Event Layer)
- Improved Hotplug
- inotify

# inotify

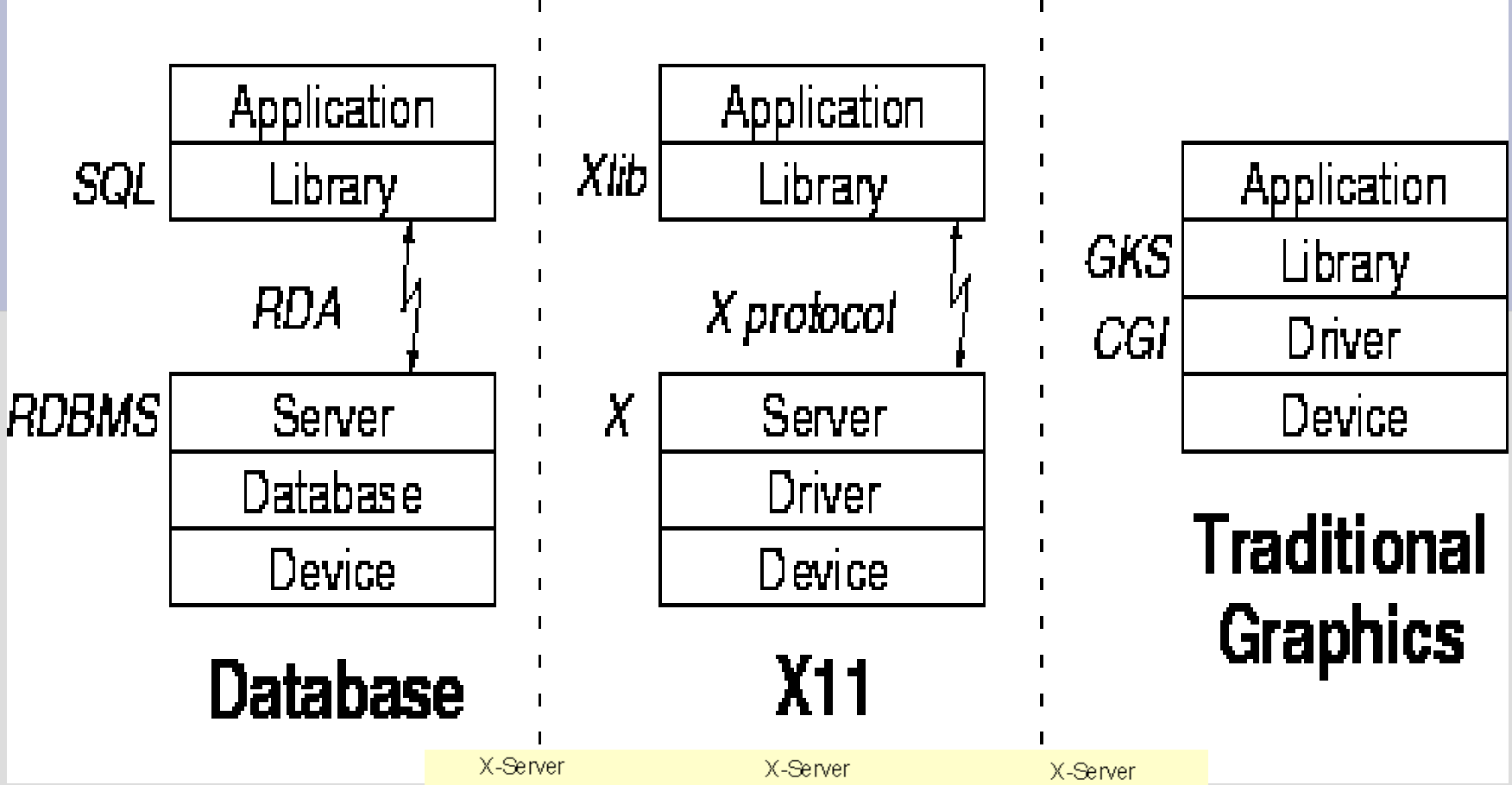
- 取代舊有的 dnotify( 原本的檔案監視機制 )

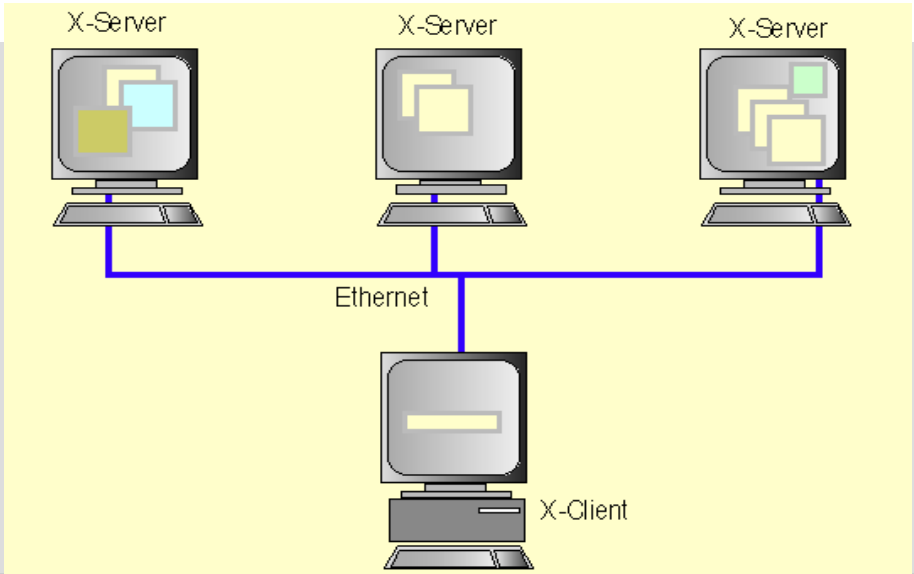
```
struct inotify_event {
    __s32       wd;     /* watch descriptor */
    __u32       mask;       /* watch mask */
    __u32       cookie;     /* cookie to synchronize two events */
    __u32       len;        /* length (including nulls) of name */
    char        name[0];    /* stub for possible name */
};
```

- 特色
  - 單一 file descriptor
  - 可配合 select() 與 poll()

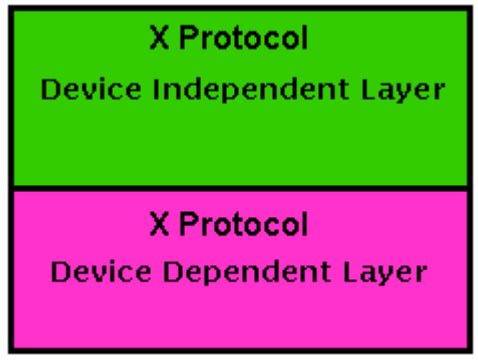
- 使得 Desktop Search 成爲可行 (Beagle)
- Indexing

用以處理桌面應用所需之大量檔案監視

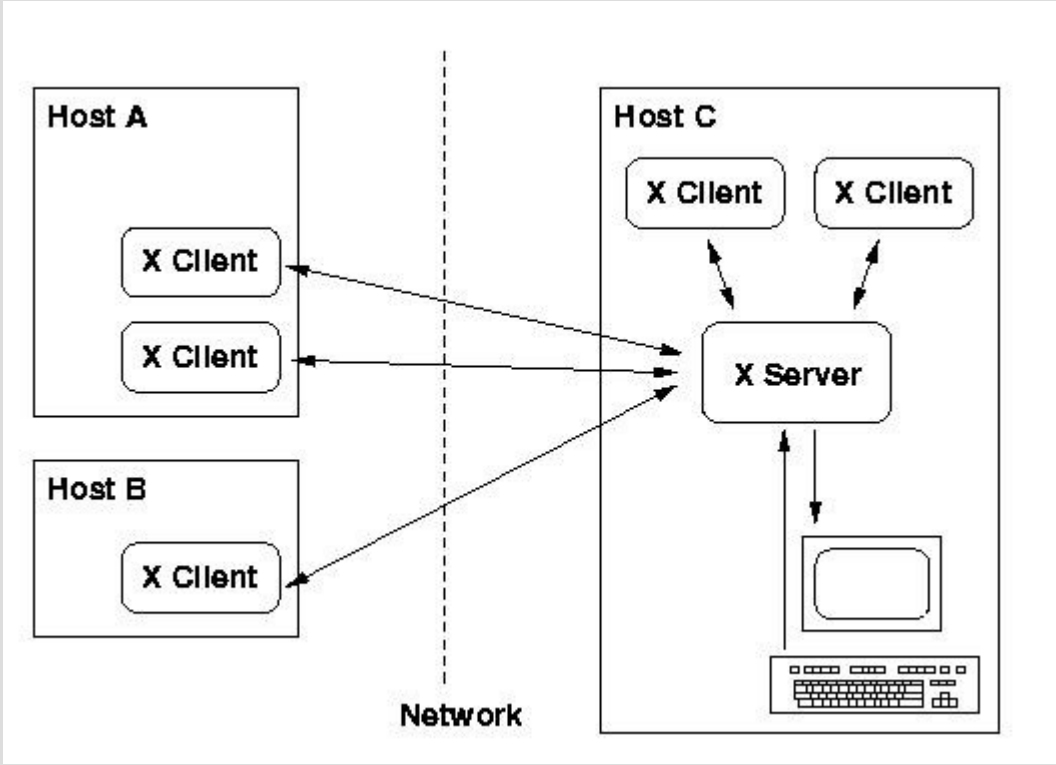

Application
SQL
Library
RDA
RDBMS
Server
Database
Device
Database
Application
Xlib
Library
X protocol
X
Server
Driver
Device
X11
Application
GKS
Library
CGI
Driver
Device
Traditional Graphics

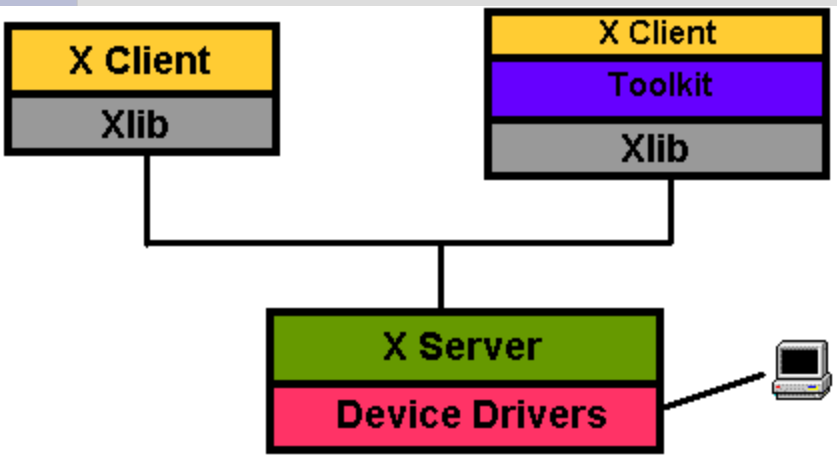

X-Server
X-Server
X-Server
Ethernet
X-Client

# X Protocol




- X 的精髓
- Message-based
- 不限定於程式語言




桌面應用時，其實是
shared memory 形式

# 令人不解的數據

- 在同樣的硬體 (1 Ghz Pentium4) 作基本繪圖運算的比較
- 比較對象

  Linux/GTK+ on xlib

  Win32/GTK+ on GDI

  native GDI on Win32

# 令人不解的數據 (2)

| | Linux/GTK+ | Win32/GTK+ | native GDI |
|---|---|---|---|
| Fill Rect | 3.1 | 19 | 3.7 |
| Line Colored | 14 | 77 | 15 |
| single pixel | 0.2 | 12 | 1.1 |
| Flip | 5103 | 16317 | 7286 |
| ... | | | |

X11 作基本繪圖操作其實很快！

單位：μs
數值越小爲佳

# X 本身不慢，但其餘元件牛步化

- Gtk+ / Qt / Pango
- 低效率的 Theme engine
- 嚴重的 client-side 與 server-side Image/Pixmap 轉換
- 缺乏一致性與可靠性的字型處理
- 「錯誤」的設計想法 (eg. I/O)
- 「自作多情」的機制 (eg. IPC)
- 「孤芳自賞」的應用程式（... 太多 ...）

# 「痛苦」指數

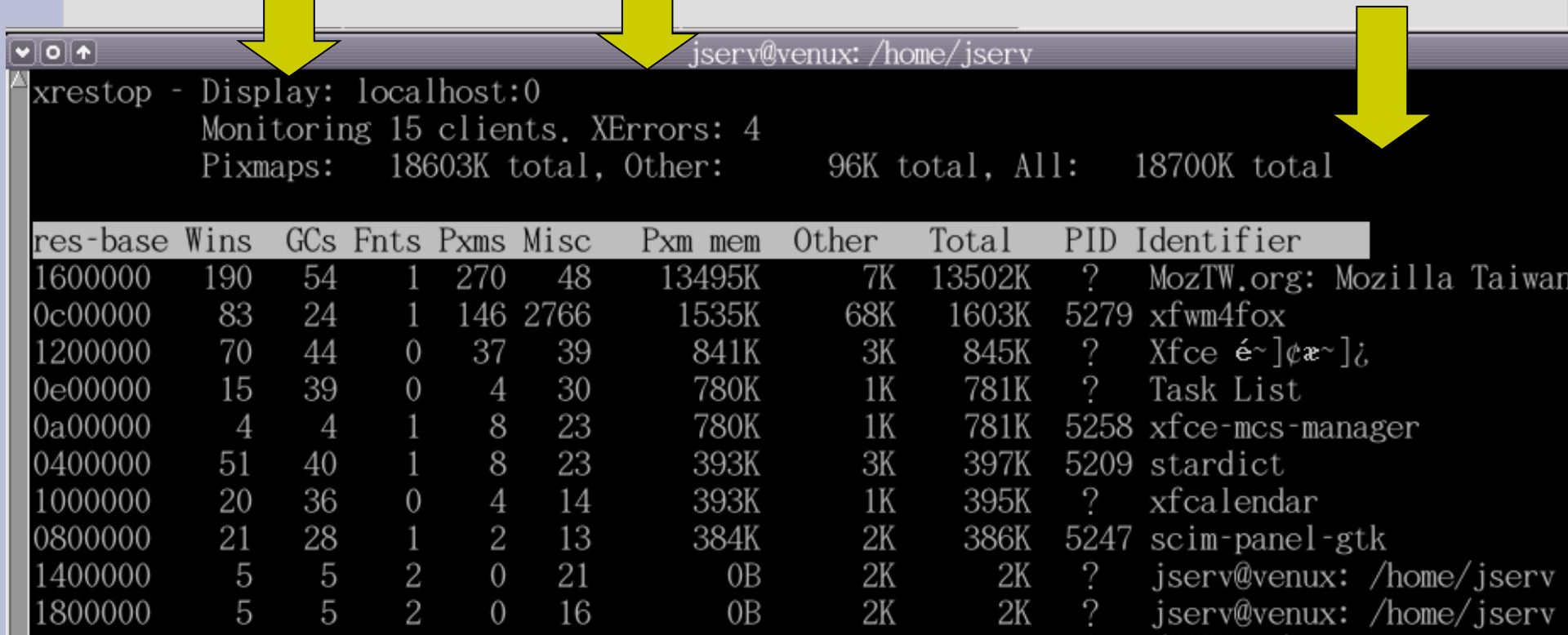
| Level | Latency | Pain Factor |
|---|---|---|
| Register | 1/3 ns | 1 |
| L1 cache | 1 ns | 3 |
| L2 cache | 5 ns | 15 |
| L3 cache | 10 ns | 30 |
| Main Memory | 50 ns | 150 |
| Ethernet | 100 ns | 300 |
| Hard disk (10,000 RPM) | 8,500,000 ns | 25,500,000 |

*...*

** Times are for a 3 GHz processor.*
*** Hierarchy latencies include miss times for previous levels, as applicable.*
**** Hard disk access time = Command Overhead + Seek Time + Settle Time + Latency*

# 觀察實際 X11 運作的情況

- 使用 xrestop 檢視運作中的系統資源分配 ...
  - Pixmaps 、 GC 、 Colormaps
  - Reference: X-Resource extension



# 試著去追蹤 Evolution Mail client

| % time | seconds | usecs/call | calls | errors | syscall |
|---|---|---|---|---|---|
| 7.10 | 0.067166 | 28 | 2373 | 72 | read |
| 3.98 | 0.037691 | 14 | 2764 | 1687 | stat64 |
| 3.47 | 0.032849 | 16 | 2046 | 769 | open |
| 1.84 | 0.017393 | 7 | 2321 | | close |
| 1.81 | 0.017100 | 248 | 69 | | select |
| 1.62 | 0.015363 | 11 | 1364 | | getdents64 |
| 100.00 | 0.331957 | | 17583 | 2567 | total |

系統呼叫的負擔不重(167 ns)，但主要是 I/O-bound，而且是來自 Gtk+ 本身，眞實時間爲 1.02s，換言之，32% 的時間耗費於系統呼叫

參考 Robert Love 的〈Fast and Slick: Optimal GNOME Programming〉

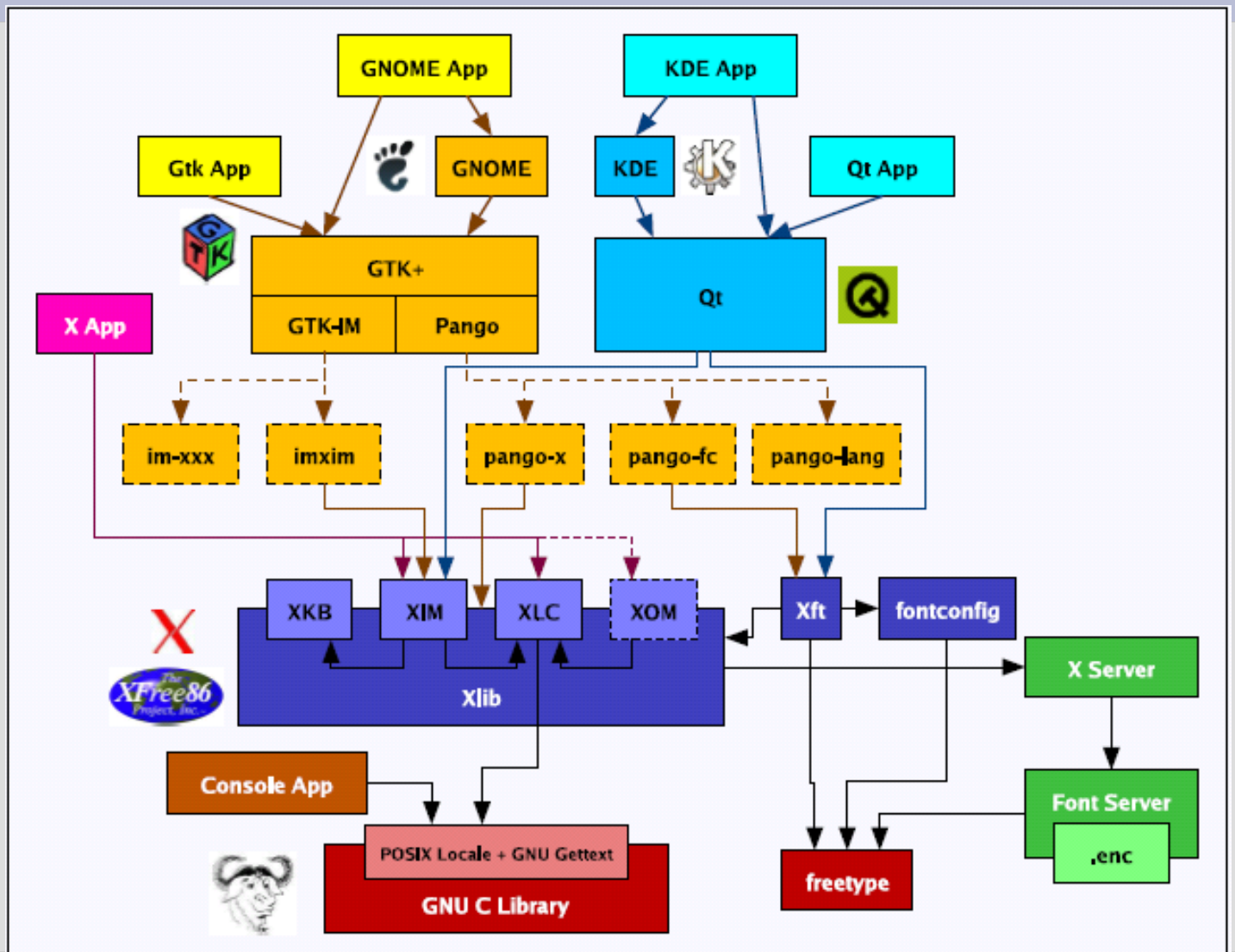
# 啊？哪來的 I/O ？

- 邪惡的 .desktop 檔
- 不全然是描述程式啓動方式，許多用以 MIME type
- 平均而言每個檔案佔 4kb ，內部尚有 L10N
- L10N 考慮國家、語系，以及方言，早已超過六千種
- 越成熟的專案，因此付出的成本越大
- PCMan, please kill them!

# 啊？哪來的 I/O ？(2)

- 愚蠢的 gconf schema
- 上百個小檔案，還以目錄形式存在
- 裡面也有 L10N
- 一般來說，大約佔 25 Mb 的空間，而 GNOME 應用程式大量使用 gconf

# Desktop 的架構


GNOME App
KDE App
Gtk App
GNOME
KDE
Qt App
GTK+
GTK-IM
Pango
Qt
X App
im-xxx
imxim
pango-x
pango-fc
pango-lang
XKB
XIM
XLC
XOM
Xft
fontconfig
X Server
Xlib
Console App
Font Server
.enc
POSIX Locale + GNU Gettext
GNU C Library
freetype

# 桌面環境成功因素

- i18n/L10n 能力
- 良好的 Framework/Toolkit 支持
- 多樣化的 profile/configuration 處理
- 高擴充性、延展性、模組化設計
- 應用程式間的互動性
- 快速更換不同外觀、環境、設定，以及各種展現
- 與週邊裝置（硬體）的整合度
  - 對應硬體需要有足夠的抽象化設計
- 依循國際標準